

Technical Note

Naive Bayes for Regression

EIBE FRANK

eibe@cs.waikato.ac.nz

LEONARD TRIGG

trigg@cs.waikato.ac.nz

GEOFFREY HOLMES

geoff@cs.waikato.ac.nz

IAN H. WITTEN

ihw@cs.waikato.ac.nz

Department of Computer Science, University of Waikato, Hamilton, New Zealand

Editor: David W. Aha

Abstract. Despite its simplicity, the naive Bayes learning scheme performs well on most classification tasks, and is often significantly more accurate than more sophisticated methods. Although the probability estimates that it produces can be inaccurate, it often assigns maximum probability to the correct class. This suggests that its good performance might be restricted to situations where the output is categorical. It is therefore interesting to see how it performs in domains where the predicted value is numeric, because in this case, predictions are more sensitive to inaccurate probability estimates.

This paper shows how to apply the naive Bayes methodology to numeric prediction (i.e., regression) tasks by modeling the probability distribution of the target value with kernel density estimators, and compares it to linear regression, locally weighted linear regression, and a method that produces “model trees”—decision trees with linear regression functions at the leaves. Although we exhibit an artificial dataset for which naive Bayes is the method of choice, on real-world datasets it is almost uniformly worse than locally weighted linear regression and model trees. The comparison with linear regression depends on the error measure: for one measure naive Bayes performs similarly, while for another it is worse. We also show that standard naive Bayes applied to regression problems by discretizing the target value performs similarly badly. We then present empirical evidence that isolates naive Bayes’ independence assumption as the culprit for its poor performance in the regression setting. These results indicate that the simplistic statistical assumption that naive Bayes makes is indeed more restrictive for regression than for classification.

Keywords: Naive Bayes, regression, model trees, linear regression, locally weighted regression

1. Introduction

Naive Bayes relies on an assumption that is rarely valid in practical learning problems: that the attributes used for deriving a prediction are independent of each other, given the predicted value. As an example where this assumption is inappropriate, suppose that the type of a fish is to be predicted from its length and weight. Given an individual fish of a particular species, its weight obviously depends greatly on its length—and vice versa. However, it has been shown that, for classification problems where the predicted value is categorical, the independence assumption is less restrictive than might be expected. For several practical classification tasks, naive Bayes produces significantly lower error rates than more sophisticated learning schemes, such as ones that learn univariate decision trees (Domingos & Pazzani, 1997).

Why does naive Bayes perform well even when the independence assumption is seriously violated? Most likely it owes its good performance to the zero-one loss function used in classification (Domingos & Pazzani, 1997). This function defines the error as the number of incorrect predictions. Unlike other loss functions, such as the squared error, it has the key property that it does not penalize inaccurate probability estimates—so long as the greatest probability is assigned to the correct class (Friedman, 1997). There is mounting evidence that this is why naive Bayes’ classification performance remains high, despite the fact that inter-attribute dependencies often cause it to produce incorrect probability estimates (Domingos & Pazzani, 1997). This raises the question of whether it can be successfully applied to non-categorical prediction problems, where the zero-one loss function is of no use. This paper investigates the application of naive Bayes to problems where the predicted value is not categorical but numeric.

Naive Bayes assigns a probability to every possible value in the target range. The resulting distribution is then condensed into a single prediction. In categorical problems, the optimal prediction under zero-one loss is the most likely value—the mode of the underlying distribution. However, in numeric problems the optimal prediction is either the mean or the median, depending on the loss function. These two statistics are far more sensitive to the underlying distribution than the most likely value: they almost always change when the underlying distribution changes, even by a small amount.¹ Therefore, when used for numeric prediction, naive Bayes is more sensitive to inaccurate probability estimates than when it is used for classification. This paper explains how it can be used for regression, and exhibits an artificial dataset where it is the method of choice. It then summarizes its performance on a set of practical learning problems. It turns out that the remarkable accuracy of naive Bayes for classification on standard benchmark datasets does not translate into the context of regression.

The use of naive Bayes for classification has been investigated extensively. The algorithm itself originated in the field of pattern recognition (Duda & Hart, 1973). Kasif *et al.* (1998) relate it to an instance-based learning algorithm, and discuss its limitations. Its surprisingly high accuracy—in comparison to more sophisticated learning methods—has frequently been noted (Cestnik, 1990; Clark & Niblett, 1989; Langley, Iba & Thompson, 1992). Domingos and Pazzani (1997) performed a large-scale comparison of naive Bayes with state-of-the-art algorithms for decision tree induction, instance-based learning, and rule induction on standard benchmark datasets, and found it to be superior to each of the other learning schemes *even on datasets with substantial attribute dependencies*.

Despite this, several researchers have tried to improve naive Bayes by deleting redundant attributes (Langley & Sage, 1994; John & Kohavi, 1997), or by extending it to incorporate simple high-order dependencies (Kononenko, 1991; Langley, 1993; Pazzani, 1996; Sahami, 1996; Friedman, Geiger & Goldszmidt, 1997). Domingos and Pazzani (1997) review these approaches in some detail, and conclude that “... attempts to build on [naive Bayes’] success by relaxing the independence assumption have had mixed results.”

Naive Bayes has previously been applied to the related problem of time series prediction by Kononenko (1998), using a regression-by-discretization approach. More specifically, he discretized the numeric target value using an “ad hoc” approach (Kononenko, 1998), and applied standard naive Bayes for classification to the discretized data. During prediction, the sum of the means of each of the pseudo-classes was output, weighted according to the class probabilities assigned by naive Bayes. According to Kononenko (1998), naive Bayes “... performed comparably to well known methods for time series prediction and sometimes even slightly better.” This paper shows that it does not perform as well in the related context of regression. To our knowledge, an extensive evaluation of naive Bayes for regression has not been published previously in the literature on machine learning, pattern recognition, or statistics.

This paper is organized as follows. In the next section we describe a method for applying naive Bayes directly to regression problems, without discretizing the target value. In Section 3 we exhibit an artificial dataset for which the independence assumption is true, and show that, on this dataset, naive Bayes performs better than two state-of-the-art methods for regression: locally weighted linear regression and model trees. In Section 4 we then turn to practical datasets to compare the predictive performance of naive Bayes to those of linear regression, locally weighted linear regression, and model trees. Section 5 investigates how well our method performs compared to standard naive Bayes, both in the case of classification, and in the case of regression. In Section 6 we present results of an experiment investigating how the independence assumption influences the performance of naive Bayes for regression. Section 7 discusses the results and draws some conclusions.

2. Naive Bayes for Regression

We address the problem of predicting a numeric target value Y , given an example E . E consists of m attributes X_1, X_2, \dots, X_m . Each attribute is either numeric, in which case it is treated as a real number, or nominal, in which case it is a set of unordered values.

