Journal of Machine Learning Research 14 (2013) 1947-1988 Submitted 6/12; Revised 1/13; Published 7/13

Alleviating Naive Bayes Attribute Independence Assumption by Attribute Weighting

Nayyar A. Zaidi NAYYAR.ZAIDI@MONASH.EDU
Faculty of Information Technology
Monash University
VIC 3800, Australia

Jesús Cerquides CERQUIDE@IIIA.CSIC.ES
IIIA-CSIC, Artificial Intelligence Research Institute
Spanish National Research Council
Campus UAB
08193 Bellaterra, Spain

Mark J. Carman MARK.CARMAN@MONASH.EDU
Geoffrey I. Webb GEOFF.WEBB@MONASH.EDU
Faculty of Information Technology
Monash University
VIC 3800, Australia

Editor: Russ Greiner

Abstract

Despite the simplicity of the Naive Bayes classifier, it has continued to perform well against more sophisticated newcomers and has remained, therefore, of great interest to the machine learning community. Of numerous approaches to refining the naive Bayes classifier, attribute weighting has received less attention than it warrants. Most approaches, perhaps influenced by attribute weighting in other machine learning algorithms, use weighting to place more emphasis on highly predictive attributes than those that are less predictive. In this paper, we argue that for naive Bayes attribute weighting should instead be used to alleviate the conditional independence assumption. Based on this premise, we propose a weighted naive Bayes algorithm, called WANBIA, that selects weights to minimize either the negative conditional log likelihood or the mean squared error objective functions. We perform extensive evaluations and find that WANBIA is a competitive alternative to state of the art classifiers like Random Forest, Logistic Regression and A1DE.

Keywords: classification, naive Bayes, attribute independence assumption, weighted naive Bayes classification

1. Introduction

Naive Bayes (also known as *simple Bayes* and *Idiot's Bayes*) is an extremely simple and remarkably effective approach to classification learning (Lewis, 1998; Hand and Yu, 2001). It infers the probability of a class label given data using a simplifying assumption that the attributes are independent given the label (Kononenko, 1990; Langley et al., 1992). This assumption is motivated by the need to estimate high-dimensional multi-variate probabilities from the training data. If there is sufficient data present for every possible combination of attribute values, direct estimation of each relevant multi-variate probability will be reliable. In practice, however, this is not the case and most com-

binations are either not represented in the training data or not present in sufficient numbers. Naive Bayes circumvents this predicament by its conditional independence assumption. Surprisingly, it has been shown that the prediction accuracy of naive Bayes compares very well with other more complex classifiers such as decision trees, instance-based learning and rule learning, especially when the data quantity is small (Hand and Yu, 2001; Cestnik et al., 1987; Domingos and Pazzani, 1996; Langley et al., 1992).

In practice, naive Bayes' attribute independence assumption is often violated, and as a result its probability estimates are often suboptimal. A large literature addresses approaches to reducing the inaccuracies that result from the conditional independence assumption. Such approaches can be placed into two categories. The first category comprises *semi-naive Bayes methods*. These methods are aimed at enhancing naive Bayes' accuracy by relaxing the assumption of conditional independence between attributes given the class label (Langley and Sage, 1994; Friedman and Goldszmidt, 1996; Zheng et al., 1999; Cerquides and De Mántaras, 2005a; Webb et al., 2005, 2011; Zheng et al., 2012). The second category comprises *attribute weighting methods* and has received relatively little attention (Hilden and Bjerregaard, 1976; Ferreira et al., 2001; Hall, 2007). There is some evidence that attribute weighting appears to have primarily been viewed as a means of increasing the influence of highly predictive attributes and discounting attributes that have little predictive value. This is not so much evident from the explicit motivation stated in the prior work, but rather from the manner in which weights have been assigned. For example, weighting by mutual information between an attribute and the class is directly using a measure of how predictive is each individual attribute (Zhang and Sheng, 2004). In contrast, we argue that the primary value of attribute weighting is its capacity to reduce the impact on prediction accuracy of violations of the assumption of conditional attribute independence.

Contributions of this paper are two-fold:

- This paper reviews the state of the art in weighted naive Bayesian classification. We provide a compact survey of existing techniques and compare them using the bias-variance decomposition method of Kohavi and Wolpert (1996). We also use Friedman test and Nemenyi statistics to analyze error, bias, variance and root mean square error.

- We present novel algorithms for learning attribute weights for naive Bayes. It should be noted that the motivation of our work differs from most previous *attribute weighting methods*. We view weighting as a way to reduce the effects of the violations of the attribute independence assumption on which naive Bayes is based. Also, our work differs from *semi-naive Bayes methods*, as we weight the attributes rather than modifying the structure of naive Bayes.

We propose a weighted naive Bayes algorithm, Weighting attributes to Alleviate Naive Bayes' Independence Assumption (WANBIA), that introduces weights in naive Bayes and learns these weights in a discriminative fashion that is minimizing either the negative conditional log likelihood or the mean squared error objective functions. Naive Bayes probabilities are set to be their maximum a posteriori (MAP) estimates.

The paper is organized as follows: we provide a formal description of the weighted naive Bayes model in Section 2. Section 3 provides a survey of related approaches. Our novel techniques for learning naive Bayes weights are described in Section 4 where we also discuss their connection with naive Bayes and Logistic Regression in terms of parameter optimization. Section 5 presents experimental evaluation of our proposed methods and their comparison with related approaches. Section 6 presents conclusions and directions for future research.

Notation	Description
$\mathrm{P}(e)$	the unconditioned probability of event e
$\mathrm{P}(e \mid g)$	conditional probability of event e given g
$\hat{\mathrm{P}}(\bullet)$	an estimate of $\mathrm{P}(\bullet)$
a	the number of attributes
n	the number of data points in $\mathcal{D}$
$\mathbf{x} = \langle x_1, \ldots, x_a \rangle$	an object (a-dimensional vector) and $\mathbf{x} \in \mathcal{D}$
$y \in \mathcal{Y}$	the class label for object $\mathbf{x}$
$\lvert\mathcal{Y}\rvert$	the number of classes
$\mathcal{D} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$	data consisting of n objects
$\mathcal{L} = \{y^{(1)}, \ldots, y^{(n)}\}$	labels of data points in $\mathcal{D}$
$\mathcal{X}_i$	discrete set of values for attribute i
$\lvert\mathcal{X}_i\rvert$	the cardinality of attribute i
$v = \frac{1}{a}\sum_i \lvert\mathcal{X}_i\rvert$	the average cardinality of the attributes

Table 1: List of symbols used

2. Weighted Naive Bayes

We wish to estimate from a training sample $\mathcal{D}$ consisting of n objects, the probability $\mathrm{P}(y \mid \mathbf{x})$ that an example $\mathbf{x} \in \mathcal{D}$ belongs to a class with label $y \in \mathcal{Y}$. All the symbols used in this work are listed in Table 1. From the definition of conditional probability we have

$$\mathrm{P}(y \mid \mathbf{x}) = \mathrm{P}(y, \mathbf{x}) / \mathrm{P}(\mathbf{x}). \tag{1}$$

As $\mathrm{P}(\mathbf{x}) = \sum_{i=1}^{|\mathcal{Y}|} \mathrm{P}(y_i, \mathbf{x})$, we can always estimate $\mathrm{P}(y|\mathbf{x})$ in Equation 1 from the estimates of $\mathrm{P}(y, \mathbf{x})$ for each class as:

$$\mathrm{P}(y, \mathbf{x}) / \mathrm{P}(\mathbf{x}) = \frac{\mathrm{P}(y, \mathbf{x})}{\sum_{i=1}^{|\mathcal{Y}|} \mathrm{P}(y_i, \mathbf{x})}. \tag{2}$$

In consequence, in the remainder of this paper we consider only the problem of estimating $\mathrm{P}(y, \mathbf{x})$.

Naive Bayes estimates $\mathrm{P}(y, \mathbf{x})$ by assuming the attributes are independent given the class, resulting in the following formula:

$$\hat{\mathrm{P}}(y, \mathbf{x}) = \hat{\mathrm{P}}(y) \prod_{i=1}^{a} \hat{\mathrm{P}}(x_i \mid y). \tag{3}$$

Weighted naive Bayes extends the above by adding a weight to each attribute. In the most general case, this weight depends on the attribute value:

$$\hat{\mathrm{P}}(y, \mathbf{x}) = \hat{\mathrm{P}}(y) \prod_{i=1}^{a} \hat{\mathrm{P}}(x_i|y)^{w_{i,x_i}}. \tag{4}$$

Doing this results in $\sum_i^a |\mathcal{X}_i|$ weight parameters (and is in some cases equivalent to a "binarized logistic regression model" see Section 4 for a discussion). A second possibility is to give a single weight per attribute:

$$\hat{\mathrm{P}}(y, \mathbf{x}) = \hat{\mathrm{P}}(y) \prod_{i=1}^{a} \hat{\mathrm{P}}(x_i \mid y)^{w_i}. \tag{5}$$

One final possibility is to set all weights to a single value:

$$\hat{\mathrm{P}}(y,\mathbf{x}) = \hat{\mathrm{P}}(y)\left(\prod_{i=1}^{a}\hat{\mathrm{P}}(x_i\,|\,y)\right)^{w}. \tag{6}$$

Equation 5 is a special case of Equation 4, where $\forall_{i,j}\ w_{ij} = w_i$, and Equation 6 is a special case of Equation 5 where $\forall_i\ w_i = w$. Unless explicitly stated, in this paper we intend the intermediate form when we refer to attribute weighting, as we believe it provides an effective trade-off between computational complexity and inductive power.

Appropriate weights can reduce the error that results from violations of naive Bayes' conditional attribute independence assumption. Trivially, if data include a set of a attributes that are identical to one another, the error due to the violation of the conditional independence assumption can be removed by assigning weights that sum to 1.0 to the set of attributes in the set. For example, the weight for one of the attributes, x_i could be set to 1.0, and that of the remaining attributes that are identical to x_i set to 0.0. This is equivalent to deleting the remaining attributes. Note that, any assignment of weights such that their sum is 1.0 for the a attributes will have the same effect, for example, we could set the weights of all a attributes to $1/a$.

Attribute weighting is strictly more powerful than attribute selection, as it is possible to obtain identical results to attribute selection by setting the weights of selected attributes to 1.0 and of discarded attributes to 0.0, and assignment of other weights can create classifiers that cannot be expressed using attribute selection.

2.1 Dealing with Dependent Attributes by Weighting: A Simple Example

This example shows the relative performance of naive Bayes and weighted naive Bayes as we vary the conditional dependence between attributes. In particular it demonstrates how optimal assignment of weights will never result in higher error than attribute selection or standard naive Bayes, and that for certain violations of the attribute independence assumption it can result in lower error than either.

We will constrain ourselves to a binary class problem with two binary attributes. We quantify the conditional dependence between the attributes using the Conditional Mutual Information (CMI):

$$I(X_1,X_2|Y) \quad = \quad \sum_{y}\sum_{x_2}\sum_{x_1}\mathrm{P}(x_1,x_2,y)\log\frac{\mathrm{P}(x_1,x_2|y)}{\mathrm{P}(x_1|y)\mathrm{P}(x_2|y)}.$$

The results of varying the conditional dependence between the attributes on the performance of the different classifiers in terms of their Root Mean Squared Error (RMSE) is shown in Figure 1.

To generate these curves, we varied the probabilities $\mathrm{P}(y|x_1,x_2)$ and $\mathrm{P}(x_1,x_2)$ and plotted average results across distinct values of the Conditional Mutual Information. For each of the 4 possible attribute value combinations $(x_1,x_2) \in \{(0,0),(0,1),(1,0),(1,1)\}$, we selected values for the class probability given the attribute value combination from the set: $\mathrm{P}(y|x_1,x_2) \in \{0.25,0.75\}$. Note that $\mathrm{P}(\neg y|x_1,x_2) = 1-P(y|x_1,x_2)$, so this process resulted in 2^4 possible assignments to the vector $\mathrm{P}(y|\bullet,\bullet)$.

We then set the values for the attribute value probabilities $\mathrm{P}(x_1,x_2)$ by fixing the marginal distributions to a half $\mathrm{P}(x_1) = \mathrm{P}(x_2) = 1/2$, and varying the correlation between the attributes using

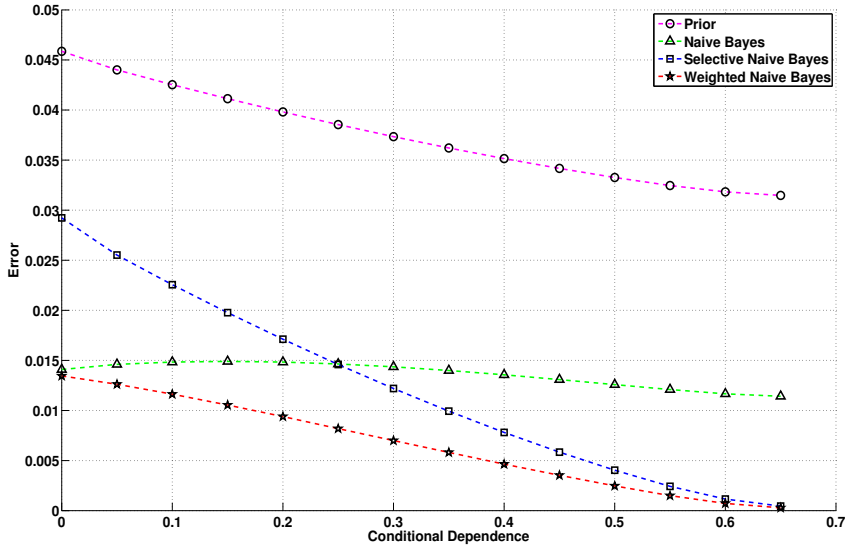


Figure 1: Variation of Error of naive Bayes, selective naive Bayes, weighted naive Bayes and classifier based only on prior probabilities of the class as a function of conditional dependence (conditional mutual information) between the two attributes.

Pearson's correlation coefficient, denoted ρ, as follows:[1]

$$\begin{aligned} P(X_1=0,X_2=0) &= P(X_1=1,X_2=1) = \frac{(1+\rho)}{4}, \\ P(X_1=0,X_2=1) &= P(X_1=1,X_2=0) = \frac{(1-\rho)}{4}, \\ & \text{where } -1 \leq \rho \leq 1. \end{aligned}$$

Note that when $\rho=-1$ the attributes are perfectly anti-correlated ($x_1=\neg x_2$), when $\rho=0$ the attributes are independent (since the joint distribution $P(x_1,x_2)$ is uniform) and when $\rho=1$ the attributes are perfectly correlated.

For the graph, we increased values of ρ in increments of 0.00004, resulting in 50000 distributions (vectors) for $P(\bullet,\bullet)$ for each vector $P(y|\bullet,\bullet)$. Near optimal weights (w_1,w_2) for the weighted naive Bayes classifier were found using grid search over the range $\{\{0.0,0.1,0.2,\ldots,0.9,1.0\} \times \{0.0,0.1,0.2,\ldots,0.9,1.0\}\}$. Results in Figure 1 are plotted by taking average across conditional mutual information values, with a window size of 0.1.

1. Note that from the definition of Pearson's correlation coefficient we have:

$$\rho = \frac{E[(X_1-E[X_1])(X_2-E[X_2])]}{\sqrt{E[(X_1-E[X_1])^2]E[(X_2-E[X_2])^2]}} = 4P(X_1=1,X_2=1)-1,$$

since $E[X_1]=E[X_2]=P(1)=1/2$ and $E[X_1X_2]=P(X_1=1,X_2=1)$.

We compare the expected RMSE of naive Bayes ($w_1 = 1, w_2 = 1$), weighted naive Bayes, naive Bayes based on feature 1 only (selective Bayes with $w_1 = 1, w_2 = 0$), naive Bayes based on feature 2 only (selective Bayes with $w_1 = 0, w_2 = 1$), and naive Bayes using only the prior (equivalent to weighted naive Bayes with both weights set to 0.0). It can be seen that when conditional mutual information (CMI) is small, naive Bayes performs better than selective naive Bayes and the prior classifier. Indeed, when CMI is 0.0, naive Bayes is optimal. As CMI is increased, naive Bayes performance deteriorates compared to selective naive Bayes. Weighted naive Bayes, on the other hand, has the best performance in all circumstances. Due to the symmetry of the problem, the two selective Bayes classifiers give exactly the same results.

Note that in this experiment we have used the optimal weights to calculate the results. We have shown that weighted naive Bayes is capable of *expressing* more accurate classifiers than selective naive Bayes. In the remaining sections we will examine and evaluate techniques for learning from data the weights those models require.

3. Survey of Attribute Weighting and Selecting Methods for Naive Bayes

Attribute weighting is well-understood in the context of nearest-neighbor learning methods and is used for reducing bias in high-dimensional problems due to the presence of redundant or irrelevant features (Friedman, 1994; Guyon et al., 2004). It is also used for mitigating the effects of the curse-of-dimensionality which results in exponential increase in the required training data as the number of features are increased (Bellman, 1957). Attribute weighting for naive Bayes is comparatively less explored.

Before discussing these techniques, however, it is useful to briefly examine the closely related area of feature selection for naive Bayes. As already pointed out, weighting can achieve feature selection by settings weights to either 0.0 or 1.0, and so can be viewed as a generalization of feature selection.

Langley and Sage (1994) proposed the Selective Bayes (SB) classifier, using feature selection to accommodate redundant attributes in the prediction process and to augment naive Bayes with the ability to exclude attributes that introduce dependencies. The technique is based on searching through the entire space of all attribute subsets. For that, they use a forward sequential search with a greedy approach to traverse the search space. That is, the algorithm initializes the subset of attributes to an empty set, and the accuracy of the resulting classifier, which simply predicts the most frequent class, is saved for subsequent comparison. On each iteration, the method considers adding each unused attribute to the subset on a trial basis and measures the performance of the resulting classifier on the training data. The attribute that most improves the accuracy is permanently added to the subset. The algorithm terminates when addition of any attribute results in reduced accuracy, at which point it returns the list of current attributes along with their ranks. The rank of the attribute is based on the order in which they are added to the subset.

Similar to Langley and Sage (1994), Correlation-based Feature Selection (CFS) used a correlation measure as a metric to determine the relevance of the attribute subset (Hall, 2000). It uses a best-first search to traverse through feature subset space. Like SB, it starts with an empty set and generates all possible single feature expansions. The subset with highest evaluation is selected and expanded in the same manner by adding single features. If expanding a subset results in no improvement, the search drops back to the next best unexpanded subset and continues from there.

The best subset found is returned when the search terminates. CFS uses a stopping criterion of five consecutive fully expanded non-improving subsets.

There has been a growing trend in the use of decision trees to improve the performance of other learning algorithms and naive Bayes classifiers are no exception. For example, one can build a naive Bayes classifier by using only those attributes appearing in a C4.5 decision tree. This is equivalent to giving zero weights to attributes not appearing in the decision tree. The Selective Bayesian Classifier (SBC) of Ratanamahatana and Gunopulos (2003) also employs decision trees for attribute selection for naive Bayes. Only those attributes appearing in the top three levels of a decision tree are selected for inclusion in naive Bayes. Since decision trees are inherently unstable, five decision trees (C4.5) are generated on samples generated by bootstrapping 10% from the training data. Naive Bayes is trained on an attribute set which comprises the union of attributes appearing in all five decision trees.

One of the earliest works on weighted naive Bayes is by Hilden and Bjerregaard (1976), who used weighting of the form of Equation 6. This strategy uses a single weight and therefore is not strictly performing attribute weighting. Their approach is motivated as a means of alleviating the effects of violations of the attribute independence assumption. Setting w to unity is appropriate when the conditional independence assumption is satisfied. However, on their data set (acute abdominal pain study in Copenhagen by Bjerregaard et al. 1976), improved classification was obtained when w was small, with an optimum value as low as 0.3. The authors point out that if symptom variables of a clinical field trial are not independent, but pair-wise correlated with independence between pairs, then $w = 0.5$ will be the correct choice since using $w = 1$ would make all probabilities the square of what they ought be. Looking at the optimal value of $w = 0.3$ for their data set, they suggested that out of ten symptoms, only three are providing independent information. The value of w was obtained by maximizing the log-likelihood over the entire testing sample.

