Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid

Ron Kohavi
Data Mining and Visualization
Silicon Graphics, Inc.
2011 N. Shoreline Blvd
Mountain View, CA 94043-1389
ronnyk@sgi.com

Abstract

Naive-Bayes induction algorithms were previously shown to be surprisingly accurate on many classification tasks even when the conditional independence assumption on which they are based is violated. However, most studies were done on small databases. We show that in some larger databases, the accuracy of Naive-Bayes does not scale up as well as decision trees. We then propose a new algorithm, NBTree, which induces a hybrid of decision-tree classifiers and Naive-Bayes classifiers: the decision-tree nodes contain univariate splits as regular decision-trees, but the leaves contain Naive-Bayesian classifiers. The approach retains the interpretability of Naive-Bayes and decision trees, while resulting in classifiers that frequently outperform both constituents, especially in the larger databases tested.

Introduction

Seeing the future first requires not only a wide-angle lens, it requires a multiplicity of lenses
—Hamel & Prahalad (1994), p. 95

Many data mining tasks require classification of data into classes. For example, loan applications can be classified into either 'approve' or 'disapprove' classes. A *classifier* provides a function that maps (classifies) a data item (instance) into one of several predefined classes (Fayyad, Piatetsky-Shapiro, & Smyth 1996). The automatic induction of classifiers from data not only provides a classifier that can be used to map new instances into their classes, but may also provide a human-comprehensible characterization of the classes. In many cases, interpretability—the ability to understand the output of the induction algorithm—is a crucial step in the design and analysis cycle. Some classifiers are naturally easier to interpret than others; for example, decision-trees (Quinlan 1993) are easy to visualize, while neural-networks are much harder.

Naive-Bayes classifiers (Langley, Iba, & Thompson 1992) are generally easy to understand and the induction of these classifiers is extremely fast, requiring only a single pass through the data if all attributes are discrete. Naive-Bayes classifiers are also very simple and easy to understand. Kononenko (1993) wrote that physicians found the induced classifiers easy to understand when the log probabilities were presented as evidence that adds up in favor of different classes.

Figure 1 shows a visualization of the Naive-Bayes classifier for Fisher's Iris data set, where the task is to determine the type of iris based on four attributes. Each bar represents evidence for a given class and attribute value. Users can immediately see that all values for petal-width and petal length are excellent determiners, while the middle range (2.95-3.35) for sepal-width adds little evidence in favor of one class or another.

Naive-Bayesian classifiers are very robust to irrelevant attributes, and classification takes into account evidence from many attributes to make the final prediction, a property that is useful in many cases where there is no "main effect." On the downside, Naive-Bayes classifiers require making strong independence assumptions and when these are violated, the achievable accuracy may asymptote early and will not improve much as the database size increases.

Decision-tree classifiers are also fast and comprehensible, but current induction methods based on recursive partitioning suffer from the fragmentation problem: as each split is made, the data is split based on the test and after two dozen levels there is usually very little data on which to base decisions.

In this paper we describe a hybrid approach that attempts to utilize the advantages of both decision-trees (*i.e.*, segmentation) and Naive-Bayes (evidence accumulation from multiple attributes). A decision-tree is built with univariate splits at each node, but with Naive-Bayes classifiers at the leaves. The final classifier resembles Utgoff's Perceptron trees (Utgoff 1988), but the induction process is very different and geared toward larger datasets.

The resulting classifier is as easy to interpret as

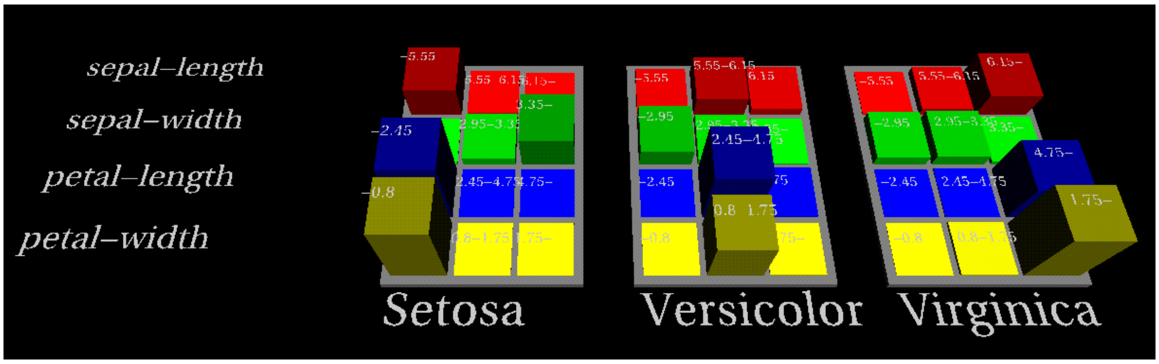

Figure 1: Visualization of a Naive-Bayes classifier for the iris dataset.