If the probability density function $p(Y|E)$ of the target value were known for all possible examples E , we could choose Y to minimize the expected prediction error. However, $p(Y|E)$ is usually not known, and has to be estimated from data. Naive Bayes achieves this by applying Bayes’ theorem and assuming independence of the attributes X_1, X_2, \dots, X_m given the target value Y . Bayes’ theorem states that

$$p(Y|E) = \frac{p(E, Y)}{\int p(E, Y)dY} = \frac{p(E|Y)p(Y)}{\int p(E|Y)p(Y)dY}, \quad (1)$$

where the likelihood $p(E|Y)$ is the probability density function (pdf) of the example E for a given target value Y , and the prior $p(Y)$ is the pdf of the target value before any examples have been seen. Naive Bayes makes the key assumption that the attributes are independent given the target value, and so Equation 1 can be written

$$p(Y|E) = \frac{p(X_1|Y)p(X_2|Y) \cdots p(X_m|Y)p(Y)}{\int p(X_1|Y)p(X_2|Y) \cdots p(X_m|Y)p(Y)dY}. \quad (2)$$

Instead of estimating the pdf $p(E|Y)$, the individual pdfs $p(X_i|Y)$ can now be estimated separately. This dimensionality reduction makes the learning problem much easier. Because the amount of data needed to obtain an accurate estimate increases with the dimensionality of the problem, $p(X_i|Y)$ can be estimated more reliably than $p(E|Y)$. However, a question remains: how harmful is the independence assumption when it is not valid for the learning problem at hand? The empirical results presented in Section 4 shed some light on this issue.

The following sections discuss how $p(X_i|Y)$ and $p(Y)$ can be estimated from a set of training examples. For $p(X_i|Y)$ there are two cases to consider: the case where attribute X_i is numeric, and the case where it is nominal.

2.1. Handling Numeric Attributes

We first discuss the estimation of $p(X|Y)$ for numeric attributes X , where we assume that they are normalized by their range, as determined using the training data. In this case, $p(X|Y)$ is a pdf involving two numeric variables. Because

$$p(X|Y) = \frac{p(X, Y)}{\int p(X, Y)dX}, \quad (3)$$

the conditional probability $p(X|Y)$ can be estimated by computing an approximation to the joint probability $p(X, Y)$. In principle, any estimator for two-dimensional pdfs can be used to model $p(X, Y)$, for example, mixture models (Ghahramani & Jordan, 1994). We have chosen the kernel density estimator

$$\hat{p}(X = x, Y = y) = \frac{1}{nh_X h_Y} \sum_{i=1}^n K\left(\frac{x - x_i}{h_X}\right) K\left(\frac{y - y_i}{h_Y}\right), \quad (4)$$

where x_i is the attribute value, y_i the target value of training example i , $K(\cdot)$ is a given kernel function, and h_X and h_Y are the kernel widths for X and Y . If either x_i or y_i is missing, the example is not included in the calculation. It can be shown that this estimate converges to the true pdf if the kernel function obeys certain smoothness properties and the kernel widths are chosen appropriately (Silverman, 1986).

A common choice for $K(\cdot)$ is the Gaussian kernel $K(t) = (2\pi)^{-1/2}e^{-t^2/2}$, and this is what we use in our experiments. Ideally, the kernel widths h_X and h_Y should be chosen so that the difference between the estimated pdf $\hat{p}(X, Y)$ and the true pdf $p(X, Y)$ is minimized. One way of measuring this difference is the expected cross-entropy between the two pdfs, an unbiased estimate of which can be obtained by leave-one-out cross-validation (Smyth et al., 1995):

$$CV_{CE} = -\frac{1}{n} \sum_{j=1}^n \log \left(\frac{1}{(n-1)h_X h_Y} \sum_{i=1, i \neq j}^n K\left(\frac{x_j - x_i}{h_X}\right) K\left(\frac{y_j - y_i}{h_Y}\right) \right) \quad (5)$$

Then, h_X and h_Y are set to $h_X = c_X/\sqrt{n}$ and $h_Y = c_Y/\sqrt{n}$, where c_X and c_Y are chosen to minimize the estimated cross-entropy. In our experiments, we use a grid search for $(c_X, c_Y) \in [0.4, 0.8] \times [0.4, 0.8]$ with a grid width of 0.1. We have tried other parameter settings for the search and found little difference in the result. Because

$$\int \hat{p}(X, Y) dX = \frac{1}{nh_Y} \sum_{i=1}^n K\left(\frac{y - y_i}{h_Y}\right), \quad (6)$$

we have all the terms needed to compute an estimate $\hat{p}(X|Y)$ of $p(X|Y)$ for a numeric attribute X .

2.2. Handling Nominal Attributes

Now consider the case where $p(X|Y)$ has to be estimated for a nominal attribute X with a set of unordered values v_1, v_2, \dots, v_o . Rather than estimating $p(X|Y)$ directly, we transform the problem into one of estimating $p(Y|X)$, the pdf of a numeric variable given a nominal one, and $p(X)$, the prior probability of a nominal attribute value. This transformation is effected by applying Bayes' theorem:

$$p(X = v_k | Y = y) = \frac{p(X = v_k)p(Y = y|X = v_k)}{\sum_{k=1}^o p(X = v_k)p(Y = y|X = v_k)} \quad (7)$$

Both $p(Y|X)$ and $p(X)$ can be estimated easily. For the former, we use the one-dimensional counterpart of the kernel density estimator from above:

$$\hat{p}(Y = y | X = v_k) = \frac{1}{n_k h_k} \sum_{i=1}^{n_k} K\left(\frac{y - y_i}{h_k}\right), \quad (8)$$

where the sum is over all n_k examples with attribute value $X = v_k$. This type of estimator was also used by John and Langley (1995) to estimate the density of numeric attributes in naive Bayes for categorical prediction. For the latter, an estimate of $p(X)$ is obtained simply by computing the proportion of examples with attribute value v_k :

$$\hat{p}(X = v_k) = \frac{n_k}{\sum_{l=1}^o n_l} \quad (9)$$

Again, the cross-validation procedure can be used to choose $h_k = c_k/\sqrt{n_k}$ for each v_k so that the estimated cross-entropy is minimized. Here we use a grid search for $c_k \in [0.4, 0.8]$ with a grid width of 0.1. This gives all the terms necessary to compute an estimate $\hat{p}(X|Y)$ of $p(X|Y)$ for a nominal attribute X .

2.3. Estimating the Prior

The procedure for estimating the prior $p(Y)$ is the same as for $p(Y|X)$ (Equation 8). The only difference is that, instead of including only examples with a particular attribute value, all n examples are used. The kernel width is chosen using the previously described cross-validation procedure.

2.4. Prediction

The optimal prediction $t(e)$ for an example e with respect to the posterior probability $p(Y|E = e)$ depends on the loss function. We consider two loss functions: the squared error and the absolute error. In either case, the predicted value should minimize the expected loss. It is easy to show that the expected squared error

$$E[(t(e) - y)^2] = \int p(Y = y|E = e)(t(e) - y)^2 dy \quad (10)$$

is minimized if the expected value of y (that is, its mean) is predicted:

$$t_{SE}(e) = \int p(Y = y|E = e)y dy. \quad (11)$$

An estimate $\hat{t}_{SE}(e)$ of this quantity can be obtained from $\hat{p}(Y = y, E = e)$. Let G be a set of equally spaced grid points in the domain of y . Suppose y_{min} is the minimum and y_{max} the maximum value for y in the training data, and let $h = (y_{max} - y_{min})/(d - 1)$ for some number of grid points d . Then we use $G = \{\dots, y_{min} - 2h, y_{min} - h, y_{min}, y_{min} + h, \dots, y_{max} - h, y_{max}, y_{max} + h, y_{max} + 2h, \dots\}$. Note that one can stop evaluating points to the left of y_{min} and to the right of y_{max} as soon as $\hat{p}(Y = y, E = e)$ becomes negligible. In our experiments, d is set to 50 and attributes X_i for which $\hat{p}(X_i|Y = y)$ is negligible for all values in G are excluded from the computation of $\hat{p}(Y = y, E = e)$. Then,

$$\hat{t}_{SE}(e) = \frac{\sum_{y \in G} \hat{p}(Y = y, E = e)y}{\sum_{y \in G} \hat{p}(Y = y, E = e)}. \quad (12)$$

The sum in the denominator approximates the integral in the denominator of Equation 2.