Zhang and Sheng (2004) used the gain ratio of an attribute with the class labels as its weight. Their formula is shown in Equation 7. The gain ratio is a well-studied attribute weighting technique and is generally used for splitting nodes in decision trees (Duda et al., 2006). The weight of each attribute is set to the gain ratio of the attribute relative to the average gain ratio across all attributes. Note that, as a result of the definition at least one (possibly many) of the attributes have weights greater than 1, which means that they are not only attempting to lessen the effects of the independence assumption—otherwise they would restrict the weights to be no more than one.

$$w_i = \frac{\mathrm{GR}(i)}{\frac{1}{a}\sum_{i=1}^{a}\mathrm{GR}(i)}. \tag{7}$$

The gain ratio of an attribute is then simply the Mutual Information between that attribute and the class label divided by the entropy of that attribute:

$$\mathrm{GR}(i) = \frac{I(X_i, Y)}{H(X_i)} = \frac{\sum_y \sum_{x_1} \mathrm{P}(x_1, y) \log \frac{\mathrm{P}(x_1, y)}{\mathrm{P}(x_1)\mathrm{P}(y)}}{\sum_{x_1} \mathrm{P}(x_1) \log \frac{1}{\mathrm{P}(x_1)}}.$$

Several other wrapper-based methods are also proposed in Zhang and Sheng (2004). For example, they use a simple hill climbing search to optimize weight $\mathbf{w}$ using Area Under Curve (AUC) as an evaluation metric. Another Markov-Chain-Monte-Carlo (MCMC) method is also proposed.

An attribute weighting scheme based on differential evolution algorithms for naive Bayes classification have been proposed in Wu and Cai (2011). First, a population of attribute weight vectors

is randomly generated, weights in the population are constrained to be between 0 and 1. Second, typical genetic algorithmic steps of mutation and cross-over are performed over the the population. They defined a fitness function which is used to determine if mutation can replace the current individual (weight vector) with a new one. Their algorithm employs a greedy search strategy, where mutated individuals are selected as offspring only if the fitness is better than that of target individual. Otherwise, the target is maintained in the next iteration.

A scheme used in Hall (2007) is similar in spirit to SBC where the weight assigned to each attribute is inversely proportional to the minimum depth at which they were first tested in an unpruned decision tree. Weights are stabilized by averaging across 10 decision trees learned on data samples generated by bootstrapping 50% from the training data. Attributes not appearing in the decision trees are assigned a weight of zero. For example, one can assign weight to an attribute i as:

$$w_i = \frac{1}{T}\sum_{t}^{T}\frac{1}{\sqrt{d_{ti}}}. \tag{8}$$

where d_{ti} is the minimum depth at which the attribute i appears in decision tree t, and T is the total number of decision trees generated. To understand whether the improvement in naive Bayes accuracy was due to attribute weighting or selection, a variant of the above approach was also proposed where all non-zero weights are set to one. This is equivalent to SBC except using a bootstrap size of 50% with 10 iterations.

Both SB and CFS are feature selection methods. Since selecting an optimal number of features is not trivial, Hall (2007) proposed to use SB and CFS for feature weighting in naive Bayes. For example, the weight of an attribute i can be defined as:

$$w_i = \frac{1}{\sqrt{r_i}}. \tag{9}$$

where r_i is the rank of the feature based on SB and CFS feature selection.

The feature weighting method proposed in Ferreira et al. (2001) is the only one to use Equation 4, weighting each attribute value rather than each attribute. They used entropy-based discretization for numeric attributes and assigned a weight to each partition (value) of the attribute that is proportional to its predictive capability of the class. Different weight functions are proposed to assign weights to the values. These functions measure the difference between the distribution over classes for the particular attribute-value pair and a "baseline class distribution". The choice of weight function reduces to a choice of baseline distribution and the choice of measure quantifying the difference between the distributions. They used two simple baseline distribution schemes. The first assumes equiprobable classes, that is, uniform class priors. In that case the weight of for value j of the attribute i can be written as:

$$w_{ij} \quad \propto \quad \left(\sum_{y} |\,\mathrm{P}(y|X_i=j) - \frac{1}{|\mathcal{Y}|}\,|^{\alpha}\right)^{1/\alpha}.$$

where $\mathrm{P}(y|X_i=j)$ denotes the probability that the class is y given that the i-th attribute of a data point has value j. Alternatively, the baseline class distribution can be set to the class probabilities across all values of the attribute (i.e., the class priors). The weighing function will take the form:

$$w_{ij} \quad \propto \quad \left(\sum_{y} |\,\mathrm{P}(y|X_i=j) - \mathrm{P}(y|X_i\neq \texttt{miss})\,|^{\alpha}\right)^{1/\alpha}.$$

where $P(y|X_i \neq \texttt{miss})$ is the class prior probability across all data points for which the attribute i is not missing. Equation 10 and 10 assume an L_α distance metric where $\alpha = 2$ corresponds to the L_2 norm. Similarly, they have also proposed to use distance based on Kullback-Leibler divergence between the two distributions to set weights.

Many researchers have investigated techniques for extending the basic naive Bayes independence model with a small number of additional dependencies between attributes in order to improve classification performance (Zheng and Webb, 2000). Popular examples of such *semi-naive Bayes methods* include Tree-Augmented Naive Bayes (TAN) (Friedman et al., 1997) and ensemble methods such as Averaged n-Dependence Estimators (AnDE) (Webb et al., 2011). While detailed discussion of these methods is beyond the scope of this work, we will describe both TAN and AnDE in Section 5.10 for the purposes of empirical comparison.

Semi-naive Bayes methods usually limit the structure of the dependency network to simple structures such as trees, but more general graph structures can also be learnt. Considerable research has been done in the area of learning general Bayesian Networks (Greiner et al., 2004; Grossman and Domingos, 2004; Roos et al., 2005), with techniques differing on whether the network structure is chosen to optimize a generative or discriminative objective function, and whether the same objective is also used for optimizing the parameters of the model. Indeed optimizing network structure using a discriminative objective function can quickly become computationally challenging and thus recent work in this area has looked at efficient heuristics for discriminative structure learning (Pernkopf and Bilmes, 2010) and at developing decomposable discriminative objective functions (Carvalho et al., 2011).

In this paper we are interested in improving performance of the NB classifier by reducing the effect of attribute independence violations through attribute weighting. We do not attempt to identify the particular dependencies between attributes that cause the violations and thus are not attempting to address the much harder problem of inducing the dependency network structure. While it is conceivable that semi-naive Bayes methods and more general Bayesian Network classifier learning could also benefit from attribute weighting, we leave its investigation to future work.

A summary of different methods compared in this research is given in Table 2.

4. Weighting to Alleviate the Naive Bayes Independence Assumption

In this section, we will discuss our proposed methods to incorporate weights in naive Bayes.

4.1 WANBIA

Many previous approaches to attribute weighting for naive Bayes have found weights using some form of mechanism that increases the weights of attributes that are highly predictive of the class and decreases the weights of attributes that are less predictive of the class. We argue that this is not appropriate. Naive Bayes delivers Bayes optimal classification if the attribute independence assumption holds. Weighting should only be applied to remedy violations of the attribute independence assumption. For example, consider the case where there are three attributes, x_1, x_2 and x_3, such that x_1 and x_2 are conditionally independent of one another given the class and x_3 is an exact copy of x_1 (and hence violates the independence assumption). Irrespective of any measure of how well these three attributes each predict the class, Bayes optimal classification will be obtained by setting the weights of x_1 and x_3 to sum to 1.0 and setting the weight of x_2 to 1.0. In contrast, a method that uses a measure such as mutual information with the class to weight the attribute will

Name	Description
	Naive Bayes.
NB	Naive Bayes Classifier.
	Weighted Naive Bayes (using Typical Feature Weighting Methods).
GRW	Use gain ratio as attribute weights in naive Bayes, shown in Equation 7 (Zhang and Sheng, 2004).
SBC	Assign weight to attribute i as given in Equation 8 where $L = 5$ with a bootstrap size of 10%. Also $d_i = 0$ if $d_i > 3$ (Ratanamahatana and Gunopulos, 2003).
MH	Assign weight to attribute i as given in Equation 8 where $L = 10$ with a bootstrap size of 50% (Hall, 2007).
SB	Use Selective Bayes method to determine the rank of individual features and assign weights according to Equation 9 (Langley and Sage, 1994).
CFS	Use correlation-based feature selection method to determine the rank of individual features and assign weights according to Equation 9 (Langley and Sage, 1994; Hall, 2007).
	Selective Naive Bayes (using Typical Feature Selection Methods).
SBC-FS	Similar to SBC except $w_i = 1$ if $w_i > 0$.
MH-FS	Similar to MH except $w_i = 1$ if $w_i > 0$ (Hall, 2007).
	Weighted Naive Bayes (Ferreira et al., 2001).
FNB-d1	Weights computed per attribute value using Equation 10 with $\alpha = 2$.
FNB-d2	Weights computed per attribute value using Equation 10 with $\alpha = 2$.
	Semi-naive Bayes Classifiers.
AnDE	Average n-Dependent Estimator (Webb et al., 2011).
TAN	Tree Augmented Naive Bayes (Friedman et al., 1997).
	State of the Art Classification Techniques.
RF	Random Forests (Breiman, 2001).
LR	Logistic Regression (Roos et al., 2005).
	Weighted Naive Bayes (Proposed Methods, will be discussed in Section 4).
$\text{WANBIA}^{\text{CLL}}$	Naive Bayes weights obtained by maximizing Conditional Log-Likelihood.
$\text{WANBIA}^{\text{MSE}}$	Naive Bayes weights obtained by minimizing Mean-Square-Error.

Table 2: Summary of techniques compared in this research.

reduce the accuracy of the classifier relative to using uniform weights in any situation where x_1 and x_3 receive higher weights than x_2.

Rather than selecting weights based on measures of predictiveness, we suggest it is more profitable to pursue approaches such as those of Zhang and Sheng (2004) and Wu and Cai (2011) that optimize the weights to improve the prediction performance of the weighted classifier as a whole.

Following from Equations 1, 2 and 5, let us re-define the weighted naive Bayes model as:

$$\hat{\mathrm{P}}(y|\mathbf{x};\pi,\Theta,\mathbf{w}) = \frac{\pi_y \prod_i \theta_{X_i=x_i|y}^{w_i}}{\sum_{y'} \pi_{y'} \prod_i \theta_{X_i=x_i|y'}^{w_i}}, \tag{10}$$

with constraints:

$$\sum_y \pi_y = 1 \;\text{ and }\; \forall_{y,i} \sum_j \theta_{X_i=x_i|y} = 1,$$

where

- $\{\pi_y, \theta_{X_i=x_i|y}\}$ are naive Bayes parameters.
- $\pi \in [0,1]^{|\mathcal{Y}|}$ is a class probability vector.
- The matrix Θ consist of class and attribute-dependent probability vectors $\theta_{i,y} \in [0,1]^{|\mathcal{X}_i|}$.
- $\mathbf{w}$ is a vector of class-independent weights, w_i for each attribute i.

Our proposed method WANBIA is inspired by Cerquides and De Mántaras (2005b) where weights of different classifiers in an ensemble are calculated by maximizing the conditional log-likelihood (CLL) of the data. We will follow their approach of maximizing the CLL of the data to determine weights $\mathbf{w}$ in the model. In doing so, we will make the following assumptions:

- Naive Bayes parameters $(\pi_y, \theta_{X_i=x_i|y})$ are fixed. Hence we can write $\hat{\mathrm{P}}(y|\mathbf{x};\pi,\Theta,\mathbf{w})$ in Equation 10 as $\hat{\mathrm{P}}(y|\mathbf{x};\mathbf{w})$.
- Weights lie in the interval between zero and one and hence $\mathbf{w} \in [0,1]^a$.

For notational simplicity, we will write conditional probabilities as $\theta_{x_i|y}$ instead of $\theta_{X_i=x_i|y}$. Since our prior is constant, let us define our supervised posterior as follows:

$$\hat{\mathrm{P}}(\mathcal{L} \,|\, \mathcal{D}, \mathbf{w}) \quad = \quad \prod_{j=1}^{|\mathcal{D}|} \hat{\mathrm{P}}(y^{(j)}|\mathbf{x}^{(j)};\mathbf{w}). \tag{11}$$

Taking the log of Equation 11, we get the Conditional Log-Likelihood (CLL) function, so our objective function can be defined as:

$$\begin{aligned} \mathrm{CLL}(\mathbf{w}) &= \log \hat{\mathrm{P}}(\mathcal{L} \,|\, \mathcal{D}, \mathbf{w}) \\ &= \sum_{j=1}^{|\mathcal{D}|} \log \hat{\mathrm{P}}(y^{(j)}|\mathbf{x}^{(j)};\mathbf{w}) \\ &= \sum_{j=1}^{|\mathcal{D}|} \log \frac{\gamma_{y\mathbf{x}}(\mathbf{w})}{\sum_{y'} \gamma_{y'\mathbf{x}}(\mathbf{w})}, \end{aligned} \tag{12}$$

where

$$\gamma_{y\mathbf{x}}(\mathbf{w}) = \pi_y \prod_i \theta_{x_i|y}^{w_i}.$$

The proposed method WANBIA$^{\mathrm{CLL}}$ is aimed at solving the following optimization problem: find the weights $\mathbf{w}$ that maximizes the objective function $\mathrm{CLL}(\mathbf{w})$ in Equation 12 subject to $0 \leq w_i \leq 1 \ \forall i$. We can solve the problem by using the L-BFGS-M optimization procedure (Zhu et al., 1997). In order to do that, we need to be able to assess $\mathrm{CLL}(\mathbf{w})$ in Equation 12 and its gradient.

Before calculating the gradient of $\mathrm{CLL}(\mathbf{w})$ with respect to $\mathbf{w}$, let us find out the gradient of $\gamma_{y\mathbf{x}}(\mathbf{w})$ with respect to w_i, we can write:

$$\begin{aligned} \frac{\partial}{\partial w_i} \gamma_{y\mathbf{x}}(\mathbf{w}) &= (\pi_y \prod_{i' \neq i} \theta_{x_{i'}|y}^{w_{i'}}) \frac{\partial}{\partial w_i} \theta_{x_i|y}^{w_i} \\ &= (\pi_y \prod_{i' \neq i} \theta_{x_{i'}|y}^{w_{i'}}) \, \theta_{x_i|y}^{w_i} \log(\theta_{x_i|y}) \\ &= \gamma_{y\mathbf{x}}(\mathbf{w}) \log(\theta_{x_i|y}). \end{aligned} \tag{13}$$

Now, we can write the gradient of $\mathrm{CLL}(\mathbf{w})$ as:

$$\frac{\partial}{\partial w_i} \mathrm{CLL}(\mathbf{w}) \quad = \quad \frac{\partial}{\partial w_i} \sum_{\mathbf{x} \in \mathcal{D}} \left(\log(\gamma_{y\mathbf{x}}(\mathbf{w})) - \log(\sum_{y'} \gamma_{y'\mathbf{x}}(\mathbf{w})) \right)$$

$$= \sum_{\mathbf{x}\in\mathcal{D}} \left(\frac{\gamma_{y\mathbf{x}}(\mathbf{w}) \log(\theta_{x_i|y})}{\gamma_{y\mathbf{x}}(\mathbf{w})} - \frac{\sum_{y'} \gamma_{y'\mathbf{x}}(\mathbf{w}) \log(\theta_{x_i|y'})}{\sum_{y'} \gamma_{y'\mathbf{x}}(\mathbf{w})} \right)$$

$$= \sum_{\mathbf{x}\in\mathcal{D}} \left(\log(\theta_{x_i|y}) - \sum_{y'} \hat{\mathrm{P}}(y'|\mathbf{x};\mathbf{w}) \log(\theta_{x_i|y'}) \right). \tag{14}$$

WANBIA$^{\mathrm{CLL}}$ evaluates the function in Equation 12 and its gradient in Equation 14 to determine optimal values of weight vector $\mathbf{w}$.

Instead of maximizing the supervised posterior, one can also minimize Mean Square Error (MSE). Our second proposed weighting scheme WANBIA$^{\mathrm{MSE}}$ is based on minimizing the MSE function. Based on MSE, we can define our objective function as follows:

$$\mathrm{MSE}(\mathbf{w}) = \frac{1}{2} \sum_{\mathbf{x}^{(j)}\in\mathcal{D}} \sum_{y} (\mathrm{P}(y|\mathbf{x}^{(j)}) - \hat{\mathrm{P}}(y|\mathbf{x}^{(j)}))^2, \tag{15}$$

where we define

$$\mathrm{P}(y|\mathbf{x}^{(j)}) = \begin{cases} 1 & \text{if } y = y^{(j)} \\ 0 & \text{otherwise} \end{cases}.$$

The gradient of $\mathrm{MSE}(\mathbf{w})$ in Equation 15 with respect to $\mathbf{w}$ can be derived as:

$$\frac{\partial \mathrm{MSE}(\mathbf{w})}{\partial w_i} = -\sum_{\mathbf{x}\in\mathcal{D}} \sum_{y} \left(\mathrm{P}(y|\mathbf{x}) - \hat{\mathrm{P}}(y|\mathbf{x}) \right) \frac{\partial \hat{\mathrm{P}}(y|\mathbf{x})}{\partial w_i}, \tag{16}$$

where

$$\frac{\partial \hat{\mathrm{P}}(y|\mathbf{x})}{\partial w_i} = \frac{\frac{\partial}{\partial w_i} \gamma_{y\mathbf{x}}(\mathbf{w})}{\sum_{y'} \gamma_{y'\mathbf{x}}(\mathbf{w})} - \frac{\gamma_{y\mathbf{x}}(\mathbf{w}) \frac{\partial}{\partial w_i} \sum_{y'} \gamma_{y'\mathbf{x}}(\mathbf{w})}{(\sum_{y'} \gamma_{y'\mathbf{x}}(\mathbf{w}))^2}$$

$$= \frac{1}{\sum_{y'} \gamma_{y'\mathbf{x}}(\mathbf{w})} \left(\frac{\partial \gamma_{y\mathbf{x}}(\mathbf{w})}{\partial w_i} - \hat{\mathrm{P}}(y|\mathbf{x}) \sum_{y'} \frac{\partial \gamma_{y'\mathbf{x}}(\mathbf{w})}{\partial w_i} \right).$$

Following from Equation 13, we can write:

$$\frac{\partial \hat{\mathrm{P}}(y|\mathbf{x})}{\partial w_i} = \frac{1}{\sum_{y'} \gamma_{y'\mathbf{x}}(\mathbf{w})} \left(\gamma_{y\mathbf{x}}(\mathbf{w}) \log(\theta_{x_i|y}) - \hat{\mathrm{P}}(y|\mathbf{x}) \sum_{y'} \gamma_{y'\mathbf{x}}(\mathbf{w}) \log(\theta_{x_i|y'}) \right)$$

$$= \hat{\mathrm{P}}(y|\mathbf{x}) \log(\theta_{x_i|y}) - \hat{\mathrm{P}}(y|\mathbf{x}) \sum_{y'} \hat{\mathrm{P}}(y'|\mathbf{x}) \log(\theta_{x_i|y'})$$

$$= \hat{\mathrm{P}}(y|\mathbf{x}) \left(\log(\theta_{x_i|y}) - \sum_{y'} \hat{\mathrm{P}}(y'|\mathbf{x}) \log(\theta_{x_i|y'}) \right). \tag{17}$$

Plugging the value of $\frac{\partial \hat{\mathrm{P}}(y|\mathbf{x})}{\partial w_i}$ from Equation 17 in Equation 16, we can write the gradient as:

$$\frac{\partial \mathrm{MSE}(\mathbf{w})}{\partial w_i} = -\sum_{\mathbf{x}\in\mathcal{D}} \sum_{y} \left(\mathrm{P}(y|\mathbf{x}) - \hat{\mathrm{P}}(y|\mathbf{x}) \right) \hat{\mathrm{P}}(y|\mathbf{x})$$

$$\left(\log(\theta_{x_i|y}) - \sum_{y'} \hat{\mathrm{P}}(y'|\mathbf{x}) \log(\theta_{x_i|y'}) \right). \tag{18}$$

WANBIA$^{\text{MSE}}$ evaluates the function in Equation 15 and its gradient in Equation 18 to determine the optimal value of weight vector $\mathbf{w}$.

