Pulmonary Embolism Classification Using CT Images (and JMP Pro!)

Marie Gaudard
North Haven Group

Abstract

This paper explores some of the new advanced predictive features of JMP Pro Version 9 in the context of a medical data mining application. Bootstrap forests, boosted trees, and neural models, as well as the use of the JMP Scripting Language, are employed in an effort to identify pulmonary embolisms using three-dimensional computed tomography angiography (CTA) data. The data we use for our training and validation sets formed the basis for the 2006 KDD (Knowledge Discovery and Data Mining) Cup competition. These data present interesting challenges: sparse and noisy data, multiple regions associated with a single pulmonary embolism, a spatial structure within patients, and non-traditional measures of sensitivity. We develop predictive models using the training data, choose final models, apply these models to the separate test set in order to assess its performance, and compare our results with KDD Cup entries.

Background for Pulmonary Embolism Study

Pulmonary Embolism

The term ***pulmonary embolism*** (***PE***) refers to a blockage of an artery in one of the lungs. Such a blockage is caused by an ***embolus***, that is, an object (usually a blood clot) that travels to the lungs through the bloodstream, lodges in an artery, and blocks it.

Incidence estimates of 69 per 100,000 (Silverstein et al., 1998) and 23 per 100,000 (Anderson et al., 1991) can be found in the literature, with incidence increasing markedly with age (Horlander et al., 2003). Anderson quotes a study that estimates that PE is the primary cause of 100,000 deaths per year in the US.

The Merck Manual of Geriatrics cites an incidence of 1.3 per 1000 in the 65 – 69 age group. The mortality rate for hospitalized patients who are 65 years and older is estimated to be 21%. The recurrence rate is estimated to be 5 to 10%, with a total one-year mortality rate of 39%.

With this degree of morbidity and mortality, it is not surprising that efforts focus on early and accurate detection of PE. Contrast-enhanced computed tomography angiography (***CTA***) has become a preferred diagnostic tool. The contrast material dissolves in the blood, causing the blood vessels to appear bright on the scan. The embolus does not absorb the contrast material, causing it to appear as a dark area.

Candidate Identification

Manual identification of the dark spots that might correspond to PEs is difficult and time-intensive. Even once likely PE-related dark spots are identified, it is difficult to determine if these spots truly reflect PEs rather than other artifacts such as vessel-boundary effects, suboptimal contrast enhancement, lymphoid tissue, flow voids in the veins, or respiratory motion artifacts (Bouma et al., 2009).

The KDD Cup training data set consists of 3038 candidates for PEs. These are classified and labeled as PEs by experts based on ***segmentation***. Segmentation utilizes algorithms to delineate and map out the arteries within the lungs (Pichon et al., 2004; Masutani et al., 2000, 2002; Sluimer et al., 2006). Segmentation is not perfect and can lead to errors. Figure 1, from Pichon et al. (2004), shows a segmented view of the arterial tree within a lung and a number of candidates that might or might not involve PEs. Figure 2, from Masutani et al. (2002), shows some examples of PEs.

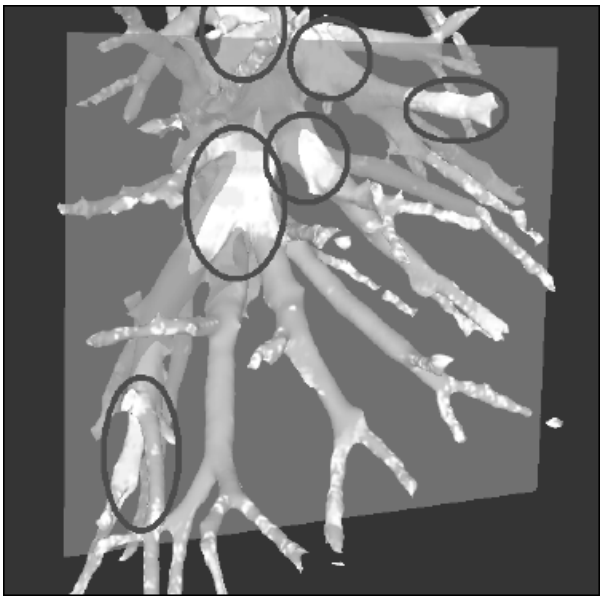

Figure 1 Candidates for PEs Identified in the Arterial Tree (Image from Pichon, et al., 2004)

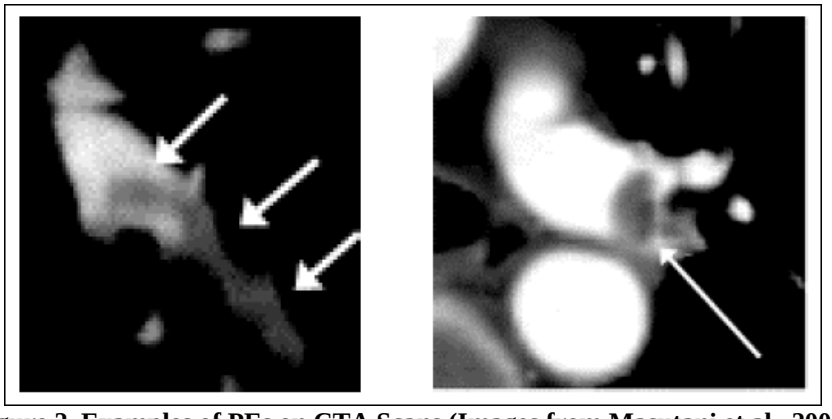

Figure 2 Examples of PEs on CTA Scans (Images from Masutani et al., 2002)

The noise in the response is one of several challenges presented by the KDD training data. A listing of challenges includes:

- The data are sparse in the sense that, although 3038 candidates are measured, only 46 patients are represented.
- Many features are highly correlated with other features.
- Many feature distributions have high skewness and kurtosis.
- There is spatial correlation. The candidates themselves are likely to be close to true PEs. Also, it is reported that some arteries are likely to produce false positive candidates.
- The data are unbalanced. There are far more negative than positive candidates.
- As already mentioned, there is noise in the labeling of candidates due to the fact that segmentation and expert judgment can lead to errors.

In developing and selecting models to meet the KDD criteria, we focus on a number of these issues. Our main emphasis, which will require us to conduct simulation studies, is on the issue of unbalanced class sizes. We touch on the issue of spatial correlation by including information about a single nearest neighbor in our list of potential model inputs. We will not be concerned with multicollinearity as we use techniques that are not sensitive to multicollinearity. Furthermore, since we are interested in a classification, and not a descriptive model, we are not concerned with which predictor of a correlated collection appears in a model.

In our study of potential algorithms, we will consider bootstrap forests, boosted trees, and neural models. In connection with the high skewness and kurtosis of many of the feature distributions, we note that decision tree models are considered fairly robust to these issues (King et al., 1995). However, neural models are suspected to be affected by highly non-normal input variable distributions.

The KDD Cup Tasks

The 2006 KDD Cup contest focused on separating true PEs from look-alikes. The data presented for the competition consist of measurements on image regions that are suspected to contain PEs. These candidates have been labeled as corresponding to PEs or not by experts using segmentation, as mentioned earlier. The focus of the contest is to develop classifiers that satisfy various conditions relative to this task.

Treatment of PEs is systemic and usually involves a regimen of blood-thinners. For this reason, the prevention of over-diagnosis is important. According to the 2006 KDD Cup guidelines, a physician reviews the results of computer-aided detection algorithms before prescribing treatment. Thus a good classifier will identify PEs with a minimum of false positives.

The 2006 KDD Cup contest involved three main tasks (KDD Cup Task description), each with three subtasks defined by false positive thresholds. In this paper, we will focus on two of the main tasks and all three false positive thresholds for these two tasks. The two main tasks are: Task 1, which involves optimizing PE sensitivity; and Task 2, which involves optimizing patient sensitivity. The subtasks require that sensitivity be optimized while maintaining no more than 2, 4, and 10 false positives per patient (subtasks (a), (b), and (c), respectively). (A detailed discussion of the contest is given in Lane et al.)

Although we will address both tasks, our strategy will be to find algorithms that have good PE sensitivity performance. We will then use these algorithms, which have been developed to classify PEs for Task 1, to classify patients for Task 2 as well. We will select three different models, each of which optimizes PE sensitivity subject to meeting one of the three false positive thresholds.

Description of Data

The training data set (available at Task Rules) consists of 3038 records with 118 measured features. These include a Patient ID and a label column, where label contains information that indentifies PEs. Thus there are 116 features that can be used for classification. These features are not described in detail, but, based on their names, one can conclude that they describe a candidate's location, size, shape, and intensity, as well as shape and intensity in a neighborhood of the candidate, and other relevant quantities.

Candidates that are associated with a PE are assigned a three-digit PE label. Candidates not associated with a PE are given the label "0". We learn that:

- There are 2675 candidates not associated with PEs.
- There are 363 candidates associated with PEs. (So only 13.6% of the candidates are associated with PEs.)
- There are 142 identified PEs in the training data. The numbers of candidates associated with each of the 142 PEs are summarized in Table 1. For example, there are exactly 3 candidates associated with each of 12 pulmonary embolisms. In 64 cases, a PE is associated with only a single candidate.

Number of Candidates	Number of PEs	Proportion
1	64	0.451
2	40	0.282
3	12	0.085
4	7	0.049
5	6	0.042
6	4	0.028
7	1	0.007
8	1	0.007
9	1	0.007
10	1	0.007
12	1	0.007
13	2	0.014
14	1	0.007
15	1	0.007
Total	**142**	

Table 1 Number of Candidates Associated with PEs

- There are 46 patients represented in the data.
- The numbers of candidates per patient range from 10 to 227, as shown in Figure 3.