If, on the other hand, the expected absolute error

$$E[|t(e) - y|] = \int p(Y = y|E = e)|t(e) - y| dy \quad (13)$$

is to be minimized, this is achieved by setting $t_{AE}(e)$ so that

$$\int_{-\infty}^{t_{AE}(e)} p(Y = y|E = e) dy = 0.5. \quad (14)$$

In this case, the optimum prediction $t_{AE}(e)$ is the median (Lehmann, 1983). Again, an estimate $\hat{t}_{AE}(e)$ can be obtained using $\hat{p}(Y = y, E = e)$ by finding the smallest y' for which

$$\frac{\sum_{y \in G, y < y'} \hat{p}(Y = y, E = e)}{\sum_{y \in G} \hat{p}(Y = y, E = e)} > 0.5, \quad (15)$$

and setting $t_{AE}(e) = y'$. For estimating t_{AE} we use $d = 100$ grid points.

3. Experiments with an Artificial Dataset

As previously noted, naive Bayes can be an excellent alternative to more sophisticated methods for categorical learning. This section exhibits an artificial dataset for which this is also the case in the regression setting. We compare naive Bayes to two state-of-the-art methods for numeric prediction: locally weighted linear regression, and model trees.

Locally weighted linear regression (LWR) is a combination of instance-based learning and linear regression (Atkeson, Moore & Schaal, 1997). Instead of performing a linear regression on the full, unweighted dataset, it performs a weighted linear regression, weighting the training instances according to their distance to the test instance at hand. In other words, it performs a local linear regression by giving those training instances higher weight that are close to the test instance. This means that a linear regression has to be done for each new test instance, which makes the method computationally quite expensive. However, it also makes it highly flexible, and enables it to approximate non-linear target functions.

There are many different ways of implementing the weighting scheme for locally weighted linear regression (Atkeson, Moore & Schaal, 1997). A common approach, which we also employ here, is to weight the training instances according to a triangular kernel centered at the test instance, setting the kernel width to the distance of the test instance’s k th nearest neighbour. The best value for k is determined using ten-fold cross-validation of the root mean squared error on the training data, for ten values of k ranging from one to the number of training instances.

For performing the linear regression, we use an implementation that eliminates redundant attributes by deleting, at each step, the attribute with the smallest contribution to the prediction. To decide when to stop deleting attributes, Akaike’s information criterion is employed (Akaike, 1973). Nominal attributes with n values are converted into $n - 1$ binary attributes using the algorithm described by Wang and Witten (1997). Missing values are globally replaced by the mode (for nominal attributes), or the mean (for numeric attributes)—derived from the training data before the linear regression is performed.²

The second state-of-the-art method is a model-tree predictor. Model trees are the counterpart of decision trees for regression tasks. They have the same structure as decision trees, with one difference: they employ a linear regression function at each leaf node to make a prediction. For our experiments we use the model tree inducer M5’ (Wang & Witten, 1997), a re-implementation of Quinlan’s M5 (Quinlan, 1992). An interesting fact is that M5’ in general produces more accurate predictions than a state-of-the-art decision tree learner when applied to *categorical* prediction tasks (Frank et al., 1998). In our implementation, missing values are globally replaced before a model tree is built. M5’ and LWR use the same methods for performing linear regression and treating nominal attributes.

Naive Bayes assumes that the attribute values are statistically independent given a particular target value. In terms of a regression problem this can be interpreted as each attribute being a function of the target value plus some noise:

$$x_1 = f_1(y) + \epsilon, \quad x_2 = f_2(y) + \epsilon, \quad \dots \tag{16}$$

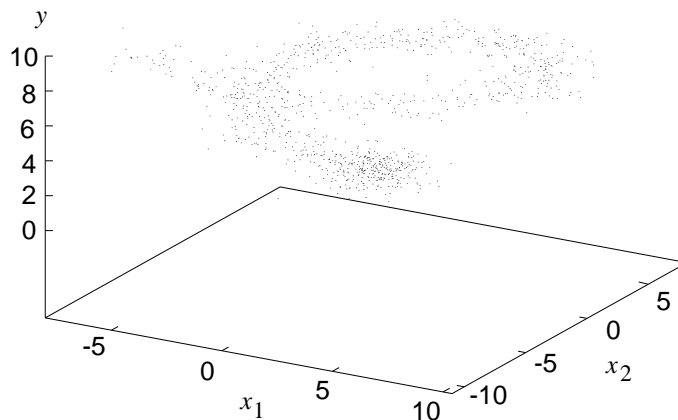


Figure 1. Spiral dataset

How does naive Bayes fare if a dataset satisfies the independence assumption? We investigated this question using artificial data. More specifically, we constructed a dataset describing a diminishing spiral in three dimensions. This spiral (using 1000 randomly generated data points) is depicted in Figure 1. The features x_1 and x_2 were generated from the target value y according to the following equations:

$$x_1 = y * \sin(y) + N(0, 1) \quad x_2 = y * \cos(y) + N(0, 1), \quad (17)$$

where $N(0, 1)$ denotes normally distributed noise with zero mean and unit variance. Note, that y is not a function of either of the two attributes x_1 and x_2 alone. However, it is—modulo noise—a function of (x_1, x_2) .

We evaluated how well naive Bayes predicts y for different training set sizes. Figure 2 shows the resulting learning curve, and, for comparison, the learning curve using LWR and unsmoothed M5' model trees. Unsmoothed model trees perform better in this domain than smoothed ones. Each point on the curve was generated by choosing 20 random training datasets and evaluating the resulting models on the 1000 independently generated examples shown in Figure 1. The error bars are 99% confidence intervals.

Figure 2 shows that naive Bayes performs significantly better than M5' on the spiral dataset, and slightly better than LWR. This demonstrates that naive Bayes can be preferable to other state-of-the-art methods for regression if the independence assumption is satisfied—a situation that might occur in practice if, for example, the attributes are readings from sensors that all measure the same target quantity, but do so in different ways. Measuring the distance of an object using ultrasonic waves on the one hand, and infrared radiation on the other hand is an example of this situation: the measurements of the two corresponding sensors are almost independent given a particular distance.