4.2 Connections with Logistic Regression

In this section, we will re-visit naive Bayes to illustrate WANBIA's connection with the logistic regression.

4.2.1 BACKGROUND: NAIVE BAYES AND LOGISTIC REGRESSION

As discussed in Section 2 and 4.1, the naive Bayes (NB) model for estimating $\mathrm{P}(y|\mathbf{x})$ is parameterized by a class probability vector $\pi \in [0,1]^{|\mathcal{Y}|}$ and a matrix Θ, consisting of class and attribute dependent probability vectors $\theta_{i,y} \in [0,1]^{|\mathcal{X}_i|}$. The NB model thus contains

$$(|\mathcal{Y}|-1)+|\mathcal{Y}|\sum_i(|\mathcal{X}_i|-1)$$

free parameters, which are estimated by maximizing the likelihood function:

$$\mathrm{P}(\mathcal{D},\mathcal{L};\pi,\Theta)=\prod_j \mathrm{P}(y^{(j)},\mathbf{x}^{(j)}),$$

or the posterior over model parameters (in which case they are referred to as maximum a posteriori or MAP estimates). Importantly, these estimates can be calculated analytically from attribute-value count vectors.

Meanwhile a multi-class logistic regression model is parameterized by a vector $\alpha \in \mathcal{R}^{|\mathcal{Y}|}$ and matrix $\mathcal{B} \in \mathcal{R}^{|\mathcal{Y}|\times a}$ each consisting of real values, and can be written as:

$$\mathrm{P}_{\mathrm{LR}}(y|\mathbf{x};\alpha,\mathcal{B})=\frac{\exp(\alpha_y+\sum_i \beta_{i,y}x_i)}{\sum_{y'}\exp(\alpha_{y'}+\sum_i \beta_{i,y'}x_i)},$$

where

$$\alpha_1=0 \;\;\&\;\; \forall_i\, \beta_{i,1}=0.$$

The constraints arbitrarily setting all parameters for $y=1$ to the value zero are necessary only to prevent over-parameterization. The LR model, therefore, has:

$$(|\mathcal{Y}|-1)\times(1+a)$$

free parameters. Rather than maximizing the likelihood, LR parameters are estimated by maximizing the conditional likelihood of the class labels given the data:

$$\mathrm{P}(\mathcal{L}|\mathcal{D};\alpha,\mathcal{B})=\prod_j \mathrm{P}(y^{(j)}|\mathbf{x}^{(j)}),$$

or the corresponding posterior distribution. Estimating the parameters in this fashion requires search using gradient-based methods.

Mathematically the relationship between the two models is simple. One can compare the models, by considering that the "multiplicative contribution" of an attribute value x_i in NB is found

by simply looking up the corresponding parameter $\theta_{X_i=x_i|y}$ in Θ, while for LR it is calculated as $\exp(\beta_{i,y}x_i)$, that is, by taking the exponent of the product of the value with an attribute (but *not value*) dependent parameter $\beta_{i,y}$ from $\mathcal{B}$.[2]

4.2.2 PARAMETERS OF WEIGHTED ATTRIBUTE NAIVE BAYES

The WANBIA model is an extension of the NB model where we introduce a weight vector $\mathbf{w} \in [0,1]^a$ containing a class-independent weight w_i for each attribute i. The model as written in Equation 10 includes the NB model as a special case (where $\mathbf{w} = \mathbf{1}$). We do not treat the NB parameters of the model as free however, but instead fix them to their MAP estimates (assuming the weights were all one), which can be computed analytically and therefore does not require any search. We then estimate the parameter vector w by maximizing the Conditional Log Likelihood (CLL) or by minimizing the Mean Squared Error (MSE).[3]

Thus in terms of the number of parameters that needs to be estimated using gradient-based search, a WANBIA model can be considered to have a free parameters, which is always less than the corresponding LR model with $(|\mathcal{Y}|-1)(1+a)$ free parameters to be estimated. Thus for a binary class problems containing only binary attributes, WANBIA has 1 less free parameter than LR, but for multi-class problems with binary attributes it results in a *multiplicative factor* of $|\mathcal{Y}|-1$ fewer parameters. Since parameter estimation using CLL or MSE (or even Hinge Loss) requires search, fewer free parameters to estimate means faster learning and therefore scaling to larger problems.

For problems containing non-binary attributes, WANBIA allows us to build (more expressive) *non-linear classifiers*, which are not possible for Logistic Regression unless one "binarizes" all attributes, with the resulting blow-out in the number of free parameters as mentioned above. One should note that LR can only operate on nominal data by binarizing it. Therefore, on discrete problems with nominal data, WANBIA offers significant advantage in terms of the number of free parameters.

Lest the reader assume that the only goal of this work is to find a more computationally efficient version of LR, we note that the real advantage of the WANBIA model is to make use of the information present in the easy to compute naive Bayes MAP estimates to guide the search toward reasonable settings for parameters of a model that is *not hampered by the assumption of attribute independence*.

A summary of the comparison of naive Bayes, WANBIA and Logistic Regression is given in Table 3.

5. Experiments

In this section, we compare the performance of our proposed methods WANBIA$^{\text{CLL}}$ and WANBIA$^{\text{MSE}}$ with state of the art classifiers, existing semi-naive Bayes methods and weighted naive Bayes methods based on both attribute selection and attribute weighting. The performance is analyzed in terms of 0-1 loss, root mean square error (RMSE), bias and variance on 73 natural domains from the UCI repository of machine learning (Frank and Asuncion, 2010). Table 4 describes the details of each data set used, including the number of instances, attributes and classes.

2. Note that unlike the NB model, the LR model does not require that the domain of attribute values be discrete. Non-discrete data can also be handled by Naive Bayes models, but a different parameterization for the distribution over attribute values must be used.
3. Note that we cannot maximize the Log Likelihood (LL) since the model is not generative.

	Naive Bayes	**WANBIA**	**Logistic Regression**
Approach	Estimate parameters by maximizing the likelihood function $P(\mathcal{D},\mathcal{L})$	Estimate parameters by maximizing conditional log-likelihood $P(\mathcal{L}\|\mathcal{D})$ or minimizing Mean-Squared-Error	Estimate parameters by maximizing conditional likelihood $P(\mathcal{L}\|\mathcal{D})$
Form	$\hat{P}(y\|\mathbf{x};\pi,\Theta)$	$\hat{P}(y\|\mathbf{x};\pi,\Theta,\mathbf{w})$	$\hat{P}(y\|\mathbf{x};\alpha,\mathcal{B})$
Formula	$\frac{\pi_y\prod_i\theta_{x_i\|i,y}}{\sum_{y'}\pi_{y'}\prod_i\theta_{x_i\|i,y'}}$	$\frac{\pi_y\prod_i\theta_{x_i\|i,y}^{w_i}}{\sum_{y'}\pi_{y'}\prod_i\theta_{x_i\|i,y'}^{w_i}}$	$\frac{\exp(\alpha_y+\sum_i\beta_{i,y}x_i)}{\sum_{y'}\exp(\alpha_{y'}+\sum_i\beta_{i,y'}x_i)}$
Constraints	$\pi\in[0,1]^{\|\mathcal{Y}\|}$, $\theta_{i,y}\in[0,1]^{\|\mathcal{X}_i\|}$, $\sum_y\pi_y=1$, $\forall_{y,i}\sum_j\theta_{X_i=x_i\|y}=1$	$\pi\in[0,1]^{\|\mathcal{Y}\|}$, $\theta_{i,y}\in[0,1]^{\|\mathcal{X}_i\|}$, $\mathbf{w}\in[0,1]^a$, $\sum_y\pi_y=1$, $\forall_{y,i}\sum_j\theta_{X_i=x_i\|y}=1$	$\alpha_1=0,\forall_i\,\beta_{i,1}=0$
No. of 'Fixed' Parameters	$(\|\mathcal{Y}\|-1)+\|\mathcal{Y}\|\sum_i(\|\mathcal{X}_i\|-1)$	$(\|\mathcal{Y}\|-1)+\|\mathcal{Y}\|\sum_i(\|\mathcal{X}_i\|-1)$	None
No. of 'Fixed' Parameters (Binary Case)	$1+(2\times a)$	$1+(2\times a)$	None
Strategy to calculate Fixed Parameters	π and Θ are fixed to their MAP estimates	π and Θ are fixed to their MAP estimates when $\mathbf{w}=\mathbf{1}$	Not applicable
No. of 'Free' Parameters	None	a	$(\|\mathcal{Y}\|-1)\times(1+a)$
No. of 'Free' Parameters (Binary case)	None	a	$(1+a)$

Table 3: Comparison of naive Bayes, weighted naive Bayes and Logistic Regression

This section is organized as follows: we will discuss our experimental methodology with details on statistics employed and miscellaneous issues in Section 5.1. Section 5.2 illustrates the impact of a single weight on bias, variance, 0-1 loss and RMSE of naive Bayes as shown in Equation 6. The performance of our two proposed weighting methods WANBIA$^{\text{CLL}}$ and WANBIA$^{\text{MSE}}$ is compared in Section 5.3. We will discuss the calibration performance of WANBIA in Section 5.4. In Section 5.5, we will discuss results when the proposed methods are constrained to learn only a single weight. In Section 5.6 and 5.7, WANBIA$^{\text{CLL}}$ and WANBIA$^{\text{MSE}}$ are compared with naive Bayes where weights are induced through various feature weighting and feature selection schemes respectively. We compare the performance of our proposed methods with per-attribute value weight learning method of Ferreira et al. (2001) in Section 5.8. We will discuss the significance of these results in Section 5.9. The performance of our proposed methods is compared with state of the art classifiers like Average n-Dependent Estimators (AnDE), Tree Augmented Networks (TAN), Random Forests (RF) and Logistic Regression (LR) in Section 5.10, 5.11 and 5.12 respectively. Results are summarized in Section 5.13.

5.1 Experimental Methodology

The experiments are conducted in the Weka work-branch (version 3.5.7) on data sets described in Table 4. Each algorithm is tested on each data set using 20 rounds of 2-fold cross validation. We employed Friedman and Nemenyi tests with a significance level of 0.05 to evaluate the performance

Domain	Case	Att	Class	Domain	Case	Att	Class
Abalone	4177	9	3	MAGIC Gamma Telescope	19020	11	2
Adult	48842	15	2	Mushrooms	8124	23	2
Annealing	898	39	6	Musk1	476	167	2
Audiology	226	70	24	Musk2	6598	167	2
Auto Imports	205	26	7	Nettalk(Phoneme)	5438	8	52
Balance Scale	625	5	3	New-Thyroid	215	6	3
Breast Cancer (Wisconsin)	699	10	2	Nursery	12960	9	5
Car Evaluation	1728	8	4	Optical Digits	5620	49	10
Census-Income (KDD)	299285	40	2	Page BlocksClassification	5473	11	5
Chess	551	40	2	Pen Digits	10992	17	10
Connect-4 Opening	67557	43	3	Pima Indians Diabetes	768	9	2
Contact-lenses	24	5	3	Pioneer	9150	37	57
Contraceptive Method Choice	1473	10	3	Poker-hand	1175067	11	10
Covertype	581012	55	7	Postoperative Patient	90	9	3
Credit Screening	690	16	2	Primary Tumor	339	18	22
Cylinder	540	40	2	Promoter Gene Sequences	106	58	2
Dermatology	366	35	6	Satellite	6435	37	6
Echocardiogram	131	7	2	Segment	2310	20	7
German	1000	21	2	Sick-euthyroid	3772	30	2
Glass Identification	214	10	3	Sign	12546	9	3
Haberman's Survival	306	4	2	Sonar Classification	208	61	2
Heart Disease (Cleveland)	303	14	2	Spambase	4601	58	2
Hepatitis	155	20	2	Splice-junction Gene Sequences	3190	62	3
Horse Colic	368	22	2	Statlog (Shuttle)	58000	10	7
House Votes 84	435	17	2	Syncon	600	61	6
Hungarian	294	14	2	Teaching Assistant Evaluation	151	6	3
Hypothyroid(Garavan)	3772	30	4	Thyroid	9169	30	20
Ionosphere	351	35	2	Tic-Tac-Toe Endgame	958	10	2
Iris Classification	150	5	3	Vehicle	846	19	4
King-rook-vs-king-pawn	3196	37	2	Volcanoes	1520	4	4
Labor Negotiations	57	17	2	Vowel	990	14	11
LED	1000	8	10	Wall-following	5456	25	4
Letter Recognition	20000	17	26	Waveform-5000	5000	41	3
Liver Disorders (Bupa)	345	7	2	Wine Recognition	178	14	3
Localization	164860	7	3	Yeast	1484	9	10
Lung Cancer	32	57	3	Zoo	101	17	7
Lymphography	148	19	4				

Table 4: Data sets

of each algorithm. The experiments were conducted on a Linux machine with 2.8 GHz processor and 16 GB of RAM.

5.1.1 TWO-FOLD CROSS-VALIDATION BIAS-VARIANCE ESTIMATION

The Bias-variance decomposition provides valuable insights into the components of the error of learned classifiers. *Bias* denotes the systematic component of error, which describes how closely the learner is able to describe the decision surfaces for a domain. *Variance* describes the component of error that stems from sampling, which reflects the sensitivity of the learner to variations in the training sample (Kohavi and Wolpert, 1996; Webb, 2000). There are a number of different bias-variance decomposition definitions. In this research, we use the bias and variance definitions of Kohavi and Wolpert (1996) together with the repeated cross-validation bias-variance estimation method proposed by Webb (2000). Kohavi and Wolpert define bias and variance as follows:

$$\text{bias}^2 = \frac{1}{2}\sum_{y\in\mathcal{Y}}\left(\text{P}(y|\mathbf{x}) - \hat{\text{P}}(y\,|\,\mathbf{x})\right)^2,$$

and

$$\text{variance} = \frac{1}{2}\left(1 - \sum_{y \in \mathcal{Y}} \hat{\mathrm{P}}(y \,|\, \mathbf{x})^2\right).$$

In the method of Kohavi and Wolpert (1996), which is the default bias-variance estimation method in Weka, the randomized training data are randomly divided into a training pool and a test pool. Each pool contains 50% of the data. 50 (the default number in Weka) local training sets, each containing half of the training pool, are sampled from the training pool. Hence, each local training set is only 25% of the full data set. Classifiers are generated from local training sets and bias, variance and error are estimated from the performance of the classifiers on the test pool. However, in this work, the repeated cross-validation bias-variance estimation method is used as it results in the use of substantially larger training sets. Only two folds are used because, if more than two are used, the multiple classifiers are trained from training sets with large overlap, and hence the estimation of variance is compromised. A further benefit of this approach relative to the Kohavi Wolpert method is that every case in the training data is used the same number of times for both training and testing.

A reason for performing bias/variance estimation is that it provides insights into how the learning algorithm will perform with varying amount of data. We expect low variance algorithms to have relatively low error for small data and low bias algorithms to have relatively low error for large data (Damien and Webb, 2002).

5.1.2 STATISTICS EMPLOYED

We employ the following statistics to interpret results:

- **Win/Draw/Loss (WDL) Record -** When two algorithms are compared, we count the number of data sets for which one algorithm performs better, equally well or worse than the other on a given measure. A standard binomial sign test, assuming that wins and losses are equiprobable, is applied to these records. We assess a difference as significant if the outcome of a one-tailed binomial sign test is less than 0.05.

- **Average -** The average (arithmetic mean) across all data sets provides a gross indication of relative performance in addition to other statistics. In some cases, we normalize the results with respect to one of our proposed method's results and plot the geometric mean of the ratios.

- **Significance (Friedman and Nemenyi) Test -** We employ the Friedman and the Nemenyi tests for comparison of multiple algorithms over multiple data sets (Demšar, 2006; Friedman, 1937, 1940). The Friedman test is a non-parametric equivalent of the repeated measures ANOVA (analysis of variance). We follow the steps below to compute results:

  - Calculate the rank of each algorithm for each data set separately (assign average ranks in case of a tie). Calculate the average rank of each algorithm.
  - Compute the Friedman statistics as derived in Kononenko (1990) for the set of average ranks:

$$F_F = \frac{(D-1)\chi_F^2}{D(g-1) - \chi_F^2}, \tag{19}$$

where

$$\chi_F^2 = \frac{12D}{g(g+1)}\left(\sum_i R_i^2 - \frac{g(g+1)^2}{4}\right),$$

g is the number of algorithms being compared, D is the number of data sets and R_i is the average rank of the i-th algorithm.

- Specify the null hypothesis. In our case the null hypothesis is that there is no difference in the average ranks.
- Check if we can reject the null hypothesis. One can reject the null hypothesis if the Friedman statistic (Equation 19) is larger than the critical value of the F distribution with $g-1$ and $(g-1)(D-1)$ degrees of freedom for $\alpha = 0.05$.
- If null hypothesis is rejected, perform Nemenyi tests which is used to further analyze which pairs of algorithms are significantly different. Let d_{ij} be the difference between the average ranks of the i^{th} algorithm and j^{th} algorithm. We assess the difference between the algorithms to be significant if $d_{ij} > \textit{critical difference}$ (CD) $= q_{0.05}\sqrt{\frac{g(g+1)}{6D}}$, where $q_{0.05}$ are the critical values that are calculated by dividing the values in the row for the infinite degree of freedom of the table of Studentized range statistics ($\alpha = 0.05$) by $\sqrt{2}$.

5.1.3 MISCELLANEOUS ISSUES

This section explains other issues related to the experiments.

- **Probability Estimates -** The base probabilities of each algorithm are estimated using m-estimation, since in our initial experiments it leads to more accurate probabilities than Laplace estimation for naive Bayes and weighted naive Bayes. In the experiments, we use $m = 1.0$, computing the conditional probability as:

$$\hat{\mathrm{P}}(x_i|y) = \frac{N_{x_i,y} + \frac{m}{|X_i|}}{(N_y - N_?) + m}, \tag{20}$$

where $N_{x_i,y}$ is the count of data points with attribute value x_i and class label y, N_y is the count of data points with class label y, $N_?$ is the number of missing values of attribute i.

