Naive Bayes for Regression

Eibe Frank, Leonard Trigg, Geoffrey Holmes and Ian H. Witten

October 6, 1998

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, and compares it to linear regression, instance-based learning, 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 model trees. The comparison with linear regression depends on the error measure: for one measure naive Bayes performs similarly, for another it is worse. Compared to instance-based learning, it performs similarly with respect to both measures. These results indicate that the simplistic statistical assumption that naive Bayes makes is indeed more restrictive for regression than for classification.

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 name 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 lower error rates than more sophisticated learning schemes, such as ones that learn decision trees.

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. 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 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. However, we demonstrate using an artificial dataset that situations exist where it is a useful tool.

The use of naive Bayes for classification has been investigated extensively. The algorithm itself originated in the field of pattern recognition (Duda & Hart, 1973). 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*. They remark that "... the reason for its good comparative performance is not that there are no attribute dependences in the data."

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 [NaiveBayes'] success by relaxing the independence assumption have had mixed results."

This paper is organized as follows. In the next section we describe a method for applying naive Bayes to regression problems. In Section 3, we compare the accuracy of the resulting procedure to those of three established methods for regression: linear regression, instance-based learning, and model trees. Section 4 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 $X_1, X_2, \ldots, X_m$. 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 $X_1, X_2, \ldots, X_m$ given the target value Y. Bayes' theorem states that

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

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

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

Instead of estimating the pdf $p(E|Y)$, the individual pdfs $p(X|Y)$ 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, $p(X|Y)$ 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 3 shed some light on this issue.

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

2.1 Handling Numeric Attributes

We first discuss the estimation of $p(X|Y)$ for numeric attributes X.[1] In this case, $p(X|Y)$ is a pdf involving two numeric variables. Since

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

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 $p(X,Y)$. We have chosen the kernel density estimator

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

where x_i is the attribute value, y_i the target value of training example i, $K(.)$ is a given kernel function, and h_X and h_Y are numeric "Bandwidths" for X and Y.[2] It can be shown that this estimate converges to the true pdf if the kernel function obeys certain smoothness properties and the bandwidths are chosen appropriately (Silverman, 1986). An alternative to the kernel density estimator is the normal distribution. However, the normal distribution can only model linear relationships between numeric variables, and in this case naive Bayes appears to offer no advantages over ordinary linear regression.

A common choice for $K(.)$ 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 bandwidths 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):

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

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.[3] Since

$$\int \hat{p}(X,Y)dX = \frac{1}{nh_X}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h_X}\right), \tag{6}$$

[1] All numeric attributes are normalized by their range, where the range is determined using the training data.

[2] If either x_i or y_i is missing, the example is not included in the calculation.

[3] 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.

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 $v_1, v_2, \ldots, v_o$. Estimating $p(X|Y)$ directly would involve calculating the distribution of a nominal variable conditioned on a numeric one. However, the problem can be transformed 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:

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

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:[4]

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

where the sum is over all n_k examples with attribute value $X = v_k$. 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.[5] For the latter, an estimate of $p(X)$ is obtained simply by computing the proportion of examples with attribute value v_k:

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

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 $p(Y|X)$. The only difference is that, instead of including only examples with a particular attribute value, all n examples are used. The bandwidth is chosen using the previously described cross-validation procedure.

[4]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.

[5]Here we use a grid search for $c_k \in [0.4, 0.8]$ with a grid width of 0.1.

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

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

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

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

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.[6] Then,

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

If, on the other hand, the expected absolute error

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

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

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

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

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

and setting $t_{AE}(e) = y'$.[7]

[6] 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 $G = \{\ldots, y_{min} - 2h, y_{min} - h, y_{min}, y_{min} + h, \ldots, y_{max} - h, y_{max}, y_{max} + h, y_{max} + 2h, \ldots\}$. 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 $\hat{p}(Y = y, E = e)$.

[7] For estimating t_{AE} we use $d = 100$ grid points.

3 Experimental Results

As previously noted, naive Bayes can be an excellent alternative to more sophisticated methods for categorical learning. This section determines whether this is also true for regression tasks. We compare naive Bayes empirically with three established methods for regression: linear regression, instance-based learning, and model trees. Results are presented for both loss functions discussed in the last section: the root mean squared error and the mean absolute error. Thirty-two standard datasets are used.