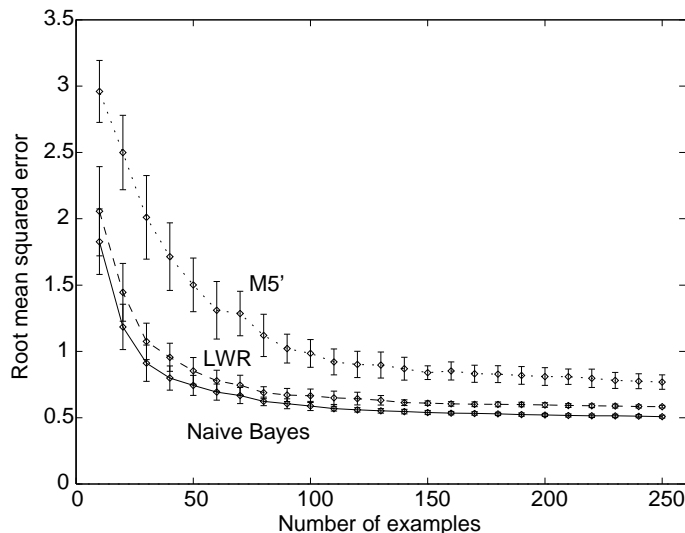


Figure 2. Learning curves for spiral dataset

4. Experiments with Practical Datasets

This section compares naive Bayes for regression to linear regression (LR), locally weighted linear regression (LWR), and M5' on practical benchmark datasets. To speed up the LWR algorithm, the kernel width is set to the distance of the furthest neighbor, which we have found to give comparable results in general. Results are presented for both loss functions discussed in Section 2: the root mean squared error and the mean absolute error. Thirty-two datasets are used. The only selection criterion was their size: we did not choose any large datasets because of the time complexity of naive Bayes for regression and locally weighted regression.

The datasets and their properties are listed in Table 1, sorted by increasing size.³ Twenty of them were used by Kilpatrick and Cameron-Jones (1998),⁴ seven are from the StatLib repository (StatLib, 1999), and the remaining five were collected by Simonoff (1996).⁵ The *pwLinear* dataset is the only artificial dataset in this set. Two of the datasets—*hungarian* and *cleveland*—are classification problems disguised as regression problems: the class is treated as an integer variable.

4.1. Results for the Relative Root Mean Squared Error

Table 2 summarizes the relative root mean squared error of all methods investigated. This measure is the root mean squared error normalized by the root mean squared error of the sample mean of Y , and expressed as a percentage. The sample mean is computed from the training data. Thus, a method that performs worse than the mean has a relative root mean squared error of more than 100 percent. Because

Table 1. Datasets used for the experiments

Dataset	Instances	Missing values (%)	Numeric attributes	Nominal attributes
schlvote ¹	38	0.4	4	1
bolts ³	40	0.0	7	0
vineyard ¹	52	0.0	3	0
elusage ¹	55	0.0	1	1
pollution ³	60	0.0	15	0
mbagrade ¹	61	0.0	1	1
sleep ³	62	2.4	7	0
auto93 ²	93	0.7	16	6
basketball ¹	96	0.0	4	0
cloud ³	108	0.0	4	2
fruitfly ²	125	0.0	2	2
echoMonths ²	130	7.5	6	3
veteran ³	137	0.0	3	4
fishcatch ²	158	6.9	5	2
autoPrice ²	159	0.0	15	0
servo ²	167	0.0	0	4
lowbwt ²	189	0.0	2	7
pharynx ²	195	0.1	1	9
pwLinear ²	200	0.0	10	0
autoHorse ²	205	1.1	17	8
cpu ²	209	0.0	6	1
bodyfat ²	252	0.0	14	0
breastTumor ²	286	0.3	1	8
hungarian ²	294	19.0	6	7
cholesterol ²	303	0.1	6	7
cleveland ²	303	0.1	6	7
autoMpg ²	398	0.2	4	3
pbc ²	418	15.6	10	8
housing ²	506	0.0	12	1
meta ²	528	4.3	19	2
sensory ³	576	0.0	0	11
strike ³	625	0.0	5	1

¹ (Simonoff, 1996)² (Kilpatrick & Cameron-Jones, 1998)³ (StatLib, 1999)

the root mean squared error was to be minimized, the results for naive Bayes were generated by predicting \hat{t}_{SE} according to Equation 12. The figures in Table 2 are averages over ten ten-fold cross-validation runs, and standard deviations of the ten are also shown. The same folds were used for each scheme. Results are marked with a “o” if they show significant improvement over the corresponding result for naive Bayes, and a “•” if they show significant degradation. Throughout, we speak of results being “significantly different” if the difference is statistically significant at the 1% level according to a paired two-sided t -test, each pair of data points consisting of the estimates obtained in one ten-fold cross-validation run for the two learning schemes being compared.

Table 2. Experimental results: relative root mean squared error, and standard deviation

Dataset	Naive Bayes	LR	LWR	M5'
schlvote	95.92± 7.2	114.23± 3.6 ●	118.81± 6.6 ●	94.00±10.2
bolts	36.27± 2.8	38.62± 1.5 ●	30.50± 2.3 ○	21.79± 3.1 ○
vineyard	66.97± 5.1	72.16± 5.6 ●	64.41± 3.8	72.11± 5.4 ●
elusage	52.78± 3.2	53.15± 3.5	55.44± 3.5 ●	50.48± 3.3
pollution	83.97± 4.6	71.51± 3.2 ○	71.09± 4.3 ○	73.98± 5.3 ○
mbagrade	92.35± 3.7	87.15± 4.0 ○	88.10± 4.2 ○	89.23± 5.0 ○
sleep	80.79± 5.4	88.96±13.7	77.32± 2.9	71.72± 4.1 ○
auto93	61.76± 5.5	59.46± 6.1	65.76± 4.3	54.65± 4.2 ○
basketball	86.53± 2.6	79.84± 1.6 ○	81.78± 1.6 ○	79.84± 1.6 ○
cloud	52.09± 2.2	38.19± 2.3 ○	41.09± 2.4 ○	38.36± 2.5 ○
fruitfly	116.42± 2.1	100.49± 0.8 ○	106.98± 1.8 ○	100.00± 0.0 ○
echoMonths	78.53± 1.5	68.25± 1.4 ○	68.04± 1.1 ○	71.01± 0.7 ○
veteran	86.32± 3.1	94.02± 3.3 ●	97.77± 4.1 ●	90.53± 2.6 ●
fishcatch	31.20± 2.2	27.25± 0.8 ○	22.45± 1.1 ○	16.23± 0.6 ○
autoPrice	41.56± 1.2	48.37± 2.3 ●	40.69± 2.2	39.82± 2.5
servo	70.40± 2.1	55.26± 1.4 ○	38.81± 1.0 ○	37.92± 3.3 ○
lowbwt	62.13± 1.0	61.49± 0.7	62.66± 1.1	62.00± 1.0
pharynx	74.25± 1.6	75.54± 1.5 ●	78.49± 2.0 ●	71.56± 1.4 ○
pwLinear	51.58± 0.7	50.51± 0.8	40.65± 0.5 ○	32.28± 0.4 ○
autoHorse	37.92± 1.9	29.55± 1.5 ○	24.81± 2.7 ○	33.32± 2.0 ○
cpu	33.56± 3.3	43.35± 3.4 ●	22.03± 2.8 ○	21.23± 2.4 ○
bodyfat	25.88± 0.7	12.38± 0.7 ○	11.88± 0.8 ○	11.15± 1.0 ○
breastTumor	100.96± 1.2	97.43± 1.2 ○	103.05± 1.2 ●	97.29± 0.6 ○
hungarian	71.28± 2.0	68.64± 0.6 ○	68.60± 0.9 ○	73.79± 1.5 ●
cholesterol	101.73± 1.2	99.78± 1.7 ○	103.89± 1.7 ●	101.62± 2.1
cleveland	74.45± 1.5	70.45± 0.7 ○	71.42± 1.1 ○	71.03± 1.0 ○
autoMpg	41.85± 0.7	37.98± 0.5 ○	33.28± 0.4 ○	35.67± 0.8 ○
pbcc	86.30± 0.9	80.32± 0.6 ○	81.38± 0.8 ○	80.83± 1.5 ○
housing	60.04± 1.6	52.68± 1.0 ○	39.94± 0.8 ○	39.84± 1.6 ○
meta	160.49±17.4	202.18±11.8 ●	160.29±10.4	150.68±32.2
sensory	92.22± 0.9	92.68± 0.5	84.87± 0.5 ○	87.84± 1.4 ○
strike	160.30±12.1	85.24± 1.9 ○	85.27± 2.1 ○	84.61± 1.9 ○