- **Numeric Values -** To handle numeric attributes we tested the following techniques in our initial experiments:
  - Quantitative attributes are discretized using three bin discretization.
  - Quantitative attributes are discretized using Minimum Description Length (MDL) discretization (Fayyad and Keki, 1992).
  - Kernel Density Estimation (KDE) computing the probability of numeric attributes as:

$$\hat{\mathrm{P}}(x_i|y) = \frac{1}{n}\sum_{\mathbf{x}^{(j)} \in \mathcal{D}} \exp\left(\frac{||x_i^j - x_i||^2}{\lambda^2}\right).$$

	NB vs. NB^w									
W/D/L	w=0.1	w=0.2	w=0.3	w=0.4	w=0.5	w=0.6	w=0.7	w=0.8	w=0.9	w=1.0
0/1 Loss	61/2/10	58/1/14	53/1/19	51/3/19	46/3/24	39/5/29	36/8/29	28/11/34	23/12/38	0/73/0
	<**0.001**	<**0.001**	<**0.001**	<**0.001**	0.011	0.532	0.457	0.525	0.072	2
	NB vs. NB^w									
W/D/L	w=0.1	w=0.2	w=0.3	w=0.4	w=0.5	w=0.6	w=0.7	w=0.8	w=0.9	w=1.0
RMSE	53/1/19	44/1/28	37/1/35	26/1/46	21/1/51	18/1/54	17/1/55	13/1/59	12/2/59	0/73/0
	<**0.001**	0.076	0.906	**0.024**	<**0.001**	<**0.001**	<**0.001**	<**0.001**	<**0.001**	2
	NB vs. NB^w									
W/D/L	w=0.1	w=0.2	w=0.3	w=0.4	w=0.5	w=0.6	w=0.7	w=0.8	w=0.9	w=1.0
Bias	59/2/12	54/1/18	51/3/19	49/3/21	48/5/20	44/4/25	43/6/24	35/9/29	29/14/30	0/73/0
	<**0.001**	<**0.001**	<**0.001**	<**0.001**	<**0.001**	**0.029**	**0.027**	0.532	1	2
	NB vs. NB^w									
W/D/L	w=0.1	w=0.2	w=0.3	w=0.4	w=0.5	w=0.6	w=0.7	w=0.8	w=0.9	w=1.0
Variance	39/1/33	38/3/32	30/7/36	31/4/38	30/6/37	31/4/38	23/11/39	22/18/33	25/18/30	0/73/0
	0.556	0.550	0.538	0.470	0.463	0.470	0.055	0.177	0.590	2

Table 5: Win/Draw/Loss comparison of NB with weighted NB of form Equation 6

  - k-Nearest neighbor (k-NN) estimation to compute the probability of numeric attributes. The probabilities are computed using Equation 20, where $N_{x_i,y}$ and N_y are calculated over a neighborhood spanning k neighbors of x_i. We use $k = 10$, $k = 20$ and $k = 30$.

  While a detailed analysis of the results of this comparison is beyond the scope of this work, we summarize our findings as follows: the k-NN approach with $k = 50$ achieved the best 0-1 loss results in terms of Win/Draw/Loss. The k-NN with $k = 20$ resulted in the best bias performance, KDE in the best variance and MDL discretization in best RMSE performance. However, we found KDE and k-NN schemes to be extremely slow at classification time. We found that MDL discretization provides the best trade-off between the accuracy and computational efficiency. Therefore, we chose to discretize numeric attributes with MDL scheme.

- **Missing Values -** For the results reported in from Section 5.2 to Section 5.9, the missing values of any attributes are incorporated in probability computation as depicted in Equation 20. Starting from Section 5.10, missing values are treated as a distinct value. The motivation behind this is to have a fair comparison between WANBIA and other state of the art classifiers, for instance, Logistic Regression and Random Forest.

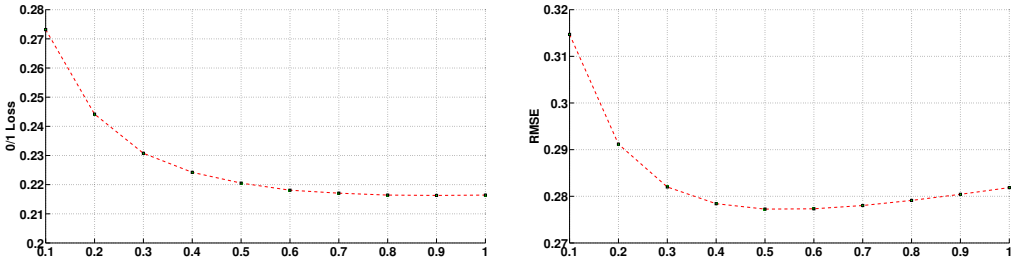
- **Notation -** We will categorize data sets in terms of their size. For example, data sets with instances ≤ 1000, > 1000 and ≤ 10000, > 10000 are denoted as bottom size, medium size and top size respectively. We will report results on these sets to discuss suitability of a classifier for data sets of different sizes.

5.2 Effects of a Single Weight on Naive Bayes

In this section, we will employ the Win/Draw/Loss record (WDL) and simple arithmetic mean to summarize the effects of weights on naive Bayes' classification performance. Table 5 compares the WDL of naive Bayes in Equation 3 with weighted naive Bayes in Equation 6 as the weight w is varied from 0.1 to 1.0. The WDL are presented for 0-1 loss, RMSE, bias and variance. The results reveal that higher value of w, for example, $w = 0.9$ results in significantly better performance in

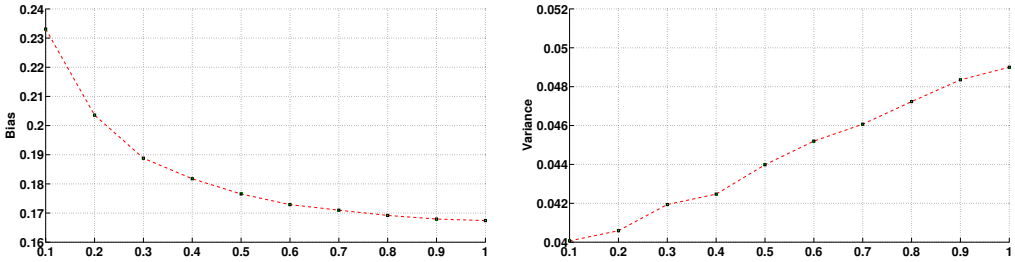
terms of 0-1 loss and RMSE and non-significantly better performance in terms of bias and variance as compared to the lower values.

Averaged (arithmetic mean) 0-1 loss, RMSE, bias and variance results across 73 data sets as a function of weight are plotted in Figure 2 and 3. As can be seen from the figures that as w



(a) (b)

Figure 2: Averaged 0-1 Loss (2(a)), RMSE (2(b)) across 73 data sets, as function of w.



(a) (b)

Figure 3: Averaged Bias (3(a)) Variance (3(b)) across 73 data sets, as function of w.

is increased from 0.1 to 1.0, bias decreases and variance increases. It is hard to characterize 0-1 loss and RMSE curves in Figure 2. 0-1 loss is decreased as we increase the value of w and is almost constant when $w > 0.7$. However, RMSE drops as w is increased to 0.5 and then increases for $w > 0.5$. As the results are averaged across 73 data sets, it is hard to say anything conclusive, however, we conjecture that the optimal value of 0.5 for w in case of RMSE metric suggests that in most data sets, only half of the attributes are providing independent information.

5.3 Mean-Square-Error versus Conditional-Log-Likelihood Objective Function

The Win/Draw/Loss comparison of our two proposed methods WANBIA$^{\text{CLL}}$ and WANBIA$^{\text{MSE}}$ is given in Table 6. It can be seen that WANBIA$^{\text{MSE}}$ has significantly better bias but significantly worst variance than WANBIA$^{\text{CLL}}$. Also, WANBIA$^{\text{MSE}}$ wins on the majority of data sets in terms of 0-1 loss and RMSE but the results are not significant. The two methods are also compared against naive Bayes in Table 7. The two versions of WANBIA win significantly in terms of bias, 0-1 loss and RMSE against naive Bayes.

	WANBIA$^{\text{CLL}}$ vs. WANBIA$^{\text{MSE}}$	
	W/D/L	*p*
Bias	19/10/44	**0.002**
Variance	42/7/24	**0.035**
0-1 Loss	26/8/39	0.136
RMSE	28/12/33	0.608

Table 6: Win/Draw/Loss: WANBIA$^{\text{CLL}}$ versus WANBIA$^{\text{MSE}}$

	WANBIA$^{\text{CLL}}$ vs. NB		**WANBIA$^{\text{MSE}}$ vs. NB**	
	W/D/L	*p*	W/D/L	*p*
Bias	55/7/11	<**0.001**	57/7/9	<**0.001**
Variance	24/8/41	**0.046**	24/7/42	**0.035**
0-1 Loss	46/8/19	**0.001**	49/7/17	<**0.001**
RMSE	55/8/10	<**0.001**	54/6/13	<**0.001**

Table 7: Win/Draw/Loss: WANBIA$^{\text{CLL}}$ versus NB, WANBIA$^{\text{MSE}}$ versus NB

Since the performance of WANBIA$^{\text{MSE}}$ and WANBIA$^{\text{CLL}}$ is similar, from the following section onwards, for simplicity we will only consider WANBIA$^{\text{MSE}}$ when comparing with other weighted NB and state of the art classification methods and denote it by WANBIA.

5.4 Comparing the Calibration of WANBIA and NB Probability Estimates

One benefit of Bayesian classifiers (and indeed also Logistic Regression) over Support Vector Machine and Decision-Tree based classifiers is that the former implicitly produce interpretable confidence values for each classification in the form of class membership probability estimates $\hat{\text{P}}(y|\mathbf{x})$. Unfortunately, the probability estimates that naive Bayes produces can often be poorly calibrated as a result of the conditional independence assumption. Whenever the conditional independence assumption is violated, which is usually the case in practice, the probability estimates tend to be more extreme (closer to zero or one) than they should otherwise be. In other words, the NB classifier tends to be more confident in its class membership predictions than is warranted given the training data. Poor calibration and over-confidence do not always affect performance in terms of 0-1 Loss, but in many applications accurate estimates of the probability of $\mathbf{x}$ belonging to class y are needed (Zadrozny and Elkan, 2002).

Since, WANBIA is based on alleviating the attribute-independence assumption, it also corrects for naive Bayes' poor calibration as can be seen in Figure 4.

Formally, we say a classifier is well-calibrated (Murphy and Winkler, 1977), if the empirical class membership probability $\tilde{\text{P}}(y|\hat{\text{P}}(y|\mathbf{x}))$ conditioned on the predicted probability $\hat{\text{P}}(y|\mathbf{x})$ converges to the latter, as the number of training examples goes to infinity. Putting it more simply, if we count the number of data points for which a classifier assigned a particular class probability of say $\hat{\text{P}}(y|\mathbf{x}) = 0.3$, then if the classifier is well-calibrated we would expect approximately 30% of these data points to be members of class y in the data set.

Figure 4 shows reliability diagrams showing the relative calibration for naive Bayes and WANBIA (DeGroot and Fienbert, 1982). Reliability diagrams plot empirical class membership probability $\tilde{\mathrm{P}}(y|\hat{\mathrm{P}}(y|\mathbf{x}))$ versus predicted class membership probability for $\hat{\mathrm{P}}(y|\mathbf{x})$ at various levels of the latter. If a classifier is well-calibrated, all points will lie on the diagonal indicating that estimates are equal to their empirical probability. In the diagrams, the empirical probability $\tilde{\mathrm{P}}(y|\hat{\mathrm{P}}(y|\mathbf{x}) = p)$ is the ratio of the number of training points with predicted probability p belonging to class y to the total number of training points with predicted probability p. Since, the number of different predicted values is large as compared to the number of data points, we can not calculate reliable empirical probabilities for each data point, but instead bin the predicted values along the x-axis. For plots in Figure 4, we have used a bin size of 0.05.

Reliability diagrams are shown for sample data sets Adult, Census-income, Connect-4, Localization, Magic, Page-blocks, Pendigits, Satellige and Sign. One can see that WANBIA often attains far better calibrated class membership probability estimates.

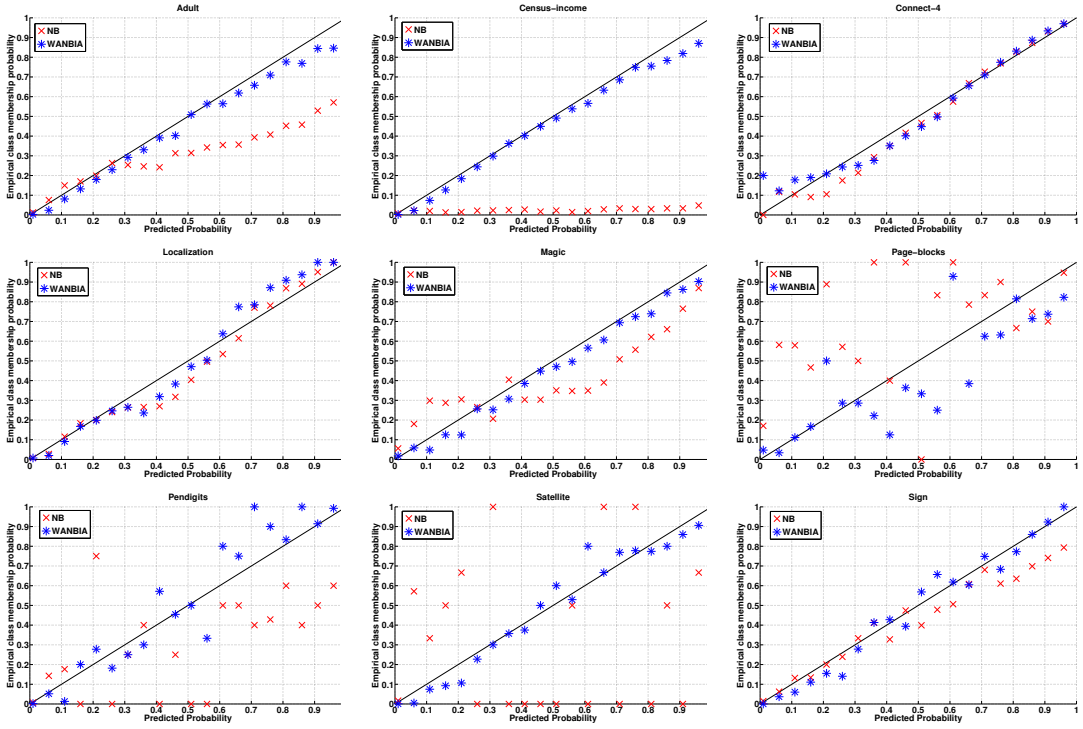


Figure 4: Reliability diagrams of naive Bayes and WANBIA on nine data sets.

5.5 Single versus Multiple Naive Bayes Weights Learning

To study the effect of single versus multiple weight learning for naive Bayes (naive Bayes in Equation 5 versus naive Bayes in Equation 6), we constrained WANBIA to learn only a single weight for all attributes. The method is denoted by WANBIA-S and compared with WANBIA and naive Bayes in Table 8.

It can be seen that learning multiple weights result in significantly better 0-1 loss, bias and RMSE as compared to learning a single weight but significantly worst variance. This is again

	vs. WANBIA		vs. NB	
	W/D/L	p	W/D/L	p
Bias	5/7/61	<**0.001**	27/18/28	1
Variance	46/7/20	**0.001**	29/21/23	0.488
0-1 Loss	17/7/49	<**0.001**	30-18/25	0.590
RMSE	19/7/47	<**0.001**	52/15/6	<**0.001**

Table 8: Win/Draw/Loss: WANBIA-S vs. WANBIA and NB

the effect of the bias-variance trade-off. Learning multiple weights result in lowering the bias but increases the variance of classification. As can be seen from the table, the performance of WANBIA-S compared to NB is fairly even in terms of 0-1 loss, bias and variance and WDL results are non-significant. However, RMSE is significantly improved as WANBIA-S improves naive Bayes probability estimates on 52 of the 73 data sets.

5.6 WANBIA versus Weighted Naive Bayes Using Feature Weighting Methods

The Win/Draw/Loss results of WANBIA against GRW, SBC, MH and CFS weighting NB techniques are given in Table 9. It can be seen that WANBIA has significantly better 0-1 loss, bias and RMSE than all other methods. Variance is, however, worst comparing to GRW, CFS and SB.

		vs. GRW	vs. SBC	vs. MH	vs. CFS	vs. SB
Bias		60/0/13	64/1/8	62/3/8	63/4/6	61/5/7
	p	<**0.001**	**0.048**	<**0.001**	**0.048**	<**0.001**
Variance		31/1/41	46/1/26	28/2/43	21/4/48	29/3/41
	p	0.288	**0.012**	0.095	**0.001**	0.188
0-1 Loss		58/0/15	66/1/6	57/2/14	50/3/20	52/3/18
	p	<**0.001**	<**0.001**	<**0.001**	<**0.001**	<**0.001**
RMSE		65/1/7	62/2/9	54/2/17	50/4/19	52/3/18
	p	<**0.001**	<**0.001**	<**0.001**	<**0.001**	<**0.001**

Table 9: Win/Draw/Loss: WANBIA versus Feature Weighting Methods

5.7 WANBIA versus Selective Naive Bayes

In this section, we will compare WANBIA performance with that of selective naive Bayes classifiers SBC-FS and MH-FS. The Win/Draw/Loss results are given in Table 10. It can be seen that WANBIA has significantly better 0-1 loss, bias and RMSE as compared to SBC-FS and MH-FS. It also has better variance as compared to the other methods.

5.8 WANBIA versus Ferreira et al. Approach

WANBIA comparison with Ferreira et al. (2001) approaches FNB-d1 and FNB-d2 in terms of 0-1 loss, RMSE, bias and variance is given in Table 11. WANBIA has significantly better 0-1 loss, bias and RMSE and non-significantly worst variance as compared to the other methods.

	vs. SBC-FS		**vs. MH-FS**	
	W/D/L	p	W/D/L	p
Bias	58/3/12	<**0.001**	58/6/9	<**0.001**
Variance	52/3/18	<**0.001**	37/6/30	0.463
0-1 Loss	65/3/5	<**0.001**	56/6/11	<**0.001**
RMSE	65/3/5	<**0.001**	64/5/4	<**0.001**

Table 10: Win/Draw/Loss: WANBIA$^{\text{CLL}}$, WANBIA$^{\text{MSE}}$ vs. SBC-FS and MH-FS

	vs. FNB-d1		**vs. FNB-d2**	
	W/D/L	p	W/D/L	p
Bias	70/2/1	<**0.001**	64/1/8	<**0.001**
Variance	27/3/43	**0.072**	30/1/42	0.194
0-1 Loss	58/2/13	<**0.001**	59/1/13	<**0.001**
RMSE	56/1/16	<**0.001**	59/1/13	<**0.001**

Table 11: Win/Draw/Loss: WANBIA vs. FNB-d1 and FNB-d2

5.9 Discussion

In this section, we discuss the significance of the results presented in the Sections 5.6, 5.7 and 5.8 using Friedman and Nemenyi tests. Following the graphical representation of Demšar (2006), we show the comparison of techniques WANBIA, GRW, SBC, MH, CFS, SB, FNB-d1, FNB-d2, SBC-FS and MH-FS against each other on each metric, that is, 0-1 loss, RMSE, bias and variance.

We plot the algorithms on a vertical line according to their ranks, the lower the better. Ranks are also displayed on a parallel vertical line. Critical difference is also plotted. Algorithms are connected by a line if their differences are not significant. This comparison involves 10 ($a = 10$) algorithms with 73 ($D = 73$) data sets. The Friedman statistic is distributed according to the F distribution with $a-1 = 9$ and $(a-1)(D-1) = 648$ degrees of freedom. The critical value of $F(9,648)$ for $\alpha = 0.05$ is 1.8943. The Friedman statistics for 0-1 loss, bias, variance and RMSE in our experiments are 18.5108, 24.2316, 9.7563 and 26.6189 respectively. Therefore, the null hypotheses were rejected. The comparison using Nemenyi test on bias, variance, 0-1 loss and RMSE is shown in Figure 5.9.

As can be seen from the figure, the rank of WANBIA is significantly better than that of other techniques in terms of the 0-1 loss and bias. WANBIA ranks first in terms of RMSE but its score is not significantly better better than that of SB. Variance-wise, FNB-d1, GRW, CFS, FNB-d2 and MH have the top five ranks with performance not significantly different among them, whereas WANBIA stands eighth, with rank not significantly different from GRW, fnbd1, MH, SB, fnbd2 and MH-FS.