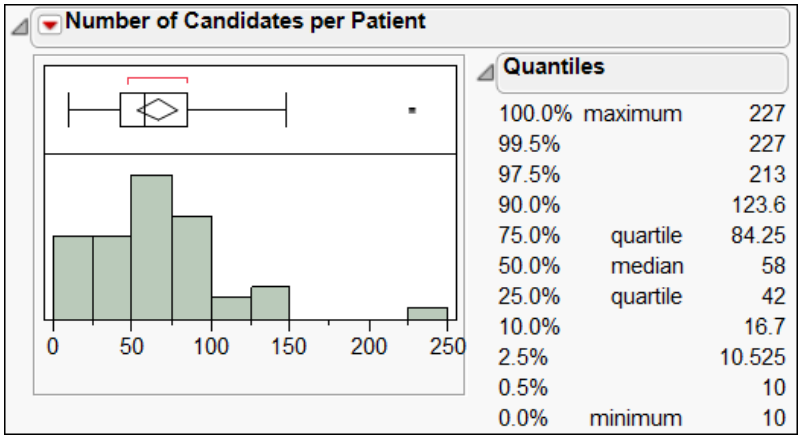

Figure 3 Number of Candidates per Patient

- There are 38 patients with at least one PE; 8 patients have no PEs.
- Each of the 46 patients has some negative candidates. These range in number from 9 to 210.
- The numbers of PEs for the 38 positive patients range from 1 to 11 (Figure 4).

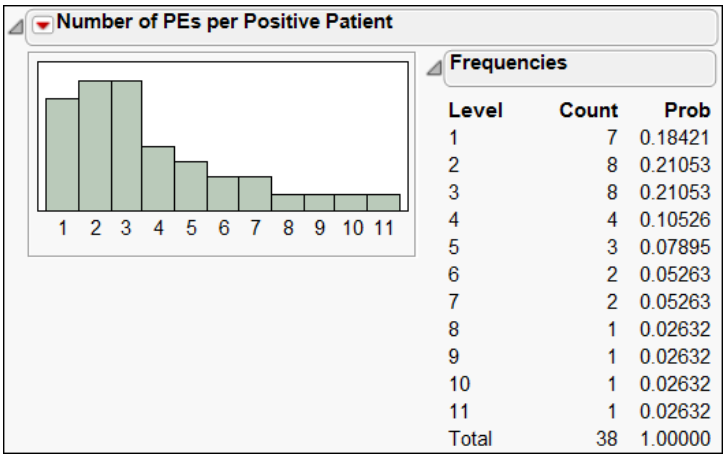

Figure 4 Number of PEs per Patient for the 38 Patients with PEs

We define a patient to be positive if that patient has at least one identified PE. Otherwise, a patient is considered negative. Table 2 summarizes the numbers of positive and negative candidates and patients, and the number of PEs.

	Candidates	PEs	Patients
Positive	363	142	38
Negative	2675	Not meaningful	8

Table 2 Summary of Positive/Negative Frequencies

Criteria for Evaluating Classifiers

Definitions

Typical criteria for evaluating medical tests include sensitivity, specificity, and false positive rate, as well as other measures. These terms are based on the terminology summarized in Figure 5.

Gold Standard	Test Outcome	
	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

Figure 5 Test Outcomes

Sensitivity = TP/(TP + FN)
Specificity = True Negative Rate = TN/(FP + TN)
False Positive Rate = 1 – Specificity = FP/(FP + TN)

These definitions don't apply directly to the pulmonary embolism detection problem, since several candidates can be associated with a single PE. For positive events, the KDD Cup contest defines criteria at the level of PEs and patients. However, a definition of false positive rate makes sense at the candidate level.

Candidate Level Measures

The false positive rate is defined by the 2006 KDD Cup rules as the average across all patients of the total number of candidates incorrectly labeled as positive within a patient. We call this measure as the ***KDD false positive rate.*** The KDD cup tasks involve defining classification systems that do not exceed three levels of KDD false positive rates: 2 per patient, 4 per patient, and 10 per patient.

In our study of classifiers, we will use the more standard definition of false positive rate. Namely, the ***false positive rate*** is the number of false positives divided by the total of false positives and true negatives. In other words, the false positive rate is the proportion of negative candidates that are classified as positive. The false positive rate is also one minus the specificity.

PE Level Measures

A PE is considered to have been detected if at least one of its associated candidates is correctly classified. The KDD Cup competition defines PE sensitivity as the mean number of PEs identified over all patients in the study. We will refer to this notion of sensitivity as ***KDD PE sensitivity***. Task 1 of the KDD Cup competition is to construct three classifiers that maximize KDD PE sensitivity subject to thresholds on the KDD false positive rates of 2, 4, and 10, respectively.

In our study, we will use the usual definition of sensitivity. Namely, we will define ***PE sensitivity*** as the number of PEs correctly identified divided by the total number of PEs exhibited by patients in the study.

Patient Level Measures

A patient is considered to be a true positive if that patient has at least one PE and at least one candidate associated with one of that patient's PEs is labeled as a PE. Task 2 in the KDD Cup contest is to construct three classifiers that maximize the number of true positive patients subject to not exceeding the KDD false positive rate thresholds of 2, 4, and 10, respectively.

Our Goal

In this paper we will address two of the three KDD tasks. Task 1 requires classifiers that optimize PE sensitivity, while Task 2 requires classifiers that optimize patient sensitivity. Both of these tasks have three subtasks, each with a firm requirement that the KDD false positive rate not be exceeded:

- Subtask (a) requires a KDD false positive rate of at most 2 per patient;
- Subtask (b) requires a KDD false positive rate of at most 4 per patient;
- Subtask (c) requires a KDD false positive rate of at most 10 per patient;

Our plan is to explore several potential classification algorithms. From these, we will select classifiers that we will propose for the two KDD Cup tasks. We will relate the performance of our classifiers to the scores obtained by KDD Cup contestants.

For better generalization, in our evaluations of algorithms, we will discuss the false positive rate as it is usually defined (see earlier). To that end, we convert the requirements of 2, 4, and 10 false positives per patient to the ratio we have defined. For us, the false positive rate is given by:

$$FP \text{ Rate} = \frac{\#FPs}{\#FPs + \#TNs}.$$

In the training data, there are 2675 candidates not associated with PEs, in other words, there are 2675 negative candidates. There are 46 patients. A KDD rate of 2 false positives per patient, then, equates to a total of 92 FPs. This evaluates to a false positive rate of:

$$\text{False Positive Rate} = \frac{\#FPs}{\#FPs + \#TNs} = \frac{92}{2675} = .0344.$$

Similar reasoning gives the other two entries in the conversion table below (Table 3).

KDD False Positive Rate	Traditional False Positive Rate
2	0.0344
4	0.0688
10	0.1720

Table 3 Conversion of KDD False Positive Rates to Traditional False Positive Rates

Scoring

The 2006 KDD Cup contest rules indicate that submissions will be evaluated based on their performance on a separate test set. The overall score on Task 1 will be the mean KDD PE sensitivity and on Task 2, the mean number of true positive patients. However, the rules indicate that, if the KDD false positive rate threshold is exceeded on any subtask, the classifier is disqualified for that entire task.

As it turns out, the organizers applied a bootstrap approach to obtain scores. A collection of 200 bootstrapped samples was selected from the test set, and entries were evaluated on all 200 samples. In computing an overall score, if a false positive threshold was exceeded on any one subtask for a bootstrap sample, scores of 0 were assigned for all three subtasks. The resulting scores were averaged. Participants were given 24 hours to submit their classifications for the test set.

Addition of Other Potential Predictors

Three New Variables

Exploratory analysis of the 116 features was conducted. As a consequence of this analysis, two nominal variables and one continuous variable were added to the original list of 116, to bring the total to 119 available features.

Three of the measured features are the x, y, and z location of each candidate. In Figure 6, we show a **Scatterplot 3D** view of the x, y, and z values, with red points representing candidates labeled as PEs and blue points representing candidates not labeled as PEs. The separation of the candidates into two groupings suggests that we can identify each candidate with one lung or the other.

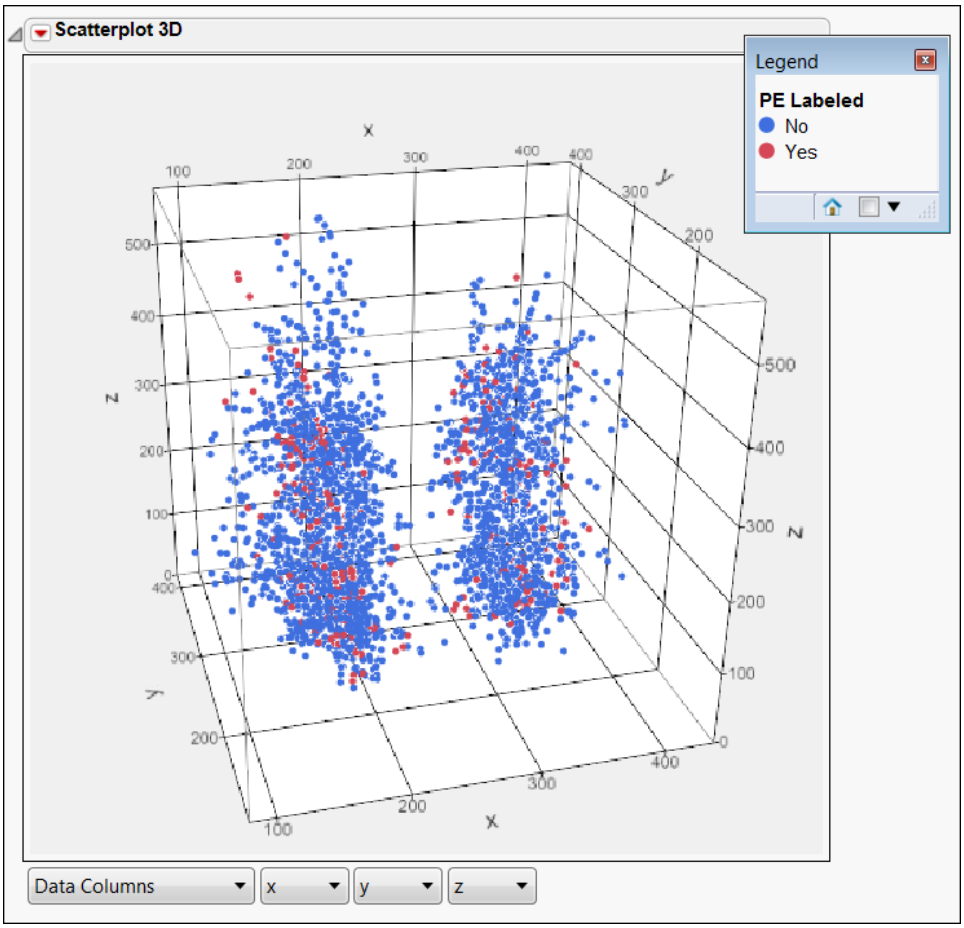

Figure 6 Scatterplot 3D View of Candidate Locations (All Patients)

However, there is another phenomenon that is noteworthy. Patient IDs are four or five-digit numbers. More specifically, each Patient ID begins with either the digits “1”, “21”, or “28”, followed by three digits. These initial digits seem to define three groups. When the spatial positions of candidates are stratified by these leading digits, it is apparent that the location scale differs significantly among the three groups.