Table 3. Results of paired *t*-tests ($p=0.01$) on relative root mean squared error results: number indicates how often method in column significantly outperforms method in row

	Naive Bayes	LR	LWR	M5'
Naive Bayes	–	18	20	23
LR	8	–	13	15
LWR	6	10	–	15
M5'	3	4	6	–

Table 3 summarizes how the different methods compare with each other. Each entry indicates the number of datasets for which the method associated with its column is significantly more accurate than the method associated with its row.

Observe from Table 3 that linear regression outperforms naive Bayes on eighteen datasets (first row, second column), whereas naive Bayes outperforms linear regression on only eight (second row, first column). M5' dominates even more strongly: it outperforms naive Bayes on twenty-three datasets, and is significantly worse on only three (*hungarian*, *vineyard*, and *veteran*). As mentioned earlier, *hungarian* is actually a classification problem disguised as a regression problem. The finding for LWR is very similar: it outperforms naive Bayes on twenty datasets, and is significantly worse on only six.

These results, and the remaining figures in Table 3, indicate that M5' and LWR are the methods of choice on datasets of this type, if it is the root mean squared error that is to be minimized. Linear regression is third. However, compared to naive Bayes and LWR, linear regression and M5' have the advantage that they produce comprehensible output and are less expensive computationally.

4.2. Results for the Mean Absolute Error

We now compare the methods with respect to their mean absolute error. As discussed in Section 2, using naive Bayes to predict the median \hat{t}_{AE} according to Equation 15 should perform better than using it to predict the mean \hat{t}_{SE} , at least if the estimated pdf $\hat{p}(Y, E)$ is sufficiently close to the true pdf. In our experiments we observed that this is not the case: the mean performs significantly better than the median on fifteen datasets, and significantly worse on only seven. This indicates that the mean is more tolerant to inaccurate estimates that occur because of inter-attribute dependencies. For the results presented here, we therefore use the mean instead of the median.

Table 4 summarizes the relative mean absolute error of the four methods. This is their mean absolute error divided by the mean absolute error of the sample mean. Again, Table 5 summarizes the results of the significance tests.

Compared to the relative root mean squared error results, Table 5 shows a slightly different picture. Here, naive Bayes performs as well as linear regression: it is significantly more accurate on thirteen datasets, and significantly less accurate on thirteen. This is not as surprising as it might seem. Because the linear regression function is derived by minimizing the root mean squared error on the training data, it fits extreme values in the dataset as closely as possible. However, the mean absolute error is relatively insensitive to extreme deviations of the predicted value from the true one. This implies that naive Bayes does not fit extreme values (and outliers) very well, but does a reasonable job on the rest of the data.

The win-loss situation between naive Bayes and LWR is almost unchanged: naive Bayes scores ten significant wins, and LWR nineteen. As in the previous results, M5' outperforms naive Bayes by a wide margin: it performs significantly better on twenty-two datasets and significantly worse on only five. Two of those five are *hungarian* and *cleveland*, the two datasets that represent classification problems disguised as regression problems. These results, and the other figures in Table 5, show that M5' and LWR's superior performance is not restricted to the root mean

Table 4. Experimental results: relative mean absolute error, and standard deviation

Dataset	Naive Bayes	LR	LWR	M5'
schlvote	90.13± 7.0	112.43± 4.3 ●	114.23± 7.1 ●	89.78± 7.4
bolts	32.27± 2.8	36.29± 1.7 ●	26.02± 1.7 ○	18.84± 3.4 ○
vineyard	64.44± 4.3	70.85± 4.6 ●	63.80± 2.9	70.55± 4.2 ●
elusage	50.66± 2.8	49.45± 2.9	52.67± 2.8 ●	48.53± 3.7
pollution	84.29± 4.0	68.50± 3.7 ○	67.99± 3.8 ○	72.07± 5.4 ○
mbagrade	93.92± 3.7	86.40± 3.8 ○	85.33± 4.2 ○	88.43± 4.8 ○
sleep	85.59± 7.1	87.94± 8.9	77.93± 4.1 ○	74.07± 4.3 ○
auto93	57.81± 4.0	57.65± 6.2	63.31± 3.7 ●	51.86± 3.9 ○
basketball	85.16± 2.8	78.88± 1.7 ○	80.85± 1.8 ○	78.88± 1.7 ○
cloud	45.43± 2.0	34.86± 1.7 ○	37.07± 1.8 ○	35.06± 1.8 ○
fruitfly	116.11± 1.9	100.37± 0.8 ○	107.36± 1.6 ○	100.00± 0.0 ○
echoMonths	72.34± 1.5	65.42± 1.5 ○	64.30± 1.2 ○	67.95± 0.7 ○
veteran	76.80± 1.8	89.68± 1.9 ●	91.00± 2.7 ●	86.54± 1.4 ●
fishcatch	22.19± 1.3	23.73± 0.6 ●	20.19± 0.7 ○	13.49± 0.4 ○
autoPrice	37.43± 0.7	42.74± 1.7 ●	35.37± 1.4 ○	33.30± 1.6 ○
servo	51.61± 0.8	54.83± 0.9 ●	34.22± 0.6 ○	27.64± 1.7 ○
lowbwt	61.38± 0.8	62.47± 0.5 ●	62.29± 1.2	63.22± 1.0 ●
pharynx	68.62± 1.2	71.00± 1.8 ●	73.59± 1.7 ●	67.32± 1.1 ○
pwLinear	50.51± 0.8	50.53± 1.0	40.25± 0.7 ○	32.05± 0.5 ○
autoHorse	28.33± 1.2	24.71± 0.9 ○	17.39± 1.2 ○	29.10± 1.2
cpu	28.11± 2.1	42.67± 1.6 ●	18.62± 1.4 ○	15.74± 1.3 ○
bodyfat	21.21± 0.3	7.53± 0.2 ○	6.65± 0.2 ○	5.47± 0.4 ○
breastTumor	104.26± 0.9	99.29± 1.6 ○	106.25± 1.4 ●	99.91± 0.6 ○
hungarian	38.08± 0.6	54.47± 0.4 ●	51.80± 0.5 ●	57.53± 1.3 ●
cholesterol	99.67± 1.0	99.52± 1.6	102.72± 1.2 ●	101.50± 1.7
cleveland	58.33± 1.1	64.89± 0.5 ●	64.54± 0.9 ●	64.81± 1.0 ●
autoMpg	36.95± 0.6	34.68± 0.4 ○	28.85± 0.3 ○	31.16± 0.6 ○
pbcc	80.15± 0.8	78.72± 0.5 ○	79.88± 0.6	78.28± 1.3 ○
housing	56.18± 0.8	51.29± 0.6 ○	39.23± 0.6 ○	37.89± 1.2 ○
meta	78.44± 3.7	146.42± 4.5 ●	104.90± 3.6 ●	79.00± 8.5
sensory	93.33± 0.9	93.92± 0.6	86.10± 0.5 ○	88.78± 1.7 ○
strike	91.45± 2.3	74.08± 1.2 ○	69.88± 0.9 ○	71.22± 1.1 ○

