Journal of Machine Learning Research (2008) Submitted ; Published

Semi-naive Bayesian Classification

Fei Zheng FEI.ZHENG@INFOTECH.MONASH.EDU
Clayton School of Information Technology
Monash University
Melbourne, Vic. 3800, Australia

Geoffrey I. Webb GEOFF.WEBB@INFOTECH.MONASH.EDU
Clayton School of Information Technology
Monash University
Melbourne, Vic. 3800, Australia

Editor:

Abstract

The success and popularity of naive Bayes (NB) has led to a field of research exploring algorithms that seek to retain its numerous strengths while reducing error by alleviating the attribute interdependence problem. These algorithms can be categorized into five groups: those that apply conventional NB to a subset of attributes, those that alter NB by allowing interdependencies between attributes, those that apply NB to a subset of the training sample, those that calibrate NB's probability estimates and those that introduce hidden variables to NB. Eighteen key algorithms are analyzed in detail. We provide comparative analysis of thirteen algorithms' features and benchmark them using error analysis based on the bias-variance decomposition and probabilistic prediction analysis based on the quadratic loss function on sixty natural domains from the UCI Machine Learning Repository. To provide a baseline for comparison, we also present comprehensive experimental results for Logistic Regression and LibSVM, a popular SVM implementation. In analyzing the results of these experiments we provide general recommendations for selection between semi-naive Bayesian methods based on the characteristics of the application to which they are applied.

Keywords: Classification, Naive Bayes, the Attribute Independence Assumption, Semi-naive Bayesian Classification

1. Introduction

Given a labeled training sample described by a set of features or attributes, *supervised classification algorithms* seek to build a *classifier* that maps a new instance to be classified (hereafter referred to as a *test instance*) into a discrete class label. In the Bayesian classifier (Duda and Hart, 1973; Lachenbruch, 1975; Aitchison and Dunsmore, 1975), a test instance is assigned the class label with the highest posterior probability. Following the classification of David (1976), approaches to estimating posterior probabilities broadly fall into two categories: firstly, the *diagnostic paradigm*, also called *discriminative classifiers*, where the parameters of the conditional distribution are directly estimated from the training sample by maximizing the conditional likelihood; and secondly, the *sampling paradigm*, also called *generative classifiers*, where the parameters of the joint distribution are estimated from the training sample by maximizing the joint likelihood and those of the conditional distribution are obtained via Bayes theorem. Two typical examples of discriminative and generative classifiers are Logistic Regression and naive Bayes (NB). Logistic Regression directly estimates the posterior probabilities by fitting the training data to a logistic curve, while NB directly estimates the prior and class conditional probabilities and obtains the posterior probabilities by applying Bayes rule. Generally, generative classifiers are easier and more efficient to train with lower variance, while discriminative classifiers tend to have lower bias and hence higher classification accuracy on large training data (Rubinstein and Hastie, 1997; Ng and Jordan, 2001; Mitchell, 2005). This paper examines recent research into variants of the generative learning algorithm NB.

NB is a very simple and computationally efficient learning algorithm that demonstrates remarkably strong accuracy over a range of application domains (Domingos and Pazzani, 1996; Mitchell, 1997; Lewis, 1998; Hand and Yu, 2001). It has many desirable features. One of these is that it performs optimal classification save only in so far as there are

- violations of its assumption that the attributes are independent of one another given the class; and
- inaccuracies in the estimation of the base probabilities from the training data.