5.10 WANBIA versus Semi-naive Bayes Classification

In this section, we will compare WANBIA with semi-naive Bayes methods Tree Augmented Naive Bayes (TAN) and Average n-Dependence Estimators (AnDE). AnDE provides a family of classification algorithms that includes naive Bayes when $n = 0$. As n increases, bias decreases and variance of classification increases. We will constrain to A1DE in this work. A1DE relaxes NB's attribute independence assumption by (only) making each attribute independent given the class and one at-

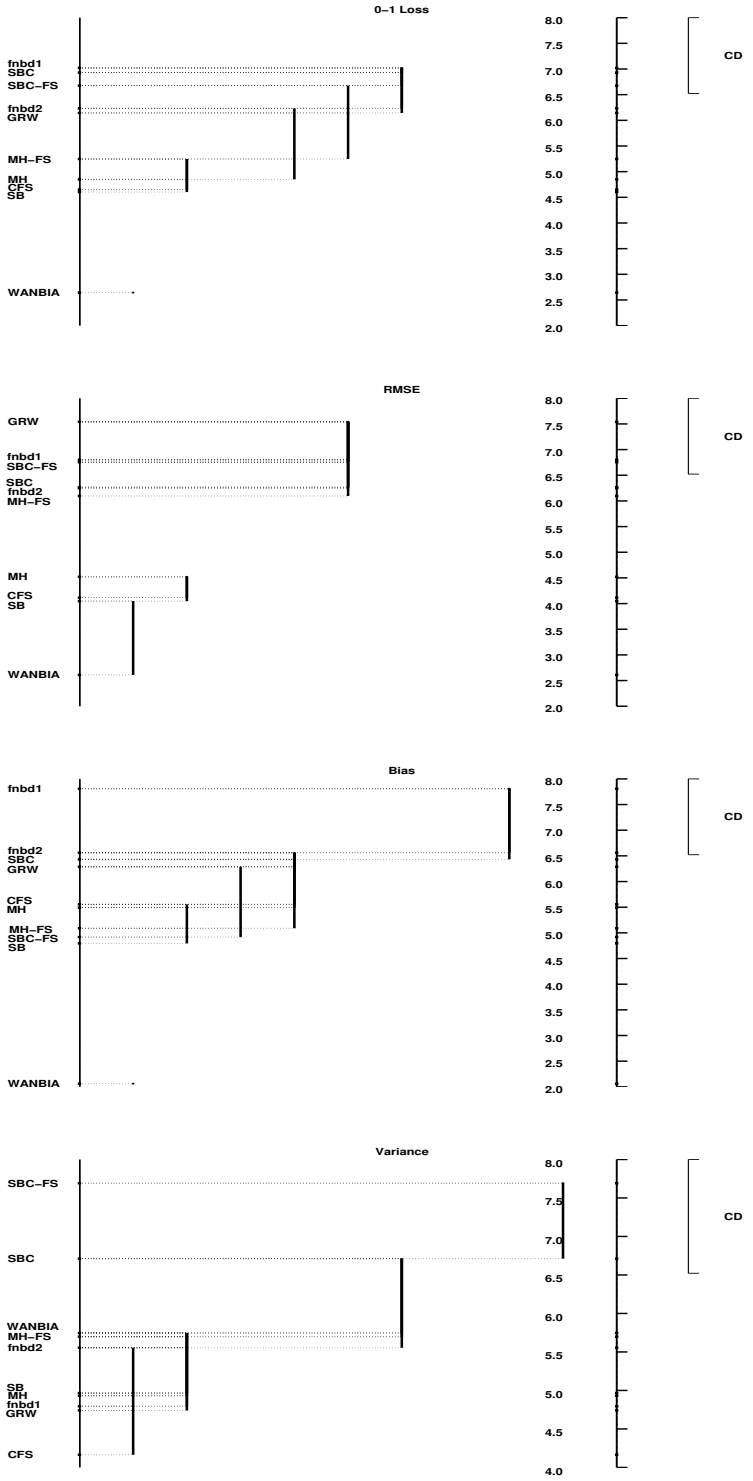


Figure 5: 0-1 Loss, RMSE, Bias, Variance comparison of 10 algorithms (GRW, SBC, MH, CFS, SB, SBC-FS, MH-FS, FNB-d1, FNB-d2, WANBIA) with the Nemenyi test on 73 data sets. CD = 1.4778.

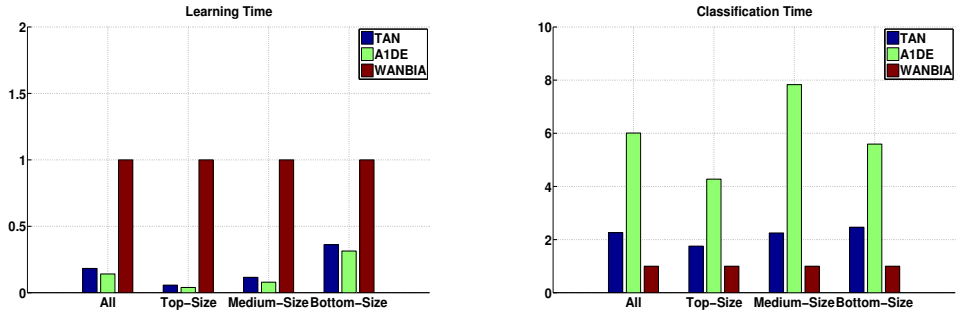


Figure 6: Averaged learning time (left) and classification time (right) of TAN, A1DE and WANBIA on all 73, Top, Medium and Bottom size data sets. Results are normalized with respect to WANBIA and geometric mean is reported.

tribute, the super-parent. This results in a one-dependence classifier. A1DE is an ensemble of these one-dependence classifiers. As A1DE is based on learning without search, every attribute takes a turn to be a super-parent. A1DE estimates by averaging over all estimates of $\mathrm{P}(y,\mathbf{x})$, that is:

$$\hat{\mathrm{P}}(y,\mathbf{x}) \quad = \quad \frac{1}{a}\sum_{i=1}^{a}\hat{\mathrm{P}}(y,x_i)\hat{\mathrm{P}}(\mathbf{x}|y,x_i).$$

Similarly, TAN augments the naive Bayes structure by allowing each attribute to depend on at most one non-class attribute. Unlike A1DE, it is not an ensemble and uses an extension of the Chow-Liu tree that uses conditional mutual information to find a maximum spanning tree as a classifier. The estimate is:

$$\hat{\mathrm{P}}(y,\mathbf{x}) \quad = \quad \hat{\mathrm{P}}(y)\prod_{i=1}^{a}\hat{\mathrm{P}}(x_i|y,\pi(x_i)),$$

where $\pi(x_i)$ is the parent of attribute x_i.

Bias-variance analysis of WANBIA with respect to TAN and A1DE is given in Table 12 showing that WANBIA has similar bias-variance performance to A1DE and significantly better variance performance to TAN with slightly worst bias. Considering, TAN is a low bias high variance learner, it should be suitable for large data. This can be seen in Table 13 where TAN has significantly better 0-1 Loss and RMSE performance than WANBIA on large data sets and significantly worst on small. The average learning and classification time comparison of WANBIA and TAN is given in Figure 6 and scatter of the actual time values is given in Figures 7 and 8. Even though, TAN is competitive to WANBIA in terms of learning time (training TAN involves a simple optimization step), we claim that WANBIA's improved performance on medium and small size data sets is very encouraging.

WANBIA has similar bias-variance profile to A1DE and one can expect it to perform more or less like A1DE. This can be seen from the 0-1 Loss and RMSE comparison results given in Table 14. Most of the results are not significant, except on large data sets where A1DE is more effective. However, this improved performance for larger data sets has a toll associated with it. As can be seen from Figure 6 (right) and Figure 8, A1DE is extremely computationally expensive especially at (critical) classification time. Even though, WANBIA and A1DE performs in a similar

	vs. TAN		vs. A1DE	
	W/D/L	p	W/D/L	p
Bias	31/2/40	0.342	35/3/35	1.09
Variance	61/2/10	<**0.001**	35/3/35	1.09

Table 12: Win/Draw/Loss: Bias-variance analysis of WANBIA, TAN and A1DE

			Size		
		All	Top	Medium	Bottom
0-1 Loss		48/2/23	2/0/10	14/1/6	32/1/7
	p	**0.004**	**0.038**	0.115	<**0.001**
RMSE		46/1/26	2/0/10	14/1/6	30/0/10
	p	**0.024**	**0.038**	0.115	**0.002**

Table 13: Win/Draw/Loss: WANBIA versus TAN

fashion on small and medium size data sets, WANBIA offers a huge improvement over the state of the art by offering a faster algorithm at classification time. Note, that training A1DE does not involve any optimization step and hence offers a fast training step as compared to other traditional learning algorithms.

			Size		
		All	Top	Medium	Bottom
0-1 Loss		31/4/38	2/1/9	10/1/10	19/2/19
	p	0.470	0.065	1.176	1.128
RMSE		30/3/40	2/0/10	9/1/11	19/2/19
	p	0.282	**0.038**	0.823	1.128

Table 14: Win/Draw/Loss: WANBIA versus A1DE

5.11 WANBIA versus Random Forest

Random Forest (RF) (Breiman, 2001) is considered to be a state of the art classification scheme. RFs consist of multiple decision trees, each tree is trained on data selected at random but with replacement from the original data (bagging). For example, if there are N data points, select N data points at random with replacement. If there are n attributes, a number m is specified, such that $m < n$. At each node of the decision tree, m attributes are randomly selected out of n and are evaluated, the best being used to split the node. Each tree is grown to its largest possible size and no pruning is done. Classifying an instance encompasses passing it through each decision tree and the output is determined by the mode of the output of decision trees. We used 100 decision trees in this work.

Bias-variance comparison of WANBIA and RF in Table 15 suggests that RF is a low bias and high variance classifier. Like TAN, one can expect it to work extremely well on large data sets. This is evident from Table 16 where 0-1 Loss and RMSE of RF and WANBIA is compared. Note, we were unable to compute results for RF on our two largest data sets Poker-hand and Covertype

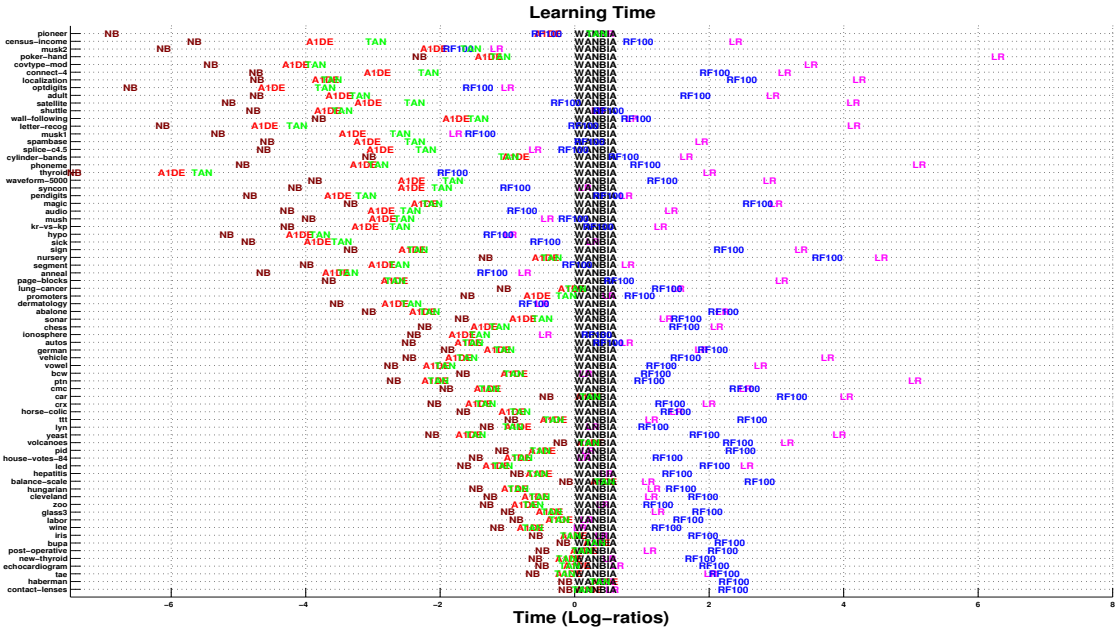


Figure 7: Learning time comparison of NB, A1DE, TAN, RF, LR and WANBIA on all 73 data sets. Results are normalized with respect to WANBIA and log-ratios are plotted.

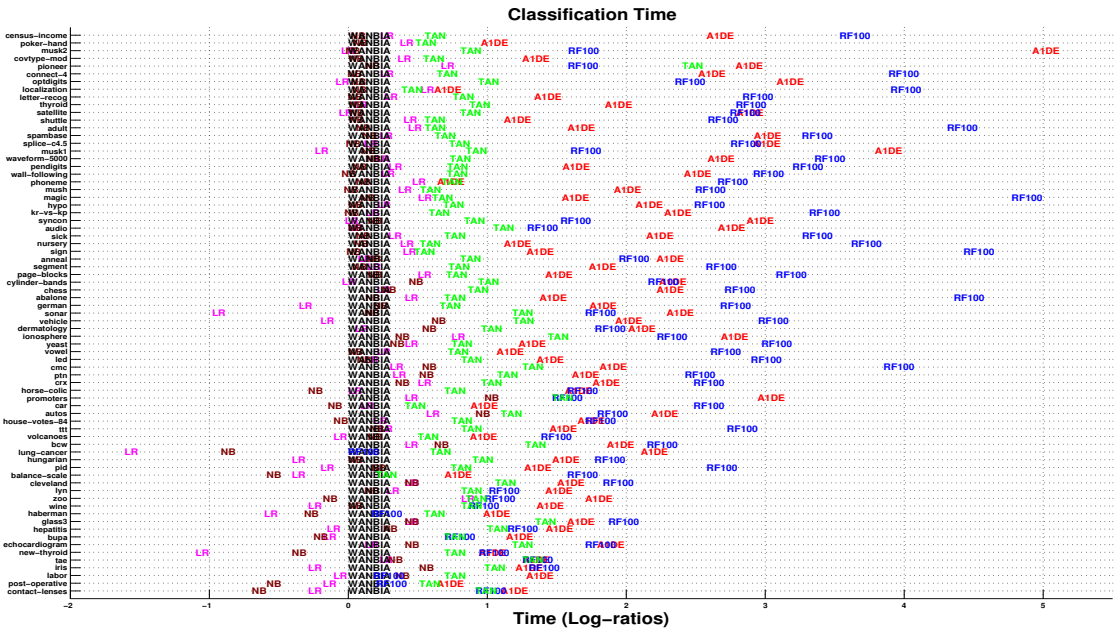


Figure 8: Classification time comparison of NB, A1DE, TAN, RF, LR and WANBIA on all 73 data sets. Results are normalized with respect to WANBIA and log-ratios are plotted.

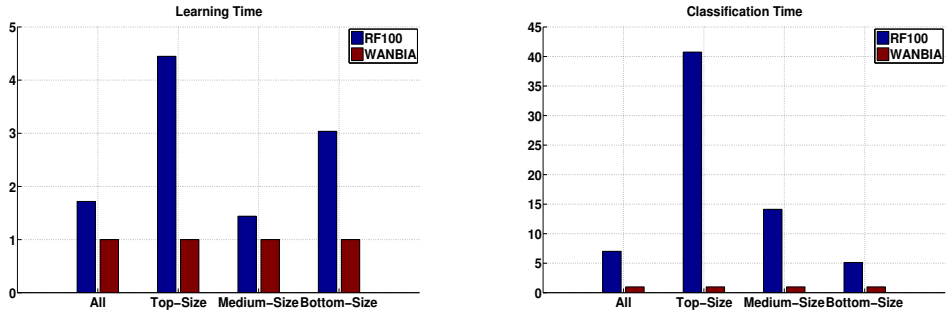


Figure 9: Averaged learning time (left) and classification time (right) of RF and WANBIA on all 73, Top, Medium and Bottom size data sets. Results are normalized with respect to WANBIA and geometric mean is reported.

(Table 4). Even with 32 GB of RAM, Weka exhausted heap memory during cross-validation experiments on RF for these data sets. However, due to its low bias one would expect RF to beat WANBIA on these two data sets, resulting in W/D/L of $2/0/10$ and $3/0/9$ on largest data sets for 0-1 Loss and RMSE with a significance of **0.038** and 0.146.

	vs. RF	
	W/D/L	p
Bias	21/2/48	**0.001**
Variance	53/3/16	$<$**0.001**

Table 15: Win/Draw/Loss: Bias-variance analysis of WANBIA and RF

			Size		
		All	Top	Medium	Bottom
0-1 Loss		38/1/32	2/0/8	7/1/13	29/0/11
	p	0.550	0.109	0.263	**0.006**
RMSE		42/1/28	3/0/7	12/1/8	27/0/13
	p	0.119	0.343	0.503	**0.038**

Table 16: Win/Draw/Loss: WANBIA versus RF

On smaller data sets WANBIA has a better 0-1 Loss performance and significantly better RMSE than RF. This is packaged with WANBIA's far superior learning and classification timings over RF as can be seen from Figures 7, 8 and 9,. This makes WANBIA an excellent alternative to RF especially for small data.

5.12 WANBIA versus Logistic Regression

In this section, we compare the performance of WANBIA with state of the art discriminative classifier Logistic Regression (LR). We implemented LR as described in Roos et al. (2005). The following

objective function is optimized:

$$\begin{aligned} \text{CLL}(\beta) &= \sum_{j=1}^{|\mathcal{D}|} \log \hat{\text{P}}(y|\mathbf{x}) \\ &= \sum_{j=1}^{|\mathcal{D}|} \left(\beta_y^T \mathbf{x} - \log \left(\sum_{k'=1}^{K} \exp(\beta_{k'}^T \mathbf{x}) \right) \right). \end{aligned}$$

The gradient of $\text{CLL}(\beta)$ is:

$$\begin{aligned} \frac{\partial \text{CLL}(\beta)}{\partial \beta_{i,k}} &= \sum_{j=1}^{|\mathcal{D}|} \delta(y=k) x_i - \frac{\exp(\beta_k^T \mathbf{x})}{\sum_{k'=1}^{K} \exp(\beta_{k'}^T \mathbf{x})} x_i \\ &= \sum_{j=1}^{|\mathcal{D}|} \left(\delta(y=k) - \hat{\text{P}}(k \,|\, \mathbf{x}) \right) x_i. \end{aligned}$$

The same L-BFGS-M optimization procedure of Zhu et al. (1997) that is used for optimizing WANBIA parameters is used to learn LR parameters. For L_2 regularization, the following objective function is optimized:

$$\text{CLL}(\beta) = \sum_{j=1}^{|\mathcal{D}|} \log \hat{\text{P}}(y|\mathbf{x}) + C\|\beta\|^2,$$

where the value of C is found using 3-fold cross validation over the training data by searching C from the list: $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}, 10^{6}\}$. The value of C resulting in lowest 0-1 error is chosen.

