National Research
Council Canada

Conseil national
de recherches Canada

Institute for
Information Technology

Institut de technologie
de l'information

NRC · CNRC

Discriminative vs. Generative Classifiers: An In-Depth Experimental Comparison using Cost Curves *

Drummond, C.
December 2005

* published as NRC/ERB-1135. 30 pages. December 2005. NRC 48480.

Canada

National Research
Council Canada

Conseil national
de recherches Canada

Institute for
Information Technology

Institut de technologie
de l'information

ERB-1135

NRC·CNRC

Discriminative vs. Generative Classifiers : An In-Depth Experimental Comparison using Cost Curves

Drummond, C.

December 2005

Canada

NRC 48480

Discriminative vs. Generative Classifiers: An In-Depth Experimental Comparison using Cost Curves.

Chris Drummond
Institute for Information Technology,
National Research Council Canada,
Ottawa, Ontario, Canada, K1A 0R6
Chris.Drummond@nrc-cnrc.gc.ca

December 20, 2005

Abstract

This technical report discusses the experimental comparison of commonly used algorithms both in their traditional discriminative form and as generative classifiers. The performance is compared using cost curves to see what benefits might be gained by using a generative classifier when the misclassification costs, and class frequencies, are unknown. There is some evidence that learning a discriminative classifier is more effective for a traditional classification task. Focusing on algorithms that have generative and discriminative forms, allows a clear comparison between these two types of classifier without being obscured by algorithmic differences. The report compares the performance of the classifiers over 16 data sets and for the full range of misclassification costs and class frequencies. The experiments show that there is some merit in using generative classifiers for cost sensitive learning but more work is needed to make them as effective as using multiple discriminative classifiers.

1 Introduction

Many experiments have shown that discriminative classifiers have better performance than generative ones on a traditional classification task [1, 2, 3, 4]. There is also some theory suggesting why this holds true, at least asymptotically [2]. The intuition is that to do more than you have to for a task is not only unnecessary it is also potentially harmful. Nevertheless the debate continues, with some research showing that the conclusion is not as simple as the discriminative classifier being always better. Some restrictions on the sort of distributions the generative model learns have been shown to improve the accuracy of classification [5] over and above that of discriminatory classifiers. Here, the intuition is that knowledge restricts the size of the hypothesis space leading to better performance. Generative classifiers are a natural way to include domain knowledge, leading some researchers to propose a hybrid of the two [6, 7].

The author of this technical report is strongly interested in cost sensitive learning. Cost sensitive learning is a research area which has grown considerably in recent years, as evidenced by various bibliographies [8, 9]. This type of learning is a natural fit with generative classifiers. Without clear knowledge of the class frequencies and misclassifications costs, a discrimination boundary cannot be constructed whereas class likelihood functions can still be learned.

This report details experiments using popular algorithms, for which various researchers have suggested simple ways of modifying them for probability estimation. This allows a natural framework for experimental comparison, where the same basic algorithm is used to generate both discriminative and generative classifiers. The performance measure used is expected cost. This measure is a very natural way to include not only the actual costs often found in industrial problems but also more qualitative human concerns. Cost curves are used to visualize the difference in expected cost between classifiers over the full range of misclassification costs and class frequencies.

2 Cost Curves

This section gives a brief introduction to cost curves [10, 11], a way to visualize classifier performance over different misclassification costs and class distributions.

The error rate of a binary classifier is a convex combination of the likelihood functions $P(-|+)$, $P(+|-)$, where $P(L|C)$ is the probability that an instance of class C is labeled L and the coefficients $P(+)$, $P(-)$ are the class priors:

$$E[Error] = \underbrace{P(-|+)}_{FN} P(+) + \underbrace{P(+|-)}_{FP} P(-)$$

Estimates of the likelihoods are the false positive (FP) and false negative (FN) rates. A straight line, such as the one in bold in Figure 1, gives the error rate on the y-axis (ignore the axis labels in parentheses for the moment), for each possible prior probability of an instance belonging to the positive class on the x-axis. If this line is completely below another line, representing a second classifier, it has a lower error rate for every probability. If they cross, each classifier is better for some range of priors. Of particular note are the two trivial classifiers, the dashed lines in the figure. One always predicts that instances are negative, the other that instances are positive. Together they form the majority classifier, the shaded triangle in Figure 1, which predicts the most common class. The figure shows that any single classifier with a non-zero error rate will always be outperformed by the majority classifier if the priors are sufficiently skewed, therefore of little use. Even a good classifier produces too many false positives when negative examples are very common [12].

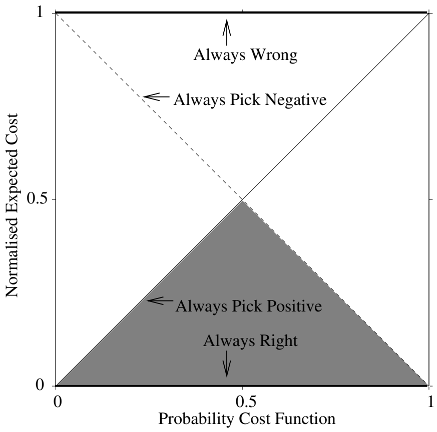

Figure 1: Visualizing Performance

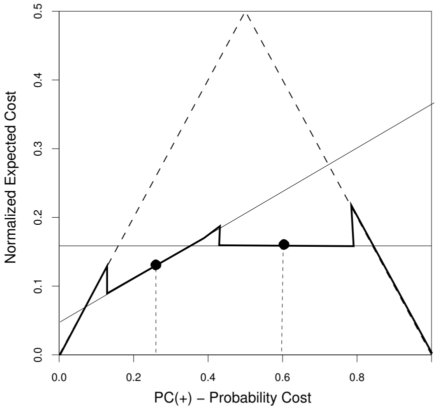

Figure 2: The Cost Curve

If misclassification costs are taken into account, expected error rate is replaced by expected cost, as defined by Equation 1. The expected cost is also a convex combination of the prior probabilities, but plotting it against the priors would produce a y-axis that no longer ranges from zero to one. The expected cost is normalized by dividing by the maximum value, given by Equation 2. The costs and priors are combined into the Probability_Cost(+) on the x-axis, as in Equation 3. Applying the same normalization factor results in an x-axis that ranges from zero to one, as in Equation 4. The positive and negative Probability_Cost(_)'s now sum to one, as was the case with the probabilities.

$$
\begin{array}{rcl}
E[Cost] & = & FN * C(-|+)P(+) + FP * C(+|-)P(-) \qquad (1) \\
max(E[Cost]) & = & C(-|+)P(+) + C(+|-)P(-) \qquad (2) \\
PC(+) & = & C(-|+)P(+) \qquad (3) \\
Norm(E[Cost]) & = & FN * PC(+) + FP * PC(-) \qquad (4)
\end{array}
$$

With this representation, the axes in Figure 1 are simply relabeled, using the text in parentheses, to account for costs. Misclassification costs and class frequencies are more imbalanced the further away from 0.5, the center of the diagram. The lines are still straight. There is still a triangular shaded region, but now representing the classifier predicting the class that has the smaller expected cost. For simplicity we shall continue to refer to it as the majority classifier.

In Figure 2 the straight continuous lines are the expected cost for discriminative classifiers for two different class frequencies, or costs, indicated by the vertical dashed lines. To build a curve requires many different classifiers, each associated with the $PC(+)$ value used to generate it. Let's assume each classifier is used in the range from half way between its $PC(+)$ value and that of its left neighbor to half way between this value and that of its right neighbor. The resulting black curve, which includes the trivial classifiers, is shown in Figure 2. It has discontinuities where the change over between classifiers occurs.

To produce a curve for a generative classifier, each instance is associated with the $PC(+)$ value at which the classifier changes the way it is labeled. If the instances are sorted according to this value, increasing $PC(+)$ values generate unique FP and TP pairs. A curve is constructed in the same way as that for the discriminative classifiers. But now there are many more points, one for each instance in the test set, typically producing a much smoother looking curve.

3 Experiments

This section discusses experiments comparing the performance of various popular algorithms, as implemented in the machine learning system called Weka [13]. The main set of experiments compares the expected cost of a single generative classifier to that of a single discriminative classifier and to a series of such classifiers trained on data sets with different class frequencies. The question it addresses is to what extent the existing variants of standard algorithms are effective for cost sensitive learning. Further experiments look at how these probability estimators might be improved, firstly by calibration and secondly by using more balanced training sets.

To produce different $PC(+)$ values, the training set is under-sampled, the number of instances of one class being reduced to produce the appropriate class distribution. This is done for 16 $PC(+)$ values, roughly uniformly covering the range 0 to 1. The FP and TP values are estimated using ten-fold stratified cross validation. The experiments use 16 data sets, 14 from the UCI collection [14] and plus two used by the author in earlier work [15].

3.1 Decision Trees

We begin with the decision tree algorithm J48, Weka's version of C4.5 [16]. Figures 3 and 4 show cost curves for the 16 data sets (the name is just above the x-axis). The gray solid curves give expected cost using the class frequency at the leaves to estimate probabilities. To interpret these graphs, let us note that, in these experiments at least, there is little or no difference between discriminative and generative classifiers for the particular $PC(+)$ value at which they were trained. The main advantage of a generative classifier is that it will operate effectively at a quite different $PC(+)$ values. The solid black curve is for 14 discriminative classifiers generated by under-sampling. It acts, essentially, as a lower bound on the expected cost of using the generative classifier. The bold black straight line is the standard classifier trained (with default settings) at the original data set frequency, indicated by the vertical line. At this frequency, the black line, the gray solid curve, and the black curve have the same expected cost (being exactly the same classifier). The cost sensitivity of the generative classifier is seen by comparing the distance of the gray curve to the straight black line and the distance to the black curve, as one moves away from the original frequency. Closer to the black curve is better.

Although close to the original frequency there is little to separate the curves, the difference grows as the distance increases. For $PC(+)$ values closer to zero and one, the solid gray curves are much better than the single discriminative classifiers and quite close to the multiple ones. Unfortunately, here the performance is worse than the majority classifier, making any gain over the discriminative classifier of dubious merit. One way to improve the probability estimates is to use Laplace correction at the leaves of an unpruned tree [17]. In Figures 3 and 4 this variant is indicated by the dashed gray curve. Generally, this improves on the standard algorithm, again it is most clear far away from the original frequency. For some data sets, e.g. letterK and Sick, it is indistinguishable from the black solid curve. But for some data sets, e.g. credit-a and hepatitis, without pruning means it is worse than the standard classifier around the original frequency.

There are two commonly used approaches to improve cost sensitivity: calibration and changing the training set distribution. Figures 5 and 6 compare the cost curves to their lower envelopes, the dashed curves. The envelopes represent perfect calibration. The sides of an envelope represent the best classifier chosen for some range of $PC(+)$ values (This is unachievable in practice as the best is only know after testing).

For both variants of generative classifier not using Laplace correction, there is a large potential improvement by using calibration. This occurs particularly close to 0 and 1 on the x axis. Although it should be stressed that this improvement only amounts to potentially doing as well as the majority classifier far away from the original frequency. Calibration has less of an effect for the Laplace correction classifier, although there are are some data sets where there is still some potential improvement. It is also noteworthy, that in many data sets the Laplace correction not only improves on the basic curves without it, it is also produces curves that are often better than the other classifiers lower envelopes. So although it may gain much of its advantage through improved calibration other factors also contribute to its success.

Figure 21 shows results for using different ratios for training the generative classifier. A ratio of one gives a balanced training set, produced by undersampling the majority class. There are two additional ratios on either side of the balanced one, twice as many of the positive class and twice as many of the negative. For many data sets, like Bupa and Hepatitis, balancing the training set makes the cost curve more symmetric and decreases the area under the cost curve. This result is similar to that obtained by Weiss and Provost [18], a balanced data set is generally better. As with Weiss and Provost's work, sometimes other ratios still offer some improvement. A fact which warrants further investigation and is the subject of future work.

3.2 Support Vector machines

The original Support Vector Machine [2], as its aim was to find a "maximum margin hyperplane", was intended purely as a discriminative classifier and had no means of producing probability estimates. The Weka system uses Platt's [19] Sequential Minimal Optimization to learn the hyperplane, which produces the black solid lines and curves in Figures 7 and 8. Platt [20] showed how a sigmoid can be fitted to the training set (or by cross validation) using cross entropy as the error measure. Figures 7 and 8 show that this variant, the gray curve, is extremely competitive with the multiple discriminative classifiers. For only a couple of data sets, letterK and credit-a, are the two discernibly different.