We defined a new feature called Patient Group; each candidate was assigned to one of three groups, with values *1*, *21*, and *28*, based on whether it came from a patient whose Patient ID began with the digits 1, 21, or 28. The plot in Figure 7 shows these three groupings of candidates.

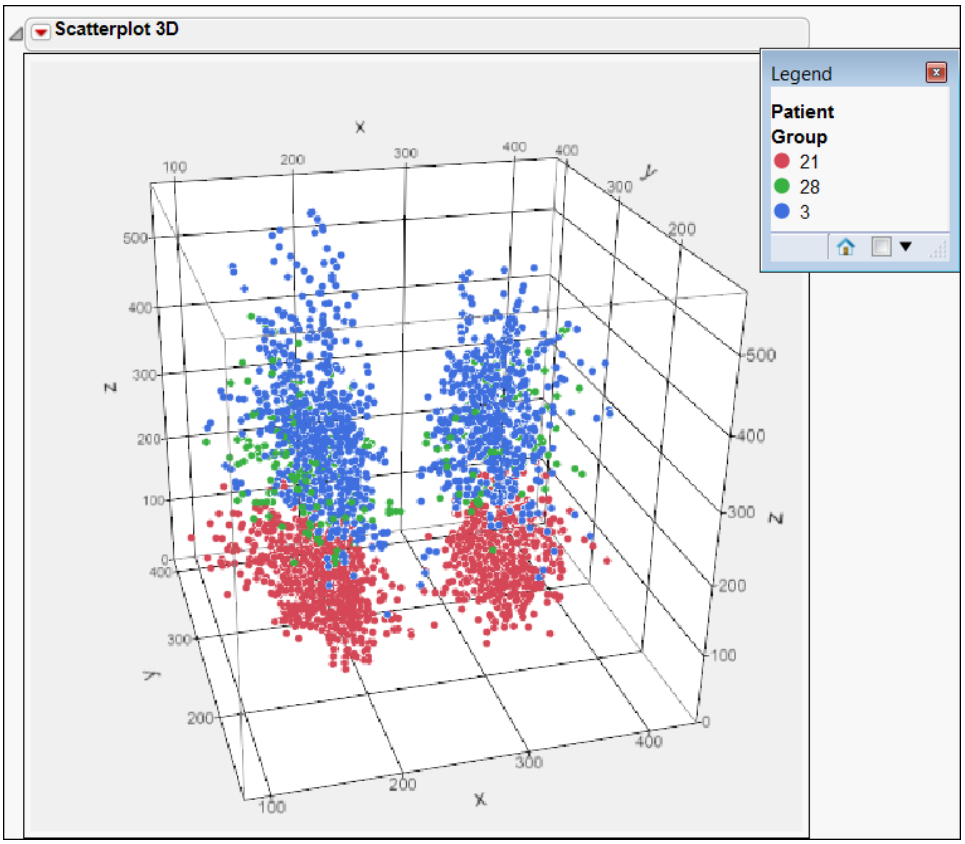

Figure 7 View of Locations for Three Groupings of Patient Codes

The greatest difference occurs along the z axis: Candidates associated with Patient Group *21* have the lowest z values, with Patient Group *28* candidates next, and with Patient Group *3* candidates having the highest values of z. One can only speculate as to the cause, but perhaps the patients differed by area studied, or perhaps the measurements were made using different equipment, different procedures, or in different facilities.

Whatever the case, it seemed prudent to add the variable Patient Group to the list of features so as to allow models to adjust for potential differences in location information for the three groups of patients.

It also seems that information on the lung in which a candidate is located might be useful, since the lung structures are not symmetric. Using a graphical approach, for each of the three patient groups, we defined a simple algorithm for assigning a candidate to one of the two lungs. We called the new feature Lung.

Along with the definition of Lung, we defined a new variable that described the x distance of a candidate to the "center" of the two lungs. Again, this was done graphically and very informally. This new feature is called x Distance to Center.

The total number of features available for classification is now 116 + 3 = 119. In the next section, we double this number.

Nearest Neighbor Features

There is evidence in the literature on pulmonary embolisms that information on neighboring candidates can be useful in classifying a candidate into the PE or non-PE class. For this reason, we wanted to consider information on neighboring candidates in classification models. We took a straightforward approach to doing this.

For each candidate, say candidate A, we identified the candidate that was closest to A in terms of Euclidean distance. We called this the "nearest neighbor" to A. We then augmented A's feature set with 119 features corresponding to A's nearest neighbor: the original 116 features; the defined features Lung and x Distance from Center; and a new feature, the distance to A's nearest neighbor, NN Distance.

With the addition of these 119 nearest neighbor features, each candidate now has 238 features available for modeling.

The Class Imbalance Challenge

Description

Medical data sets used in classification efforts are typically unbalanced. For example, medical scenarios often involve disease diagnosis, where occurrence of the disease is rare in the general population. This results in a two-class problem where one class is represented by a large number of observations and another by relatively few observations. Usually, interest is focused on detecting instances of the smaller, disease-occurrence class.

Class imbalance can be problematic in terms of machine learning algorithms, which tend to assume that classes are balanced. These algorithms tend to weight observations equally and so give a greater "voice" to the larger class.

In a class-imbalanced dataset, other factors that can affect the performance of a classification algorithm include sample size and separability (Sun et al., 2007):

- Sample Size. If the rare class is represented by a large enough sample size, the effect of imbalance is mitigated, since there is more information about the smaller class that can be used in modeling to distinguish it from the larger class.
- Separability. The effect of separability relates to the degree of "overlap" between the features corresponding to the smaller and larger class. If there is little overlap, then imbalance may not be an issue since there are characteristics that easily distinguish the two classes. However, as the degree of overlap in the feature sets increases, imbalance can prove problematic.

Dealing with Class Imbalance

Approaches to addressing the class imbalance problem are discussed in Sun et al. (2007). The authors distinguish two basic approaches: Data-level approaches and algorithm-level approaches. Data-level approaches include various resampling strategies, for example, over-sampling the small class or under-sampling the large class. Algorithm-level

approaches adjust the classification algorithm itself to better deal with the class imbalance problem.

Sun et al. (2007) also discuss cost-sensitive learning approaches, which assume a specific structure of misclassification costs, such as a loss function, reflecting that misclassifying members of the smaller population is more costly than misclassifying members of the prevalent population. Examples of such approaches include weighting the data space, building the loss function into the classifier, and assigning observations to the lowest cost class using Bayes risk theory.

The topic of imbalance is also discussed in Hastie et al. (2001, p. 277), where, in the context of tree-based methods, two approaches are mentioned. One is to apply Bayes rule to terminal nodes to classify observations in a node in a fashion that minimizes loss. The other is to incorporate loss into the tree-growing algorithm by weighting an observation according to its misclassification cost.

Decision Trees and the Effect of Weighting

In our situation, we do not know the loss incurred as a result of misclassification. But we have criteria that must be met, and these criteria implicitly define a loss function. We will explore the idea of weighting observations while fitting tree-based algorithms. Our strategy will be to explore a range of weights in order to determine an optimal weight and algorithm. Specifically, we will test the use of weight functions in conjunction with the partition platform to see if we can obtain higher sensitivity while maintaining the false positive rate at a low level.

Let's begin with a look at how a weight function affects the partition algorithm. Recall that PE Labeled is the Yes/No indicator of whether or not a PE is present. For the purpose of illustration we define a weight function that gives candidates associated with PEs a weight of five, compared to a weight of one for observations not associated with PEs (Figure 8).

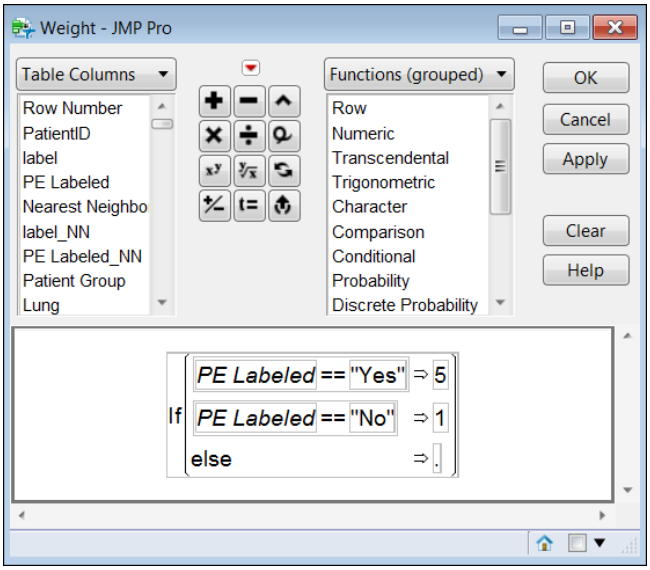

Figure 8 Definition of Weight Column in Formula Editor

In the **Partition** platform launch dialog (found under **Analyze > Modeling**), we request a **Decision Tree** with these entries:

- The new variable, Weight, entered as **Weight**;
- PE Labeled as **Y, Response**;
- All 238 features as **X, Factor**.

The report opens to the view shown in Figure 9.