decision-trees and Naive-Bayes. The decision-tree segments the data, a task that is consider an essential part of the data mining process in large databases (Brachman & Anand 1996). Each segment of the data, represented by a leaf, is described through a Naive-Bayes classifier. As will be shown later, the induction algorithm segments the data so that the conditional independence assumptions required for Naive-Bayes are likely to be true.

The Induction Algorithms

We briefly review methods for induction of decision-trees and Naive-Bayes.

Decision-tree (Quinlan 1993; Breiman *et al.* 1984) are commonly built by recursive partitioning. A univariate (single attribute) split is chosen for the root of the tree using some criterion (*e.g.*, mutual information, gain-ratio, gini index). The data is then divided according to the test, and the process repeats recursively for each child. After a full tree is built, a pruning step is executed, which reduces the tree size. In the experiments, we compared our results with the C4.5 decision-tree induction algorithm (Quinlan 1993), which is a state-of-the-art algorithm.

Naive-Bayes (Good 1965; Langley, Iba, & Thompson 1992) uses Bayes rule to compute the probability of each class given the instance, assuming the attributes are conditionally independent given the label. The version of Naive-Bayes we use in our experiments was implemented in $\mathcal{MLC}$++ (Kohavi *et al.* 1994). The data is pre-discretized using the an entropy-based algorithm (Fayyad & Irani 1993; Dougherty, Kohavi, & Sahami 1995). The probabilities are estimated directly from data based directly on counts (without any corrections, such as Laplace or m-estimates).

Accuracy Scale-Up: the Learning Curves

A Naive-Bayes classifier requires estimation of the conditional probabilities for each attribute value given the label. For discrete data, because only few parameters need to be estimated, the estimates tend to stabilize quickly and more data does not change the underlying model much. With continuous attributes, the discretization is likely to form more intervals as more data is available, thus increasing the representation power. However, even with continuous data, the discretization is global and cannot take into account attribute interactions.

Decision-trees are non-parametric estimators and can approximate any "reasonable" function as the database size grows (Gordon & Olshen 1984). This theoretical result, however, may not be very comforting if the database size required to reach the asymptotic performance is more than the number of atoms in the universe, as is sometimes the case. In practice, some parametric estimators, such as Naive-Bayes, may perform better.

Figure 2 shows learning curves for both algorithms on large datasets from the UC Irvine repository[1] (Murphy & Aha 1996). The learning curves show how the accuracy changes as more instances (training data) are shown to the algorithm. The accuracy is computed based on the data not used for training, so it represents the true generalization accuracy. Each point was computed as an average of 20 runs of the algorithm, and 20 intervals were used. The error bars show 95% confidence intervals on the accuracy, based on the left-out sample.

In most cases it is clear that even with much more

[1] The Adult dataset is from the Census bureau and the task is to predict whether a given adult makes more than $50,000 a year based attributes such as education, hours of work per week, *etc.*.