Figures 9 and 10 show there is typically little difference between the cost curves (solid lines) and their lower envelopes (dashed lines), so calibrating the classifier should have little effect. This is not surprising as fitting a sigmoid is, itself, a form of calibration. Although the sigmoid only has two degrees of freedom, one can see more flexible schemes are unlikely to improve calibration much. This may be why no real benefit was seen using isotonic regression [21].

There is one data set, LetterK, that shows a large difference in expected cost. This is an extremely imbalanced domain and by training the classifier on a balanced data set, the black curves in Figure 21, considerable improvement is gained. So to get an improvement for all class frequencies and misclassification costs a balanced training set is better. But the diagrams suggest, although further investigations needed, that if the intended $PC(+)$ values were restricted to be reasonably close to the original frequency value that a different training set distribution might be better.

3.3 Neural Networks

Weka implements the traditional PDP algorithm [22] which is trained using back propagation and minimizes the squared error of the network output. This can be used as a discrimination classifier or, by using the standard sigmoid output of the network, as a probability estimator. As Figures 11 and 12 show, much like the standard decision tree, it improves on a single discriminative classifier but mainly where the majority classifier is dominant. However using the sigmoid output directly as a probability estimator results in a large performance reduction with respect to the multiple discriminative classifiers.

Figures 13 and 14 show that there is typically a large separation between the basic cost curve and its lower envelope suggesting that much of this performance difference is due to poor calibration. Figure 22 shows that balancing the training set offers some improvement but it is in correct calibration where there is most potential gain. It is noteworthy that the Weka algorithm minimizes squared error. Minimizing cross entropy, like the generative version of the Support Vector Machine, should produce better probability estimates [23]. Further work is needed to see if using cross-entropy as an error measure is sufficient to remove the performance difference or if calibration is still necessary.

3.4 Nearest Neighbor

The nearest neighbor algorithm used is Weka's implementation of IBk, the instance based learning algorithm of Aha and Kibler [24]. Figures 15 and 16 show four variants of IBk. The solid curves are for $K = 5$, a neighborhood of 5 instances is used for classification. One nearest neighbor $K = 1$, the dashed lines, is included as a reference point as it has often performed in the past very competitively with other algorithms. With only a single neighbor, it assigns probabilities of zero or one to instances. This produces roughly a straight line, for probability estimation, with some variation due to the variation in test set folds. The reasonably small neighborhood of 5 instances was chosen somewhat arbitrarily but often has good performance, very close to that of the multiple discriminative classifiers. It is also usually better than the 1 nearest neighbor version.

Figures 17 and 18 show that for large ranges of $PC(+)$ values there is little difference between the curve and the lower envelope, between the gray and black lines of the same type. It only at the extremes, close to one and zero, where the difference becomes apparent and calibration would offer some advantage. Most of the time the use of a distance function (1/distance and 1-distance) did not improve the probability estimates and on occasion made them a lot worse. Only for the "sick" data set did there seem to be any advantage. These nearest neighbor algorithms, probably due to their local quality, seem to derive little benefit from balancing the training set, as shown in Figure 21.

3.5 Naive Bayes

Naive Bayes is really the quintessential generative classifier, and there is no discriminative version. It is included here only to investigate how well calibrated are the different variants of naive Bayes. Figures 19 and 20 shows different ways that Naive Bayes represents continuous variables: a Gaussian approximation, a kernel approximation or discretization. The Gaussian variant certainly fares the worst. For many data sets, the gray curve extends way outside the majority classifier and is only useful for a small range of misclassification costs and class distributions. For many data sets (colic, credit-a, glass2, sleepBR2, sonar) the deviations are very large. The kernel approximation is much better but even this does badly on a couple of data sets, oil2 and glass2. The supervised discretization method works the best.

Where there are large deviations, by the Gaussian and Kernel approximation methods, the distance to their respective lower envelopes (the black lines of the same type) suggests much of this problem is due to poor calibration. The supervised discretization method of Fayyad and Irani [25] does fairly well without calibration. In fact, it is often better than the lower envelopes of the other methods. How it gains this advantage is the subject of future work.

4 Discussion and Future Work

In summary, the sigmoid variant for the Support Vector Machine, with a balanced training set, was extremely effective as a generative classifier. IBk with a neighborhood of five was also effective. Decision trees with Laplace correction and, to lesser extent, the Multilayer Perceptron faired reasonably and both showed potential for improvement. Generally, although balancing the training set is useful, calibration offers the most potential benefit. It is notably inherent in the Support Vector Machine sigmoid fitting procedure. Whether or not simple fitting procedures would offer the same degree of improvement for the other algorithms is worth investigating further.

The experiments were carried out using Weka's default setting for each algorithm, and with some settings chosen somewhat arbitrarily. Many more experiments are needed, with algorithms with different settings for these values, before strong conclusions can be reached. There are also other algorithms, and variants of the algorithms discussed in this technical report, worth exploring. A very large scale experiment, using cost curves, will be carried out in the future to make these comparison more comprehensive.

In this report, a curve made up of 16 discriminative classifiers has been used as a "gold standard". A good generative classifier is assumed to be one whose performance is close to this "gold standard". But to get good cost sensitive performance, one could simply use the 16 classifiers. The main advantage of the generative classifier is that it is a single classifier, reducing learning time and storage considerably. Another advantage is that a single classifier is may be more understandable. Yet neither the Support Vector Machine nor the Multilayer Perceptron is easily understandable without extra processing. Even for J48 as the generative classifier is unpruned, it is more complex than any single discriminative classifier. It may be possible that a few, a lot less than the 16, judiciously chosen, discriminative classifiers would be very competitive. A tree with a stable splitting criterion but variable cost sensitive pruning [15] would have identical lower branches for all $PC(+)$ values, making a collection of trees more easily understandable.

5 Conclusions

This report experimentally compared the performance of discriminative and generative classifiers for cost sensitive leaning. It showed that variants of commonly used algorithms produced reasonably effective generative classifiers. Where the classifiers were less effective, simple techniques such as choosing the right training set distribution or calibrating the posterior probbaility show potential to improve their performance considerably.

References

[1] Rubinstein, Y.D., Hastie, T.: Discriminative vs informative learning. In: Knowledge Discovery and Data Mining. (1997) 49–53

[2] Vapnik, V.: Statistical Learning Theory. Wiley (1998)

[3] Jebara, T., Pentland, A.: Maximum conditional likelihood via bound maximization and the cem algorithm. In: Advances in Neural Information Processing Systems 11. MIT Press, (1999) 494–500

[4] Nigam, K., Lafferty, J., McCallum, A.: Using maximum entropy for text classification. In: Proceedings of the IJCAI99 Workshop on Machine Learning for Information Filtering. (1999) 61–67

[5] Tong, S., Koller, D.: Restricted Bayes optimal classifiers. In: Proceedings of the 17th National Conference on Artificial Intelligence. (2000) 658–664

[6] Jaakkola, T.S., Haussler, D.: Exploiting generative models in discriminative classiers. In: Advances in Neural Information Processing Systems. MIT Press (1999) 487–493

[7] Bouchard, G., Triggs, B.: The tradeoff between generative and discriminative classifiers. In: Proceedings of the 16th Symposium in Computational Statistics. (2004) 697–704

[8] Boz, O.: Cost-sensitive learning bibliography. `http://home.ptd.net/$\sim$olcay/cost-sensitive.html` (2002)

[9] Japkowicz, N.: The class imbalance problem. `http://www.site.uottawa.ca/$\sim$nat/Research/class\_imbalance\_bibli.h%tml` (2002)

[10] Drummond, C., Holte, R.C.: Explicitly representing expected cost: An alternative to ROC representation. In: Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining. (2000) 198–207

[11] Drummond, C., Holte, R.C.: Cost curves: An improved method for visualizing classifier performance. Machine Learning (In Press)

[12] Axelsson, S.: The base-rate fallacy and its implications for the difficulty of intrusion detection. In: In Proceedings of the 6th ACM Conference on Computer and Communications Security. (1999) 1–7

[13] Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2005)

[14] Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html` (1998)

[15] Drummond, C., Holte, R.C.: Exploiting the cost (in)sensitivity of decision tree splitting criteria. In: Proceedings of the 17th International Conference on Machine Learning. (2000) 239–246

[16] Quinlan, J.R.: C4.5 Programs for Machine Learning. Morgan Kaufmann (1993)

[17] Provost, F., Domingos, P.: Tree induction for probability-based ranking. Machine Learning **52** (2003)

[18] Weiss, G.M., Provost, F.J.: Learning when training data are costly: The effect of class distribution on tree induction. Journal of Artificial Intelligence Research **19** (2003) 315–354

[19] Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods - Support Vector Learning. MIT Press (1998)

[20] Platt, J.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: Advances in Large-Margin Classifiers. MIT Press (2000) 61–74

[21] Zadrozny, B., Elkan, C.: Transforming classifier scores into accurate multiclass probability estimates. In: Proceedings of the Eighth International Conference on Knowledge Discovery & Data Mining. (2002)

[22] Rumelhart, D.E., McClelland, J.L.: Parallel distributed processing: explorations in the microstructure of cognition. MIT Press (1986)

[23] Bishop, C.M.: Neural networks for pattern recognition. OUP (1996)

[24] Aha, D., Kibler, D.: Instance-based learning algorithms. Machine Learning **6** (1991) 37–66

[25] Fayyad, U.M., Irani, K.B.: On the handling of continuous-valued attributes in decision tree generation. Machine Learning **8** (1992) 87–102