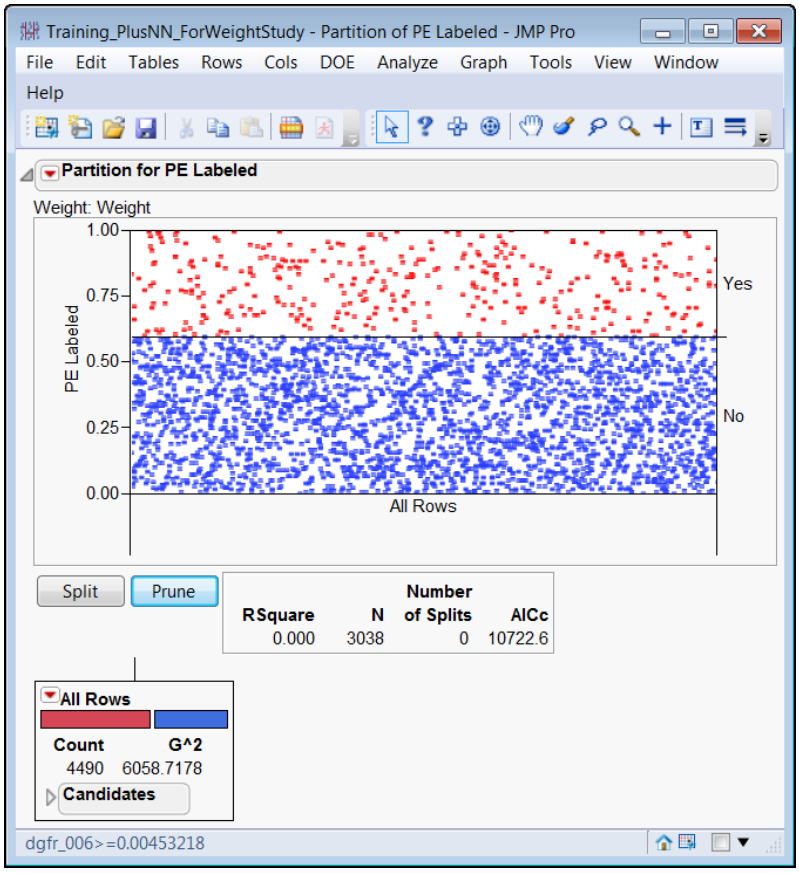

Figure 9 Initial View of Partition Report

Note that we have **N** = 3038 records. However, in the opening node, we see a **Count** of 4490. This is a consequence of the fact that the 363 candidates associated with PEs receive a weight of 5, while the 2675 negative candidates have a weight of 1: 5*363 + 1* 2675 = 4490.

The first split (Figure 10) is on the variable dgfr_006. We will illustrate how this split is obtained.

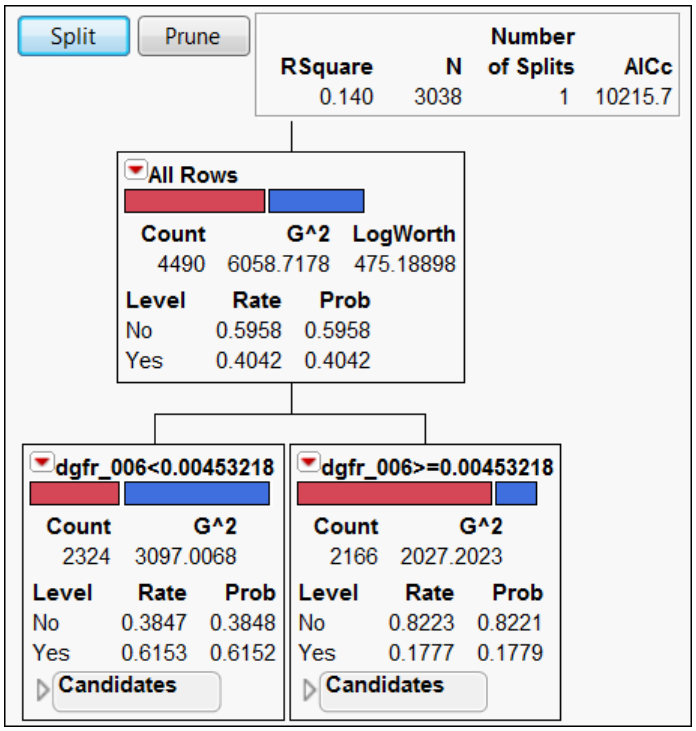

Figure 10 Split on Initial Node

To this end, we define an indicator variable called 1st_Split_WithWeight to have the value *A* if dgfr_006 < 0.00453218 and *B* if dgfr_006 >= 0.00453218. (This is a bit more complicated, because there are tied values at 0.00453218. So, in our definition of *A* and *B* we used the rows that the partition algorithm selected, by choosing **Select Rows** from the node menus.) If we run **Fit Y by X** with PE Labeled as **X** and 1st_Split_WithWeight as **Y**, ignoring the weighting variable, we obtain the contingency table in Figure 11, where both **Count** and **Col %** are displayed.

1st_Split_Wit...

PE Labeled	Count Col %	A	B	
	No	894 75.76	1781 95.86	2675
	Yes	286 24.24	77 4.14	363
		1180	1858	3038

Figure 11 Split Variable versus PE Labeled, No Weight Column

If we use the weighting variable in the **Fit Y by X** launch menu, we obtain the table in Figure 12. Note that the weighting has increased the observed counts in the PE Labeled *Yes* category by a factor of 5, thereby giving these candidates more importance.

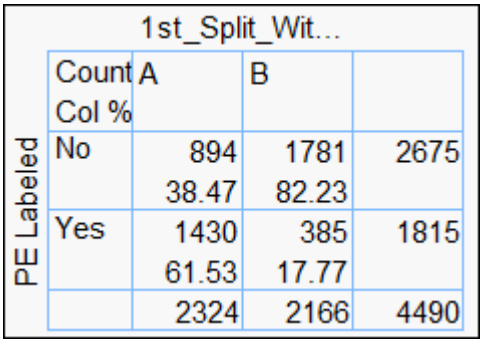

Figure 12 Split Variable versus PE Labeled, with Weight Column

When the response is nominal, at any given point in the tree, the partition algorithm decides which variable to split and where to split its values using the likelihood ratio ChiSquare statistic (G^2) computed from contingency tables such as the one above. (The actual decision is based on the LogWorth statistic, which is a function of G^2.) When a weight variable is used, that variable changes the proportions of observations in the various cells. The computed likelihood ratio statistic, G^2, is based on this weighted number of observations. Using the weight variable defined above, a split on dgfr_006 at 0.00453218 gives the largest value of the LogWorth statistic among all input variables and all splits.

Now let's compare the optimal first split for an unweighted decision tree model to that of the weighted model. Is no weight is employed in the partition model, the first split occurs again on dgfr_006, but the cut point is now at -0.0201714 (Figure 13). The LogWorth criterion has determined that a split of this variable at this value is best among all possible splits.

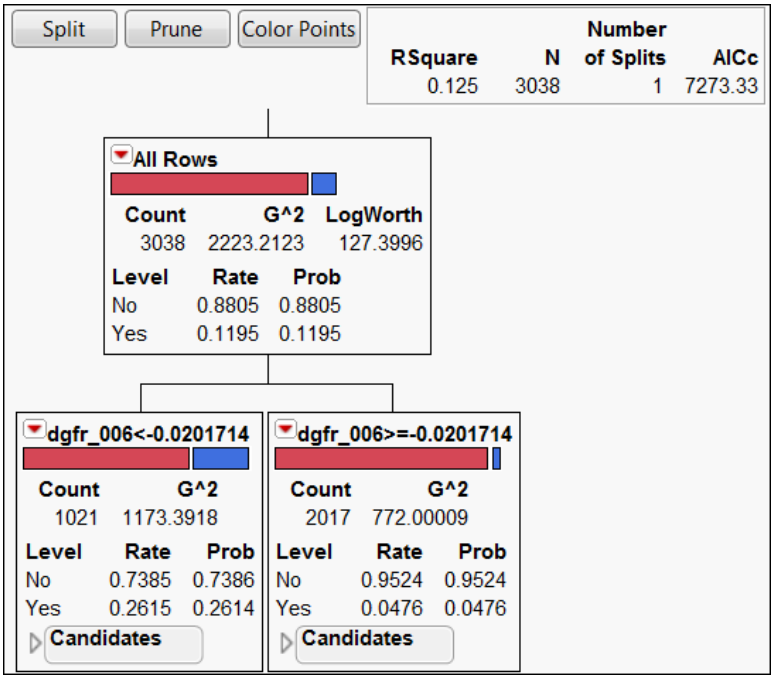

Figure 13 First Split for Model with No Weight

The contingency table corresponding to this split of dgfr_006 is given in Figure 14. Also given are the associated column percents. Note that the column percentage of PE candidates in the *A* portion of the split, dgfr_006 < -0.00201714 is 26.15; this means that candidates in the A portion of the split are given a predicted probability of 26.15 of being PEs. The percentage of candidates in the *B* portion that are associated with PEs, dgfr_006 >= -0.00201714, is 4.76, so these candidates are given a 0.0476 probability of being PEs. Since both of these probabilities are less than 0.50, if we were to have to classify observations with no further splits, all of our candidates would be classified as negatives, namely, as not corresponding to PEs.

1st_Split_No...

PE Labeled	Count Col %	A	B	
	No	754 73.85	1921 95.24	2675
	Yes	267 26.15	96 4.76	363
		1021	2017	3038

Figure 14 Split Variable for No Weight Partition versus PE Labeled

By comparison, with the weighting, the first split gives candidates in the A node a 61.53% probability of being associated with PEs (positives), and candidates in the B node an 82.23% probability of *not* being associated with PEs (negatives).

With respect to the partition algorithm, note that placing a higher weight on the class of positives than on the class of negatives is similar to oversampling the smaller class. By giving the smaller class higher weight than the prevalent class, the ChiSquare-based partition criterion gives the smaller class greater emphasis than it would otherwise receive. This has the potential to increase the true positive rate, but can also increase the false positive rate. The question is whether we can find a good balance between the two.

Simulations Using the Partition Platform

The Plan

We would like to determine if weighting the smaller class can give improved performance in terms of PE sensitivity and false positive rates. We would also like to identify a "best" weighting factor. We will present the results of two studies of the effect of weighting, one study involving bootstrap forests and the other involving boosted trees.

Both studies follow the same format. In the study of bootstrap forests, the smaller class is assigned weights varying from 0.25 to 5.00, incremented in amounts of 0.25. In the study of boosted trees, the weights vary from 0.25 to 8.00. (The weights of .25, .50, and .75 are included simply to see what the effect of underweighting the smaller group might be.) For each weight, 100 simulations are run using each of the bootstrap forest or boosted tree model classes. Each instance of a simulation involves fitting the model to a random sample of 70% of the data and then calculating validation statistics on the 30% holdback

sample. The random training and validation samples are selected anew for each simulation, so that cross-validation statistics are obtained.

The performance statistics are the PE sensitivity and the false positive rate. From the 100 simulated values, we estimate the mean and standard deviation of these quantities. Code for the simulations was written in JSL.