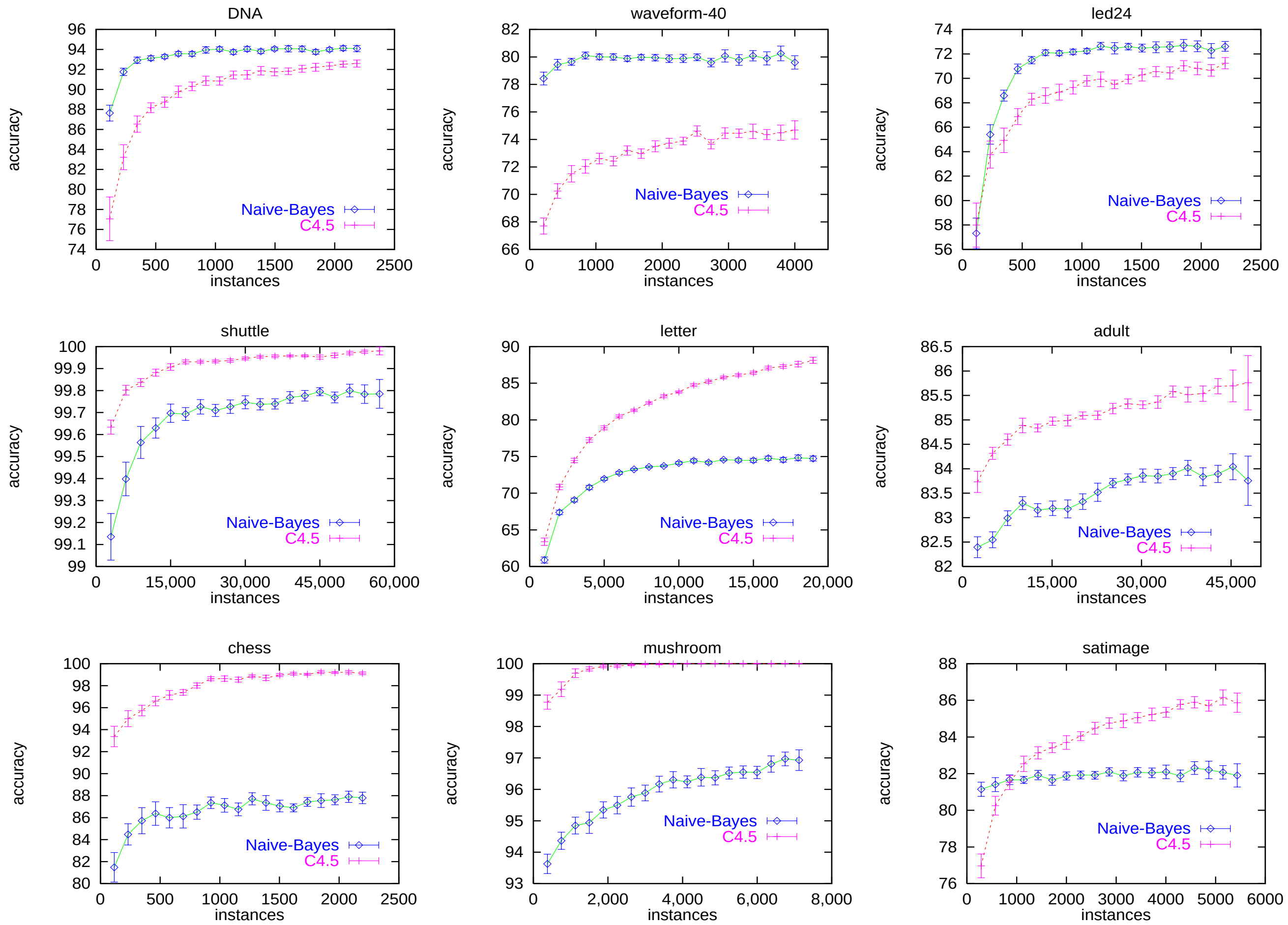

Figure 2: Learning curves for Naive-Bayes and C4.5. The top three graphs show datasets where Naive-Bayes outperformed C4.5, and the lower six graphs show datasets where C4.5 outperformed Naive-Bayes. The error bars are 95% confidence intervals on the accuracy.

data, the learning curves will not cross. While it is well known that no algorithm can outperform all others in all cases (Wolpert 1994), our world does tend to have some smoothness conditions and algorithms can be more successful than others in practice. In the next section we show that a hybrid approach can improve both algorithms in important practical datasets.

NBTree: The Hybrid Algorithm

The NBTree algorithm we propose is shown in Figure 3. The algorithm is similar to the classical recursive partitioning schemes, except that the leaf nodes created are Naive-Bayes categorizers instead of nodes predicting a single class.

A threshold for continuous attributes is chosen using the standard entropy minimization technique, as is done for decision-trees. The utility of a node is computed by discretizing the data and computing the 5-fold cross-validation accuracy estimate of using Naive-Bayes at the node. The utility of a split is the weighted sum of the utility of the nodes, where the weight given to a node is proportional to the number of instances that go down to that node.

Intuitively, we are attempting to approximate whether the generalization accuracy for a Naive-Bayes classifier at each leaf is higher than a single Naive-Bayes classifier at the current node. To avoid splits with little value, we define a split to be *significant* if the relative (not absolute) reduction in error is greater than 5% and there are at least 30 instances in the node.

Direct use of cross-validation to select attributes has not been commonly used because of the large overhead involved in using it in general. However, if the data is discretized, Naive-Bayes can be cross-validated in time that is linear in the number of instances, number of attributes, and number of label values. The reason is

Input: a set T of labelled instances.
Output: a decision-tree with naive-bayes categorizers at the leaves.