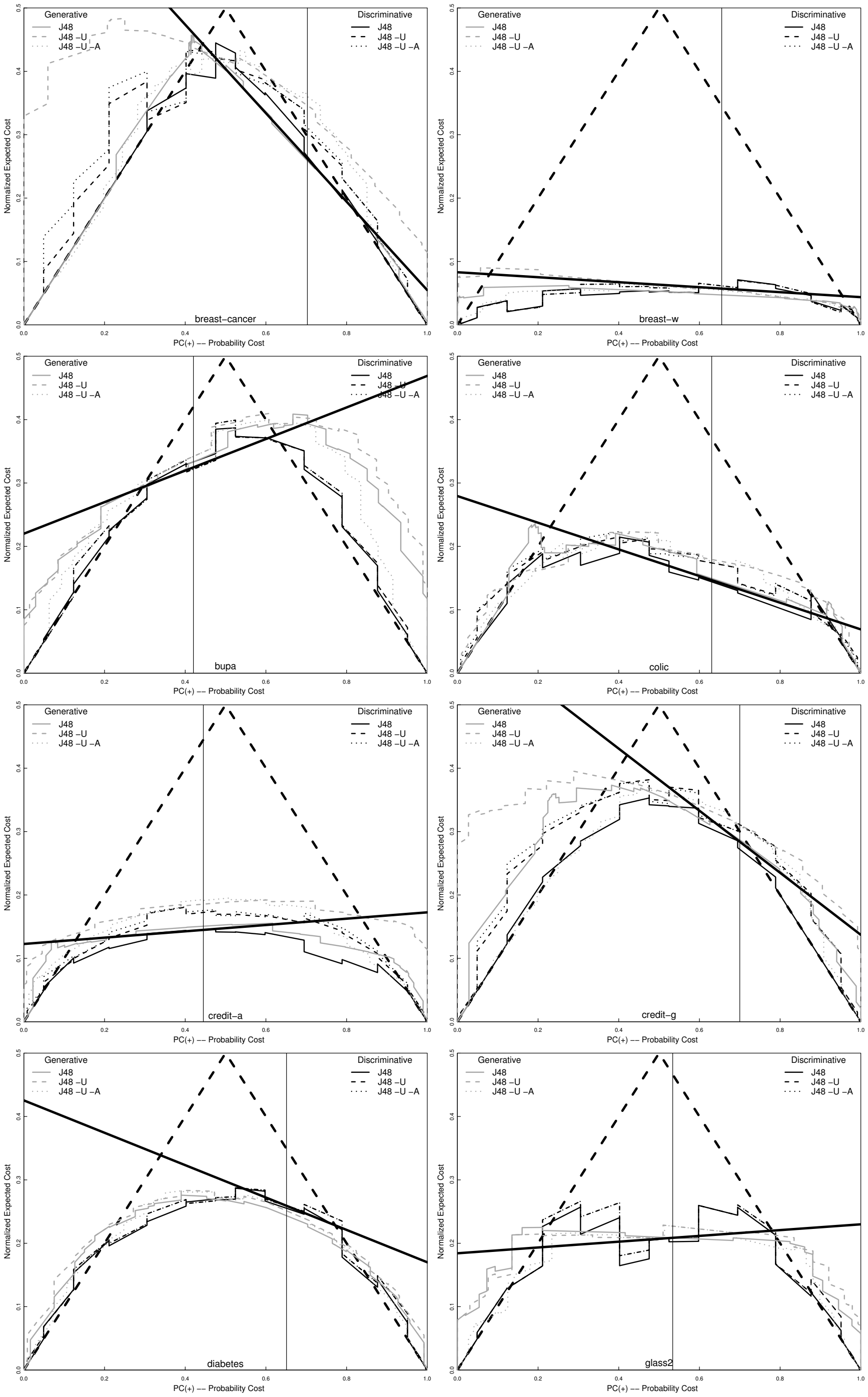

Figure 3: (a) Probability Estimation vs Undersampling: J48

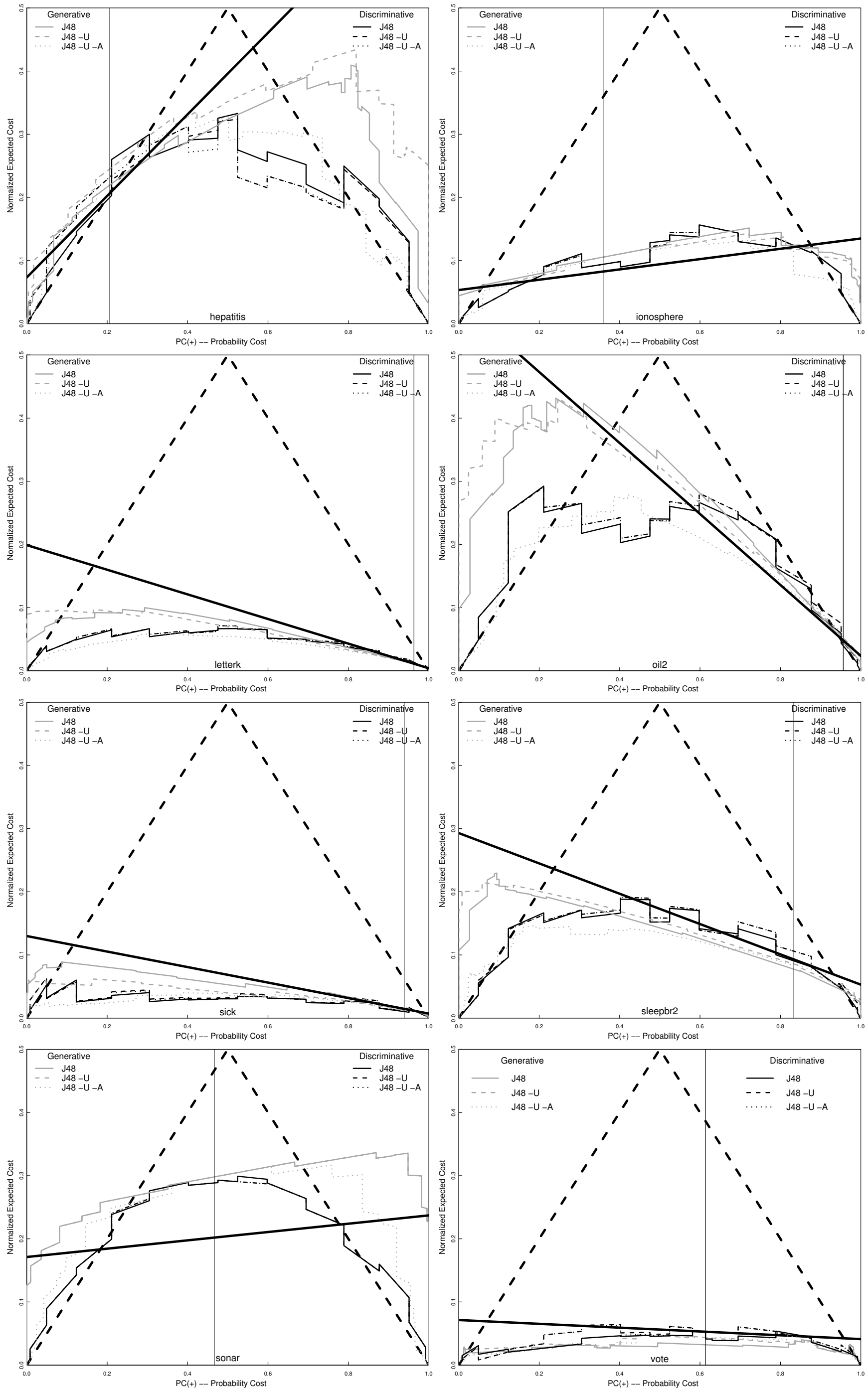

Figure 4: (b) Probability Estimation vs Undersampling: J48

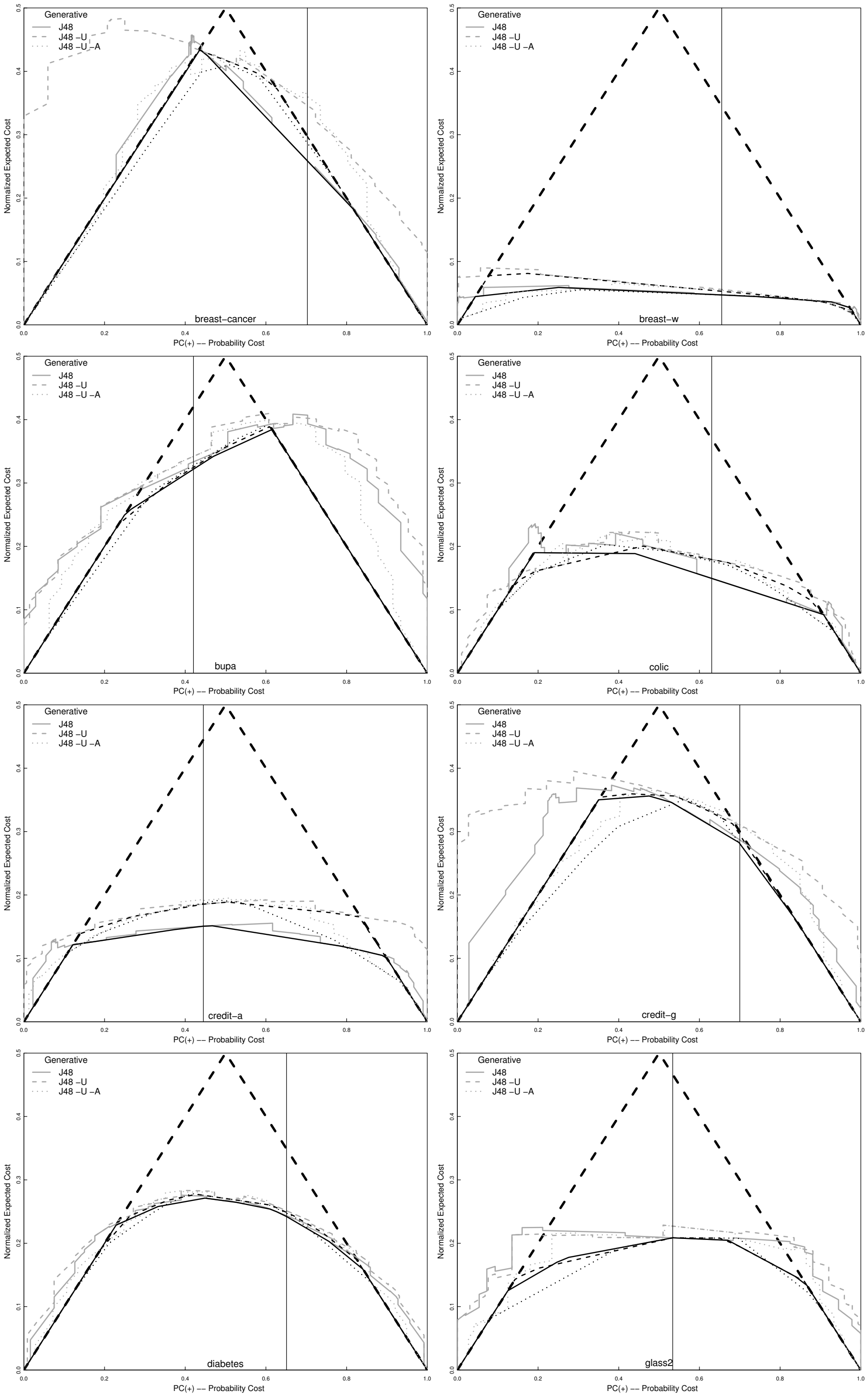

Figure 5: (a) Calibration of Probability Estimate: J48

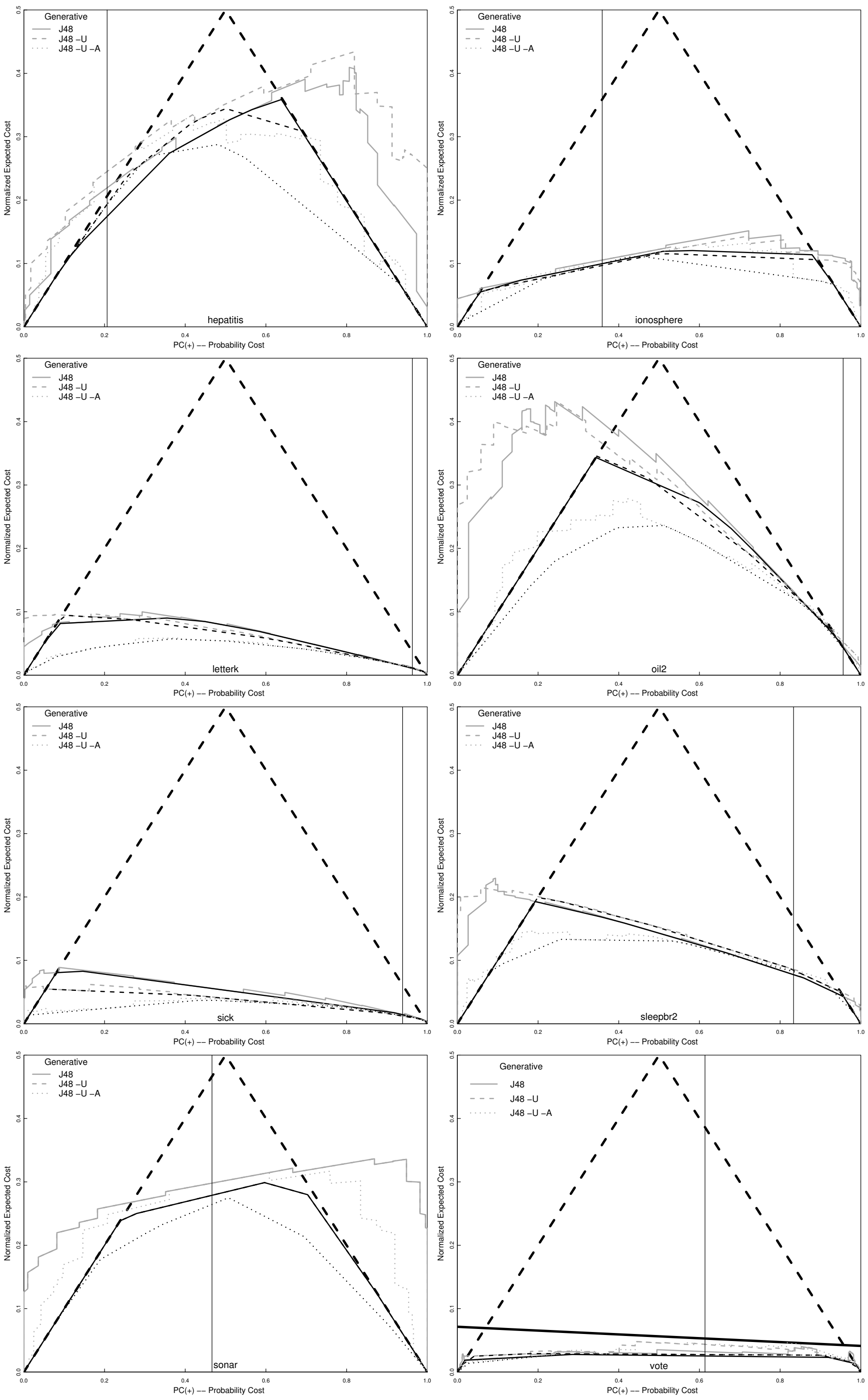