3.1 Experimental conditions

The first established method, multiple linear regression (LR), is a widely-applied prediction method. We use an implementation that eliminates redundant attributes using backward elimination. At each step it deletes 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).

The second established method is instance-based learning. To represent the class of instance-based learning algorithms, we use a distance-weighted k-nearest-neighbor method, which derives its prediction by weighting the target values of the k nearest training instances according to the inverse of their distance to the test instance. The best k ($k \in \{1, 2, \ldots, 20\}$) is chosen using leave-one-out cross-validation on the training data, minimizing either the estimated root mean squared error (kNN-RMSE) or the mean absolute error (kNN-MAE). The distance metric is the one employed by IB1 (Aha et al., 1991).

The third established method is the 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).

The datasets and their properties are listed in Table 1, sorted by increasing size. Twenty of them were used by Kilpatrick and Cameron-Jones (1998), seven are from the StatLib repository (StatLib, 1998), and the remaining five were collected by Simonoff (1996).[8] 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.

[8]Simonoff's datasets can also be found at the StatLib repository.

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
baskball[1]	96	0.0	4	0
cloud[3]	108	0.0	4	2
fruitfly[2]	125	0.0	2	2
echoMonths[2]	131	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	10
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, 1998)

3.2 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.[9] Thus, a method that performs worse than the mean has a relative root mean squared error of more than 100 percent. Because 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. For the same reason, the results for the k-nearest-

[9]The sample mean is computed from the test data.

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

Dataset	Naive Bayes	LR		kNN-RMSE		M5′	
schlvote	263.70±87.2	233.31±93.3		267.91±97.1		164.24±70.8	○
bolts	57.61±10.7	53.37±10.6		77.95±10.9	•	32.28±7.6	○
vineyard	82.90±9.6	77.37±9.7		75.52±8.1	○	77.21±9.9	
elusage	65.51±6.3	53.21±5.9	○	70.94±7.0	•	49.36±3.9	○
pollution	97.03±11.1	98.86±9.2		85.99±8.4	○	79.09±8.5	○
mbagrade	103.84±6.8	85.63±4.0	○	118.54±8.3	•	85.63±4.0	○
sleep	95.01±10.5	82.08±7.6	○	90.88±8.5		92.48±25.8	
auto93	66.62±6.4	67.25±5.7		71.77±4.2		61.79±4.8	
baskball	92.03±4.0	80.11±2.1	○	90.02±3.6		81.35±2.3	○
cloud	56.55±2.9	39.71±2.2	○	75.46±2.8	•	40.07±2.0	○
fruitfly	123.54±4.6	100.16±0.5	○	120.30±4.2	○	100.38±0.6	○
echoMonths	81.63±2.2	68.93±1.2	○	72.61±2.2	○	68.50±1.4	○
veteran	93.66±5.7	97.39±7.0	•	106.15±8.5	•	93.70±6.0	
fishcatch	32.14±2.2	30.76±2.3		34.82±4.2		16.61±0.7	○
autoPrice	43.63±1.6	49.28±3.4	•	46.05±2.2	•	37.95±2.5	○
servo	75.07±1.8	62.71±9.6	○	57.21±7.0	○	38.35±2.2	○
lowbwt	64.32±1.3	62.77±2.0		70.02±1.8	•	62.27±1.1	○
pharynx	87.38±2.9	79.33±2.6	○	80.23±1.6	○	72.86±2.1	○
pwLinear	53.53±1.1	51.08±1.4	○	54.80±1.6	•	33.19±0.8	○
autoHorse	39.11±2.0	54.20±5.4	•	44.81±1.4	•	31.72±2.5	○
cpu	35.91±4.4	52.17±8.4	•	38.57±5.9		18.99±2.6	○
bodyfat	26.73±0.6	12.50±0.7	○	36.72±0.9	•	10.49±0.9	○
breastTumor	103.04±1.5	97.23±1.2	○	105.98±1.2	•	97.03±1.1	○
hungarian	73.04±2.3	74.16±0.7		72.67±1.7		76.95±2.1	•
cholesterol	103.90±1.5	99.68±1.7	○	102.69±1.1	○	103.54±1.9	
cleveland	76.00±2.0	70.54±1.0	○	73.49±1.4	○	74.51±2.1	
autoMpg	42.44±0.7	38.43±0.9	○	42.62±1.1		36.37±0.6	○
pbc	87.33±1.1	80.48±0.7	○	87.73±1.2		86.22±1.4	
housing	61.00±1.6	52.78±1.1	○	45.14±1.5	○	39.20±1.9	○
meta	238.24±71.7	281.85±51.1		240.99±47.0		200.17±75.3	○
sensory	93.11±0.9	94.58±1.6		90.07±0.7	○	86.10±0.9	○
strike	162.37±12.9	84.55±1.7	○	86.59±2.3	○	84.47±1.2	○