Numerous generative techniques have sought to enhance the accuracy of NB by relaxing the assumption of conditional independence between the attributes given the class (Hilden and Bjerregaard, 1976; Kittler, 1986; Kononenko, 1991; Langley, 1993; Langley and Sage, 1994; Kohavi, 1996; Pazzani, 1996; Sahami, 1996; Singh and Provan, 1996; Friedman et al., 1997; Webb and Pazzani, 1998; Keogh and Pazzani, 1999; Zheng and Webb, 2000; Webb, 2001; Xie et al., 2002; Zadrozny and Elkan, 2002; Zhang et al., 2004; Webb et al., 2005; Acid et al., 2005; Cerquides and Mántaras, 2005a,b; Zhang et al., 2005; Zheng and Webb, 2006; Langseth and Nielsen, 2006; Zheng and Webb, 2007). This paper examines these methods which we call *semi-naive Bayesian methods*. We provide a broad classification of the methods into five groups. Of these, we examine seventeen representative semi-naive Bayesian methods, with detailed time and space complexity analysis. To gain a better understanding of the strengths and limitations of these algorithms, we perform error analysis based on the bias-variance decomposition (Kohavi and Wolpert, 1996), probabilistic prediction analysis based on the quadratic loss function and training and classification time analysis on sixty natural domains from the UCI Machine Learning Repository (Newman et al., 1998).

There are seven main differences between this work and the earlier comparative study of Zheng and Webb (2005). This paper compares thirteen semi-naive Bayesian methods, including recent methods, on sixty data sets using the repeated cross-validation bias-variance estimation method proposed by Webb (2000), while the earlier paper compares eight semi-naive Bayesian methods on thirty-four data sets with a smaller average size using the bias-variance decomposition method proposed by Kohavi and Wolpert (1996). Because the repeated cross-validation bias-variance estimation method results in the use of substantially larger training sets, we believe it is preferable to the method of Kohavi and Wolpert. To provide a baseline for comparison, this study also presents experimental results for Logistic Regression and LibSVM. In addition, this work employs m-estimation to estimate probabilities, rather than Laplace estimation, and evaluates the probabilistic prediction of each method by using the quadratic loss function. Furthermore, the Friedman test and Nemenyi test are employed to analyze error, bias, variance and root mean squared error as they might be more appropriate than large numbers of binomial sign tests for comparison of multiple algorithms over multiple data sets. Due to these differences, the results of this paper and the earlier paper differ slightly, those of the current study being indicative of the use of larger training sets.

The rest of the paper is organized as follows. Section 2 reviews classification with NB. Section 3 describes generic strategies for relaxing the independence assumption and provide details of seventeen representative semi-naive Bayesian methods, including algorithm description and complexity analysis. Logistic Regression and its extensions are briefly described in Section 4. Section 5 contains the experimental results of NB, twelve semi-naive Bayesian methods, Logistic Regression and LibSVM on a wide range of domains and general guidelines for selection between semi-naive Bayesian methods. Conclusions are provided in Section 6. Finally, the Appendix presents detailed error, bias, variance and root mean squared error results.

2. Naive Bayes: A Generative Classifier

Supervised classification learning is the process of predicting a discrete class label $y \in \{c_1, \dots, c_k\}$ for a test instance $\mathbf{x} = \langle x_1, \dots, x_n \rangle$ from a labelled training sample, where x_i is the value of the *ith* attribute X_i and c_i is the *ith* value of the class variable Y. The Bayesian classifier (Duda and Hart, 1973; Lachenbruch, 1975; Aitchison and Dunsmore, 1975) predicts the class label for $\mathbf{x}$ by selecting

$$\begin{aligned} \operatorname*{argmax}_y P(y \,|\, \mathbf{x}) &= \operatorname*{argmax}_y P(y, \mathbf{x})/P(\mathbf{x}) && (1) \\ &= \operatorname*{argmax}_y P(y, \mathbf{x}). && (2) \end{aligned}$$

The equality holds between (1) and (2) due to $P(\mathbf{x})$ being invariant across values of y. Where estimates of $P(y \,|\, \mathbf{x})$ are required rather than a simple classification, these can be obtained by normalization,

$$\hat{P}(y \,|\, \mathbf{x}) = \frac{\hat{P}(y, \mathbf{x})}{\sum_{i=1}^{k} \hat{P}(c_i, \mathbf{x})}, \qquad (3)$$

where $\hat{P}(\cdot)$ represents an estimate of $P(\cdot)$.