Figure 6: (b) Calibration of Probability Estimate: J48

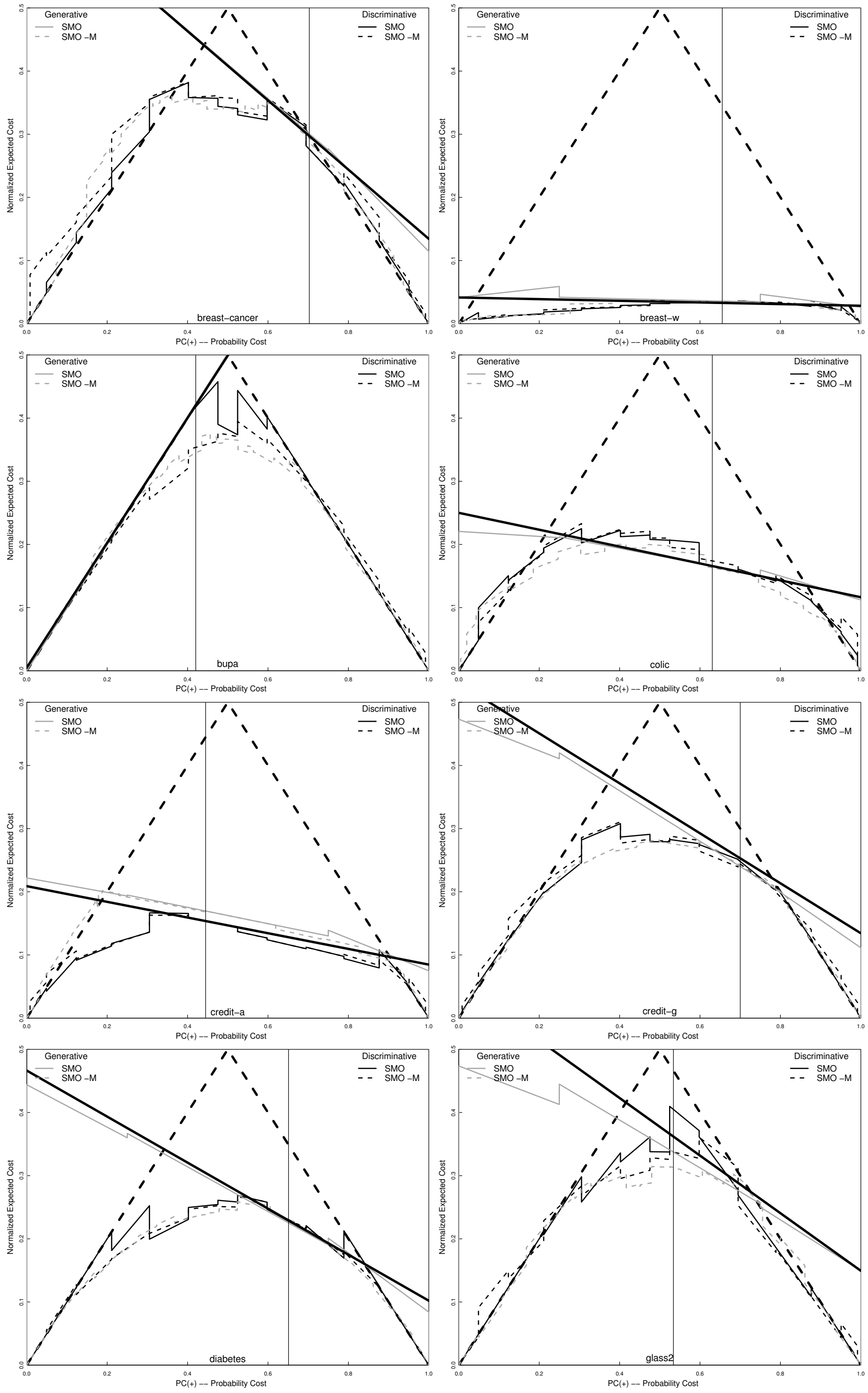

Figure 7: (a) Probability Estimation vs Undersampling: SMO

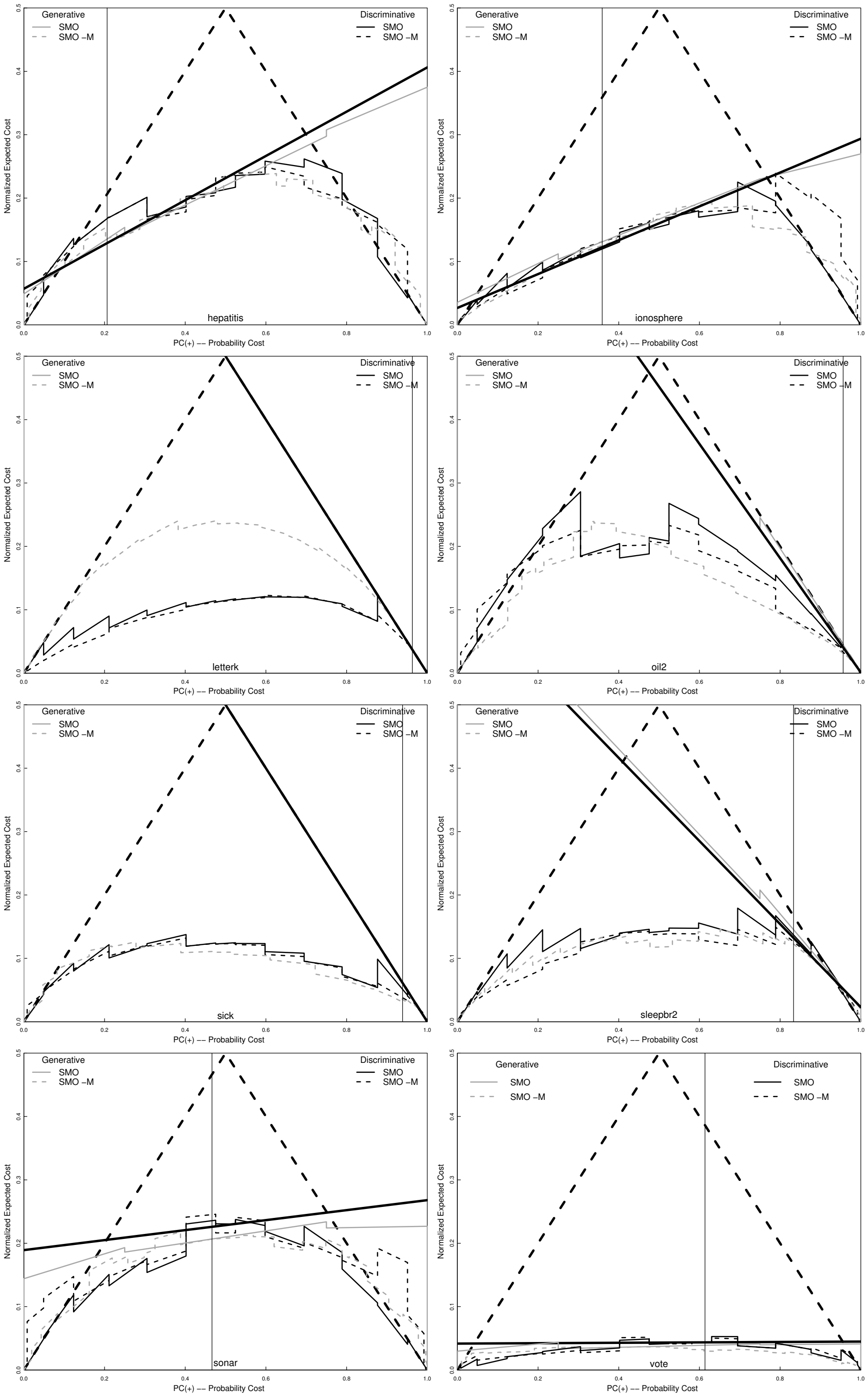

Figure 8: (b) Probability Estimation vs Undersampling: SMO

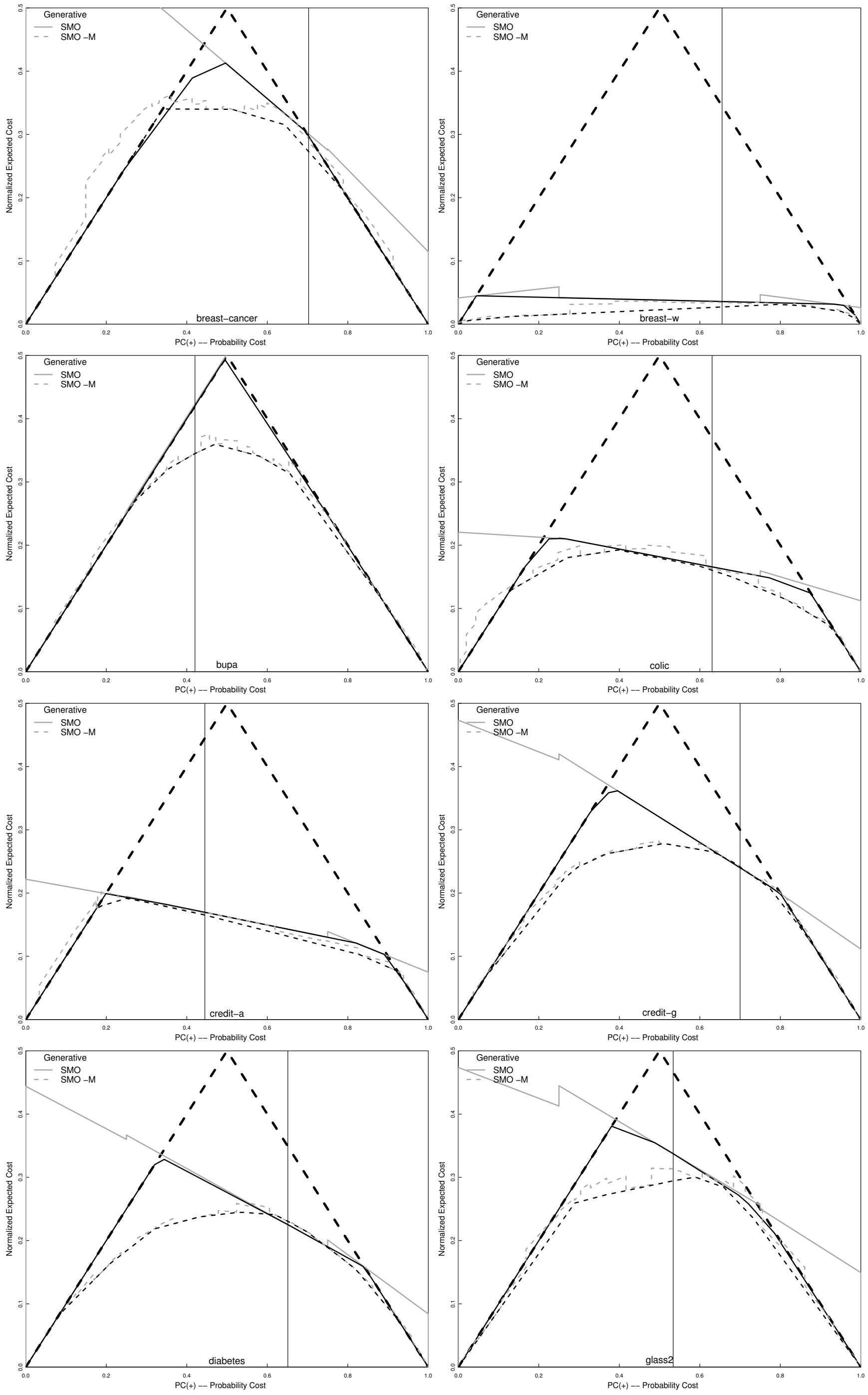

Figure 9: (a) Calibration of Probability Estimate: SMO