neighbor method were obtained by optimizing the leave-one-out estimate of the root mean squared error (kNN-RMSE). 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 "○" 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 3 summarizes how the different methods compare with each other. Each entry indicates the number of datasets for which the method associated

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	kNN-RMSE	M5$'$
Naive Bayes	–	**18**	**11**	**24**
LR	**4**	–	7	17
kNN-RMSE	**11**	17	–	24
M5$'$	**1**	4	1	–

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 four (second row, first column). M5$'$ dominates even more strongly: it outperforms naive Bayes on twenty-four datasets, and is significantly worse on only one. As mentioned earlier, this dataset (*hungarian*) is actually a classification problem disguised as a regression problem. However, compared to kNN-RMSE, naive Bayes performs relatively well: both methods score eleven significant wins against each other. These results, and the remaining figures in Table 3, present strong evidence that M5$'$ is the method of choice on datasets of this type, if it is the root mean squared error that is to be minimized. Linear regression is the runner-up. Compared to naive Bayes and kNN-RMSE, linear regression and M5$'$ have the further advantage that they produce comprehensible output and are less expensive computationally.

3.3 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 eight. 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. The results for the k-nearest-neighbor method were obtained by optimizing the leave-one-out estimate of the mean absolute error (kNN-MAE). 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:

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

Dataset	Naive Bayes	LR		kNN-MAE		M5′	
schlvote	249.71±88.6	347.09±130.1	•	225.52±83.4		223.48±100.3	
bolts	53.63±9.0	67.27±16.7	•	67.80±9.4	•	36.19±10.0	∘
vineyard	80.87±8.7	89.15±10.2	•	74.09±7.4	∘	86.47±11.6	
elusage	63.57±5.6	58.61±6.6	∘	66.00±6.0	•	53.67±3.4	∘
pollution	98.86±14.0	110.73±13.0	•	85.76±13.6	∘	85.40±12.4	∘
mbagrade	101.90±7.9	93.51±8.3	∘	112.10±10.6	•	93.51±8.3	∘
sleep	96.65±12.3	92.83±11.1		95.17±14.1		104.61±24.1	
auto93	61.15±4.5	70.66±5.8	•	65.08±4.1		59.98±4.2	
baskball	89.96±3.3	83.27±2.5	∘	88.83±2.5		84.50±2.4	∘
cloud	49.60±2.5	39.18±2.4	∘	68.10±2.7	•	38.84±2.2	∘
fruitfly	122.10±4.4	105.39±3.4	∘	122.67±4.1		105.90±3.2	∘
echoMonths	75.04±2.7	72.27±2.1	∘	68.97±2.5	∘	66.54±2.3	∘
veteran	82.56±3.7	99.02±6.4	•	101.56±6.5	•	93.13±6.1	•
fishcatch	23.28±1.2	30.05±2.6	•	21.31±1.6	∘	15.31±0.6	∘
autoPrice	40.50±2.4	46.04±3.2	•	38.56±2.2	∘	34.30±2.5	∘
servo	55.77±1.6	66.42±8.2	•	53.40±4.7		30.23±1.6	∘
lowbwt	63.40±1.4	65.31±2.3		65.61±1.7	•	63.88±1.4	
pharynx	80.05±2.2	78.20±2.6		74.76±1.6	∘	71.88±2.2	∘
pwLinear	52.35±0.9	52.38±1.4		53.11±1.4		34.03±1.0	∘
autoHorse	29.37±1.4	50.71±6.2	•	29.53±1.3		26.87±1.6	∘
cpu	31.29±3.0	56.63±7.4	•	27.22±2.5	∘	17.34±1.7	∘
bodyfat	21.92±0.3	7.74±0.1	∘	34.95±1.0	•	5.51±0.1	∘
breastTumor	104.88±1.2	99.61±1.8	∘	106.57±1.6	•	100.13±1.8	∘
hungarian	39.81±1.0	93.72±3.2	•	49.22±2.9	•	57.72±3.2	•
cholesterol	101.91±1.6	101.83±2.2		102.37±1.4		105.57±2.3	•
cleveland	59.37±1.7	66.21±1.1	•	62.91±1.6	•	64.89±1.2	•
autoMpg	37.55±0.6	35.64±0.9	∘	36.82±0.8		32.07±0.7	∘
pbc	81.55±1.4	80.88±1.1	∘	86.22±1.5	•	83.66±1.3	•
housing	56.74±0.7	51.76±0.5	∘	40.76±1.0	∘	36.43±1.2	∘
meta	134.28±23.0	236.82±35.2	•	163.12±21.4	•	119.84±20.4	∘
sensory	94.02±1.0	96.25±1.5	•	91.21±0.8	∘	88.24±1.1	∘
strike	93.63±2.9	75.46±1.6	∘	68.42±1.0	∘	70.83±1.5	∘