Let $\mathbf{X} = \langle X_1, \dots, X_n \rangle$ be the instance variable and v_i the cardinality of the domain of X_i, the number of values that is may assume. There are $\prod_{i=1}^{n} v_i$ combinations of attribute values. To accurately estimate $P(Y, \mathbf{X}) = P(Y)P(\mathbf{X} \,|\, Y)$, we need to estimate $k(\prod_{i=1}^{n} v_i - 1)$ parameters, each requiring sufficient examples to support reliable estimation (Mitchell, 2005). It is clearly unrealistic to do this directly in most real world domains as, if the number of attributes is large, some instances are unlikely to occur in the given training data, and hence it is impossible to obtain the estimate of $P(Y, \mathbf{X})$ directly from the training sample.

Naive Bayes (NB) (Ohmann et al., 1988; Kononenko, 1990; Langley et al., 1992; Langley and Sage, 1994) circumvents this problem by making the assumption that the attributes are independent given the class. Consequently, the number of parameters to estimate $P(Y, \mathbf{X})$ is dramatically reduced to $k \sum_{i=1}^{n}(v_i - 1)$. Under this conditional independence assumption NB estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y) \prod_{i \in N} \hat{P}(x_i \mid y), \tag{4}$$

where $N = \{1, \cdots, n\}$. For the history of NB, we refer the reader to Hand and Yu (2001), which provides a comprehensive history of NB with many further references.

The joint likelihood of training data is maximized by directly estimating probabilities using frequency counts over the training data. In NB, the class Y is qualitative and an attribute X_i can be either qualitative or quantitative. For qualitative attributes, $P(y)$ is estimated by the frequency of instances with value y (indicated as $F(y)$), and $P(x_i \mid y)$ is estimated by the frequency of instances with y and x_i (indicated as $F(y, x_i)$) divided by that of instances with y. To avoid the problems that result from zero frequencies and zero probabilities, smoothing methods, such as *Laplace estimation* and *m-estimation* (Cestnik, 1990), are employed. Using Laplace estimation, we have

$$\hat{P}(y) = \frac{F(y) + 1}{t + k}$$

and

$$\hat{P}(x_i \mid y) = \frac{F(y, x_i) + 1}{F(y) + v_i},$$

where t is the number of training examples. Using m-estimation, we have

$$\hat{P}(y) = \frac{F(y) + \frac{m}{k}}{t + m}$$

and

$$\hat{P}(x_i \mid y) = \frac{F(y, x_i) + \frac{m}{v_i}}{F(y) + m},$$

where m is a constant.

For quantitative attributes, one common approach to representing the distributions $P(X_i \mid Y)$ is to assume that X_i has a Gaussian distribution whose mean and variance depends on Y. Previous research (Dougherty, Kohavi, and Sahami, 1995) shows that the classification errors of NB with discretization methods employed in their study are lower than that of NB with the assumption that quantitative attributes have a Gaussian distribution. Theoretical analysis on why discretization is effective on NB can be found in Hsu et al. (2000, 2003b) and Yang and Webb (2003). For this reason, in this paper, quantitative attributes are discretized prior to classification with NB and its extensions.

At training time, NB generates a one-dimensional table of class probability estimates, indexed by class, and a two-dimensional table of conditional attribute-value probability estimates, indexed by class and attribute-value. The resulting space complexity is $O(knv)$, where v is the mean number of values per attribute. The time complexity of initializing the tables is $O(knv)$ and that of calculating the estimates is $O(tn)$. Since t is usually substantially greater than k and v, the overall training time complexity is $O(tn)$. In the following paper, if the time complexity of table initialization is lower than that of other operations in most cases, we only discuss the latter. At classification time, to classify a single example has time complexity $O(kn)$ using the tables formed at training time with space complexity $O(knv)$.

3. Semi-naive Bayesian methods

In practical scenarios, the attribute independence assumption is often violated. There have been many attempts to further improve NB's accuracy by alleviating the attribute interdependence problem while at the same time retaining its simplicity and efficiency. Domingos and Pazzani (1996) point out that interdependence between attributes will not affect NB's classification accuracy, so long as it can generate the correct ranks of conditional probabilities for the classes. However, the success of semi-naive Bayesian methods show that appropriate relaxation of the attribute independence assumption is effective at improving its accuracy. Further, in many applications it is desirable to obtain accurate estimates of the conditional class probability rather than a simple classification, and hence mere correct ranking will not suffice.