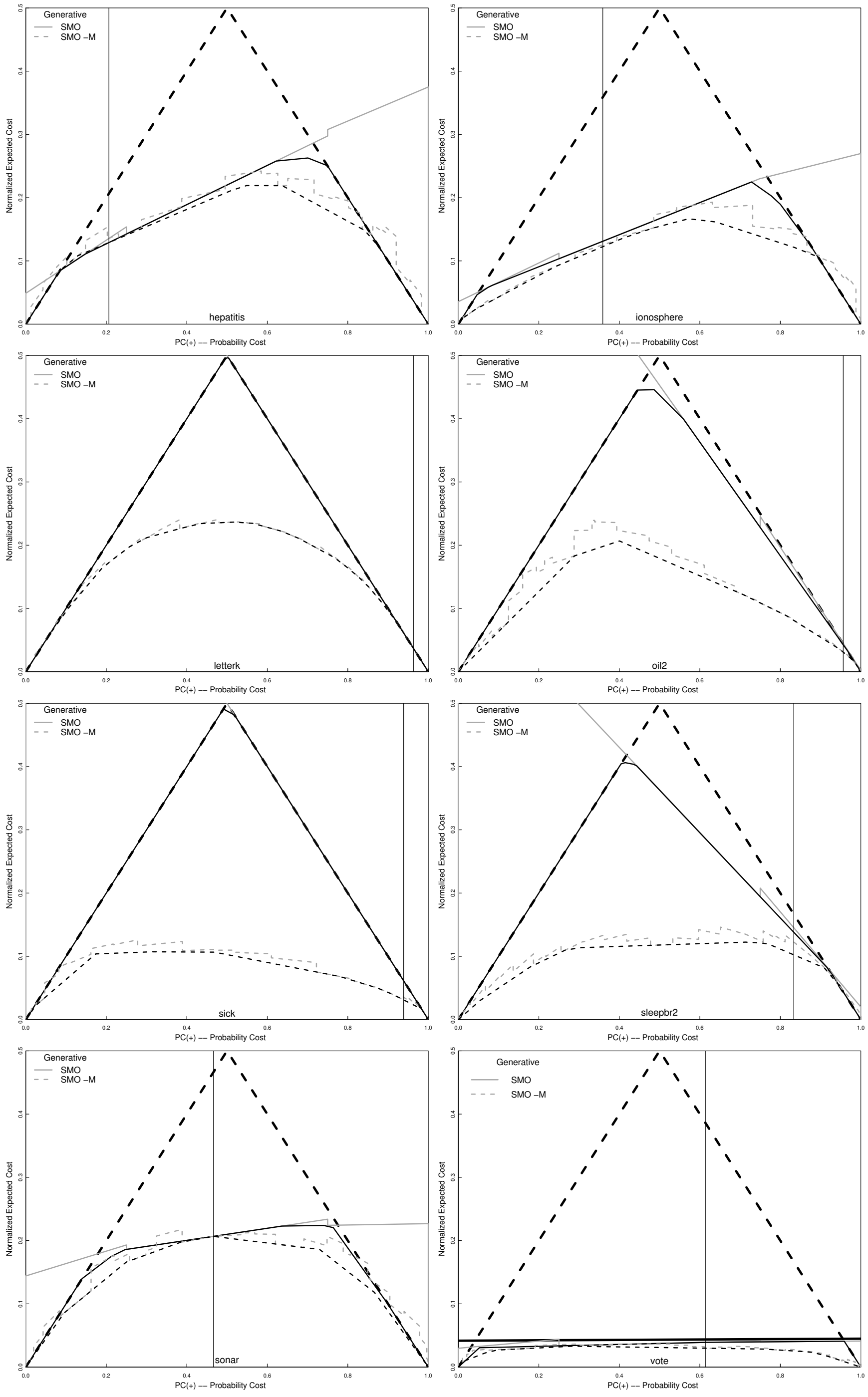

Figure 10: (b) Calibration of Probability Estimate: SMO

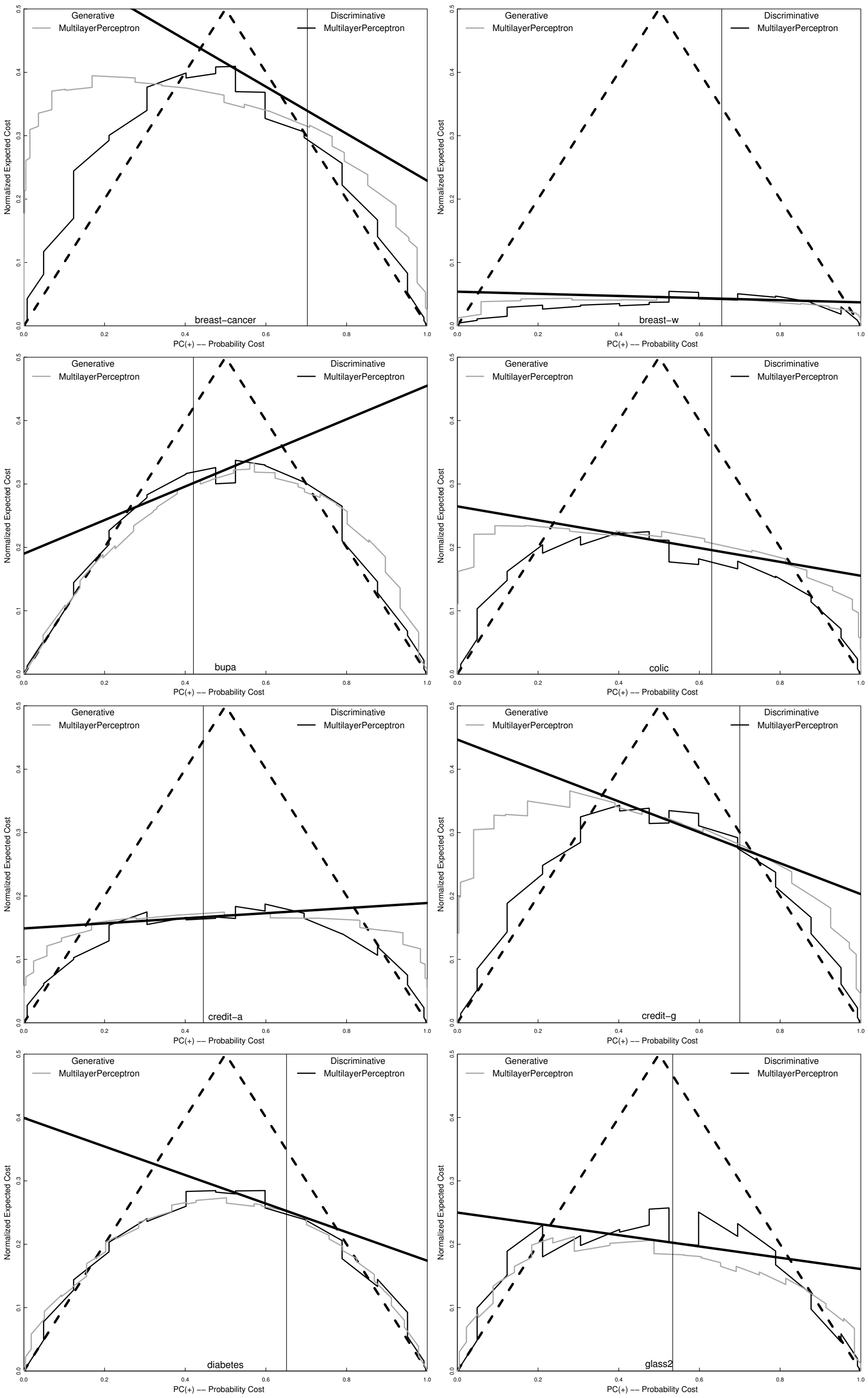

Figure 11: (a) Probability Estimation vs Undersampling: MultilayerPerceptron

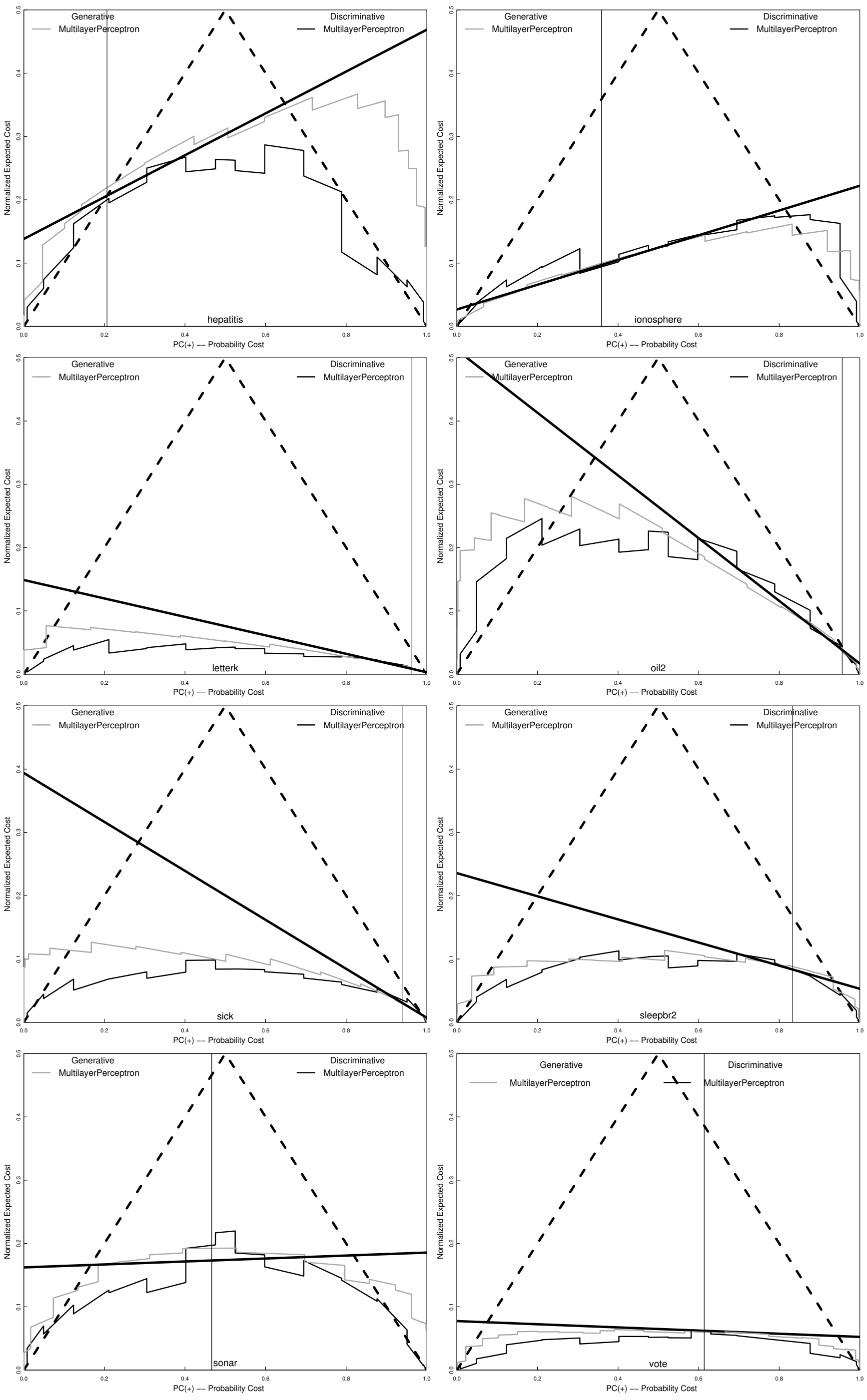

Figure 12: (b)Probability Estimation vs Undersampling: MultilayerPerceptron