it is significantly more accurate on fifteen datasets, and significantly less accurate on twelve. This is not as surprising as it might seem. Since 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.[10]

The win-loss situation between naive Bayes and kNN-MAE is virtually unchanged: naive Bayes scores twelve significant wins, and kNN-MAE ten. As in

[10] A comparison of Tables 3 and 5 shows that the same can be said of the k-nearest-neighbor method.

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	kNN-MAE	M5′
Naive Bayes	–	**12**	**10**	**22**
LR	**15**	–	17	25
kNN-MAE	**12**	9	–	24
M5′	**5**	3	5	–

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′'s superior performance is not restricted to the root mean squared error: it outperforms the other methods with respect to the mean absolute error too.

3.4 Results for classification problems, using the same methodology

There remains the possibility that the disappointing performance of naive Bayes for regression on these standard datasets is due to a fundamental flaw in our methodology for deriving naive Bayes models for numeric prediction problems. To test this hypothesis we applied it to a set of benchmark *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.[11] 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 that represent classification problems.[12] As before, these error rates were estimated using ten

[11] Because the model is to minimize the squared error of the probability estimates, we let it predict the mean according to Equation 12.

[12] 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.

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

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 (1993) and the standard naive Bayes procedure for classification on these datasets. Our implementation of naive Bayes for classficiation 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 suggest that there is no fundamental flaw in our methodology. They support our claim that it is indeed impossible to apply naive Bayes as successfully to standard regression problems as to classification tasks.

3.5 Experiments with an Artificial Dataset

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

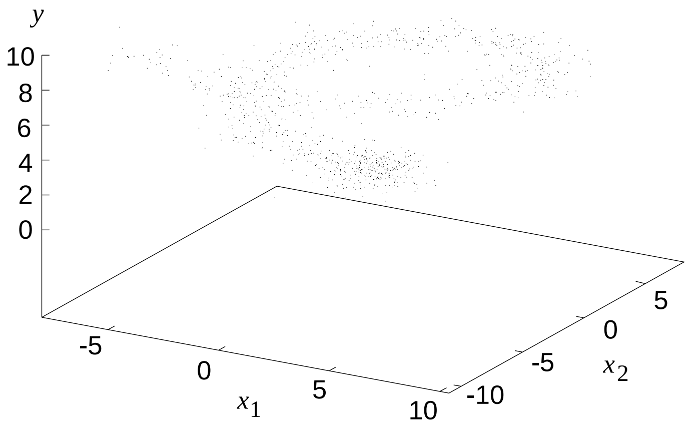

Figure 1: Spiral dataset

as each attribute being a function of the target value plus some noise:

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

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:

$$x_1 = y * sin(y) + N(0,1) \qquad x_2 = y * cos(y) + N(0,1), \tag{17}$$

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 unsmoothed M5′ model trees.[13] 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. This demonstrates that naive Bayes can be preferable to M5′ 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

[13] Unsmoothed model trees perform better in this domain than smoothed ones.