Table 17 compares the bias and variance of WANBIA with respect to LR and regularized LR. Like RF, LR is a low bias classifier. Regularizing LR reduces its variance at the expense of increasing its bias. However, it is encouraging to see that WANBIA still has lower variance than regularized LR winning on 47, drawing on five and losing only on 20 data sets.

	vs. LR		**vs. Regularized LR**	
	W/D/L	p	W/D/L	p
Bias	18/3/51	$<$**0.001**	10/2/60	$<$**0.001**
Variance	50/5/17	$<$**0.001**	47/5/20	**0.001**

Table 17: Win/Draw/Loss: Bias-variance analysis of WANBIA, LR and regularized LR

The error of WANBIA is compared with LR in Table 18. It can be seen that LR is superior to WANBIA on large data sets. Regularized LR is very difficult to beat as can be seen from Table 19. Regularized LR results in significantly better performance on not only large but also on medium size data sets. However, WANBIA still maintains its effectiveness on small data. This is, again, extremely encouraging. The comparison of LR and WANBIA's learning and classification time is given in Figures 7, 8 and 10. Note, that we have only reported timing results for un-regularized LR, which is more efficient than regularized LR. The regularized results are given in the appendix, but are not compared as, due to their computational intensity, they were computed on a Grid comprising

			Size		
		All	Top	Medium	Bottom
0-1 Loss		32/3/37	0/1/11	6/1/13	26/1/13
	p	0.630	<**0.001**	0.167	0.053
RMSE		40/4/28	0/1/11	8/1/11	32/2/6
	p	0.181	<**0.001**	0.647	<**0.001**

Table 18: Win/Draw/Loss: WANBIA versus LR

			Size		
		All	Top	Medium	Bottom
0-1 Loss		20/2/50	0/1/11	2/1/17	21/0/19
	p	<**0.001**	<**0.001**	<**0.001**	0.874
RMSE		30/2/40	0/1/11	3/1/16	27/0/13
	p	0.282	<**0.001**	**0.004**	**0.038**

Table 19: Win/Draw/Loss: WANBIA versus Regularized LR

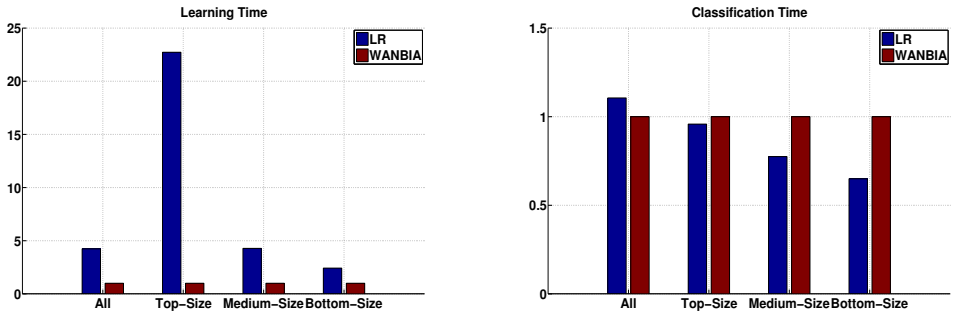


Figure 10: Averaged learning time (left) and classification time (right) of LR and WANBIA on all 73, Top, Medium and Bottom size data sets. Results are normalized with respect to WANBIA and geometric mean is reported.

computers with 4 GB of RAM and 2.0 Ghz processor. Since the environment was not controlled, we do not include them in our comparison.

Logistic Regression's better performance on large data sets and marginally better performance on medium size data sets has a computational cost associated with it. This can be seen in Figure 7. A cross-validation procedure to tune C parameters, as required for regularization, increases already relatively high costs to new heights. Therefore, WANBIA can be viewed as a substitute over LR for medium size and regularized LR on smaller data sets.

5.13 Summary of Experimental Results

We summarize our results as follows:

- WANBIA is shown to greatly improve upon naive Bayes' generalization performance. This performance gain is attributed to the fact that WANBIA successfully alleviates the conditional attribute independence assumption. Traditionally, NB is preferred for small and medium size data due to its high bias and low variance. Our results suggest that WANBIA is likely to be preferable to NB for small and medium size data sets.

- WANBIA has significantly better 0-1 loss, RMSE, bias and variance than most existing weighted naive Bayes schemes based on attribute selection and attribute weighting. As a result, WANBIA sets a new standard for attribute weighing for naive Bayes.

- WANBIA is competitive to state-of-the-art semi-naive Bayes methods TAN and A1DE. WANBIA has an edge over TAN on medium and small data sets, whereas its computational efficiency makes it a useful alternative over A1DE. However, it is credible that WANBIA's strategy for alleviating NB's attribute independence assumption is complementary to A1DE and TAN, allowing both to be applied to obtain even better classification accuracy.

- WANBIA performs significantly better on smaller data sets compared to Random Forest with 100 decision trees. While Random Forest is likely to be more accurate on larger data sets, WANBIA provides a computationally efficient alternative that may be attractive when computational burden is an issue.

- WANBIA is more accurate than both regularized and unregularized Logistic Regression on smaller data. Also, for multi-class and multi-valued data, WANBIA is based on optimizing far fewer parameters and, therefore, is computationally more efficient than LR.

6. Conclusions and Further Research

In this work we have introduced weighting schemes to incorporate weights in naive Bayes. Our work has been primarily motivated by the observation that naive Bayes conditional attribute independence assumption is often violated and, therefore, it is useful to alleviate it. We build an argument that in current research, weighting in naive Bayes has been viewed as a way of enhancing the impact of attributes that are highly correlated with the class. We argue that weighting provides a natural framework for alleviating the attribute independence assumption. Our two proposed naive Bayes weighting methods WANBIA$^{\text{CLL}}$ and WANBIA$^{\text{MSE}}$ fix naive Bayes' parameters to be the MAP estimates and learn weights by maximizing conditional log-likelihood and minimizing Mean Square Error respectively. This scheme results in the need to optimize significantly fewer parameters than LR.

Conditional log likelihood and mean square error are not the only objective functions that can be optimized to learn weights for WANBIA. One can use, for instance, Hinge Loss (generally used with SVM) and exponential loss (boosting). Another alternative is using a different form of mean square error that is $\frac{1}{2}\sum_{x^{(j)} \in |\mathcal{D}|}(1-\hat{\mathrm{P}}(y|\mathbf{x}))^2$ instead of $\frac{1}{2}\sum_{x^{(j)} \in |\mathcal{D}|}\sum_{y}(\mathrm{P}(y|\mathbf{x})-\hat{\mathrm{P}}(y|\mathbf{x}))^2$. A comparison of WANBIA trained using these objective functions has been left to future work.

In interpreting the results presented in this work, it is important to keep in mind that attribute weighting and semi-naive Bayesian relaxation of the attribute independence assumption are mutually compatible. It remains a direction for future research to explore techniques for attribute weighting in the context of semi-naive Bayes classifiers.

We have constrained ourselves in this work to weighting of the form of Equation 5. It will be interesting to optimize weights as in Equation 4, that is, optimizing a weight for each attribute's value. A next step will be to learn a weight for each attribute value per class and a weight for the prior probabilities. Such a variant of WANBIA would have the same number of parameters to optimize as Logistic Regression. For example, for i-th attribute and y-th class, weight term constitutes $\beta_{i,y}$ for LR and $w_i \log \theta_{x_i|y}$ for WANBIA.

In conclusion, with modest computation, WANBIA substantially decreases the bias of naive Bayes without unduly increasing its variance. The resulting classifier is highly competitive with the state of the art when learning from small and medium size data sets.

Acknowledgments

This research has been supported by the Australian Research Council under grant DP110101427 and Asian Office of Aerospace Research and Development, Air Force Office of Scientific Research under contract FA23861214030. The authors would like to thank Mark Hall for providing the code for CFS and MH. The authors would also like to thank anonymous reviewers for their insightful comments that helped improving the paper tremendously.

Appendix A. Code and Detailed Results

The code of the methods proposed in this work can be obtained from the website, `http://sourceforge.net/projects/rawnaivebayes/`. This appendix presents the detailed results for Error (Table 20), RMSE (Table 21), Bias (Table 22), Variance (Table 23), Train time (Table 24) and Test time (Table 25).

References

R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

B. Bjerregaard, S. Brynitz, J. Holst-Christensen, E. Kalaja, J. Lund-Kristensen, J. Hilden, F. T. deDombal, and J. C. Horrocks. Computer-aided diagnosis of acute abdomen: a system from leeds used on copenhagen patients. In *In Decision Making and Medical Care: Can Information Science Help*. North-Holland Publishing Company, 1976.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

A. Carvalho, T. Roos, A. Oliveira, and P. Myllymaki. Discriminative learning of Bayesian networks via factorized conditional log-likelihood. *Journal of Machine Learning Research*, 2011.

J. Cerquides and R. L. De Mántaras. Tan classifiers based on decomposable distributions. *Machine Learning*, 59(3):323–354, 2005a.

J. Cerquides and R. L. De Mántaras. Robust Bayesian linear classifier ensembles. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 70–81, 2005b.

	A1DE	LR	LR-Reg	NB	WANBIA^{CLL}	WANBIA^{MSE}	RF100	TAN
Mush	0.0003	0	0	0.0261	0.0012	0.001	0	0.0004
Shuttle	0.0012	0.0004	0.0004	0.004	0.0014	0.0012	0.0009	0.001
Pioneer	0.0025	0.0079	0.0079	0.0036	0.0002	0.0001	0.0008	0.0057
Syncon	0.0033	0.0102	0.0107	0.0133	0.0114	0.0118	0.0127	0.0115
Hypo	0.0115	0.0062	0.0049	0.0146	0.0083	0.0082	0.0122	0.0187
Wine	0.0174	0.0185	0.0199	0.014	0.0079	0.0112	0.0211	0.0225
Anneal	0.0188	0.0117	0.0094	0.0361	0.0199	0.019	0.0122	0.0266
Pendigits	0.0197	0.0413	0.0288	0.1179	0.1038	0.1018	0.0339	0.0411
Dermatology	0.0213	0.0288	0.031	0.0201	0.019	0.021	0.0367	0.0434
Sick	0.0263	0.0269	0.0256	0.0312	0.0263	0.0264	0.0263	0.0272
Page-blocks	0.0305	0.0368	0.0318	0.0609	0.0373	0.0351	0.0309	0.0412
Optdigits	0.033	0.0533	0.0396	0.0763	0.0642	0.0658	0.0458	0.0487
Bcw	0.0371	0.0471	0.0409	0.0266	0.0343	0.0368	0.0386	0.0513
Segment	0.0388	0.055	0.0479	0.0752	0.0502	0.0505	0.0413	0.0528
Splice-c4.5	0.0404	0.0692	0.045	0.0463	0.0411	0.04	0.0489	0.0613
New-thyroid	0.0423	0.0514	0.0528	0.0374	0.0398	0.0412	0.0479	0.0514
Musk2	0.0438	0.0207	0.0143	0.0784	0.0399	0.04	0.0385	0.0494
Labor	0.0465	0.0737	0.0816	0.0544	0.0544	0.0561	0.0939	0.0842
Wall-following	0.0479	0.0118	0.0085	0.0957	0.0219	0.0206	0.0216	0.0693
House-votes-84	0.0555	0.0493	0.0461	0.0976	0.0452	0.0463	0.0416	0.0649
Iris	0.0577	0.057	0.0573	0.0553	0.053	0.0543	0.056	0.0587
Zoo	0.0629	0.0663	0.0683	0.0713	0.0723	0.0757	0.0743	0.0931
Spambase	0.0662	0.0626	0.0588	0.0979	0.0626	0.0611	0.0575	0.0689
Ionosphere	0.0701	0.0869	0.0818	0.0868	0.0779	0.0746	0.0766	0.0781
Nursery	0.0744	0.0747	0.0747	0.0979	0.0979	0.0979	0.0248	0.0686
Thyroid	0.0752	0.0683	0.0642	0.1116	0.1061	0.0994	0.075	0.0855
Musk1	0.0869	0.0877	0.0811	0.1325	0.0649	0.0666	0.0683	0.0763
Kr-vs-kp	0.0915	0.0277	0.0286	0.1267	0.0696	0.0622	0.0128	0.0772
Census-income	0.0986	0.0433	0.0433	0.2355	0.0462	0.0461	0.0494	0.0574
Letter-recog	0.1025	0.1639	0.1495	0.2563	0.2484	0.2475	0.0902	0.151
Car	0.1069	0.0742	0.0733	0.156	0.156	0.156	0.0772	0.081
Satellite	0.1092	0.1807	0.1175	0.1751	0.1553	0.1514	0.1085	0.1179
Chess	0.1318	0.1233	0.1242	0.1364	0.1338	0.132	0.1074	0.106
Waveform-5000	0.1427	0.147	0.1386	0.1918	0.1556	0.1565	0.1558	0.1825
Crx	0.1427	0.1564	0.1396	0.1449	0.138	0.1402	0.1581	0.1651
Sonar	0.144	0.1784	0.1637	0.1519	0.1688	0.1671	0.1704	0.1683
Adult	0.1491	0.1274	0.1274	0.159	0.1306	0.1303	0.1466	0.1387
Hungarian	0.1592	0.1811	0.181	0.1585	0.1701	0.1692	0.1874	0.166
Hepatitis	0.1619	0.1923	0.17	0.1574	0.1452	0.149	0.1606	0.1713
Lyn	0.1669	0.1986	0.1858	0.1666	0.1801	0.1797	0.1909	0.2257
Magic	0.1696	0.1538	0.1537	0.2169	0.172	0.1716	0.1674	0.1619
Cleveland	0.171	0.1863	0.1766	0.1693	0.1764	0.179	0.1908	0.1909
Glass3	0.1724	0.2007	0.1776	0.1871	0.1757	0.1778	0.1951	0.1846
Autos	0.1983	0.2154	0.2059	0.2554	0.23	0.2302	0.1937	0.2437
Promoters	0.1986	0.1241	0.1302	0.1387	0.1358	0.1363	0.1519	0.2325
Horse-colic	0.2107	0.2726	0.1798	0.2126	0.1622	0.1659	0.1789	0.2236
Pid	0.2193	0.2197	0.2198	0.2215	0.2151	0.2152	0.2536	0.2249
Vowel	0.2199	0.2413	0.2371	0.3931	0.3617	0.3646	0.1674	0.2781
Cylinder-bands	0.237	0.24	0.2277	0.2559	0.2591	0.2659	0.2702	0.3761
Covtype-mod	0.2413	0.2571	0.2571	0.3117	0.2912	0.2907		0.2512
Connect-4	0.244	0.2425	0.2425	0.2792	0.2727	0.2726	0.1875	0.2368
Ttt	0.2502	0.0247	0.0181	0.2902	0.2731	0.2714	0.0765	0.2484
German	0.2535	0.2575	0.2526	0.2532	0.257	0.2571	0.2684	0.2838
Phoneme	0.263	0.2544	0.2068	0.3035	0.2587	0.2607	0.1789	0.3484
Led	0.265	0.2694	0.2659	0.2632	0.2633	0.2648	0.2802	0.2702
Balance-scale	0.2682	0.2655	0.2626	0.2594	0.2594	0.2594	0.271	0.2661
Haberman	0.2714	0.2708	0.2709	0.2714	0.2714	0.2714	0.2709	0.2722
Vehicle	0.2761	0.2845	0.2723	0.3765	0.3289	0.3288	0.2742	0.2764
Sign	0.279	0.32	0.3204	0.3593	0.3589	0.3568	0.2038	0.2747
Audio	0.3288	0.2677	0.2396	0.3305	0.292	0.2942	0.3009	0.3361
Volcanoes	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309
Echocardiogram	0.3328	0.3355	0.3351	0.3206	0.3298	0.3302	0.3489	0.3477
Contact-lenses	0.3563	0.2958	0.3604	0.3438	0.3417	0.3458	0.3438	0.4292
Localization	0.359	0.4584	0.4584	0.4939	0.4902	0.49	0.2976	0.3564
Post-operative	0.375	0.425	0.3006	0.3728	0.3611	0.3528	0.3972	0.3706
Bupa	0.3843	0.3843	0.3967	0.3843	0.3843	0.3843	0.3817	0.3843
Yeast	0.4086	0.4064	0.4059	0.4115	0.4068	0.4084	0.421	0.4096
Abalone	0.4562	0.4623	0.4656	0.4794	0.4647	0.4643	0.4823	0.4687
Poker-hand	0.4643	0.4988	0.4988	0.4988	0.4988	0.4988		0.329
Cmc	0.4791	0.447	0.4478	0.4828	0.4695	0.4654	0.4976	0.465
Tae	0.5146	0.5351	0.5334	0.5182	0.5189	0.5096	0.547	0.5344
Lung-cancer	0.5281	0.5578	0.5953	0.5203	0.5484	0.5563	0.6	0.4969
Ptn	0.5383	0.6444	0.5476	0.5388	0.5456	0.542	0.6	0.5872
Mean	0.1781	0.1817	**0.1736**	0.2033	0.1886	0.1885	0.1688	0.1890
Mean Rank	3.7465	4.5273	**3.5000**	5.7328	4.2534	4.2808	4.3356	5.6232