Bootstrap Forests

JMP's ***bootstrap forest*** is an algorithm that averages predictions from a large number of trees. Each tree is fit to a bootstrap sample (random and with replacement) of observations; each split utilizes only a random sample of potential predictors. The algorithm is capable of using validation to terminate growth. Otherwise, trees are grown until a pre-specified number of trees is reached.

In our simulation, for each value of the weight, we fit 100 bootstrap forest models to the response PE Labeled using our entire set of 238 features. **Early Stopping** was selected. Other than that, the default settings in JMP 9 Edition 7, shown in Figure 15, were utilized.

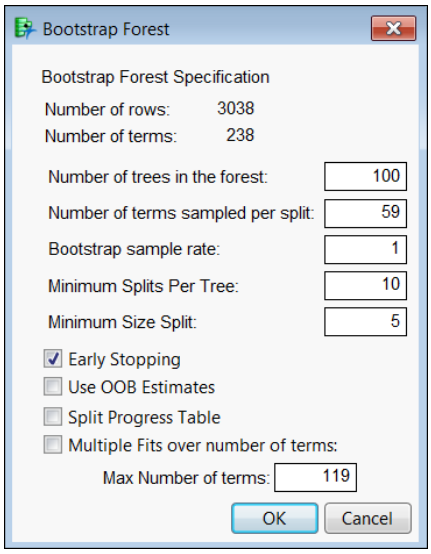

Figure 15 Settings for Bootstrap Forest for 238 Predictors

Prior to each bootstrap forest fit, a 30% random holdback sample was selected; the 70% portion of the observations was used as a training set. The model was fit to the training portion of the data. Once the model was obtained, both the true positive rate for PEs and the false positive rate for candidates were computed by applying the model to the holdback sample. In all, 100 such models were fit for each value of weight.

For each value of weight, the means and standard deviations of these 100 values were computed. Figure 16 shows the mean true positive rates for PEs (PE sensitivity) as a function of the weights applied to the set of positive candidates, while Figure 17 shows the false positive rates as a function of the weights. Both plots show +/-2 standard

deviation limits; these are intended to give a sense of the amount of variation in bootstrap forest fits at each weight value.

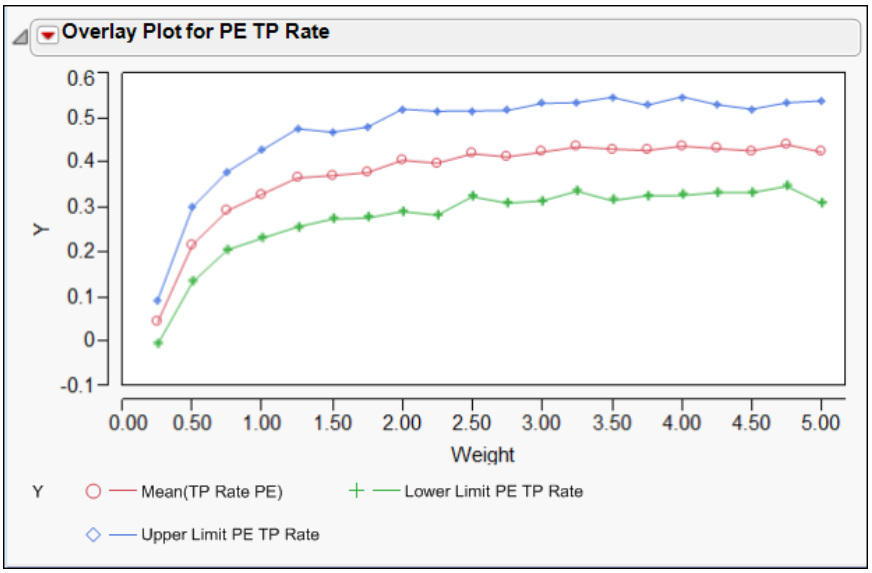

Figure 16 PE Sensitivities for Bootstrap Forest Models by Weight

A weight of 1.0 represents equal weighting of all observations. Weighting the smaller class more than the larger class clearly improves sensitivity. Note that the true positive rate increases until the weight is about 3.0, and then begins to level off. In the interval from 3 to 4, the mean true positive rate averages 0.431.

What about the false positive rate? Figure 17 shows the false positive rate as a function of the weight. A horizontal line has been drawn at the most stringent threshold, a desired false positive limit of 0.034. We see that, for all weights considered, all mean false positive rates, as well as +/-2 standard deviation intervals for these, fall below the level of 0.034.

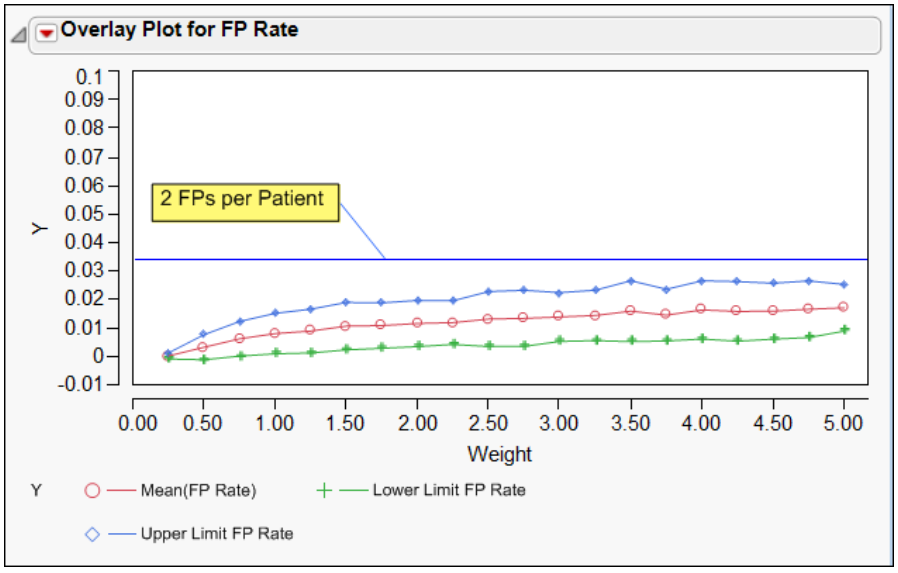

Figure 17 False Positive Rates for Bootstrap Forest Models by Weight

We conclude that, if we want to use a bootstrap forest model to optimize PE sensitivity, we need not be concerned with the false positive rate. We should simply choose a weight that will optimize sensitivity. A choice of 3.5 for the weight appears reasonable. The mean PE sensitivity at this weight, based on our 100 simulations, is 0.430 and the standard deviation is 0.058. Results are summarized Table 4.

Weight	Mean(TP Rate PE)	Lower Limit PE TP Rate	Upper Limit PE TP Rate	Mean(FP Rate)	Lower Limit FP Rate	Upper Limit FP Rate
3.50	0.430	0.315	0.546	0.016	0.005	0.027

Table 4 Summary of Simulation Performance for Bootstrap Forest Model with Weight 3.5

Boosted Trees

The term ***boosting*** connotes fitting models adaptively based on the residuals of previous models. The "boost" comes from increasing weights on misclassified observations and decreasing weights on correctly classified observations. Using this approach, subsequent classifiers are made to focus on the observations that previous classifiers failed to classify correctly. The results of all classifiers are usually combined in a weighted fashion to provide a single final classifier.

In JMP's implementation, a large, additive decision tree is fit using a sequence of smaller trees. Each tree in the sequence is fit on scaled residuals derived from the previous tree. If a validation column is provided or if sufficiently many rows are excluded, the algorithm has the option of terminating tree growth at a point where the validation statistic no longer improves. For a categorical response such as ours, the final predicted probability is a logistic transformation of the sum of the trees that fit the residuals at each stage.

In our simulation study for boosted trees, for each value of weight, we fit 100 boosted trees to the response PE Labeled using our entire set of 238 features. The default settings, shown in Figure 18, were utilized.

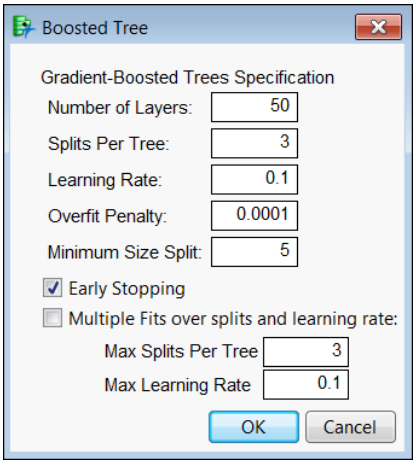

Figure 18 Default Settings for Boosted Tree Models

For each weight and for each boosted tree, the true positive rate for PEs and the false positive rate were computed by applying the models that were fit to the holdout sample. The means and standard deviations of these 100 values were computed. The plot in Figure 19 shows the mean true positive rates for PEs (PE sensitivity) as a function of the weights applied to the set of positive candidates, while the plot In Figure 20 shows the false positive rates as a function of the weights. Both plots show +/-2 standard deviation limits; recall that these are only intended to give a sense of the amount of variation that we might experience in boosted tree fits at each weight value.

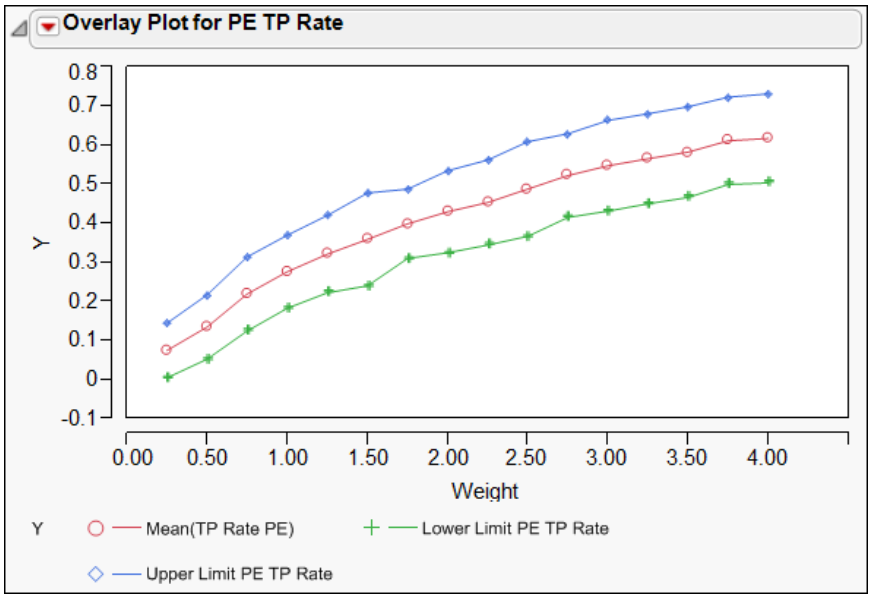

Figure 19 PE Sensitivities for Boosted Tree Models by Weight

Recall that a weight of 1.00 corresponds to a model where no weighting is applied. The true positive rate increases as the weight increases, and it does so much more dramatically than in the case of bootstrap forests. Not unexpectedly, the standard deviation increases slightly as well.