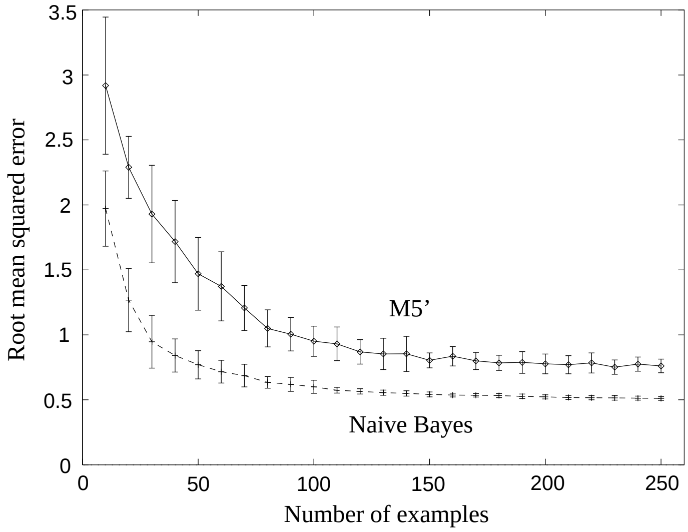

Figure 2: Learning curves for spiral dataset

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.

4 Conclusion

This paper has shown how naive Bayes can be applied to regression. As discussed in the introduction, previous work suggests that its remarkable performance for standard classification problems may not translate into the numeric 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, and to the k-nearest-neighbor method with respect to both the absolute and squared error. However, it performs worse than linear regression with respect to the squared error, and almost uniformly worse than the model tree algorithm M5$'$. Of the four methods for regression that we included in our experimental comparison, the model tree inducer M5$'$ emerged as the clear winner. It outperforms the other methods with respect to both error measures by a large margin on the standard datasets we used. However, results on artificial data show that naive Bayes can perform significantly better than M5$'$ if the independence assumption is satisfied. As we argued, there are practical applications where this is the case.

References

Aha, D. W., Kibler, D. & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning, 6(1)*, 37–66.

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In *2nd International Symposium on Information Theory* (pp. 267–281). Budapest: Akadémiai Kiadó.

Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning. In *Proceedings of the 9th European Conference on Artificial Intelligence* (pp. 147–149). London: Pitman.

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 classfiication learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence* (pp. 1022–1027). San Francisco: Morgan Kaufmann.

Frank, E., Wang, Y., Inglis, S., Holmes, G. & Witten, I. H. (1998). Using model trees for classification. *Machine Learning, In Press.*

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.

John, G. H. & Kohavi, R. (1997). Wrappers for feature subset selection. *Artificial Intelligence.*

John, G. H. & Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (pp. 338–345). San Mateo: Morgan Kaufmann.

Kilpatrick, D. & Cameron-Jones, M. (1998). Numeric prediction using instance-based learning with encoding length selection. In *Progress in Connectionist-Based Information Systems.* Singapore: Springer-Verlag.

Kononenko, I. (1991). Semi-naive bayesian classifier. In *Proceedings of the 6th European Working Session on Learning* (pp. 206–219). Berlin: Springer-Verlag.

Langley, P. (1993). Induction of recursive bayesian classifiers. In *Proceedings of the 8th European Conference on Machine Learning* (pp. 153–164). Vienna: Springer-Verlag.

Langley, P., Iba, W. & Thompson, K. (1992). An analysis of bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 223–228). San Jose: AAAI Press.

Langley, P. & Sage, S. (1994). Induction of selective bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence* (pp. 399–406). Seattle: Morgan Kaufmann.

Lehmann, E. (1983). *Theory of Point Estimation*. New York: Wiley.

Pazzani, M. (1996). *Learning from Data: Artificial Intelligence and Statistics V*, chapter Searching for Dependencies in Bayesian Classifiers, (pp. 239–248). New York: Springer Verlag.

Quinlan, J. (1992). Learning with continuous classes. In *Proceedings of the Australian Joint Conference on Artificial Intelligence* (pp. 343–348). Singapore: World Scientific.

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: AAAI Press.

Silverman, B. (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 Twelfth International Conference on Machine Learning* (pp. 506–514). San Francisco: Morgan Kaufmann.

StatLib (1998). Department of Statistics, Carnegie Mellon University.

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*. Prague: University of Economics, Faculty of Informatics and Statistics.