Table 5. Results of paired t -tests ($p=0.01$) on relative mean absolute error results: number indicates how often method in column significantly outperforms method in row

	Naive Bayes	LR	LWR	M5'
Naive Bayes	–	13	19	22
LR	13	–	17	16
LWR	10	9	–	19
M5'	5	5	8	–

squared error: they outperform the other methods with respect to the mean absolute error too.

5. Comparison with Standard Naive Bayes

There remains the possibility that the disappointing performance of naive Bayes for regression on the numeric benchmark datasets is due to a fundamental flaw in our methodology for deriving the naive Bayes models. To test this hypothesis, we compared it to standard naive Bayes for classification applied to (a) a set of benchmark *classification* problems, and (b) the numeric datasets from Section 4. For (a) we need a way to apply naive Bayes for regression to classification problems, and for (b) a way to apply naive Bayes for classification to regression problems. Fortunately, there exist standard procedures for solving these two problems. We first discuss point (a) in the next section, and then proceed to investigate point (b).

5.1. Classification Problems

We used a standard technique for transforming a classification problem with n classes into n regression problems. Each of the n new datasets contains the same number of instances as the original, with the class value set to 1 or 0 depending on whether that instance has the appropriate class or not. In the next step, a naive Bayes model is trained on each of these new *regression* datasets. For a specific instance, the output of one of these models constitutes an approximation to the probability that this instance belongs to the associated class. Because the model is to minimize the squared error of the probability estimates, we let it predict the mean according to Equation 12. In the testing process, an instance of unknown class is processed by each of the naive Bayes models, the result being an approximation to the probability that it belongs to that class. The class whose naive Bayes model gives the highest value is chosen as the predicted class.

Table 6 shows error rates for twenty-three UCI datasets (Blake, Keogh & Merz, 1998) that represent classification problems.⁶ As before, these error rates were estimated using ten ten-fold cross-validation runs. As well as naive Bayes for regression, we also ran the state-of-the-art decision tree learner C4.5 Revision 8 (Quinlan, 1993) with default parameter settings and the standard naive Bayes procedure for classification on these datasets. Our implementation of naive Bayes for classification discretizes numeric attributes using Fayyad and Irani's (1993) method, ignores missing values, and employs the Laplace estimator to avoid zero counts (Domingos & Pazzani, 1997).

The results in Table 6 show that C4.5 performs significantly better than naive Bayes for regression on five datasets, and significantly worse on eleven. Compared to naive Bayes for classification it performs significantly better on seven datasets, and worse on nine. These results show that naive Bayes for regression applied to classification problems performs comparably, or even slightly better than the standard approach to naive Bayes for classification.

Table 6. Experimental results: percentage of correct classifications, and standard deviation

Dataset	Instances	Classes	C4.5	Naive Bayes for regression	Naive Bayes for classification
anneal	898	5	98.7±0.3	98.1±0.2 ●	95.6±0.3 ●
audiology	226	24	76.3±1.4	71.8±1.4 ●	70.7±1.4 ●
australian	690	2	85.5±0.7	85.2±0.3	85.9±0.5
autos	205	6	80.0±2.5	71.9±1.5 ●	64.5±2.1 ●
balance-scale	625	3	77.6±0.9	91.3±0.3 ○	71.8±0.5 ●
breast-cancer	286	2	73.2±1.7	72.5±0.6	72.6±0.5
breast-w	699	2	94.9±0.4	96.7±0.1 ○	97.1±0.1 ○
glass	163	2	78.1±1.8	78.3±1.0	80.4±1.5
heart-c	303	2	76.7±1.7	83.1±0.8 ○	83.2±0.6 ○
heart-h	294	2	79.8±0.8	83.9±0.9 ○	84.2±0.3 ○
heart-statlog	270	2	78.3±1.9	82.1±0.4 ○	82.8±0.7 ○
hepatitis	155	2	79.7±1.2	85.0±0.4 ○	83.7±0.5 ○
horse-colic	368	2	85.4±0.3	78.7±0.6 ●	79.7±0.6 ●
ionosphere	351	2	89.4±1.3	90.7±0.4	89.2±0.6
iris	150	3	94.4±0.6	96.0±0.3 ○	92.9±1.0 ●
labor	57	2	77.2±4.1	92.3±2.2 ○	89.0±1.7 ○
lymphography	148	4	75.8±2.9	80.3±0.8 ○	84.6±1.3 ○
pima-indians	768	2	74.5±1.4	75.3±0.5	75.1±0.6
primary-tumor	339	21	41.8±1.0	47.8±0.9 ○	48.7±1.3 ○
sonar	208	2	75.0±3.0	76.5±0.8	76.5±1.3
soybean	683	19	91.5±0.6	92.5±0.6 ○	92.7±0.2 ○
vote	435	2	96.3±0.6	90.2±0.2 ●	90.2±0.1 ●
zoo	101	7	91.1±1.2	91.5±1.4	92.9±1.6

5.2. Regression Problems

We now turn to a comparison of naive Bayes for regression and standard naive Bayes for classification on the set of benchmark regression problems from Section 4. In order to apply naive Bayes for classification to those datasets, they have to be transformed into classification problems. We do this using the regression-by-discretization strategy applied by Kononenko (1998). In other words, we discretize the target value into a set of intervals, and apply naive Bayes for classification to the discretized data. For prediction, the intervals' mean values are weighted according to the class probabilities output by the naive Bayes model. However, instead of Kononenko's *ad hoc* method for discretizing the target value, our implementation finds the best equal-width discretization by performing ten-fold cross-validation on the training data, and choosing the number of intervals with minimum root mean squared error.

Table 7 shows that both naive Bayes for regression and the discretization approach perform worse than M5' with respect to the relative root mean squared error. M5' performs significantly better than naive Bayes for regression on twenty-three datasets, and significantly worse on three. Compared to naive Bayes for classification M5' performs significantly better on twenty-three datasets, and worse on four.