Previous semi-naive Bayesian methods can be roughly subdivided into five groups. The first group applies NB to a subset of attributes generated by deleting attributes (Kittler, 1986; Langley and Sage, 1994). The second group adds explicit interdependencies between attributes. Sahami (1996) introduces the terminology of the *z-dependence Bayesian classifier*, in which each attribute depends upon the class and at most z other attributes. Within this framework, NB is a 0-dependence estimator. The majority of semi-naive Bayesian methods that add explicit interdependencies between attributes establish 1-dependence classifiers (Friedman et al., 1997; Keogh and Pazzani, 1999; Webb et al., 2005; Cerquides and Mántaras, 2005a; Zheng and Webb, 2006, 2007). Two exceptions are NBTree (Kohavi, 1996) and Lazy Bayesian Rules (LBR) (Zheng and Webb, 2000), both of which may add any number of dependencies for an attribute. The third group applies NB to a subset of training instances (Langley, 1993; Frank et al., 2003). Note that the second and third groups are not mutually exclusive. For example, NBTree and LBR classify instances by applying NB to a subset of training instances, and hence they can also be categorized to the third group. The fourth group performs adjustments to the output of NB without altering its direct operation (Hilden and Bjerregaard, 1976; Webb and Pazzani, 1998; Platt, 1999; Zadrozny and Elkan, 2001, 2002; Gama, 2003). The fifth group introduces hidden variables to NB (Kononenko, 1991; Pazzani, 1996; Zhang et al., 2004, 2005; Langseth and Nielsen, 2006).

It is also useful to distinguish between *eager learning* methods (Hilden and Bjerregaard, 1976; Kittler, 1986; Kononenko, 1991; Langley, 1993; Langley and Sage, 1994; Kohavi, 1996; Pazzani, 1996; Friedman et al., 1997; Webb and Pazzani, 1998; Keogh and Pazzani, 1999; Platt, 1999; Zadrozny and Elkan, 2001, 2002; Gama, 2003; Zhang et al., 2004, 2005; Webb et al., 2005; Cerquides and Mántaras, 2005a; Langseth and Nielsen, 2006; Zheng and Webb, 2007), which perform learning at training time, and *lazy learning* methods (Zheng and Webb, 2000; Frank et al., 2003; Zheng and Webb, 2006), which defer learning until classification time.

3.1 Applying NB to a Subset of Attributes

In NB, all the attributes are used during prediction, and hence all have some influence on classification. When two attributes are strongly correlated, the influence from these two attributes may be given too much weight, and the influence of the other attributes may be reduced, which can result in prediction bias. Deleting one of these attributes may have the effect of alleviating the problem. In addition, irrelevant attributes may also degrade the performance of NB, in effect increasing variance without decreasing bias. Hence their removal is also useful.

3.1.1 Backward Sequential Elimination

The *wrapper approach* uses accuracy estimates of a target learning algorithm as an evaluation function to measure the effectiveness of alternative attribute subsets (Kohavi and John, 1997). Backward Sequential Elimination (BSE) (Kittler, 1986) employs a simple heuristic wrapper approach to detecting and repairing harmful interdependencies. It uses Leave-one-out cross validation error as a selection criterion. Starting from the full set of attributes, BSE operates by iteratively removing successive attributes, each time removing the attribute whose elimination best reduces training set error. It terminates the process when there is no accuracy improvement. Independence is assumed among the attributes in the resulting attribute subset given the class. It uses (4), where the set of indices of the resulting attribute subset is substituted for N, to estimate $P(y, \mathbf{x})$.