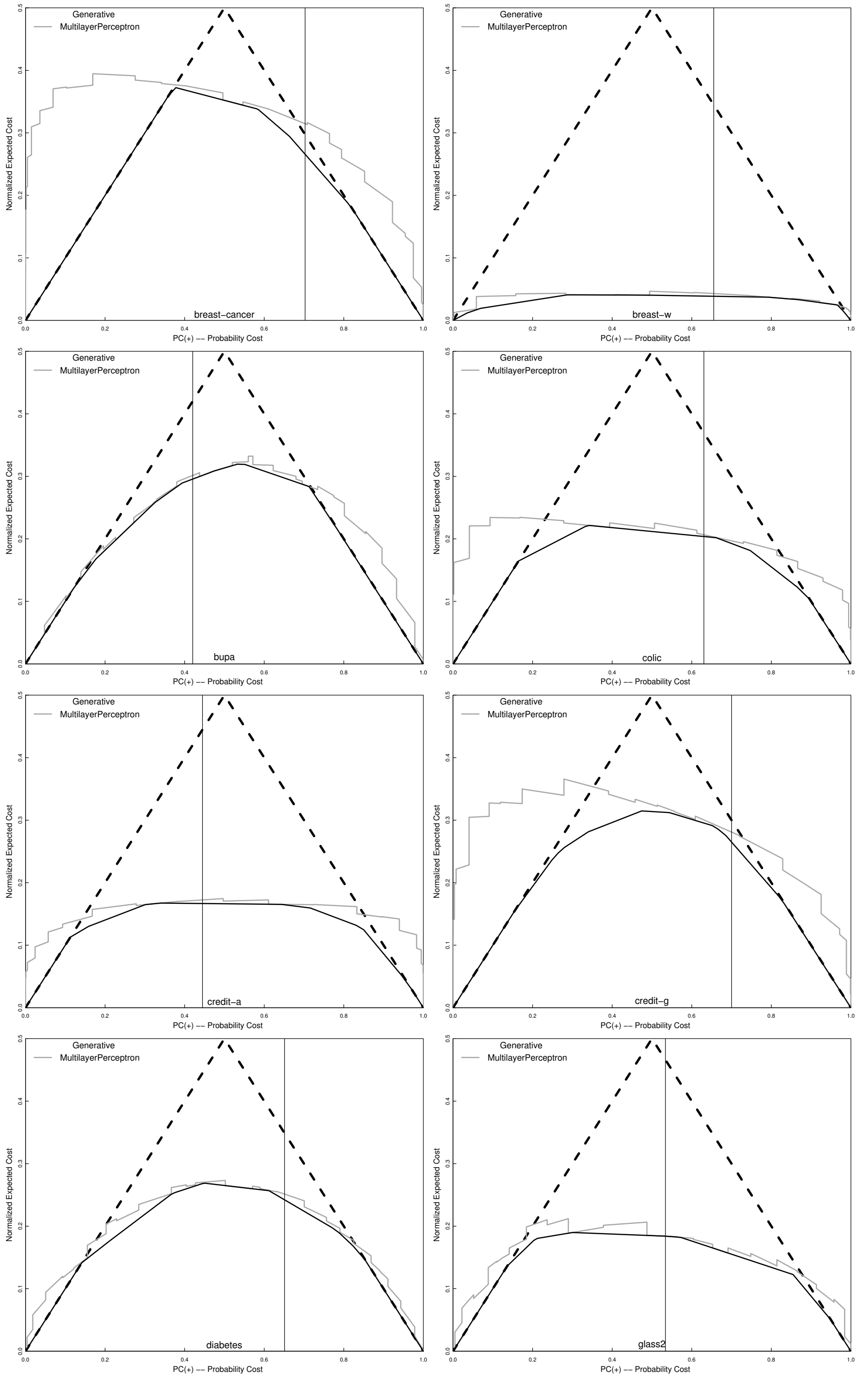

Figure 13: (a) Calibration of Probability Estimate: MultilayerPerceptron

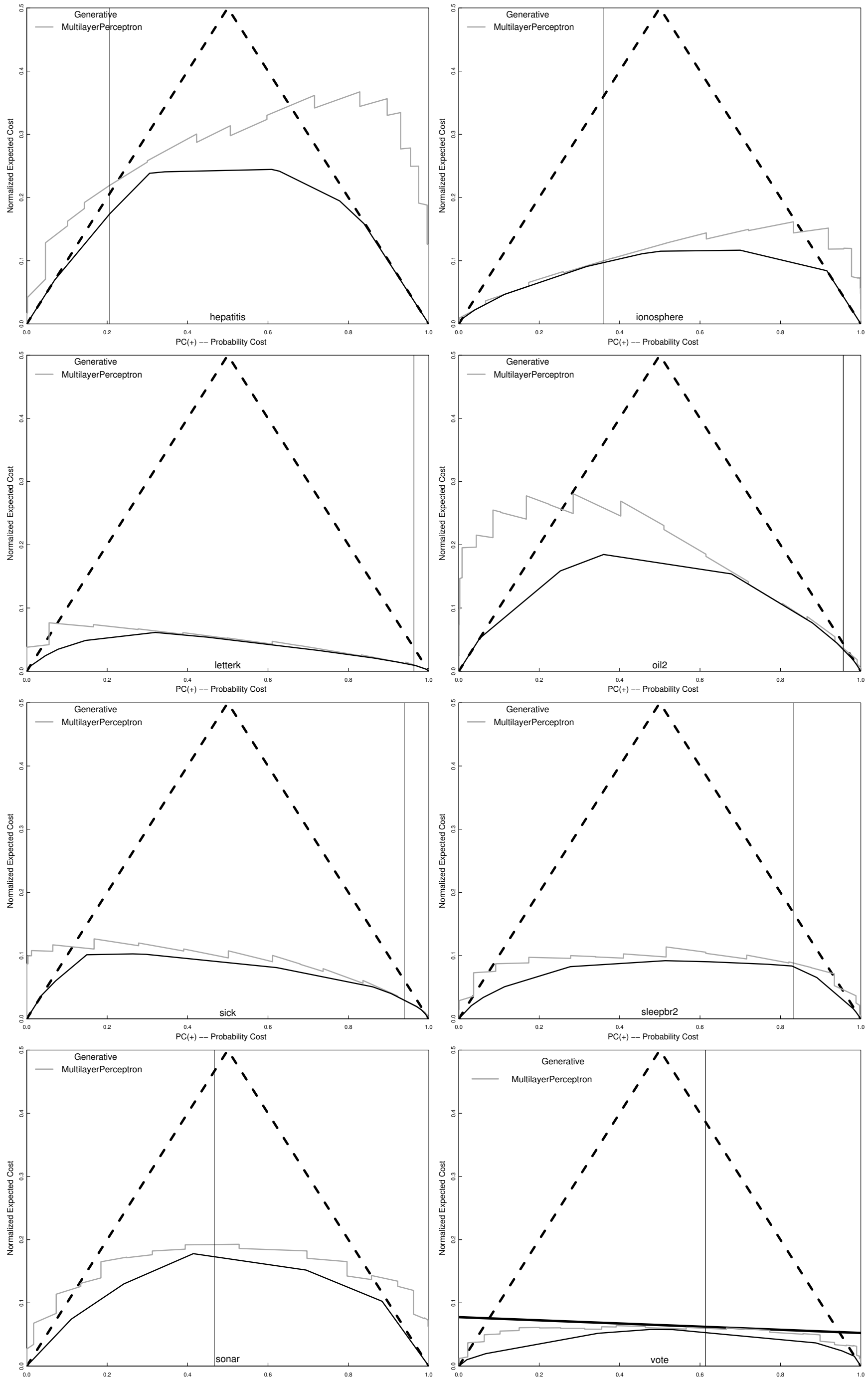

Figure 14: (b) Calibration of Probability Estimate: MultilayerPerceptron

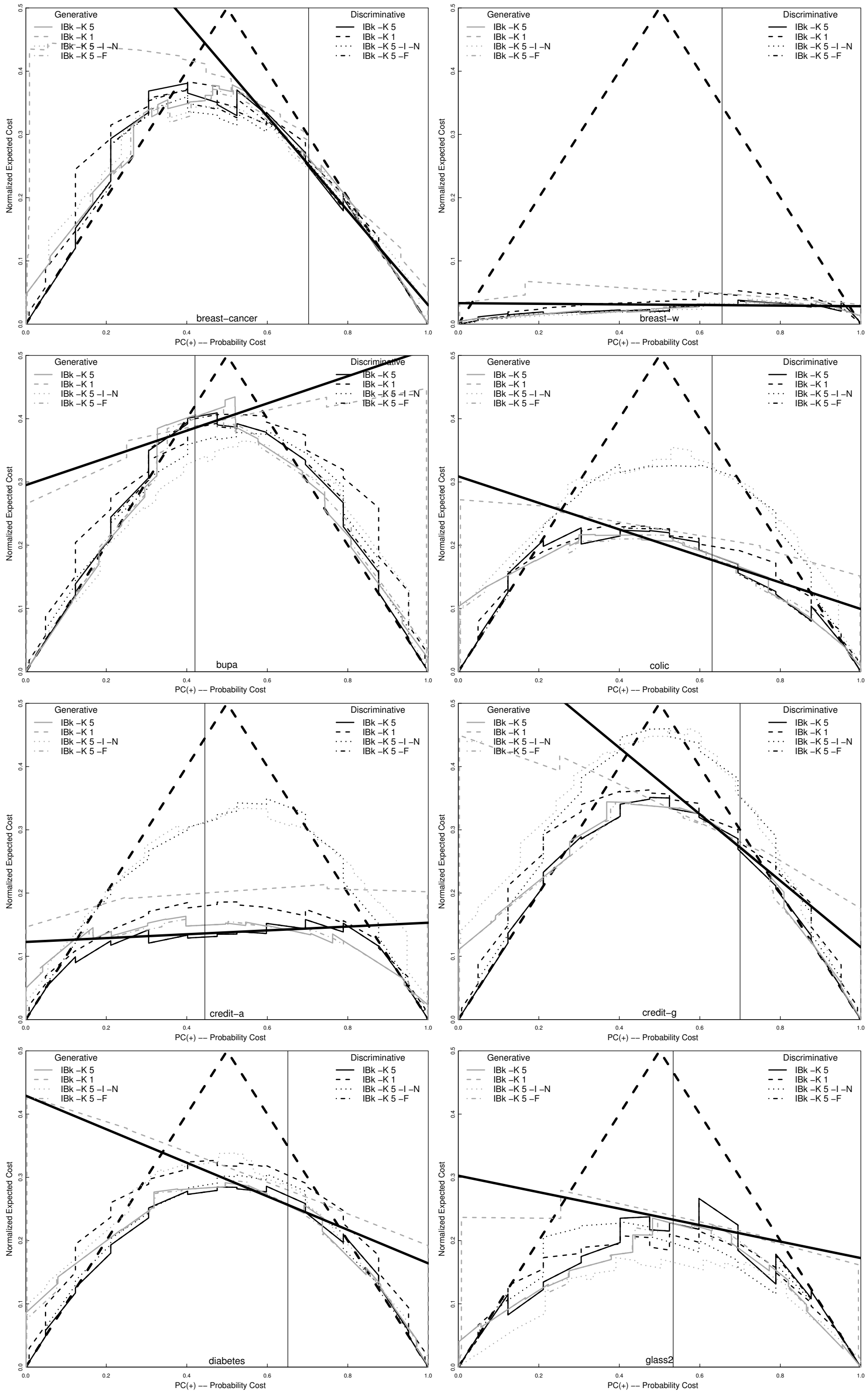

Figure 15: (a) Probability Estimation vs Undersampling:IBK