1. For each attribute X_i, evaluate the utility, $u(X_i)$, of a split on attribute X_i. For continuous attributes, a threshold is also found at this stage.
2. Let $j = \arg\max_i(u_i)$, *i.e.*, the attribute with the highest utility.
3. If u_j is not significantly better than the utility of the current node, create a Naive-Bayes classifier for the current node and return.
4. Partition T according to the test on X_j. If X_j is continuous, a threshold split is used; if X_j is discrete, a multi-way split is made for all possible values.
5. For each child, call the algorithm recursively on the portion of T that matches the test leading to the child.

Figure 3: The NBTree algorithm. The utility $u(X_i)$ is described in the text.

that we can remove the instances, update the counters, classify them, and repeat for a different set of instances. See Kohavi (1995) for details.

Given m instances, n attributes, and ℓ label values, the complexity of the attribute selection phase for discretized attributes is $O(m \cdot n^2 \cdot \ell)$. If the number of attributes is less than $O(\log m)$, which is usually the case, and the number of labels is small, then the time spent on attribute selection using cross-validation is less than the time spent sorting the instances by each attribute. We can thus expect NBTree to scale up well to large databases.

Experiments

To evaluate the NBTree algorithm we used a large set of files from the UC Irvine repository. Table 1 describes the characteristics of the data. Artificial files (*e.g.*, monk1) were evaluated on the whole space of possible values; files with over 3,000 instances were evaluated on a left out sample which is of size one third of the data, unless a specific test set came with the data (*e.g.*, shuttle, DNA, satimage); other files were evaluated using 10-fold cross-validation. C4.5 has a complex mechanism for dealing with unknown values. To eliminate the effects of unknown values, we have removed all instances with unknown values from the datasets prior to the experiments.

Figure 4 shows the absolute differences between the accuracies for C4.5, Naive-Bayes, and NBTree. Each line represents the accuracy difference for NBTree and one of the two other methods. The average accuracy for C4.5 is 81.91%, for Naive-Bayes it is 81.69%, and for NBTree it is 84.47%.

Absolute differences do not tell the whole story because the accuracies may be close to 100% in some cases. Increasing the accuracy of medical diagnosis from 98% to 99% may cut costs by half because the number of errors is halved. Figure 5 shows the ratio of errors (where error is 100%-accuracy). The shuttle dataset, which is the largest dataset tested, has only 0.04% absolute difference between NBTree and C4.5, but the error decreases from 0.05% to 0.01%, which is a huge relative improvement.

The number of nodes induced by NBTree was in many cases significantly smaller than that of C4.5. For example, for the letter dataset, C4.5 induced 2109 nodes while NBTree induced only 251; in the adult dataset, C4.5 induced 2213 nodes while NBTree induced only 137; for DNA, C4.5 induced 131 nodes and NBTree induced 3; for led24, C4.5 induced 49 nodes, while NBTree used a single node. While the complexity of each leaf in NBTree is higher, ordinary trees with thousands of nodes could be extremely hard to interpret.

Related Work

Many attempts have been made to extend Naive-Bayes or to restrict the learning of general Bayesian networks. Approaches based on feature subset selection may help, but they cannot increase the representation power as was done here, thus we will not review them.

Kononenko (1991) attempted to join pairs of attributes (make a cross-product attribute) based on statistical tests for independence. Experimentation results were very disappointing. Pazzani (1995) searched for attributes to join based on cross-validation estimates.

Recently, Friedman & Goldszmidt (1996) showed how to learn a Tree Augmented Naive-Bayes (TAN), which is a Bayes network restricted to a tree topology. The results are promising and running times should scale up, but the approach is still restrictive. For example, their accuracy for the Chess dataset, which contains high-order interactions is about 93%, much lower then C4.5 and NBTree, which achieve accuracies above 99%.

Conclusions

We have described a new algorithm, NBTree, which is a hybrid approach suitable in learning scenarios when many attributes are likely to be relevant for a classification task, yet the attributes are not necessarily conditionally independent given the label.