At training time BSE generates two tables of probability estimates as NB does. As it performs leave-one-out cross validation to select the subset of attributes, it must also store the training data, with additional space complexity $O(tn)$. Keogh and Pazzani (1999) speed up the process of evaluating the classifiers by using a two-dimensional table, indexed by instance and class, to store the probability estimates. Each entry in the table is the estimate of the posterior probabilities that instance $\mathbf{x}$ belongs to class y. It is straightforward to update these to exclude or include the contribution of a specific attribute x_i by dividing or multiplying the entry $\langle \mathbf{x}, y \rangle$ by $\hat{P}(x_i|y)$. Hence, leave-one-out cross validation can be performed by simply taking each attribute x_i in turn, excluding or including x_i in the table of posterior probabilities, and then classifying $\mathbf{x}$. The resulting space complexity is $O(tn + tk + knv)$. The time complexity of a single leave-one-out cross validation is reduced from $O(tkn)$ to $O(tk)$ by using the speed up strategy. Therefore, the time complexity of attribute selection is $O(tkn^2)$, as leave-one-out cross validation will be performed at most $O(n^2)$ times.

BSE has identical time and space complexity to NB at classification time, although it may in practice result in significant speed-up if many attributes are deleted.

3.1.2 Forward Sequential Selection

Forward Sequential Selection (FSS) (Langley and Sage, 1994) uses the reverse search direction to BSE. Starting from the empty set of attributes, it operates by iteratively adding successive attributes, each time adding the attribute whose addition most improves training set accuracy. It performs selection repeatedly while the accuracy is not reduced. It also uses (4) to estimate $P(y, \mathbf{x})$, where the set of indices of added attributes is substituted for N. FSS has identical training and classification complexity to BSE.

3.2 Altering NB by Allowing Interdependencies between Attributes

The addition of explicit arcs between attributes allows interdependencies between attributes to be addressed directly. However, techniques for learning unrestricted Bayesian networks often fail to achieve lower error than NB and sometimes lead to higher error (Friedman, Geiger, and Goldszmidt, 1997). Two possible reasons for this are that the large number of parameters that must be estimated for full Bayesian networks lead to high variance and that full Bayesian networks are oriented toward estimating any marginal probability rather than the being specifically focused on the task of estimating the conditional probabilities of the class attribute given a full set of other attribute values.

3.2.1 Tree Augmented Naive Bayes and SuperParent TAN

Tree Augmented Naive Bayes (TAN) (Friedman et al., 1997) allows each attribute to depend on at most one non-class attribute. Based on this representation, they extended a method first proposed by Chow and Liu (1968) and utilized conditional mutual information to efficiently find a maximum spanning tree as a classifier. As each attribute depends on at most one other non-class attribute, TAN is a 1-dependence classifier. The parent of each attribute X_i is indicated as $\pi(X_i)$ and the parent of the class is $\emptyset$. It estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y) \prod_{i \in N} \hat{P}(x_i \,|\, y, \pi(x_i)), \tag{5}$$

where $\pi(x_i)$ is a value of $\pi(X_i)$.

At training time TAN generates a one-dimensional table of class probability estimates, and a three-dimensional table containing a conditional probability estimate for each attribute-value, conditioned on each other attribute-value and each class, space complexity $O(k(nv)^2)$. The time complexity of forming the three dimensional probability table is $O(tn^2)$, as we need to update each entry for every combination of the two attribute-values for every instance. Creating the conditional mutual information matrix requires consideration for each pair of attributes of every pairwise combination of their respective values in conjunction with each class. The resulting time complexity is $O(k(nv)^2)$. The parent function is then generated by establishing a maximal spanning tree, time complexity $O(n^2 \log n)$. At classification time, to classify a single example has time complexity $O(kn)$. The three dimensional conditional probability table formed at training time can be compressed by storing probability estimates for each attribute-value conditioned by the parent selected for that attribute and the class. Hence, the space complexity is $O(knv^2)$.

SuperParent TAN (SP-TAN) (Keogh and Pazzani, 1999), a variant of TAN, uses the same representation as TAN, but utilizes a wrapper approach to construct the parent function. It uses leave-one-out cross validation error as a criterion to add arcs. At any stage during this process, all attributes that are yet to be assigned a non-class parent are called *orphans*. An attribute that is made the parent of all orphans (other than itself) is called a *SuperParent*. There are two steps to add an arc. First, the node that if made SuperParent most improves accuracy is identified. Next, SP-TAN assesses the effect of adding a single arc from this node to each orphan (without adding arcs to other orphans). The orphan whose adoption most improves accuracy is called the *FavoriteChild*. This Parent-FavoriteChild pair is then added to the current network. SP-TAN stops adding arcs when there is no further accuracy improvement or the number of orphans is one.