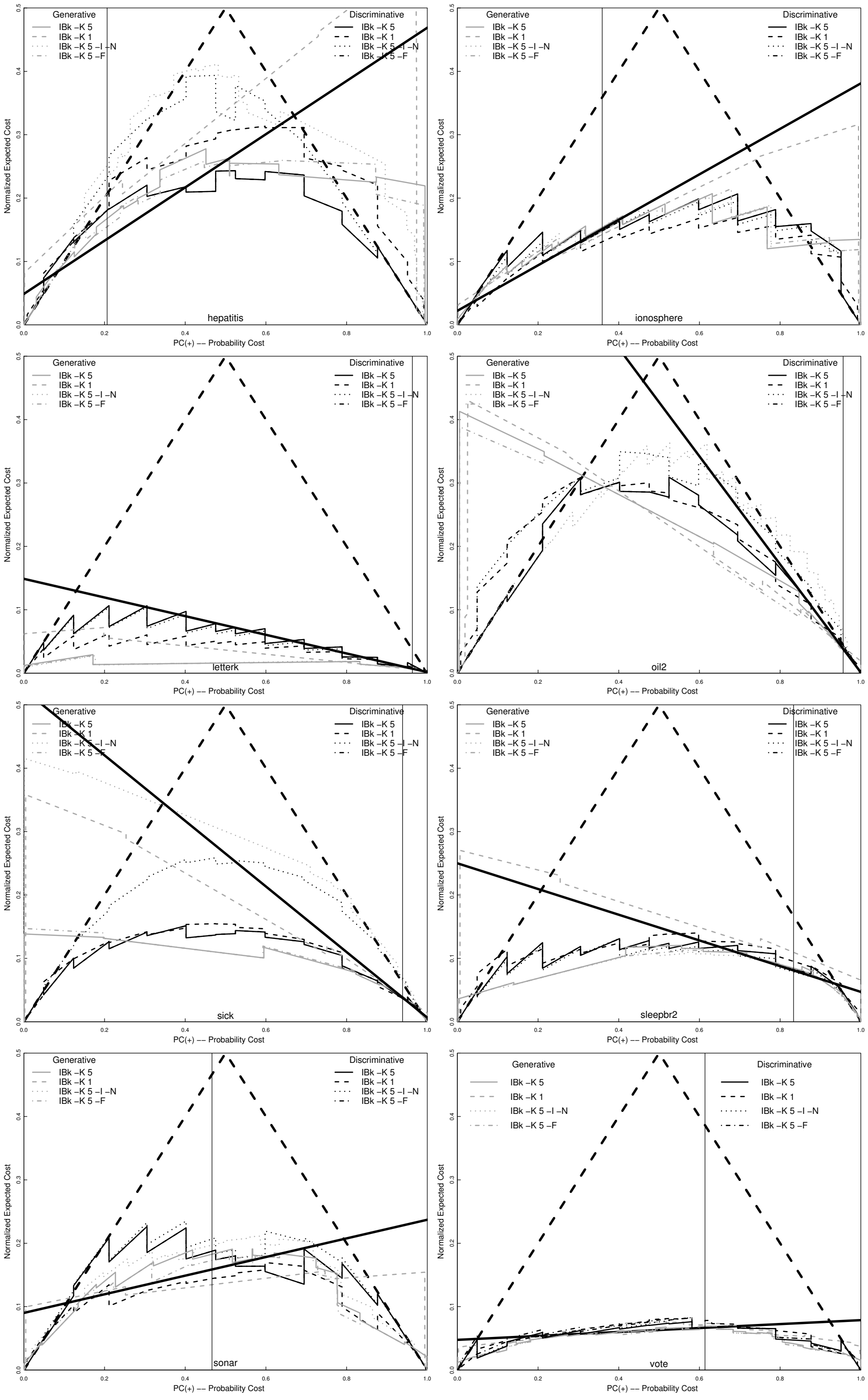

Figure 16: (b)Probability Estimation vs Undersampling: IBk

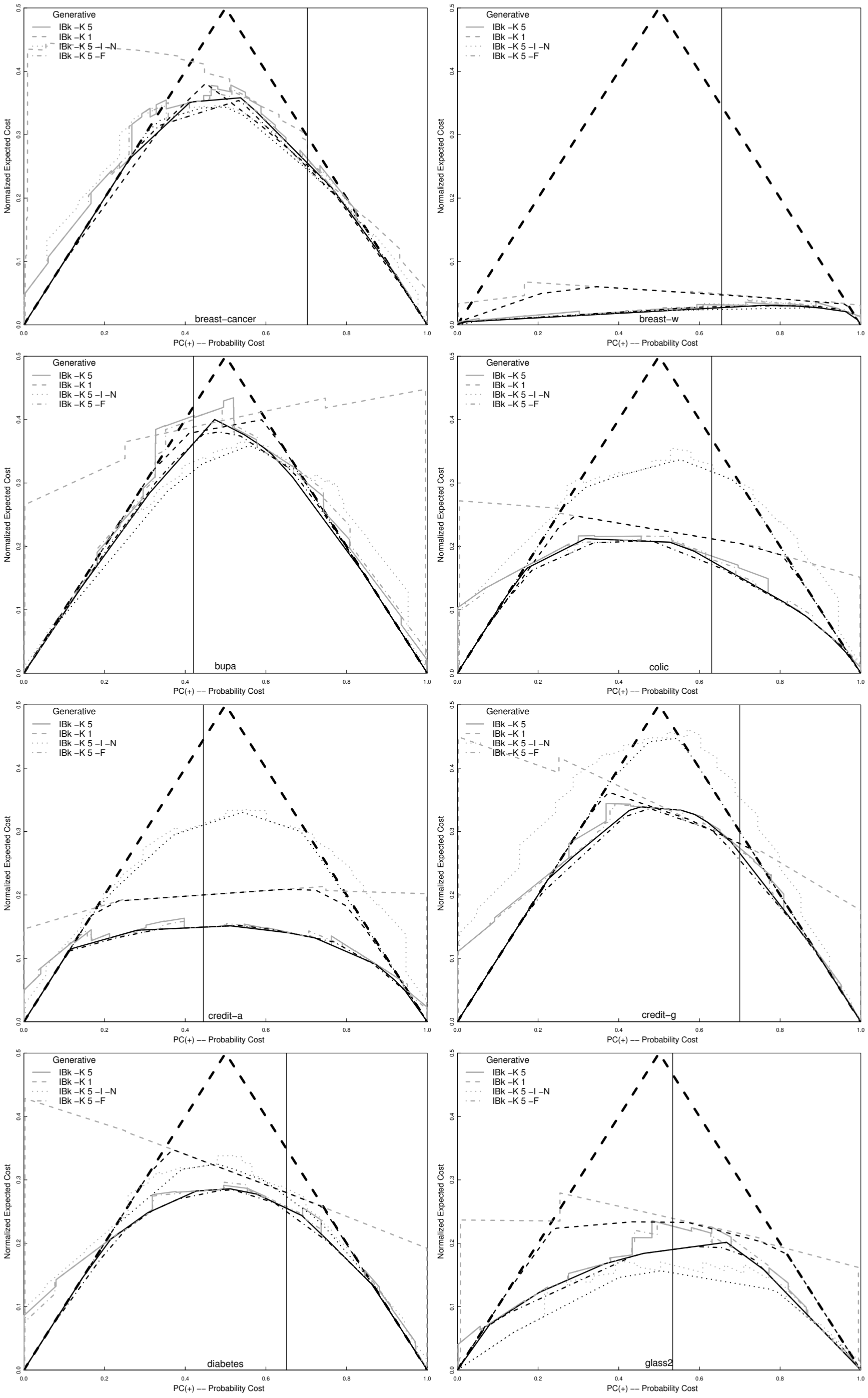

Figure 17: (a) Calibration of Probability Estimate: IBk

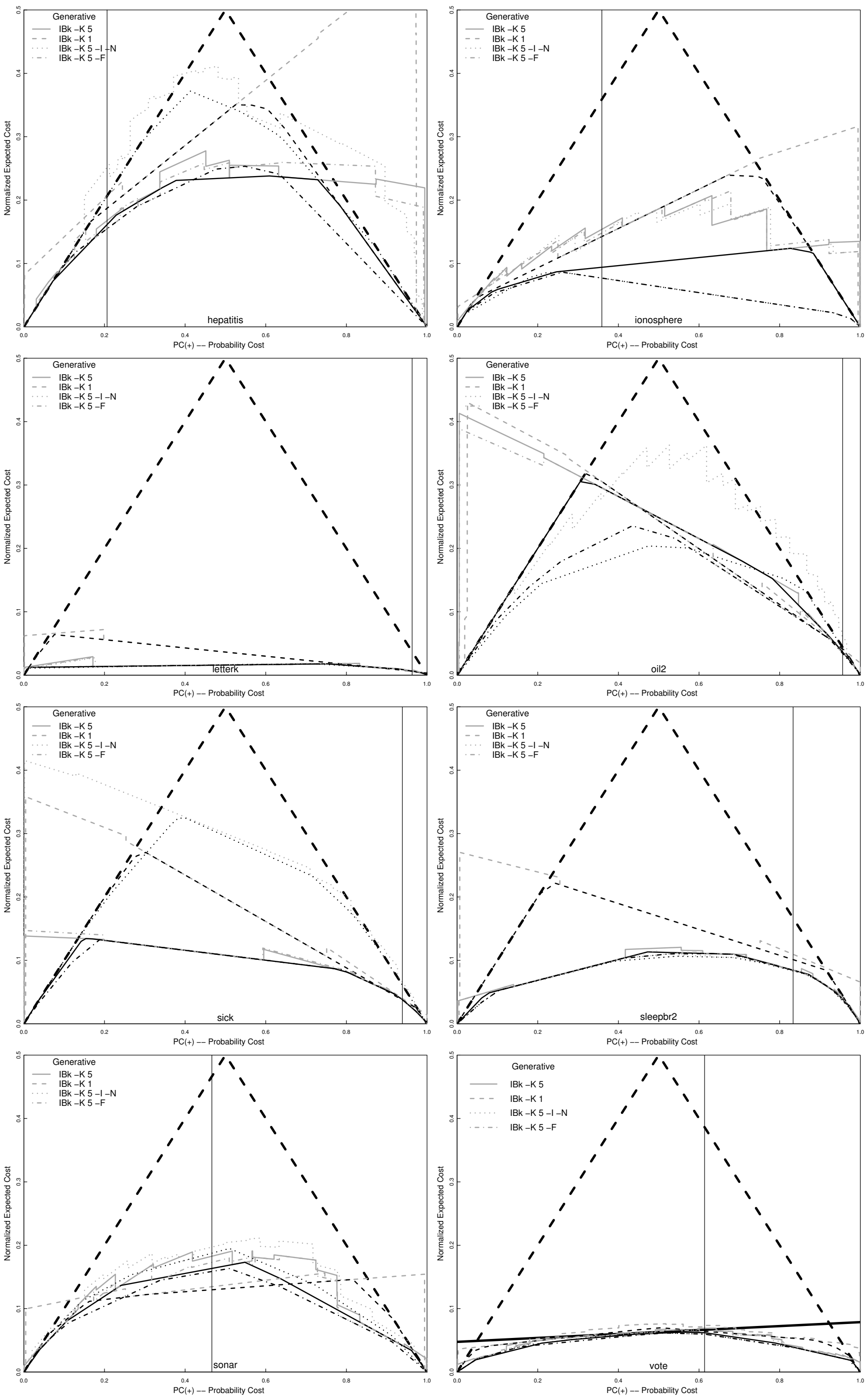

Figure 18: (b) Calibration of Probability Estimate: IBk

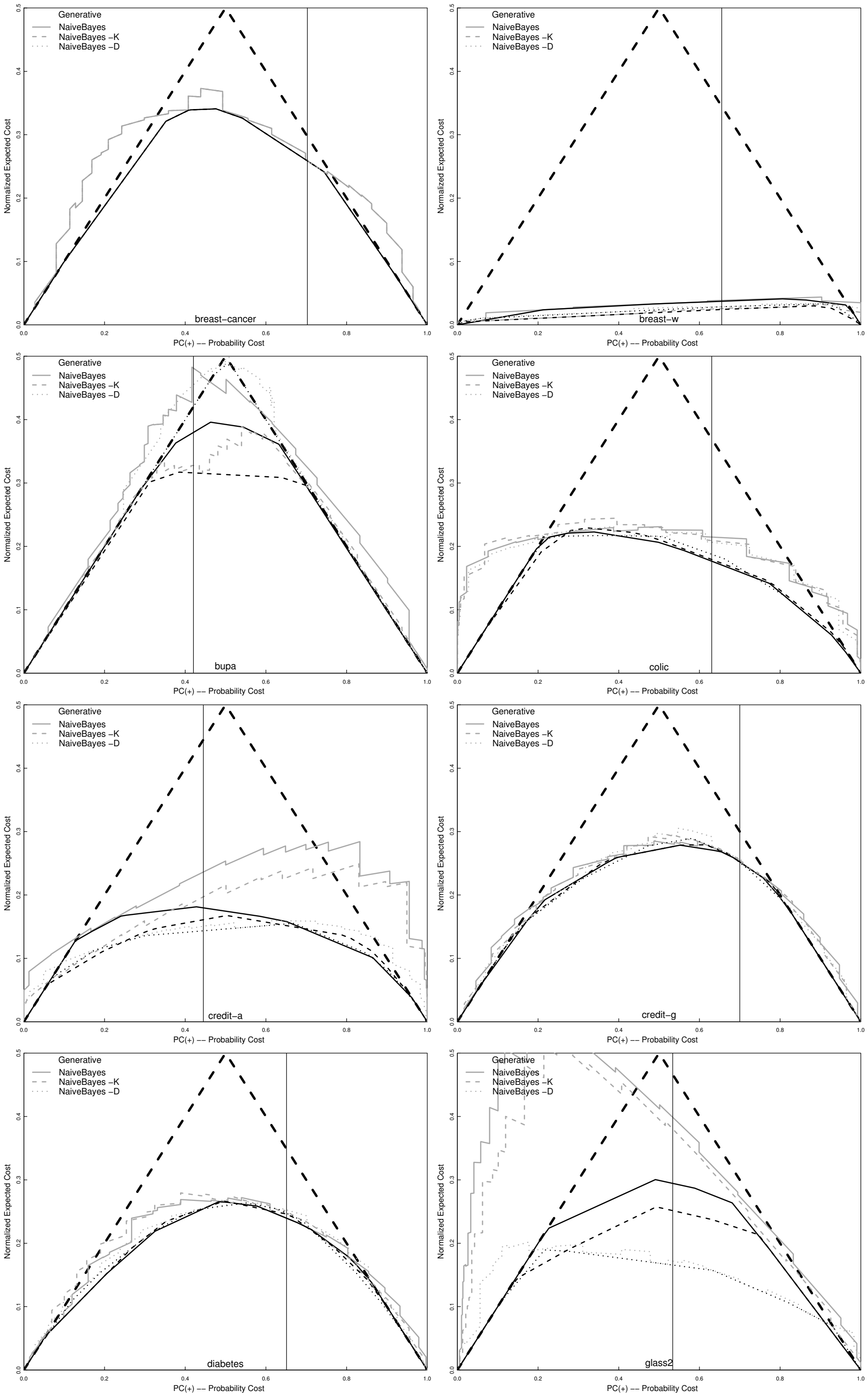