Table 20: Error

	A1DE	**LR**	**LR-Reg**	**NB**	**WANBIA^{CLL}**	**WANBIA^{MSE}**	**RF100**	**TAN**
Pioneer	0.0086	0.0156	0.0140	0.0102	0.0025	0.0017	0.0361	0.0129
Mush	0.0136	0.007	0.007	0.14	0.0403	0.0379	0.009	0.0167
Shuttle	0.0167	0.0111	0.0107	0.0309	0.0178	0.0169	0.0142	0.0154
Syncon	0.0299	0.0526	0.0551	0.0632	0.0554	0.0578	0.1145	0.0557
Pendigits	0.0556	0.0888	0.0672	0.1418	0.1256	0.1248	0.0979	0.0802
Hypo	0.0713	0.0549	0.0473	0.0775	0.0642	0.0637	0.0715	0.0885
Dermatology	0.0722	0.0931	0.0954	0.0713	0.069	0.0723	0.1303	0.1016
Anneal	0.0725	0.0609	0.0537	0.0958	0.0735	0.0741	0.0691	0.0797
Optdigits	0.0747	0.0999	0.0787	0.1159	0.099	0.1007	0.1494	0.0906
Letter-recog	0.0754	0.1001	0.0914	0.1193	0.1136	0.1135	0.0896	0.0916
Thyroid	0.0759	0.0754	0.0706	0.097	0.0867	0.0854	0.077	0.0805
Phoneme	0.0881	0.0998	0.0782	0.0951	0.086	0.0865	0.0731	0.0986
Wine	0.0917	0.1042	0.1251	0.0819	0.0656	0.0736	0.1311	0.1105
Segment	0.0957	0.1234	0.1017	0.1357	0.1032	0.1045	0.1061	0.1097
Page-blocks	0.0987	0.1116	0.0986	0.1431	0.1038	0.1027	0.0974	0.1165
Zoo	0.1171	0.1333	0.1279	0.1247	0.1253	0.1264	0.1279	0.1381
New-thyroid	0.14	0.1797	0.1741	0.1327	0.1411	0.1425	0.156	0.1577
Splice-c4.5	0.1435	0.212	0.1955	0.1536	0.1462	0.1447	0.2599	0.176
Wall-following	0.1461	0.0734	0.0598	0.2081	0.0901	0.0897	0.1206	0.1762
Audio	0.1484	0.146	0.13	0.1486	0.1354	0.135	0.1361	0.1414
Sick	0.1551	0.1483	0.1445	0.1681	0.1458	0.1452	0.1487	0.1499
Nursery	0.1583	0.1464	0.1464	0.1771	0.1771	0.1771	0.101	0.1425
Iris	0.1628	0.19	0.2162	0.165	0.1609	0.1637	0.1813	0.1678
Vowel	0.169	0.2057	0.1831	0.2206	0.2116	0.2123	0.1581	0.1886
Bcw	0.1766	0.2143	0.1844	0.1586	0.1649	0.1737	0.1796	0.1996
Satellite	0.1776	0.2449	0.1685	0.2374	0.1926	0.1916	0.1682	0.1838
Labor	0.1792	0.2567	0.2682	0.1961	0.2082	0.2143	0.2824	0.2481
Ptn	0.1816	0.2329	0.1809	0.1824	0.1811	0.1811	0.1899	0.1829
Led	0.1995	0.2025	0.2067	0.1988	0.199	0.1991	0.2091	0.2034
Musk2	0.2041	0.14	0.11	0.2766	0.1733	0.1786	0.1752	0.2169
House-votes-84	0.207	0.2191	0.1926	0.2987	0.185	0.1873	0.1846	0.2253
Car	0.2085	0.1655	0.1628	0.2293	0.2293	0.2293	0.1782	0.1849
Localization	0.2091	0.233	0.233	0.2386	0.2381	0.2381	0.1939	0.2095
Covtype-mod	0.2183	0.2259	0.2259	0.2494	0.239	0.2389		0.2243
Autos	0.22	0.244	0.2202	0.2512	0.2281	0.2284	0.2041	0.2371
Poker-hand	0.2217	0.2382	0.2382	0.2382	0.2382	0.2382		0.2124
Spambase	0.2323	0.2182	0.2115	0.2949	0.22	0.22	0.2215	0.2396
Yeast	0.2333	0.2348	0.2343	0.2341	0.2338	0.2338	0.2421	0.2353
Ionosphere	0.2529	0.29	0.2649	0.2868	0.2533	0.2595	0.2403	0.2651
Lyn	0.2542	0.3101	0.2729	0.2585	0.2539	0.2551	0.2701	0.2886
Waveform-5000	0.2586	0.2647	0.2671	0.3274	0.2697	0.2698	0.3036	0.2941
Kr-vs-kp	0.2715	0.1533	0.1547	0.3049	0.269	0.2673	0.1268	0.2383
Musk1	0.2731	0.2926	0.2568	0.3468	0.2263	0.2292	0.262	0.2515
Census-income	0.278	0.1807	0.1807	0.4599	0.1867	0.1866	0.1928	0.2083
Glass3	0.294	0.3453	0.3138	0.3118	0.3032	0.3061	0.3146	0.3034
Vehicle	0.3046	0.3235	0.2989	0.3867	0.3238	0.3239	0.3016	0.3045
Chess	0.3081	0.3121	0.3017	0.3143	0.3113	0.3113	0.2771	0.2787
Balance-scale	0.3229	0.3128	0.3672	0.3276	0.3276	0.3276	0.3181	0.3242
Adult	0.3247	0.2967	0.2967	0.3405	0.3	0.2999	0.3274	0.3091
Volcanoes	0.326	0.326	0.326	0.326	0.326	0.326	0.326	0.326
Crx	0.3353	0.3483	0.3319	0.3414	0.3194	0.3213	0.3437	0.354
Sonar	0.3367	0.4163	0.3586	0.3507	0.3365	0.3364	0.3518	0.3366
Connect-4	0.3382	0.3361	0.3361	0.3592	0.3559	0.3559	0.3057	0.3322
Magic	0.3491	0.337	0.3372	0.3916	0.3568	0.3567	0.3571	0.3425
Hungarian	0.3503	0.3781	0.3678	0.3659	0.3428	0.3436	0.369	0.3441
Sign	0.3508	0.3731	0.3734	0.3968	0.3899	0.3897	0.3104	0.3499
Cleveland	0.354	0.3729	0.3606	0.3642	0.353	0.355	0.3696	0.372
Hepatitis	0.3544	0.4358	0.3725	0.3589	0.3353	0.3368	0.3375	0.359
Pid	0.3907	0.3888	0.3896	0.3949	0.3884	0.3888	0.4247	0.3911
Promoters	0.3948	0.3271	0.3281	0.333	0.3277	0.3253	0.3983	0.444
Contact-lenses	0.395	0.4356	0.431	0.3846	0.3845	0.3853	0.4098	0.4477
Ttt	0.4037	0.1385	0.1293	0.4336	0.4262	0.4254	0.2916	0.4098
Horse-colic	0.4115	0.5142	0.3703	0.4227	0.3577	0.36	0.3762	0.4219
German	0.4168	0.4234	0.4145	0.4202	0.4151	0.4156	0.4211	0.4469
Abalone	0.4198	0.4208	0.4368	0.4641	0.4206	0.4204	0.4539	0.4268
Haberman	0.4212	0.4213	0.4248	0.4212	0.4212	0.4212	0.4214	0.4213
Post-operative	0.4281	0.4777	0.4085	0.4233	0.4191	0.4174	0.4399	0.4204
Cmc	0.4349	0.4288	0.4289	0.4463	0.4312	0.4309	0.4739	0.4358
Cylinder-bands	0.4451	0.4831	0.4161	0.4661	0.4587	0.4621	0.4157	0.4794
Echocardiogram	0.4506	0.467	0.4553	0.4491	0.4459	0.4461	0.4574	0.4627
Bupa	0.4863	0.4863	0.4878	0.4863	0.4863	0.4863	0.4862	0.4861
Tae	0.5093	0.553	0.5023	0.5135	0.5075	0.4979	0.4939	0.4857
Lung-cancer	0.5698	0.5945	0.5689	0.564	0.5707	0.5711	0.4732	0.5033
Mean	**0.2461**	0.2544	0.2403	0.2705	0.2462	0.2468	0.2469	0.2528
Mean Rank	**3.6712**	5.1301	3.7397	6.0684	3.8082	4.0342	4.4657	5.0821

Table 21: RMSE

	A1DE	LR	LR-Reg	NB	WANBIA^{CLL}	WANBIA^{MSE}	RF100	TAN
Mush	0.0002	0	0	0.023	0.001	0.001	0	0.0001
Pioneer	0.0003	0.0024	0.0028	0.0011	0	0	0	0.001
Shuttle	0.0007	0.0002	0.0003	0.003	0.0011	0.0009	0.0006	0.0006
Syncon	0.002	0.005	0.0054	0.0104	0.0068	0.007	0.008	0.0055
Hypo	0.0084	0.0029	0.0026	0.0101	0.0061	0.0061	0.0083	0.012
Dermatology	0.0104	0.0135	0.0168	0.0108	0.0071	0.0082	0.019	0.013
Wine	0.012	0.0115	0.0132	0.0118	0.003	0.0039	0.01	0.0102
Anneal	0.0124	0.005	0.0042	0.0256	0.0135	0.012	0.006	0.0137
Pendigits	0.0133	0.0167	0.0162	0.1081	0.0939	0.0915	0.0216	0.0296
Labor	0.0168	0.0318	0.039	0.0167	0.0175	0.0167	0.0409	0.0201
Page-blocks	0.0231	0.0228	0.0244	0.0525	0.0327	0.0304	0.0217	0.0336
Optdigits	0.0233	0.0237	0.0232	0.0666	0.0513	0.0503	0.0294	0.0348
Sick	0.0237	0.0224	0.0219	0.0284	0.0227	0.0231	0.0194	0.023
Segment	0.0253	0.0289	0.0284	0.0633	0.0396	0.0391	0.0253	0.0334
Musk2	0.0265	0.0101	0.0065	0.0718	0.0299	0.0274	0.028	0.0386
Bcw	0.0274	0.0284	0.0301	0.0249	0.0269	0.0275	0.0301	0.0254
Wall-following	0.0285	0.0059	0.004	0.0854	0.0165	0.0156	0.0122	0.0499
New-thyroid	0.0303	0.0298	0.0306	0.0295	0.0273	0.0283	0.0285	0.0268
Splice-c4.5	0.0307	0.0362	0.0326	0.0382	0.0316	0.0294	0.0272	0.038
Zoo	0.0334	0.0332	0.0321	0.0394	0.0363	0.0384	0.0356	0.0468
House-votes-84	0.0466	0.0271	0.0262	0.0913	0.0364	0.0358	0.0327	0.0444
Iris	0.0466	0.0401	0.0442	0.0503	0.0402	0.0412	0.0398	0.0464
Musk1	0.0578	0.0463	0.0431	0.1165	0.0394	0.0399	0.0328	0.0543
Ionosphere	0.0579	0.057	0.056	0.0807	0.0592	0.0542	0.0624	0.0647
Thyroid	0.0595	0.0434	0.0453	0.0979	0.0944	0.0851	0.0516	0.0634
Spambase	0.0601	0.0452	0.0482	0.0949	0.0551	0.0532	0.0432	0.0588
Car	0.0605	0.0523	0.0519	0.1076	0.1076	0.1076	0.0389	0.0474
Nursery	0.0656	0.0684	0.0682	0.0904	0.0904	0.0904	0.0086	0.0543
Letter-recog	0.0684	0.0967	0.1013	0.2196	0.2134	0.2122	0.049	0.1041
Kr-vs-kp	0.0716	0.0192	0.0197	0.1067	0.0567	0.0519	0.0063	0.0619
Satellite	0.0831	0.0855	0.09	0.1684	0.1428	0.1386	0.0874	0.09
Census-income	0.0862	0.041	0.041	0.2319	0.0454	0.0453	0.0416	0.052
Promoters	0.0872	0.0585	0.0613	0.0683	0.0604	0.0599	0.0552	0.0773
Chess	0.0943	0.0772	0.0813	0.0989	0.0974	0.0955	0.0548	0.0551
Vowel	0.0985	0.1	0.1121	0.2287	0.222	0.2224	0.0756	0.1069
Autos	0.1164	0.1087	0.1068	0.1791	0.1427	0.1384	0.1111	0.1464
Waveform-5000	0.1176	0.1112	0.1134	0.1828	0.1404	0.1403	0.1114	0.1212
Sonar	0.1179	0.101	0.1016	0.1314	0.1137	0.1094	0.1045	0.1044
Crx	0.1206	0.1103	0.1079	0.1253	0.1108	0.1106	0.117	0.1185
Lyn	0.1234	0.1155	0.1183	0.1257	0.1169	0.1132	0.1288	0.1232
Hepatitis	0.1242	0.1007	0.1012	0.1294	0.0958	0.095	0.1071	0.1112
Glass3	0.1378	0.1289	0.1226	0.1597	0.1424	0.1425	0.1348	0.1269
Adult	0.1401	0.1207	0.1208	0.1544	0.127	0.1267	0.1109	0.1263
Hungarian	0.1426	0.1291	0.1354	0.1487	0.1373	0.1361	0.1346	0.1166
Phoneme	0.1465	0.1264	0.1264	0.1965	0.1758	0.1763	0.1102	0.1877
Cleveland	0.1505	0.1357	0.1438	0.1548	0.1423	0.1406	0.1304	0.1416
Cylinder-bands	0.1522	0.1464	0.1464	0.1814	0.142	0.1456	0.208	0.2912
Magic	0.1605	0.1446	0.1443	0.2115	0.1656	0.1651	0.1244	0.147
Horse-colic	0.1619	0.1403	0.1344	0.182	0.1275	0.1293	0.1345	0.1452
Balance-scale	0.1721	0.17	0.1665	0.1633	0.1633	0.1633	0.1731	0.1707
Pid	0.1979	0.1895	0.1872	0.2047	0.1873	0.1886	0.1802	0.1816
Ttt	0.1996	0.0171	0.0162	0.2493	0.2354	0.2349	0.027	0.1701
Vehicle	0.1998	0.1765	0.1821	0.3066	0.2463	0.2446	0.1827	0.196
German	0.1998	0.1932	0.1953	0.2101	0.2058	0.2053	0.197	0.1917
Contact-lenses	0.209	0.158	0.2094	0.2069	0.2015	0.2042	0.1748	0.3408
Haberman	0.2106	0.2124	0.2206	0.2106	0.2106	0.2106	0.2107	0.2107
Audio	0.219	0.1478	0.1424	0.2185	0.1758	0.1744	0.173	0.1662
Covtype-mod	0.2208	0.2474	0.2474	0.3034	0.2867	0.2858		0.2299
Connect-4	0.225	0.2346	0.2346	0.2643	0.2628	0.2627	0.1427	0.2226
Led	0.2278	0.2269	0.2245	0.2262	0.2264	0.2265	0.2278	0.2257
Echocardiogram	0.2447	0.2372	0.239	0.2578	0.235	0.2329	0.2256	0.2356
Sign	0.257	0.2927	0.2924	0.3432	0.3411	0.3388	0.154	0.2505
Post-operative	0.2915	0.2715	0.2848	0.2805	0.2792	0.2766	0.3007	0.2685
Localization	0.3179	0.4368	0.4368	0.4717	0.4693	0.4698	0.2047	0.3097
Tae	0.3305	0.3306	0.3311	0.3649	0.3636	0.3583	0.3315	0.3385
Volcanoes	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309
Bupa	0.3396	0.3396	0.3146	0.3396	0.3396	0.3396	0.3451	0.3396
Abalone	0.3425	0.3457	0.3406	0.4201	0.3795	0.3785	0.3257	0.3361
Lung-cancer	0.3473	0.3714	0.384	0.352	0.3495	0.3525	0.3804	0.2834
Yeast	0.3672	0.3651	0.3655	0.3745	0.373	0.3722	0.336	0.3495
Ptn	0.3826	0.3667	0.3681	0.384	0.386	0.3791	0.3876	0.3708
Cmc	0.3907	0.3689	0.3677	0.4237	0.3941	0.3908	0.3383	0.3425
Poker-hand	0.4423	0.4988	0.4988	0.4988	0.4988	0.4988		0.2356
Mean	0.1366	0.1293	0.1305	0.1677	0.1486	0.1477	**0.1151**	0.1334
Mean Rank	4.7397	3.3561	3.5958	7.0136	5.0273	4.5342	**3.3424**	4.3901

Table 22: Bias

	A1DE	LR	LR-Reg	NB	WANBIA^{CLL}	WANBIA^{MSE}	RF100	TAN
Volcanoes	0	0	0	0	0	0	0	0
Mush	0.0001	0	0	0.0031	0.0002	0	0	0.0003
Shuttle	0.0005	0.0002	0.0002	0.001	0.0003	0.0003	0.0003	0.0004
Syncon	0.0013	0.0051	0.0054	0.0029	0.0047	0.0048	0.0047	0.006
Pioneer	0.0022	0.0055	0.002	0.0024	0.0002	0.0001	0.0007	0.0046
Sick	0.0026	0.0045	0.0037	0.0028	0.0036	0.0032	0.0069	0.0041
Hypo	0.0031	0.0033	0.0023	0.0045	0.0021	0.0021	0.0039	0.0067
Wine	0.0054	0.007	0.0068	0.0023	0.0049	0.0073	0.0111	0.0123
Spambase	0.0061	0.0174	0.0106	0.003	0.0075	0.0079	0.0143	0.0101
Anneal	0.0063	0.0067	0.0051	0.0105	0.0064	0.007	0.0062	0.0129
Pendigits	0.0063	0.0247	0.0126	0.0097	0.0099	0.0103	0.0124	0.0114
Page-blocks	0.0074	0.014	0.0073	0.0083	0.0046	0.0047	0.0092	0.0076
Nursery	0.0089	0.0063	0.0065	0.0074	0.0074	0.0074	0.0162	0.0143
House-votes-84	0.0089	0.0222	0.0199	0.0063	0.0088	0.0106	0.0089	0.0206
Adult	0.009	0.0066	0.0066	0.0045	0.0036	0.0036	0.0357	0.0124
Magic	0.0091	0.0092	0.0094	0.0054	0.0064	0.0065	0.043	0.0149
Optdigits	0.0096	0.0297	0.0164	0.0097	0.0128	0.0155	0.0165	0.0139
Bcw	0.0097	0.0187	0.0109	0.0017	0.0074	0.0093	0.0085	0.0259
Splice-c4.5	0.0097	0.0331	0.0125	0.0081	0.0095	0.0106	0.0217	0.0234
Dermatology	0.0109	0.0153	0.0142	0.0093	0.0119	0.0128	0.0178	0.0305
Iris	0.011	0.0169	0.0131	0.0051	0.0128	0.0131	0.0162	0.0123
New-thyroid	0.012	0.0216	0.0222	0.0079	0.0124	0.0129	0.0194	0.0246
Ionosphere	0.0122	0.0299	0.0258	0.0061	0.0187	0.0205	0.0142	0.0134
Census-income	0.0124	0.0023	0.0023	0.0036	0.0008	0.0008	0.0078	0.0055
Segment	0.0135	0.0261	0.0195	0.0119	0.0107	0.0115	0.016	0.0194
Thyroid	0.0157	0.0249	0.0189	0.0137	0.0117	0.0143	0.0234	0.0221
Hungarian	0.0166	0.052	0.0455	0.0098	0.0327	0.0331	0.0528	0.0494
Musk2	0.0173	0.0106	0.0078	0.0066	0.0099	0.0126	0.0105	0.0108
Connect-4	0.0189	0.0079	0.0079	0.0149	0.0098	0.0099	0.0448	0.0142
Wall-following	0.0194	0.0059	0.0045	0.0103	0.0054	0.005	0.0094	0.0194
Kr-vs-kp	0.0199	0.0085	0.009	0.02	0.0129	0.0103	0.0065	0.0153
Covtype-mod	0.0205	0.0097	0.0097	0.0082	0.0045	0.0049		0.0213
Cleveland	0.0205	0.0506	0.0327	0.0145	0.0341	0.0384	0.0603	0.0493
Pid	0.0214	0.0302	0.0326	0.0168	0.0278	0.0266	0.0734	0.0432
Poker-hand	0.022	0	0	0	0	0		0.0935
Sign	0.022	0.0273	0.028	0.0161	0.0178	0.0179	0.0498	0.0242
Crx	0.0221	0.0462	0.0317	0.0196	0.0272	0.0297	0.0411	0.0467
Waveform-5000	0.0251	0.0358	0.0253	0.009	0.0152	0.0162	0.0443	0.0613
Satellite	0.0261	0.0952	0.0275	0.0067	0.0124	0.0129	0.0211	0.0279
Sonar	0.0261	0.0773	0.0621	0.0206	0.055	0.0577	0.066	0.0639
Musk1	0.0291	0.0414	0.0379	0.016	0.0255	0.0267	0.0355	0.0219
Zoo	0.0295	0.0331	0.0362	0.0319	0.036	0.0374	0.0387	0.0463
Labor	0.0297	0.0419	0.0426	0.0377	0.0369	0.0394	0.053	0.0641
Letter-recog	0.0341	0.0672	0.0482	0.0367	0.035	0.0353	0.0413	0.0469
Glass3	0.0346	0.0718	0.055	0.0275	0.0333	0.0353	0.0603	0.0576
Led	0.0372	0.0425	0.0414	0.037	0.0369	0.0383	0.0523	0.0445
Chess	0.0375	0.0461	0.043	0.0374	0.0363	0.0366	0.0526	0.0509
Hepatitis	0.0378	0.0915	0.0688	0.028	0.0493	0.0541	0.0535	0.0601
Localization	0.041	0.0217	0.0216	0.0222	0.0209	0.0202	0.0929	0.0467
Yeast	0.0413	0.0413	0.0404	0.037	0.0338	0.0362	0.0849	0.0602
Lyn	0.0435	0.0831	0.0675	0.0408	0.0632	0.0665	0.0621	0.1025
Bupa	0.0448	0.0448	0.0821	0.0448	0.0448	0.0448	0.0366	0.0448
Car	0.0464	0.0219	0.0214	0.0484	0.0484	0.0484	0.0383	0.0336
Horse-colic	0.0488	0.1322	0.0453	0.0307	0.0347	0.0366	0.0445	0.0785
Ttt	0.0506	0.0075	0.0018	0.0409	0.0377	0.0365	0.0495	0.0783
German	0.0537	0.0643	0.0573	0.0431	0.0511	0.0517	0.0714	0.0921
Haberman	0.0608	0.0584	0.0503	0.0608	0.0608	0.0608	0.0602	0.0615
Vehicle	0.0763	0.1079	0.0901	0.0699	0.0826	0.0842	0.0915	0.0803
Autos	0.0819	0.1067	0.0991	0.0763	0.0873	0.0918	0.0825	0.0973
Post-operative	0.0835	0.1535	0.0158	0.0923	0.0819	0.0761	0.0965	0.1021
Cylinder-bands	0.0848	0.0936	0.0813	0.0745	0.117	0.1203	0.0622	0.0849
Echocardiogram	0.0881	0.0983	0.0961	0.0628	0.0948	0.0973	0.1233	0.1121
Cmc	0.0885	0.0781	0.0801	0.0591	0.0754	0.0747	0.1593	0.1225
Balance-scale	0.0961	0.0955	0.0962	0.0962	0.0962	0.0962	0.0979	0.0954
Audio	0.1097	0.1199	0.0972	0.112	0.1162	0.1199	0.1279	0.1699
Promoters	0.1113	0.0656	0.0689	0.0704	0.0755	0.0764	0.0967	0.1552
Abalone	0.1136	0.1166	0.125	0.0594	0.0852	0.0858	0.1566	0.1327
Phoneme	0.1165	0.128	0.0804	0.107	0.0829	0.0844	0.0687	0.1606
Vowel	0.1214	0.1413	0.125	0.1643	0.1397	0.1422	0.0918	0.1712
Contact-lenses	0.1473	0.1378	0.151	0.1368	0.1401	0.1417	0.169	0.0884
Ptn	0.1557	0.2777	0.1796	0.1548	0.1595	0.1629	0.2124	0.2164
Lung-cancer	0.1808	0.1864	0.2113	0.1683	0.1989	0.2037	0.2196	0.2135
Tae	0.184	0.2045	0.2024	0.1533	0.1552	0.1513	0.2156	0.196
Mean	0.0415	0.0525	0.0430	**0.0357**	0.0400	0.0409	0.0537	0.0556
Mean Rank	3.8150	5.7054	4.6712	**2.8082**	3.0821	3.9520	5.6917	6.2739