NBTree induces highly accurate classifiers in practice, significantly improving upon both its constituents

Dataset	No attrs	Train size	Test size	Dataset	No attrs	Train size	Test size	Dataset	No attrs	Train size	Test size
adult	14	30,162	15,060	breast (L)	9	277	CV-10	breast (W)	10	683	CV-10
chess	36	2,130	1,066	cleve	13	296	CV-10	crx	15	653	CV-10
DNA	180	2,000	1,186	flare	10	1,066	CV-10	german	20	1,000	CV-10
glass	9	214	CV-10	glass2	9	163	CV-10	heart	13	270	CV-10
ionosphere	34	351	CV-10	iris	4	150	CV-10	led24	24	200	3000
letter	16	15,000	5,000	monk1	6	124	432	mushroom	22	5,644	3,803
pima	8	768	CV-10	primary-tumor	17	132	CV-10	satimage	36	4,435	2,000
segment	19	2,310	CV-10	shuttle	9	43,500	14,500	soybean-large	35	562	CV-10
tic-tac-toe	9	958	CV-10	vehicle	18	846	CV-10	vote	16	435	CV-10
vote1	15	435	CV-10	waveform-40	40	300	4,700				

Table 1: The datasets used, the number of attributes, and the training/test-set sizes (CV-10 denotes 10-fold cross-validation was used).

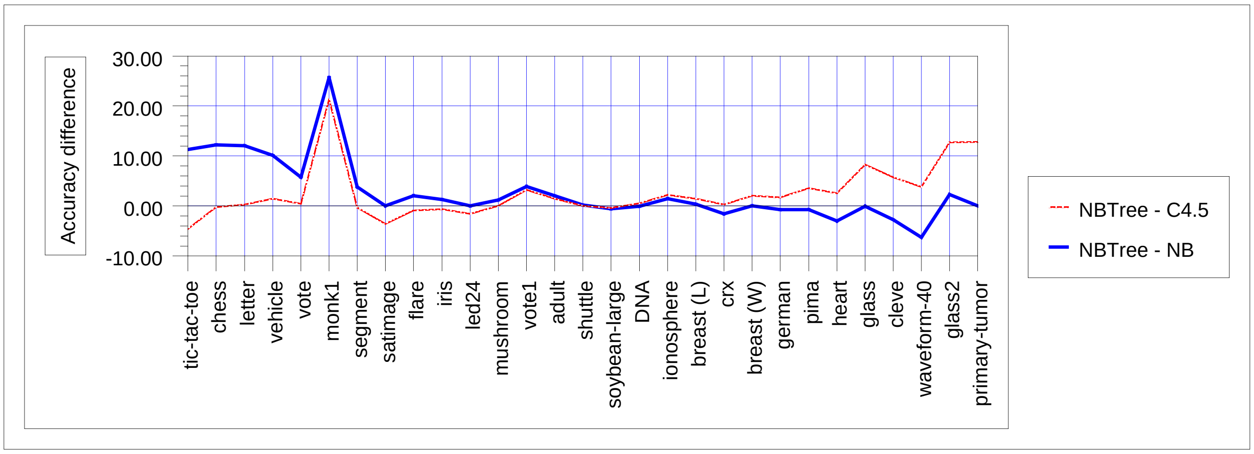

Figure 4: The accuracy differences. One line represents the accuracy difference between NBTree and C4.5 and the other between NBTree and Naive-Bayes. Points above the zero show improvements. The files are sorted by the difference of the two lines so that they cross once.

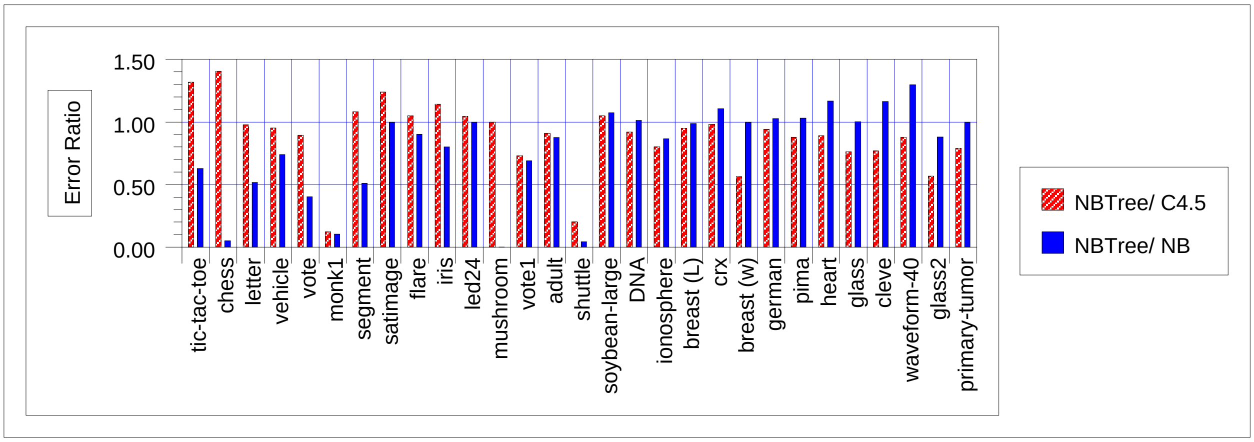

Figure 5: The error ratios of NBTree to C4.5 and Naive-Bayes. Values less than one indicate improvement.