Figure 19: (a) Calibration of Probability Estimate: NaiveBayes

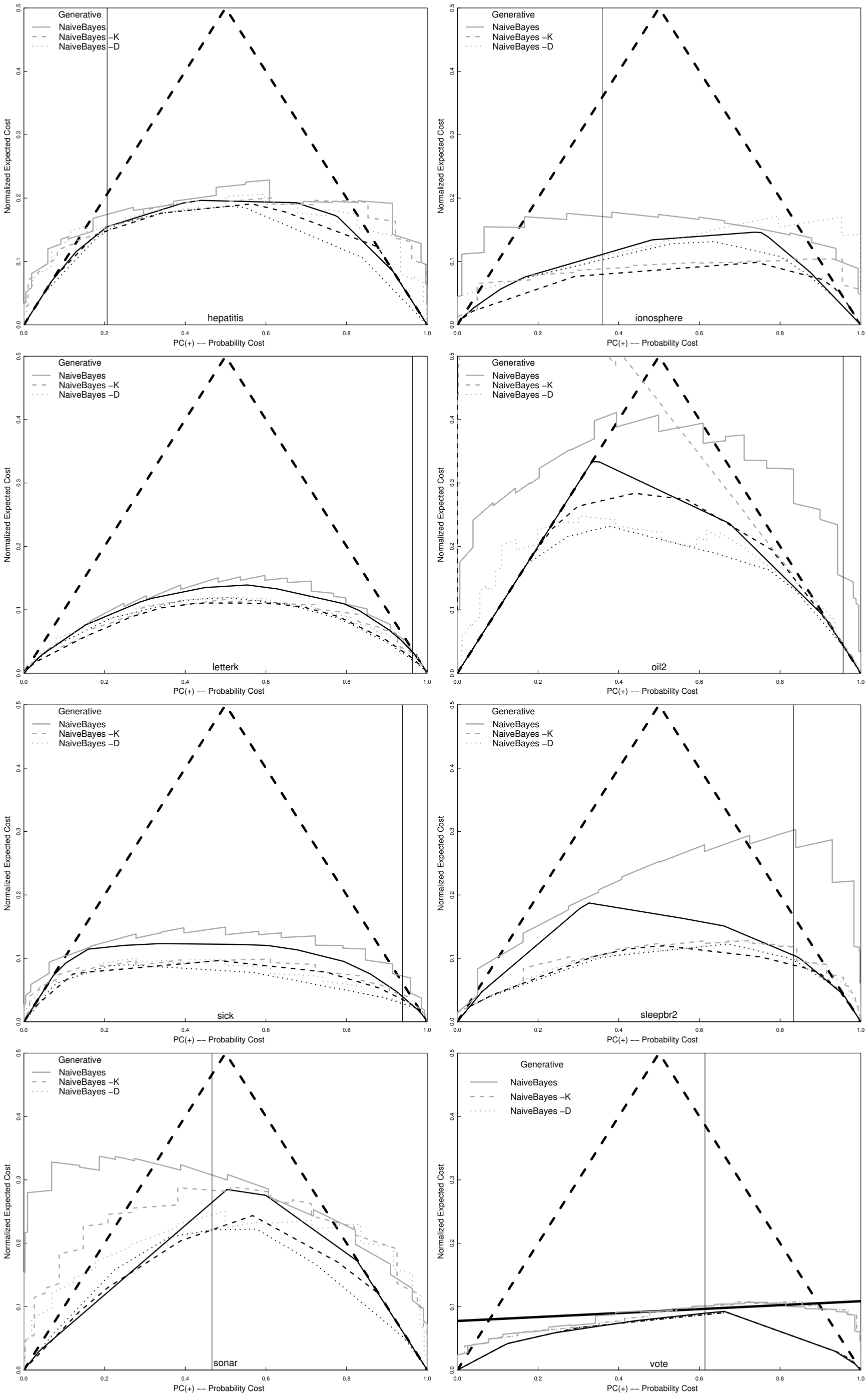

Figure 20: (b) Calibration of Probability Estimate: NaiveBayes

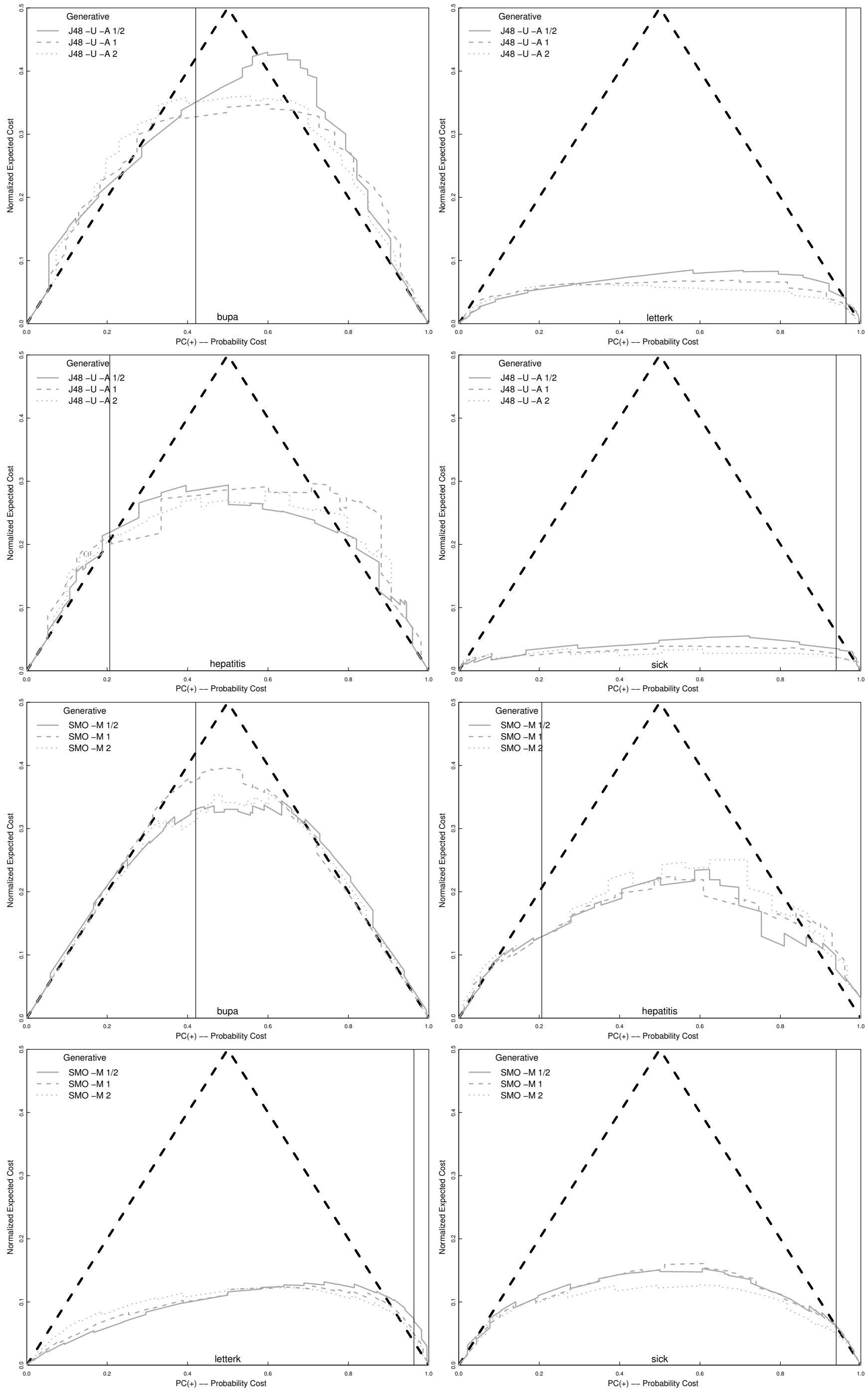

Figure 21: (a) Probability Estimates using new Ratios

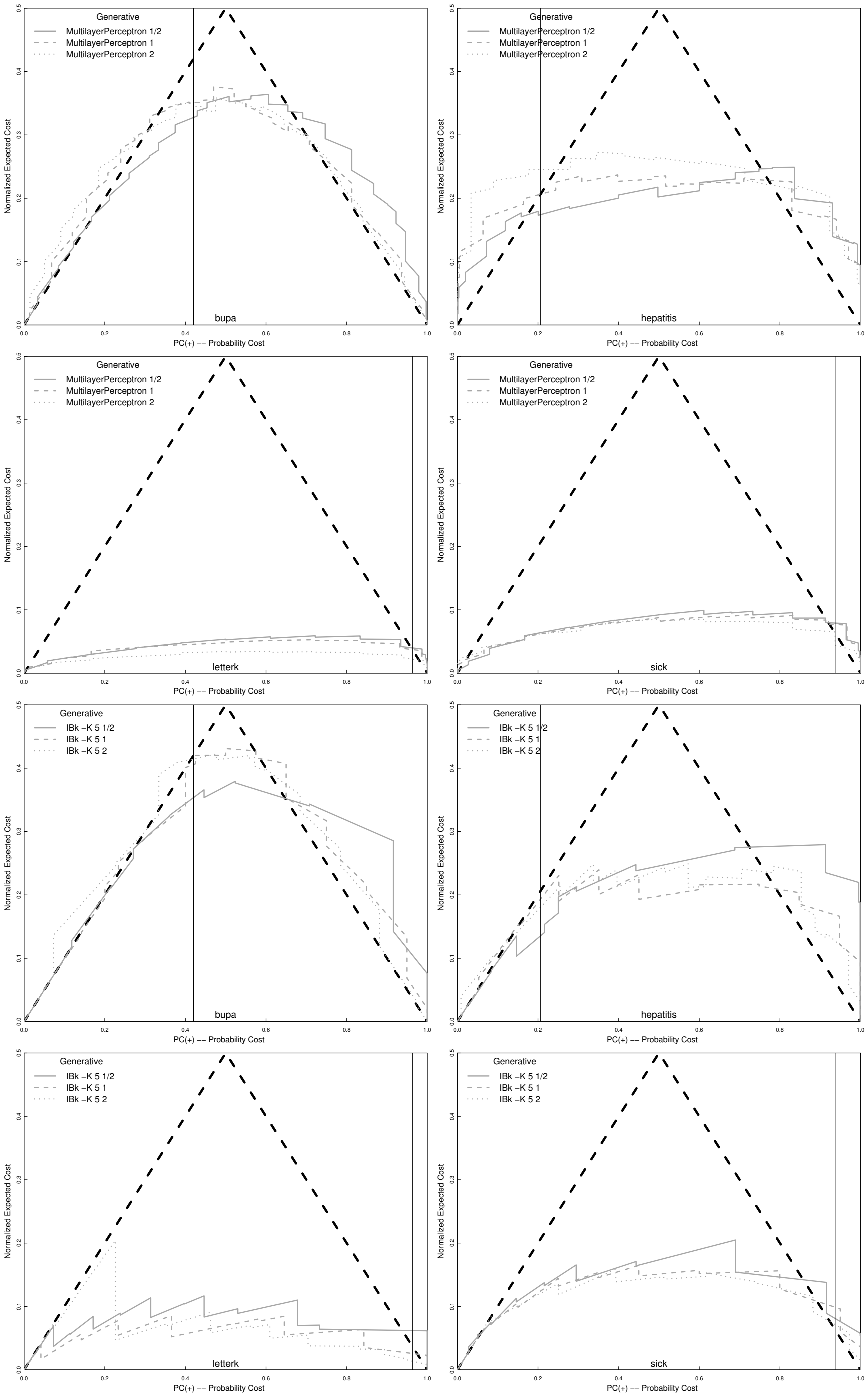

Figure 22: (b) Probabilioty Estimates using new ratios