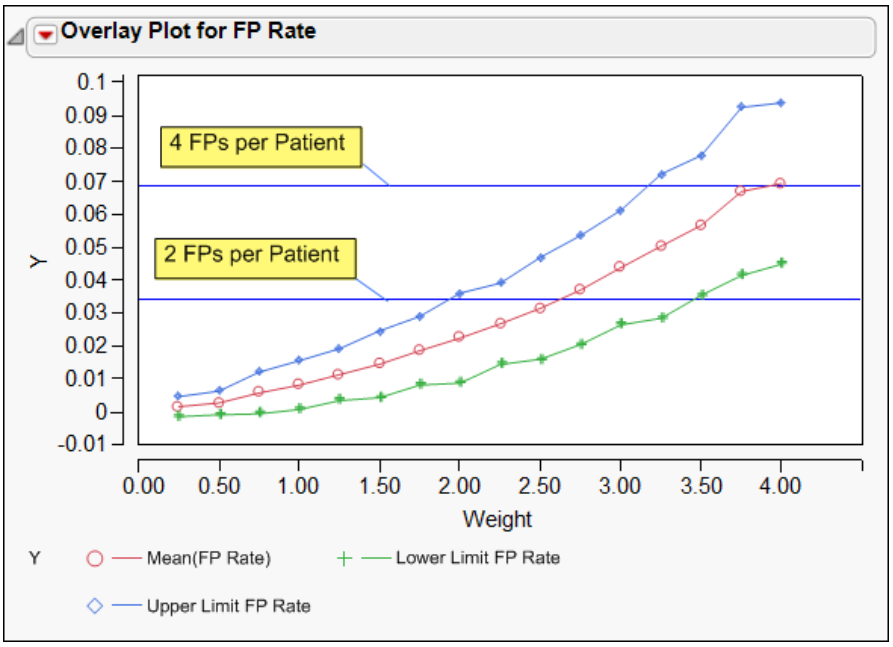

Figure 20 False Positive Rates for Boosted Tree Models by Weight

As one would expect, the false positive rates increase as the weight increases. The standard deviations increase fairly quickly.

Recall that each (a) subtask requires a model that has high PE sensitivity subject to a false positive rate that does not exceed 0.0344 (2 false positives per patient). A horizontal line is drawn at 0.0344 in Figure 20 for this false positive rate. For the (a) subtasks, if we were to fit a boosted tree model with a weight of 1.75 on the positive candidates, we would likely be able to maintain a false positive rate below 0.0344 while achieving a PE sensitivity generally exceeding 0.31. The expected patient sensitivity at a weight of 1.75 is 0.398 and the standard deviation is 0.044. At this weight, the mean false positive rate is 0.019 and the standard deviation is 0.0052. However, this performance is not as strong as that of the weight 3.5 bootstrap forest model, where the PE sensitivity is estimated at 0.430, and where the risk of false positives at that weight is lower.

Figure 20 also shows a horizontal that delineates the threshold for the (b) subtasks, where we need to meet a false positive rate of 0.0688 (4 false positives per patient). A boosted tree model with a weight of 3.25 would appear to meet the false positive requirement with high probability and to do well in terms of PE sensitivity. The mean PE sensitivity for this model is 0.565 with a standard deviation of 0.058. The two standard deviation interval for the false positive rate ranges from 0.029 to 0.072. So, although this interval exceeds 0.0688, it only does so slightly.

Boosted Trees – Extending the Weight Range

In order to address the upper limit on false positives, namely 10 per patient, which we equate to a false positive rate of 0.1720, we extended the range of weights tested with the boosted tree strategy. Specifically, we tested weight ratios of up to 8 to 1, where we weight the positive records 8 times as much as the negative records.

Figures 21 and 22 give plots showing the PE sensitivities and false positive rates for this extended range. Each plot shows the mean and two standard deviation limits computed based on 100 simulations at each weight value.

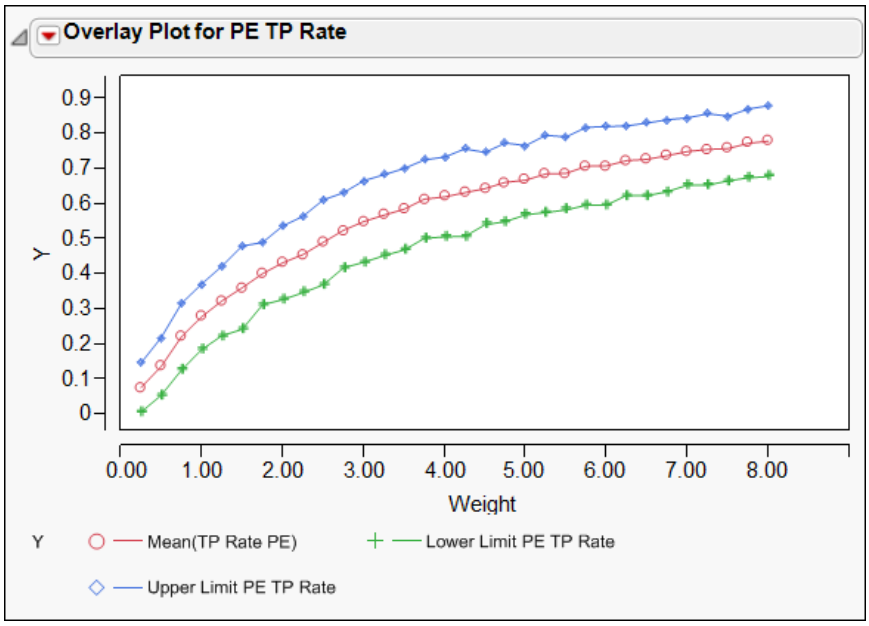

Figure 21 PE Sensitivities for Boosted Tree Models by Weight

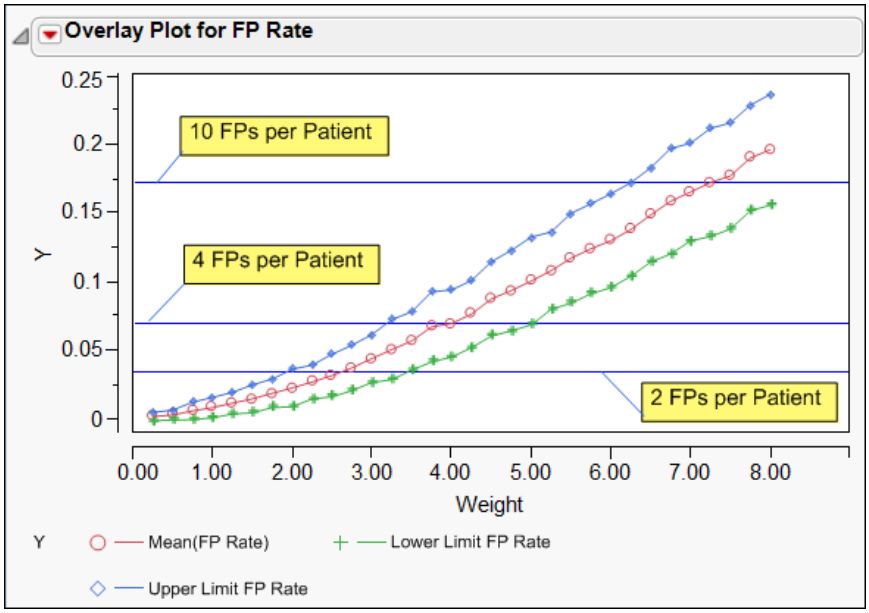

Figure 22 False Positive Rates for Boosted Tree Models by Weight

These plots suggest that the use of boosted trees with weighting is promising in terms of meeting the false positive limits and obtaining good PE sensitivity. Specifically, we see that:

- A weight of about 1.75 will optimize sensitivity while maintaining a false positive threshold of about 2 false positives per patient (0.0344 specificity);
- A weight of about 3.25 will optimize sensitivity while maintaining a false positive threshold of about 4 false positives per patient (0.0688 specificity);

- A weight of about 6.25 will optimize sensitivity while maintaining a false positive threshold of about 10 false positives per patient (0.1720 specificity).

More specifically, the 100 simulations at each of the three selected weight values give the results in Table 5.

Weight	Mean(TP Rate PE)	Lower Limit PE TP Rate	Upper Limit PE TP Rate	Mean(FP Rate)	Lower Limit FP Rate	Upper Limit FP Rate
1.75	0.398	0.309	0.487	0.019	0.008	0.029
3.25	0.565	0.449	0.680	0.050	0.029	0.072
6.25	0.720	0.621	0.819	0.138	0.104	0.172

Table 5 Summary of "Best" Boosted Tree Models

A Brief Exploration of Neural Models

We also considered a number of neural fits. It is not possible to directly include weights or loss functions in the neural fit platform, and so simulations over ranges of weights were not conducted. Since neural models might be sensitive to many irrelevant predictors, we explored two predictor sets:

- All 238 predictors,
- The subset of 50 top absolute t-statistic predictors plus Patient Group and Lung.

More specifically, the reduced set of predictors was obtained as follows. The t-statistic for a test of the differences in means of each continuous predictor by candidate class (positive or negative) was obtained. The absolute values of these t-statistics were ranked and the top 50 predictors were selected for inclusion in a reduced predictor set. The categorical variables Patient Group and Lung were also included in this set, simply because it was thought that they might be useful. We refer to this subset as the ***50 Predictor Set*** (although it contains 52 features in all).

For each neural fit, we fit either 50 or 100 models, again using a cross-validation approach where a holdback sample of 30% of candidates was randomly excluded prior to fitting each model, and where the PE sensitivity and false positive rate were computed for the holdback sample. In the **Neural** platform, found under **Analyze > Modeling**, we used **KFold** as the **Validation Method.** Other settings were set at their default levels as shown in Figure 23, with the exception of the **Hidden Layer Structure**, which we varied in a non-methodological fashion.