in many cases. Although no classifier can outperform others in all domains, NBTree seems to work well on real-world datasets we tested and it scales up well in terms of its accuracy. In fact, for the three datasets over 10,000 instances (adult, letter, shuttle), it outperformed both C4.5 and Naive-Bayes. Running time is longer than for decision-trees and Naive-Bayes alone, but the dependence on the number of instances for creating a split is the same as for decision-trees, $O(m \log m)$, indicating that the running time can scale up well.

Interpretability is an important issue in data mining applications. NBTree segments the data using a univariate decision-tree, making the segmentation easy to understand. Each leaf is a Naive-Bayes classifiers, which can also be easily understood when displayed graphically, as shown in Figure 1. The number of nodes induced by NBTree was in many cases significantly smaller than that of C4.5.

Acknowledgments We thank Yeo-Girl (Yogo) Yun who implemented the original CatDT categorizer in $\mathcal{MLC}$++. Dan Sommerfield wrote the Naive-Bayes visualization routines in $\mathcal{MLC}$++.

References

Brachman, R. J., and Anand, T. 1996. The process of knowledge discovery in databases. In *Advances in Knowledge Discovery and Data Mining*. AAAI Press and the MIT Press. chapter 2, 37–57.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees*. Wadsworth International Group.

Dougherty, J.; Kohavi, R.; and Sahami, M. 1995. Supervised and unsupervised discretization of continuous features. In Prieditis, A., and Russell, S., eds., *Machine Learning: Proceedings of the Twelfth International Conference*, 194–202. Morgan Kaufmann.

Fayyad, U. M., and Irani, K. B. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1022–1027. Morgan Kaufmann Publishers, Inc.

Fayyad, U. M.; Piatetsky-Shapiro, G.; and Smyth, P. 1996. From data mining to knowledge discovery: An overview. In *Advances in Knowledge Discovery and Data Mining*. AAAI Press and the MIT Press. chapter 1, 1–34.

Friedman, N., and Goldszmidt, M. 1996. Building classifiers using bayesian networks. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. To appear.

Good, I. J. 1965. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. M.I.T. Press.

Gordon, L., and Olshen, R. A. 1984. Almost sure consistent nonparametric regression from recursive partitioning schemes. *Journal of Multivariate Analysis* 15:147–163.

Hamel, G., and Prahalad, C. K. 1994. *Competing for the Future*. Harvard Business School Press and McGraw Hill.

Kohavi, R.; John, G.; Long, R.; Manley, D.; and Pfleger, K. 1994. MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence*, 740–743. IEEE Computer Society Press. `http://www.sgi.com/Technology/mlc`.

Kohavi, R. 1995. *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. Ph.D. Dissertation, Stanford University, Computer Science department.
ftp://starry.stanford.edu/pub/ronnyk/teza.ps.

Kononenko, I. 1991. Semi-naive bayesian classifiers. In *Proceedings of the sixth European Working Session on Learning*, 206–219.

Kononenko, I. 1993. Inductive and bayesian learning in medical diagnosis. *Applied Artificial Intelligence* 7:317–337.

Langley, P.; Iba, W.; and Thompson, K. 1992. An analysis of bayesian classifiers. In *Proceedings of the tenth national conference on artificial intelligence*, 223–228. AAAI Press and MIT Press.

Murphy, P. M., and Aha, D. W. 1996. UCI repository of machine learning databases.
`http://www.ics.uci.edu/~mlearn`.

Pazzani, M. 1995. Searching for attribute dependencies in bayesian classifiers. In *Fifth International Workshop on Artificial Intelligence and Statistics*, 424–429.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Los Altos, California: Morgan Kaufmann Publishers, Inc.

Utgoff, P. E. 1988. Perceptron trees: a case study in hybrid concept representation. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 601–606. Morgan Kaufmann.

Wolpert, D. H. 1994. The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework. In Wolpert, D. H., ed., *The Mathemtatics of Generalization*. Addison Wesley.