Table 7. Experimental results: relative root mean squared error, and standard deviation

Dataset	M5'	Naive Bayes for regression	Naive Bayes for classification
schlvote	94.00±10.2	95.92±7.2	104.71±6.2
bolts	21.79±3.1	36.27±2.8 ●	38.55±4.5 ●
vineyard	72.11±5.4	66.97±5.1 ○	60.47±5.0 ○
elusage	50.48±3.3	52.78±3.2	54.89±3.5 ●
pollution	73.98±5.3	83.97±4.6 ●	92.45±4.5 ●
mbagrade	89.23±5.0	92.35±3.7 ●	95.78±1.1 ●
sleep	71.72±4.1	80.79±5.4 ●	77.20±3.2 ●
auto93	54.65±4.2	61.76±5.5 ●	69.07±5.9 ●
basketball	79.84±1.6	86.53±2.6 ●	98.32±2.6 ●
cloud	38.36±2.5	52.09±2.2 ●	64.71±6.0 ●
fruitfly	100.00±0.0	116.42±2.1 ●	102.45±0.6 ●
echoMonths	71.01±0.7	78.53±1.5 ●	67.51±1.2 ○
veteran	90.53±2.6	86.32±3.1 ○	102.25±3.5 ●
fishcatch	16.23±0.6	31.20±2.2 ●	41.35±2.6 ●
autoPrice	39.82±2.5	41.56±1.2	40.05±1.6
servo	37.92±3.3	70.40±2.1 ●	53.35±1.7 ●
lowbwt	62.00±1.0	62.13±1.0	63.23±1.1 ●
pharynx	71.56±1.4	74.25±1.6 ●	75.63±1.4 ●
pwLinear	32.28±0.4	51.58±0.7 ●	69.03±1.0 ●
autoHorse	33.32±2.0	37.92±1.9 ●	40.27±2.1 ●
cpu	21.23±2.4	33.56±3.3 ●	71.06±10.7 ●
bodyfat	11.15±1.0	25.88±0.7 ●	56.64±2.2 ●
breastTumor	97.29±0.6	100.96±1.2 ●	96.45±0.7
hungarian	73.79±1.5	71.28±2.0 ○	70.63±1.8 ○
cholesterol	101.62±2.1	101.73±1.2	100.30±1.0
cleveland	71.03±1.0	74.45±1.5 ●	70.49±0.9
autoMpg	35.67±0.8	41.85±0.7 ●	47.42±0.6 ●
pbcc	80.83±1.5	86.30±0.9 ●	85.68±0.9 ●
housing	39.84±1.6	60.04±1.6 ●	59.08±1.8 ●
meta	150.68±32.2	160.49±17.4	111.98±3.6 ○
sensory	87.84±1.4	92.22±0.9 ●	90.65±0.6 ●
strike	84.61±1.9	160.30±12.1 ●	88.20±0.8 ●

This result and the findings from the previous section suggest that there is no fundamental flaw in our methodology. They support our conjecture that it is indeed impossible to apply naive Bayes as successfully to standard regression problems as to classification tasks.

6. Isolating the Independence Assumption

In order to test if the independence assumption is the main reason for the relatively poor performance of naive Bayes for regression, we designed an experiment to isolate its influence on the results. To this end, we created a version of the model tree inducer M5', which assumes that the attributes contribute independently to the prediction. More specifically, if there are m attributes, we generate m datasets, each consisting of one of the attributes and the class, and build m model trees from

them—each of the trees basing its prediction on just one attribute. Assuming that the attributes contribute independently to the final prediction equates to taking a weighted average of these individual model trees, and using it for prediction.

The naive approach is to use a straight, unweighted average. However, the predictive performance of the combined trees can be dramatically improved by weighting their predictions according to their performance on the training data—otherwise, adding non-informative attributes has a strongly detrimental effect. In our experiments, we obtained best results by weighting according to the relative root mean squared error. More specifically, if m_i is the prediction of the model tree for attribute i , and $RRMS_i$ is the tree’s root relative mean squared error on the training data, then the final prediction m is

$$m = \frac{\sum_i (100 - RRMS_i) * m_i}{\sum_i 100 - RRMS_i}. \quad (18)$$

In the following, we will call this prediction method M5’Independent. Of course, more sophisticated weighting schemes might produce better results. However, we are only interested in the relative performance of M5’Independent when compared to M5’, and, hence, this scheme is sufficient for our purpose.

Our goal is provide further evidence for our claim that the independence assumption of naive Bayes for regression is the reason for its comparably high prediction error. To this end, we compare the difference in performance between M5’Independent and M5’ to the difference in performance between naive Bayes for regression and M5’. If the former is predictive of the latter we have evidence that the independence assumption is indeed the culprit, because the difference between M5’Independent and M5’ is that the former cannot exploit attribute dependencies when generating its predictions.

Figure 3 shows the difference in relative root mean squared error for the two combinations. As in Section 4, ten ten-fold cross-validation runs were used to estimate the error. One point corresponds to one of twenty six datasets from Table 1. We did not include those datasets from Table 1 for which any of the three learning schemes involved had greater than 100% relative root mean squared error (schlvote, fruitfly, breastTumor, cholesterol, meta, strike) because in those cases it is obvious that our particular implementation of the learning algorithm overfits the data, and we wanted to prevent implementation issues from obfuscating the effect of the independence assumption.

Figure 3 also shows the result of a linear regression on the datapoints. The correlation coefficient for the data is 0.77. Assuming that both performance differences are normally distributed, this correlation is statistically significant at the 0.0001-level, providing strong evidence that there is a linear relationship between them, and that the independence assumption is indeed responsible for the large performance difference between M5’ and naive Bayes for regression.

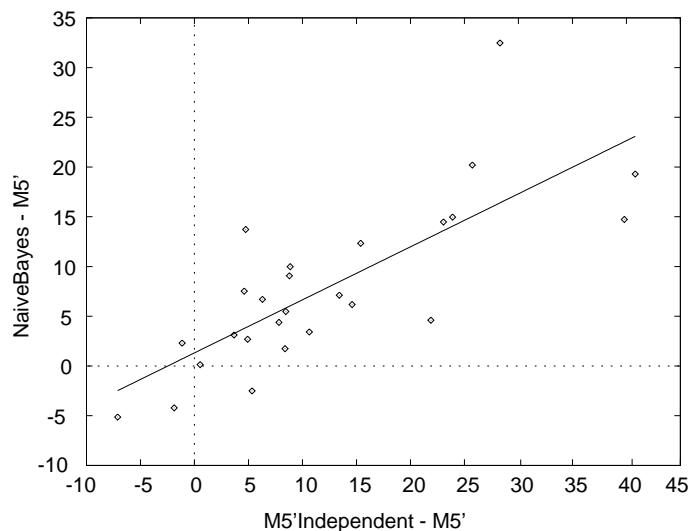


Figure 3. Comparing the difference in relative root mean squared error between M5' and M5'Independent, and M5' and naive Bayes, on the 26 practical datasets for which none of the three methods had greater than 100% relative root mean squared error.

7. Conclusions

This paper has shown how naive Bayes can be applied to regression problems by modeling the distribution of the target value using kernel density estimators. As discussed in the introduction, previous work suggests that the remarkable performance of naive Bayes for standard classification problems may not translate into the regression context. Our experimental results confirm this hypothesis. On a set of standard datasets, naive Bayes performs comparably to linear regression with respect to the absolute error of the predictions, but worse with respect to the squared error, and almost uniformly worse than locally weighted linear regression and the model tree algorithm M5'. This is the case even though our implementations of the latter three methods use a rather crude technique for dealing with missing values.