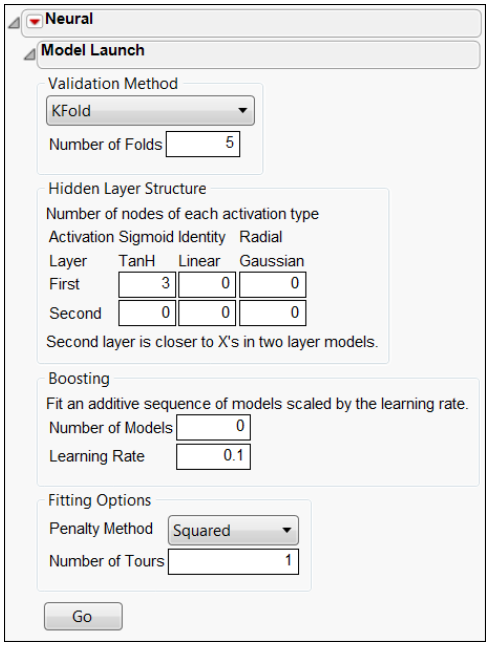

Figure 23 Control Panel for Neural Modeling

In all, six models were considered:

- All Predictors, One Hidden Layer, 3 TanH Activation Functions
- 50 Predictor Set, One Hidden Layer, 3 TanH Activation Functions
- All Predictors, Two Hidden Layers, 3 TanH Activation Functions Each
- All Predictors, Two Hidden Layers, 10 TanH Activation Functions Each
- 50 Predictor Set, One Hidden Layer, 3 Gaussian Activation Functions
- All Predictors, Two Hidden Layers, 3 Gaussian Activation Functions Each

Table 6 gives the mean true positive PE rates and the mean false positive rates for these models, as well as their standard deviations. Note that, due to time issues, 100 simulations were run for the one-layer models while 50 were run for the two-layer and Gaussian models.

Model	N	Mean(TP Rate PE)	Std Dev(TP Rate PE)	Mean(FP Rate)	Std Dev(FP Rate)
AllPredictors_OneLayer_3TanH	100	0.472	0.092	0.054	0.015
50Predictors_OneLayer_3TanH	100	0.450	0.074	0.037	0.012
All Predictors_TwoLayers_3TanHEach	100	0.468	0.099	0.056	0.019
All Predictors_TwoLayers_10TanHEach	50	0.460	0.052	0.071	0.013
50Predictors_OneLayer_3Gaussian	50	0.223	0.171	0.025	0.019
AllPredictors_TwoLayers_3GaussianEach	50	0.040	0.098	0.012	0.019

Table 6 Summary of Performance of Neural Models

Figures 24 and 25 show the means and two standard deviation limits for PE sensitivities and false positives based on the simulations. Although these ranges are not strictly comparable, given that certain models were not replicated as often as others, these plots do serve our purposes in terms of deciding whether to pursue the use of neural models for our classification tasks.

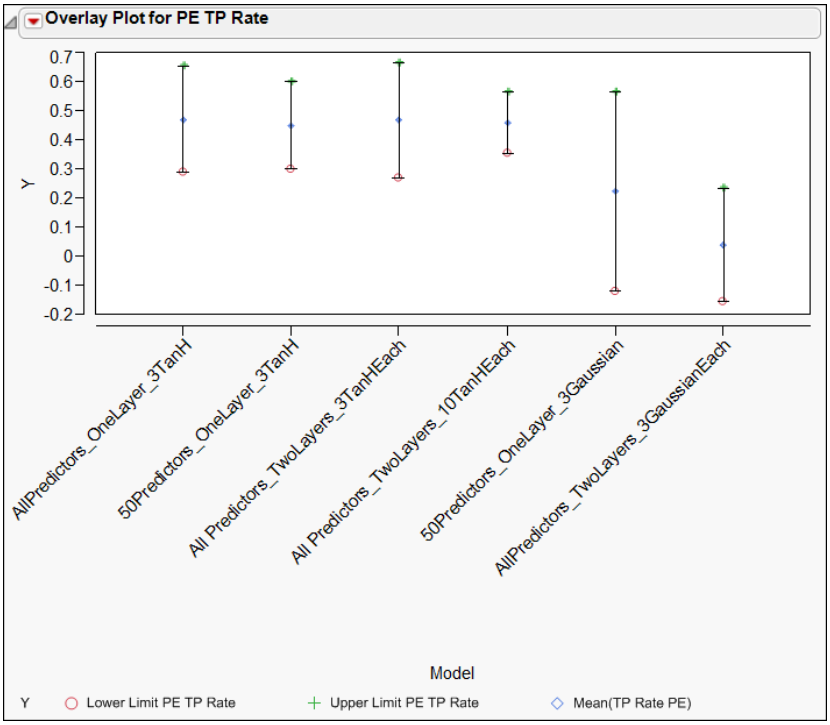

Figure 24 Mean PE Sensitivities for Neural Models, with 2 St Dev Limits

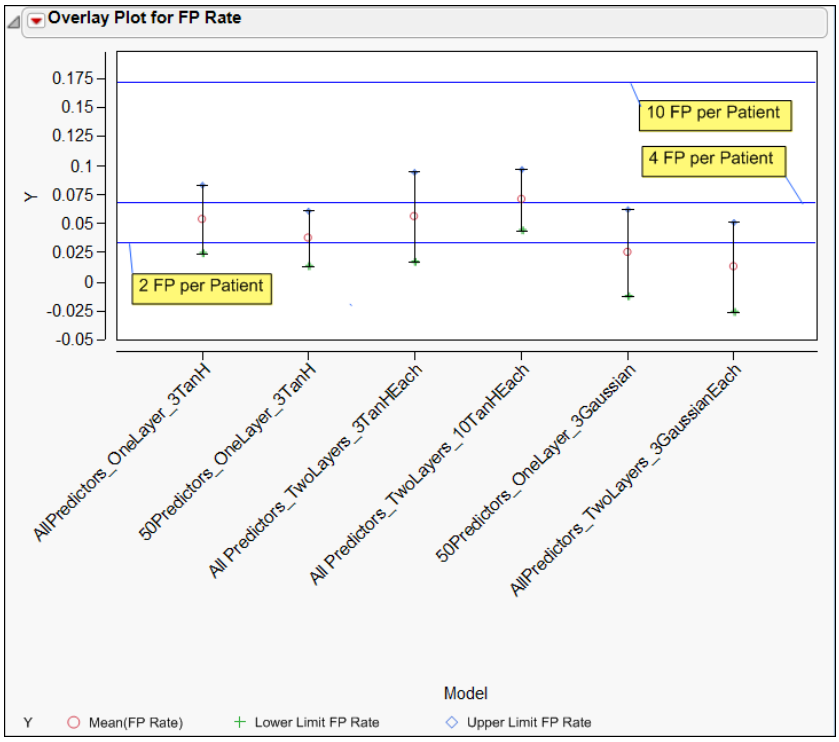

Figure 25 Mean False Positive Rates for Neural Models, with 2 St Dev Limits

The one- and two-layer Gaussian activation function models perform very poorly in terms of sensitivity. The sensitivity range for the one-layer models with Gaussian activation functions is very wide. In terms of performance, the Gaussian activation

function models don't compare with the boosted tree models identified earlier, and so are dismissed from further consideration.

This leaves four TanH-based models. None of these provides a range of false positive rates that meets the threshold of two false positives per patient most of the time. However, the 50-predictor TanH model does meet the four false positives per patient threshold. But its sensitivity ranges from 0.301 to 0.598, which does not compare well with the weight 3.25 boosted tree model we have identified. All the neural models meet the most relaxed false positive rate of ten per patient, but their sensitivities are below the estimated sensitivity of the 6.25 weighted boosted tree model.

Choice of Models and Evaluation

Choice of Models

Because of the low sensitivities of the neural models that met the false positive thresholds, we decided against selecting any of these to meet the contest requirements. For the contest selection at the 2 false positives per patient threshold, we selected the bootstrap forest class of models, applied with a weight of 3.5 on positive candidates. For the higher two thresholds, we selected boosted tree models, with appropriate weights as described earlier.

The most promising class of models and weights for each threshold are listed Table 7. We fit these models to the entire set of training data using the default settings and obtained formulas for each of these. The formulas were then copied into a data table containing the test data, allowing us to predict classes for test data candidates.

Model Class	Weight	Mean(TP Rate PE)	Lower Limit PE TP Rate	Upper Limit PE TP Rate	Mean(FP Rate)	Lower Limit FP Rate	Upper Limit FP Rate
Bootstrap Forest	3.50	0.430	0.315	0.546	0.016	0.005	0.027
Boosted Tree	3.25	0.565	0.449	0.680	0.050	0.029	0.072
Boosted Tree	6.25	0.720	0.621	0.819	0.138	0.104	0.172

Table 7 Model Choices and Performance Estimates

KDD Scoring

The test data consists of 1279 candidates and involves 21 patients. There are 1142 negative candidates, with the remaining 137 candidates involved in 58 PEs.

In the KDD Cup contest, the scoring was performed as follows. Contestants submitted their classifications for each task and subtask for the test data. Because the test set was rather small, a bootstrap resampling of the test set was used in computing scores.

Specifically, 200 bootstrap samples of 1279 candidates were selected from the test set. Rather than sampling uniformly with replacement from the 1279 candidates, the organizers decided on a hierarchical sampling strategy. They first sampled from the 21 patients uniformly (with replacement). Then, for the selected patient, they sampled from that patient's candidates uniformly (with replacement). The choice of hierarchical sampling was motivated by a desire to respect the within patient correlation structure (Lane et al.)

Incidentally, the hierarchical sampling strategy for the bootstrap samples had the effect of producing sets of 1279 candidates with a lower representation of PEs than seen in the test set. The distribution of PE counts in the 200 bootstrap samples is shown in Figure 26. Whereas there were 58 PEs in the test set, the count in the bootstrap samples averaged 43.16 PEs, and the counts ranged from 36 to 51.