Table 23: Variance

	A1DE	LR	LR-Reg	NB	WANBIA^{CLL}	WANBIA^{MSE}	RF100	TAN
Contact-lenses	0.075	0.117	13.765	0.067	0.063	0.074	0.621	0.071
Haberman	0.098	0.114	2.67	0.088	0.08	0.077	0.655	0.096
Tae	0.098	0.895	39.861	0.086	0.111	0.131	0.965	0.097
Echocardiogram	0.098	0.198	7.756	0.086	0.108	0.116	0.733	0.092
New-thyroid	0.099	0.2	13.392	0.093	0.121	0.132	0.679	0.1
Post-operative	0.099	0.294	14.513	0.083	0.111	0.106	0.762	0.101
Bupa	0.099	0.112	3.486	0.097	0.084	0.087	0.693	0.101
Iris	0.099	0.166	6.629	0.088	0.115	0.123	0.661	0.099
Wine	0.1	0.231	10.128	0.096	0.189	0.236	0.735	0.108
Labor	0.1	0.166	16.176	0.085	0.132	0.154	0.663	0.105
Glass3	0.1	0.556	48.909	0.089	0.159	0.178	1.018	0.108
Zoo	0.104	0.368	17.779	0.088	0.193	0.268	0.737	0.124
Cleveland	0.106	0.662	41.424	0.105	0.2	0.234	1.258	0.119
Hungarian	0.107	0.94	67.985	0.1	0.269	0.319	1.233	0.118
Balance-scale	0.108	0.232	10.895	0.1	0.091	0.086	1.072	0.115
Hepatitis	0.111	0.335	17.759	0.095	0.2	0.231	0.843	0.114
Led	0.113	5.203	325.834	0.105	0.341	0.444	2.831	0.13
House-votes-84	0.115	0.341	16.883	0.103	0.246	0.329	1.039	0.129
Pid	0.116	0.253	14.32	0.107	0.225	0.23	2.152	0.118
Volcanoes	0.12	2.48	109.075	0.114	0.111	0.116	1.045	0.123
Yeast	0.121	33.068	1091.32	0.111	0.598	0.714	3.914	0.14
Lyn	0.121	0.407	24.48	0.094	0.262	0.345	0.883	0.118
Ttt	0.121	0.577	62.348	0.113	0.202	0.203	2.26	0.127
Horse-colic	0.124	1.581	78.819	0.108	0.376	0.385	1.373	0.147
Crx	0.125	4.24	171.602	0.108	0.668	0.637	2.097	0.145
Car	0.128	6.549	469.286	0.117	0.123	0.127	2.556	0.137
Cmc	0.129	6.639	243.93	0.117	0.535	0.579	5.766	0.139
Ptn	0.131	181.979	3127.046	0.094	0.965	1.278	3.086	0.144
Bcw	0.132	0.423	20.418	0.106	0.312	0.398	1.059	0.138
Vowel	0.137	18.835	1901.16	0.114	0.921	1.313	3.79	0.164
Vehicle	0.14	37.456	1051.993	0.115	0.729	0.962	3.98	0.167
German	0.145	3.33	326.224	0.12	0.556	0.559	3.468	0.168
Autos	0.156	1.821	212.885	0.096	0.675	0.93	1.22	0.175
Ionosphere	0.16	0.611	36.326	0.113	0.363	1.045	1.151	0.218
Chess	0.161	5.658	308.214	0.123	0.65	0.756	3.069	0.212
Sonar	0.162	1.51	150.955	0.12	0.356	0.429	1.794	0.227
Abalone	0.168	16.021	883.628	0.146	1.953	1.972	14.309	0.195
Dermatology	0.183	1.792	168.475	0.111	1.84	3.228	1.403	0.243
Promoters	0.192	0.616	24.846	0.106	0.395	0.411	0.86	0.31
Lung-cancer	0.192	1.035	65.012	0.098	0.311	0.246	0.772	0.214
Page-blocks	0.22	77.192	3062.031	0.166	3.405	3.923	6.038	0.233
Anneal	0.235	4.306	555.919	0.124	7.262	10.031	2.345	0.294
Segment	0.256	11.027	1455.738	0.136	4.472	5.516	4.545	0.347
Nursery	0.268	43.87	2885.603	0.237	0.476	0.507	17.253	0.307
Sign	0.271	97.373	3309.138	0.257	2.234	3.712	29.188	0.308
Sick	0.274	18.226	1497.482	0.195	13.786	15.394	7.901	0.412
Hypo	0.293	7.452	809.711	0.193	15.659	21.443	5.492	0.415
Kr-vs-kp	0.303	27.203	1483.335	0.195	6.525	8.376	9.458	0.533
Mush	0.349	4.495	587.965	0.235	3.97	7.46	5.825	0.506
Audio	0.379	31.293	6576.503	0.125	4.846	8.245	3.003	0.616
Magic	0.408	84.379	4453.728	0.32	4.507	4.668	56.51	0.48
Pendigits	0.462	37.11	4665.818	0.248	13.162	19.172	25.167	0.735
Syncon	0.466	6.798	816.926	0.135	5.481	6.478	2.129	0.769
Waveform-5000	0.474	107.506	7163.793	0.237	5.051	6.559	19.181	0.917
Thyroid	0.555	1839.199	42695.323	0.273	187.224	272.806	35.287	0.915
Phoneme	0.565	2430.18	78868.508	0.15	12.683	15.977	36.345	0.736
Cylinder-bands	0.623	8.705	1006.751	0.119	1.788	1.828	3.009	0.586
Splice-c4.5	0.664	7.369	963.412	0.229	9.889	14.683	11.431	1.376
Spambase	0.665	106.084	6027.538	0.256	12.773	17.797	17.634	1.417
Musk1	0.707	3.631	837.027	0.188	11.641	23.526	4.597	1.508
Letter-recog	0.788	5570.466	892.459	0.363	73.676	96.965	87.782	1.336
Wall-following	0.876	13.34	2915.697	0.198	4.933	6.254	12.4	1.276
Shuttle	0.993	61.266	5237.291	0.756	37.254	47.905	63.185	1.291
Satellite	1.02	1534.966	9646.135	0.255	16.169	26.975	18.788	2.135
Adult	1.025	720.394	31035.38	0.74	33.376	41.734	199.569	1.465
Optdigits	1.375	50.882	6635.504	0.303	116.538	152.671	28.835	3.203
Localization	1.977	6113.835	226842.296	1.834	92.014	98.267	938.448	2.259
Connect-4	4.695	2218.933	95760.528	1.748	91.768	108.129	691.22	10.542
Covtype-mod	10.856	25589.054	519389.101	7.295	683.568	843.389		15.292
Poker-hand	15.15	32887.718	751710.215	12.815	59.807	66.984		18.964
Musk2	19.881	56.031	19571.881	0.583	54.759	198.24	27.884	36.344
Census-income	20.808	11189.489	438360.993	7.233	1082.365	1129.399	2307.386	50.237
Pioneer	309.15	835.65	8835.12	0.328	658.631	564.602	296.181	671.052
Mean	5.50	1262.10	14865.08	**0.58**	45.85	53.01	71.17	11.43
Mean Rank	2.2941	6.4109	7.9726	**1.1369**	4.0958	5.1095	5.8356	3.1438

Table 24: Train time

	A1DE	LR	LR-Reg	NB	WANBIA^{CLL}	WANBIA^{MSE}	RF100	TAN
Contact-lenses	0.006	0.0015	0.003	0.001	0.003	0.002	0.005	0.005
Post-operative	0.017	0.0075	0.009	0.005	0.004	0.009	0.011	0.015
Labor	0.018	0.0045	0.006	0.007	0.005	0.005	0.006	0.01
Iris	0.02	0.0045	0.015	0.01	0.007	0.006	0.022	0.016
Tae	0.021	0.0075	0.0105	0.008	0.009	0.006	0.021	0.021
New-thyroid	0.023	0.003	0.0135	0.006	0.007	0.009	0.023	0.018
Echocardiogram	0.024	0.0045	0.015	0.006	0.006	0.004	0.022	0.013
Bupa	0.028	0.0075	0.024	0.007	0.01	0.009	0.018	0.018
Hepatitis	0.029	0.006	0.012	0.009	0.009	0.007	0.022	0.019
Glass3	0.029	0.009	0.021	0.009	0.008	0.006	0.039	0.023
Haberman	0.029	0.006	0.0225	0.008	0.008	0.011	0.013	0.019
Wine	0.031	0.006	0.018	0.008	0.006	0.008	0.019	0.018
Zoo	0.033	0.0135	0.0105	0.005	0.004	0.006	0.016	0.014
Lyn	0.033	0.0105	0.015	0.009	0.007	0.008	0.022	0.018
Cleveland	0.036	0.012	0.021	0.012	0.006	0.008	0.05	0.023
Balance-scale	0.036	0.012	0.045	0.01	0.012	0.018	0.087	0.022
Pid	0.039	0.009	0.0285	0.013	0.009	0.011	0.145	0.023
Hungarian	0.039	0.006	0.033	0.009	0.008	0.009	0.053	0.022
Lung-cancer	0.041	0	0.0015	0.002	0.003	0.005	0.005	0.009
Bcw	0.045	0.0105	0.0225	0.013	0.012	0.007	0.06	0.025
Volcanoes	0.048	0.018	0.0495	0.023	0.02	0.02	0.08	0.033
Tt	0.051	0.015	0.0675	0.014	0.014	0.012	0.183	0.027
House-votes-84	0.052	0.012	0.015	0.009	0.011	0.01	0.055	0.021
Autos	0.053	0.0105	0.018	0.015	0.009	0.006	0.036	0.018
Car	0.053	0.024	0.057	0.019	0.022	0.022	0.264	0.033
Promoters	0.057	0.0045	0.012	0.008	0.005	0.003	0.013	0.013
Horse-colic	0.057	0.012	0.018	0.009	0.012	0.012	0.058	0.024
Crx	0.058	0.0165	0.0375	0.014	0.009	0.01	0.12	0.026
Ptn	0.06	0.0165	0.201	0.02	0.011	0.012	0.135	0.035
Cmc	0.061	0.0135	0.0405	0.017	0.012	0.01	0.471	0.035
Led	0.062	0.018	0.054	0.017	0.014	0.016	0.29	0.036
Vowel	0.064	0.027	0.1125	0.022	0.019	0.022	0.298	0.045
Yeast	0.068	0.03	0.081	0.027	0.02	0.02	0.391	0.042
Ionosphere	0.073	0.0105	0.015	0.007	0.01	0.005	0.046	0.021
Dermatology	0.075	0.0105	0.036	0.017	0.01	0.01	0.059	0.026
Vehicle	0.075	0.009	0.1965	0.02	0.016	0.011	0.21	0.036
Sonar	0.079	0.003	0.0255	0.009	0.007	0.008	0.044	0.026
German	0.085	0.0105	0.093	0.018	0.011	0.015	0.218	0.029
Abalone	0.091	0.0345	0.1155	0.026	0.026	0.023	1.793	0.046
Chess	0.101	0.0135	0.045	0.014	0.011	0.011	0.165	0.026
Cylinder-bands	0.103	0.0105	0.033	0.017	0.012	0.011	0.095	0.028
Page-blocks	0.165	0.066	0.2535	0.046	0.041	0.04	0.868	0.081
Segment	0.192	0.039	0.1545	0.035	0.034	0.034	0.446	0.07
Anneal	0.193	0.0225	0.0645	0.024	0.023	0.021	0.147	0.047
Sign	0.256	0.105	0.2835	0.07	0.065	0.071	5.943	0.114
Nursery	0.291	0.138	0.4845	0.099	0.094	0.095	3.54	0.159
Sick	0.307	0.048	0.21	0.038	0.036	0.036	0.943	0.072
Audio	0.343	0.024	0.114	0.024	0.023	0.024	0.087	0.068
Syncon	0.351	0.0195	0.0855	0.023	0.02	0.02	0.092	0.046
Kr-vs-kp	0.359	0.042	0.1365	0.036	0.037	0.037	1.02	0.066
Hypo	0.395	0.0615	0.1905	0.05	0.045	0.05	0.603	0.099
Magic	0.409	0.1455	0.4125	0.096	0.092	0.088	10.413	0.161
Mush	0.439	0.093	0.261	0.063	0.061	0.065	0.788	0.109
Phoneme	0.465	0.3885	1.2345	0.259	0.24	0.245	3.489	0.479
Wall-following	0.716	0.081	0.2115	0.061	0.063	0.064	1.179	0.13
Pendigits	0.808	0.231	0.6675	0.178	0.173	0.173	4.234	0.352
Waveform-5000	0.848	0.078	0.291	0.073	0.064	0.064	1.834	0.133
Musk1	0.929	0.0165	0.0675	0.023	0.021	0.021	0.104	0.049
Splice-c4.5	1.088	0.066	0.192	0.058	0.057	0.059	0.924	0.122
Spambase	1.223	0.0825	0.2685	0.073	0.068	0.066	1.727	0.123
Adult	1.458	0.4635	1.3635	0.318	0.295	0.301	22.379	0.522
Shuttle	1.709	0.831	2.4285	0.561	0.55	0.558	7.409	0.965
Satellite	1.989	0.114	0.39	0.123	0.122	0.122	1.901	0.273
Thyroid	2.261	0.372	0.9375	0.359	0.361	0.357	5.804	0.853
Letter-recog	2.664	0.906	0.918	0.697	0.702	0.699	11.936	1.484
Localization	3.515	3.1875	8.5485	1.947	1.879	1.895	93.887	2.782
Optdigits	5.117	0.2145	0.657	0.24	0.232	0.235	2.463	0.598
Connect-4	12.134	1.2315	2.4525	0.972	0.965	0.978	47.681	1.849
Pioneer	23.738	2.8455	7.3455	1.649	1.453	1.462	7.09	16.145
Covtype-mod	25.162	10.251	10.251	7.245	7.112	7.196		12.344
Musk2	33.621	0.2325	0.714	0.24	0.245	0.245	1.193	0.549
Poker-hand	39.21	21.9015	21.9015	15.605	15.603	15.118		24.526
Census-income	50.673	4.839	10.836	3.92	3.933	3.845	131.887	6.667
Mean	2.945	0.679	1.041	0.489	**0.481**	0.475	5.319	1.000
Mean Rank	7.1438	2.9794	5.9383	2.9315	**2.1027**	2.2191	7.2465	5.4383

Table 25: Test time

B. Cestnik, I. Kononenko, and I. Bratko. Assistant 86: A knowledge-elicitation tool for sophisticated users. In *Proceedings of the Second European Working Session on Learning*, pages 31–45. Wulmslow, UK: Sigma Press, 1987.

B. Damien and G. I. Webb. The need for low bias algorithms in classification learning from small data sets. In *Proceedings of the Sixth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD) 2002*, pages 62–73, Berlin, 2002. Springer-Verlag.

M. H. DeGroot and S. E. Fienbert. The comparison and evaluation of forecasters. *Statistician*, 32 (1):12–22, 1982.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

P. Domingos and M. J. Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 105–112. Morgan Kaufmann, 1996.

R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley and Sons, 2006.

U. M. Fayyad and B. I. Keki. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8(1):87–102, 1992.

J. T. A. S. Ferreira, D. G. T. Denison, and D. J. Hand. Weighted naive Bayes modelling for data mining, 2001.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

J. Friedman. Flexible metric nearest neighbor classification. Technical report, Department of Statistics, Stanford University, 1994.

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *The American Statistical Association*, 32(200):675–701, 1937.

M. Friedman. A comparison of alternative tests of significance for the problem of *m* rankings. *The American Statistical Association*, 11(1):86–92, 1940.

N. Friedman and M. Goldszmidt. Building classifiers using Bayesian networks. In *AAAI-96*, pages 1277–1284, 1996.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29 (2):131–163, 1997.

R. Greiner, W. Zhou, X. Su, and B. Shen. Structural extensions to logistic regression: Discriminative parameter learning of belief net classifiers. *Journal of Machine Learning Research*, 2004.

D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *International Conference on Machine Learning*, 2004.

I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh. *Feature Extraction, Foundation and Applications*. Springer, 2004.

M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 359–366. Morgan Kaufmann, 2000.

M. A. Hall. A decision tree-based attribute weighting filter for naive Bayes. *Knowledge-Based Systems*, 20:120–126, March 2007.

D. J. Hand and K. Yu. Idiot's Bayes - not so stupid after all? *International Statistical Review*, 69 (3):385–398, 2001.

J. Hilden and B. Bjerregaard. Computer-aided diagnosis and the atypical case. In *In Decision Making and Medical Care: Can Information Science Help*, pages 365–378. North-Holland Publishing Company, 1976.

R. Kohavi and D. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 275–283. San Francisco: Morgan Kaufmann, 1996.

I. Kononenko. Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition. *Current Trends in Knowledge Acquisition*, 1990.

P. Langley and S. Sage. Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406. Morgan Kaufmann, 1994.

P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 223–228. AAAI Press and MIT Press, 1992.

D. D. Lewis. Naive Bayes at forty: The independence assumption in information retrieval. In *ECML-98: Proceedings of the Tenth European Conference on Machine Learning*, pages 4–15, Berlin, April 1998. Springer.

A. Murphy and R. Winkler. Reliability of subjective probability forecasts of precipitation and temperature. *Applied Statistics*, 26(1):41–47, 1977.

F. Pernkopf and J. A. Bilmes. Efficient heuristics for discriminative structure learning of Bayesian network classifiers. *Journal of Machine Learning Research*, 2010.

C. A. Ratanamahatana and D. Gunopulos. Feature selection for the naive Bayesian classifier using decision trees. *Applied Artificial Intelligence*, pages 475–487, 2003.

T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri. On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, 59(3):267–296, 2005.

G. I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):159–196, 2000.

G. I. Webb, J. Boughton, and Z. Wang. Not so naive Bayes: Averaged one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005.

G. I. Webb, J. Boughton, F. Zheng, K. M. Ting, and H. Salem. Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive Bayesian classification. *Machine Learning*, pages 1–40, 2011.

J. Wu and Z. Cai. Attribute weighting via differential evolution algorithm for attribute weighted naive Bayes (wnb). *Journal of Computational Information Systems*, 7(5):1672–1679, 2011.

B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *ACM SIGKDD, International Conference on Knowledge Discovery and Data Mining*, 2002.

H. Zhang and S. Sheng. Learning weighted naive Bayes with accurate ranking. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM -04)*, pages 567–570, 2004.

F. Zheng, G. I. Webb, P. Suraweera, and L. Zhu. Subsumption resolution: An efficient and effective technique for semi-naive Bayesian learning. *Machine Learning*, 87(1):93–125, 2012.

Z. Zheng and G.I. Webb. Lazy learning of Bayesian rules. *Machine Learning*, 41(1):53–84, 2000.

Z. Zheng, G. I. Webb, and K.M. Ting. Lazy Bayesian Rules: A lazy semi-naive Bayesian learning technique competitive to boosting decision trees. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-99)*, pages 493–502. Morgan Kaufmann, 1999.

C Zhu, R. H. Byrd, and J. Nocedal. L-bfgs-b: Algorithm 778: L-bfgs-b, fortran routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.