We also show that standard naive Bayes applied to regression problems by discretizing the target value performs similarly badly. In conjunction with our experimental results that isolate the independence assumption as the culprit for naive Bayes' poor performance, this finding leads us to conclude that naive Bayes should only be applied to regression problems when the independence assumption holds.

Acknowledgments

Many thanks to the anonymous reviewers whose comments greatly improved this paper. The lowbwt data set is originally from Appendix 1 of Hosmer, D.W. and Lemeshow, S. (1989), Applied Logistic Regression, John Wiley and Sons, New York.

The breast-cancer, lymphography, and primary-tumor data originates from the University Medical Center, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data. The heart-disease datasets are originally from Andras Janosi, M.D., from the Hungarian Institute of Cardiology, Budapest; William Steinbrunn, M.D., from the University Hospital, Zurich, Switzerland; Matthias Pfisterer, M.D., from the University Hospital, Basel, Switzerland; and Robert Detrano, M.D., Ph.D., from the V.A. Medical Center, Long Beach and Cleveland Clinic Foundation.

Notes

1. Note, however, that in the case of a nearly symmetric bimodal distribution, the position of the mode could be more sensitive to a small distribution change than the position of the mean. Fortunately in a classification task, if the true distribution is similarly balanced, the choice between the two is of little relevance to the true error rate.
2. This is a rather crude technique for dealing with missing values. However, it does not impact on any of the conclusions drawn in this paper. The artificial dataset from this section does not contain any missing values.
3. All the datasets can be obtained from the authors upon request.
4. Of these, the *auto93*, *fishcatch*, and *fruitfly* datasets are from the Journal of Statistics Education Data Archive ([gopher://jse.stat.ncsu.edu]), the *bodyfat* dataset from the StatLib repository (StatLib, 1999), and the *pbc*, *pharynx*, and *lowbwt* datasets from a dataset collection at the University of Massachusetts Amherst ([<http://www-unix.oit.umass.edu/~statdata>]). The remaining datasets are from the UCI repository (Blake, Keogh & Merz, 1998).
5. Simonoff's datasets can also be found at the StatLib repository.
6. Following Holte (1993), the *glass* dataset has classes 1 and 3 combined and classes 4 to 7 deleted, and the *horse-colic* dataset has attributes 3, 25, 26, 27, 28 deleted with attribute 24 being used as the class. We also deleted all identifier attributes from the datasets.

References

- Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In *Proceedings of the 2nd International Symposium on Information Theory* (pp. 267–281). Budapest: Akadémiai Kiadó.
- Atkeson, C. G., Moore, A. W. & Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, 11, 11–73.
- Blake, C., Keogh, E. & Merz, C. J. (1998). *UCI Repository of Machine Learning Databases*. Irvine, CA: University of California, Department of Information and Computer Science. [<http://www.ics.uci.edu/~mlearn/MLRepository.html>].
- Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning. In *Proceedings of the 9th European Conference on Artificial Intelligence* (pp. 147–149). Stockholm, Sweden: Pitman, London.
- Clark, P. & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3(4), 261–283.
- Domingos, P. & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2/3), 103–130.
- Duda, R. & Hart, P. (1973). *Pattern Classification and Scene Analysis*. New York: Wiley.
- Fayyad, U. M. & 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* (pp. 1022–1027). Chambery, France: Morgan Kaufmann, San Mateo, CA.
- Frank, E., Wang, Y., Inglis, S., Holmes, G. & Witten, I. H. (1998). Using model trees for classification. *Machine Learning*, 32(1), 63–76.

- Friedman, J. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1, 55–77.
- Friedman, N., Geiger, D. & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29(2/3), 131–163.
- Ghahramani, Z. & Jordan, M. I. (1994). Supervised learning from incomplete data via an EM approach. In *Advances in Neural Information Processing Systems 6* (pp. 120–127). San Mateo, CA: Morgan Kaufmann.
- Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11, 63–91.
- John, G. H. & Kohavi, R. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2), 273–324.
- John, G. H. & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence* (pp. 338–345). Montreal, Quebec: Morgan Kaufmann, San Mateo, CA.
- Kasif, S., Salzberg, S., Waltz, D., Rachlin, J. & Aha, D. W. (1998). A probabilistic framework for memory-based reasoning. *Artificial Intelligence*, 104(1-2), 297–312.
- Kilpatrick, D. & Cameron-Jones, M. (1998). Numeric prediction using instance-based learning with encoding length selection. In *Progress in Connectionist-Based Information Systems* (pp. 984–987). Dunedin, New Zealand: Springer-Verlag, Singapore.
- Kononenko, I. (1991). Semi-naive Bayesian classifier. In *Proceedings of the 6th European Working Session on Learning* (pp. 206–219). Porto, Portugal: Springer-Verlag, Berlin.
- Kononenko, I. (1998). Personal communication.
- Langley, P. (1993). Induction of recursive Bayesian classifiers. In *Proceedings of the 8th European Conference on Machine Learning* (pp. 153–164). Vienna, Austria: Springer-Verlag, Berlin.
- Langley, P., Iba, W. & Thompson, K. (1992). An analysis of Bayesian classifiers. In *Proceedings of the 10th National Conference on Artificial Intelligence* (pp. 223–228). San Jose, CA: AAAI Press, Menlo Park, CA.
- Langley, P. & Sage, S. (1994). Induction of selective Bayesian classifiers. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence* (pp. 399–406). Seattle, WA: Morgan Kaufmann, San Mateo, CA.
- Lehmann, E. L. (1983). *Theory of Point Estimation*. New York: Wiley.
- Pazzani, M. (1996). Searching for dependencies in Bayesian classifiers. In *Learning from Data: Artificial Intelligence and Statistics V* (pp. 239–248). New York: Springer-Verlag.
- Quinlan, J. R. (1992). Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence* (pp. 343–348). Hobart, Australia: World Scientific, Singapore.
- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.
- Sahami, M. (1996). Learning limited dependence Bayesian classifiers. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* (pp. 335–338). Portland, OR: AAAI Press, Menlo Park, CA.
- Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. New York: Chapman and Hall.
- Simonoff, J. S. (1996). *Smoothing Methods in Statistics*. New York: Springer-Verlag.
- Smyth, P., Gray, A. & Fayyad, U. M. (1995). Retrofitting decision tree classifiers using kernel density estimation. In *Proceedings of the 12th International Conference on Machine Learning* (pp. 506–514). Tahoe City, CA: Morgan Kaufmann, San Francisco, CA.
- StatLib (1999). Department of Statistics, Carnegie Mellon University. [<http://lib.stat.cmu.edu>].
- Wang, Y. & Witten, I. H. (1997). Induction of model trees for predicting continuous classes. In *Proceedings of the Poster Papers of the European Conference on Machine Learning* (pp. 128–137). Prague: University of Economics, Faculty of Informatics and Statistics, Prague.

Received: 20 April, 1999

Accepted: 5 August, 1999

Final Manuscript: 12 August, 1999