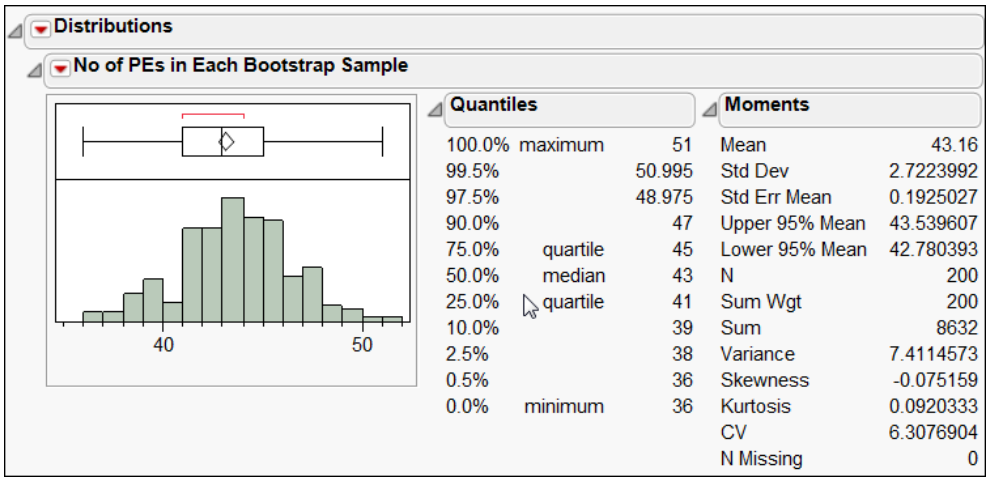

Figure 26 Distribution of Counts of PEs in 200 Bootstrap Samples

In scoring the entries, the organizers computed the subtask KDD false positive rate for each of the 200 bootstrap samples. If a subtask failed to meet the false positive threshold, that subtask's sensitivity score was set to 0. Furthermore, in obtaining an overall average sensitivity score for each task, if any subtask did not qualify on a sample, then all three subtasks were assigned a sensitivity score of 0. The final overall scores were averages of the KDD PE sensitivity values (for Task 1) and counts of true positive patients (for Task 2).

Applying this strategy in scoring our three models, we obtain the subtask scores in Table 8. Our overall scores are given in Table 9.

N Rows	Mean(Task 1a)	Mean(Task 1b)	Mean(Task 1c)	Mean(Task 2a)	Mean(Task 2b)	Mean(Task 2c)
200	0.94	1.19	1.47	12.04	12.74	14.88

Table 8 Subtask Scores Using Selected Models

Overall Scores	
Task 1	**Task 2**
1.19	13.14

Table 9 Overall Scores Using Selected Models

In the actual KDD Cup contest, there were 65 entries for Task 1 and 64 for Task 2. For comparison, the results for the highest-ranked ten entries for Tasks 1 and 2 are given in Figures 27 and 28. We tie for seventh place on Task 1 and we rank fifth on Task 2.

KDD Cup 2006 Task 1 results

Below are the results. The ID is the one used to submit answers with.

Download ID	T1a Mean snstvity	T1b Mean snstvity	T1c Mean snstvity	Final score
ef76e0193ef8372113a26b9be97368d3	1.17	1.31	1.58	1.35
d0b226d5cb7acf1960c3c366bba827f6	1.00	1.25	1.58	1.28
2b33b8ba98ec223548eab4e64ae88dc4	0.93	1.36	1.51	1.27
8e3126f2523ecc6acbc87bff0e3d49c4	0.97	1.34	1.40	1.24
f0edf80fda6fbba3e06b24eb9b3df1fc	1.08	1.25	1.38	1.24
9b5ae2264450b107c5ee952ec5a72eeb	1.04	1.20	1.39	1.21
3ac232f02b893cb57a4487c4a5330ea4	0.90	1.24	1.43	1.19
4b27bd0bf34dc747b0211623c2eb65a6	0.75	1.15	1.55	1.15
2781ba0557656 1b182e2809ee9768ccb	0.84	1.12	1.47	1.14
c671036b3aafa55d0dde37175e7fca96	0.83	1.06	1.52	1.14

Figure 27 KDD Cup Highest-Ranked Scores on Task 1

KDD Cup 2006 Task 2 results

Here are the results. Be sure to use your Dowload ID to look your results up. This is the second ID, the one you used to submit your answers.

Download ID	T2a Mean snstvity	T2b Mean snstvity	T2c Mean snstvity	Final score
2b33b8ba98ec223548eab4e64ae88dc4	11.50	14.34	14.90	13.58
3ac232f02b893cb57a4487c4a5330ea4	11.56	13.74	15.39	13.56
656708047c26984c5bc03e7450ebd90b	11.18	13.74	15.39	13.44
d0b226d5cb7acf1960c3c366bba827f6	12.02	13.74	14.33	13.36
8e3126f2523ecc6acbc87bff0e3d49c4	10.98	13.99	14.07	13.01
c671036b3aafa55d0dde37175e7fca96	11.11	12.48	14.46	12.68
3c5362554b598ca99371d8f9520b61fa	11.78	12.37	13.15	12.43
b4e3aa99276f8c677d4d7dea4dc1f6b5	11.19	11.95	13.97	12.37
2781ba0557656 1b182e2809ee9768ccb	10.68	12.24	13.94	12.28
92bbb197efdf5ea61ddf64bdf55e13a6	10.79	12.55	13.45	12.26

Figure 28 KDD Cup Highest-Ranked Scores on Task 2

Performance on Test Data

How do our models perform when they are applied directly to the test data? Table 10 shows the computed measures of PE sensitivity and false positive rate for our three models applied to the test data.

Model for…	PE Sensitivity	FP Rate
2 FP per Patient	0.500	0.012
4 FP per Patient	0.621	0.042
10 FP per Patient	0.776	0.122

Table 10 Usual Measures of Sensitivity and False Positive Rate for Selected Models

Patient-level confusion matrices are given in Table 11. Note that the models for the 4 and 10 KDD false positive rates have no false negatives.

Test Set Label	Model for 2 FP per Patient		Model for 4 FP per Patient		Model for 10 FP per Patient	
	Positive	Negative	Positive	Negative	Positive	Negative
Positive	16	3	19	0	19	0
Negative	2	0	2	0	2	0

Table 11 Patient-Level Confusion Matrices for Selected Models

Conclusion

There are many other approaches to modeling that we might have tried. There is evidence in the literature that bootstrap trees do not perform well in the presence of many irrelevant predictors (Gashler et al., 2008). We could have attempted fits on smaller sets of the more relevant predictors. There are many other forms of neural models that we could have considered, and we could have utilized cut-offs based on ROC curves to meet the false positive thresholds.

Certainly much more could have been done as well in terms of data preparation and defining potential predictors. We made no attempt to remove outliers or to transform highly skewed distributions, which might have helped with neural fits. We could have refined our approach to integrating information about nearest neighbors, identifying more than the single nearest neighbor and perhaps using only subsets of the nearest neighbor feature sets.

If this had been an actual task, rather than a contest, we would have leveraged subject matter knowledge in sorting out relevant predictors and in defining appropriate transformations of predictors. Knowledge of the type that false positives often occur in certain arteries would have been integrated. In fact, it would have likely been possible to resolve the mystery about patient groupings on the x, y, z, locations, which might have greatly supported the modeling effort.

Overall, though, we conclude that the strategy of weighting the positive class in conjunction with using bootstrap forest and boosted tree models in JMP has provided a solid solution to the task set by the KDD Cup organizers. With the exciting new data-mining capabilities in JMP 9 and its flexible scripting language, we were able to achieve a very respectable level of success in classifying PE candidates with a reasonable amount of effort and with no subject matter knowledge.

References

Anderson F, Wheeler H, Goldberg R, et al. A population-based perspective of the hospital incidence and case-fatality rates of deep vein thrombosis and pulmonary embolism. The Worcester DVT Study. *Archives Of Internal Medicine* [serial online]. May 1991;151(5):933-938. Available from: MEDLINE, Ipswich, MA. Accessed August 17, 2010.

Bouma H, Sonnemans J, Vilanova A, Gerritsen F. Automatic Detection of Pulmonary Embolism in CTA Images. *IEEE Transactions on Medical Imaging* [serial online]. August 2009;28(8):1223-1230. Available from: Academic Search Premier, Ipswich, MA. Accessed August 17, 2010.

Gashler M, Giraud-Carrier C, Martinez T. Decision Tree Ensemble: Small Heterogeneous Is Better Than Large Homogeneous. Machine Learning and Applications, 2008. ICMLA '08. Seventh International Conference on. Dec. 2008; 11-13:900-905. ISBN: 978-0-7695-3495-4.

Hastie T, Tibshirani R, Friedman J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2001;Springer.

Horlander K, Mannino D, Leeper K. Pulmonary Embolism Mortality in the United States, 1979-1998. *Arch Intern Med*. 2003;163:1711-1717. (Full Text)

King R, Feng C, Sutherland A. StatLog: Comparison of Classification Algorithms on Large Real-World Problems. *Applied Artificial Intelligence*. 1995;9;289-333.

Lane T, Rao B, Bi J, et al. On the Medical Frontier: The 2006 KDD Cup Competition and Results. *SIGKDD Explorations*. 8(2):39-46. (Full Text)

Masutani Y, MacMahon H, and Doi K. Computer-assisted detection of pulmonary embolism. *Proc. SPIE (Medical Imaging 2000)*, pt. II, vol. 3979:944–950.

Masutani Y, MacMahon H, and Doi K. Computerized Detection of Pulmonary Embolism in Spiral CT Angiography Based on Volumetric Image Analysis. *IEEE Transactions on Medical Imaging*. December 2002;21(12):1517-1523.

Pichon E, Novak C, Kiraly A, Naidich D. A novel method for pulmonary emboli visualization from high-resolution CT images. *Proc. SPIE*. 2004;5367:161–170.

Silverstein M, Heit J, Mohr D, Petterson T, O'Fallon W, Melton L. Trends in the incidence of deep vein thrombosis and pulmonary embolism: a 25-year population-based study. *Archives of Internal Medicine*. March 23, 1998;158(6):585-593. Available from: CINAHL Plus with Full Text, Ipswich, MA. Accessed August 17, 2010. (Full Text)

Sluimer I, Schilham A, Prokop M, van Ginneken, B. Computer Analysis of Computed Tomography Scans of the Lung: A Survey. *IEEE Transactions on Medical Imaging*. April(2006);25(4):385-405.

Sun Y, Kamel M, Wong A, Wang Y. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition* [serial online]. December 2007;40(12):3358-3378. Available from: Academic Search Premier, Ipswich, MA. Accessed August 17, 2010.