For ease of understanding, we use a hypothetical example with 4 attributes (X_1, X_2, X_3 and X_4) to explain this process. The network is initialized to NB and the set of orphans is initialized to $\{X_1, X_2, X_3, X_4\}$, as shown in Figure 1 (a). Each attribute is temporarily made

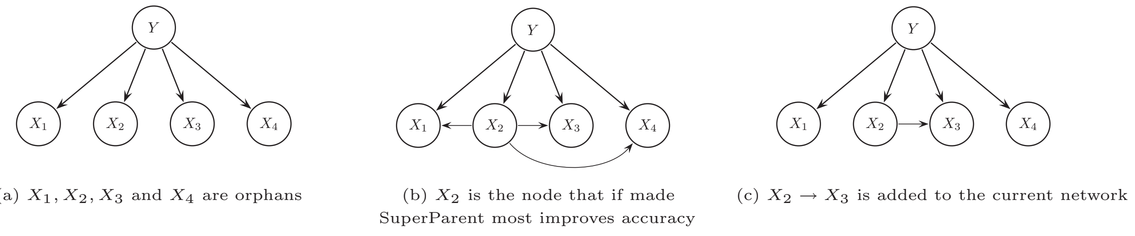

Y
X_1
X_2
X_3
X_4
(a) X_1, X_2, X_3 and X_4 are orphans
(b) X_2 is the node that if made SuperParent most improves accuracy
(c) $X_2 \rightarrow X_3$ is added to the current network

Figure 1: The process of adding an arc for SP-TAN

the parent of all orphans (other than itself) in turn and the effect on classification accuracy is assessed. In this example (Figure 1 (b)), X_2 is the node that if made SuperParent most improves accuracy. Then, SP-TAN finds the FavoriteChild of X_2. If the addition of the arc from X_2 to X_3 most improves accuracy, X_3 is identified as the FavoriteChild of X_2, and this arc is added to the current network (Figure 1 (c)). This process is terminated when there is no accuracy improvement or the number of orphans is one. SP-TAN also uses (5) to classify an instance.

Under different criteria for establishing parent functions, TAN tends to add $n-1$ arcs, while SP-TAN may have fewer arcs between attributes. As the selection of root does not affect the log-likelihood of the network, TAN randomly selects a root attribute and directs all edges away from it. In contrast, SP-TAN makes the direction from SuperParents to their FavoriteChilds.

At training time SP-TAN needs additional space complexity $O(tn)$ for storing the training data compared with TAN. The selection of a single SuperParent is order $O(tkn^2)$, and the selection of the FavoriteChild is order $O(tkn)$, which is achieved by using the speed up strategy mentioned in Section 3.1.1. Keogh and Pazzani (1999) proposed another optimization to speed up the evaluating process by sorting instances according to whether they are classified correctly and testing the misclassified instances first. Hence, once the number of the misclassified instances is larger than the current best-so-far error, we can stop testing the classifier. These optimizations are effective in practice. The time complexity of forming the parent function is $O(tkn^3)$, as $O(n)$ arcs are added. SP-TAN has identical classification time and space complexity to TAN.

3.2.2 NBTree

NBTree (Kohavi, 1996) seeks to combine the advantages of NB and decision trees. It partitions the training data using a tree structure and establishes a local NB in each leaf. NBTree uses 5-fold cross validation accuracy estimation as the splitting criterion. Each value of a splitting attribute has its own subtree. The utility of a node is the 5-fold cross validation accuracy of NB at this node. The utility of a split on an attribute equals the weighted sum of the utility of nodes generated by the split. NBTree partitions the training sample according to the test on the attribute that has the highest utility, out of those that are substantially better than the utility of the current node. A split is defined to be substantially better if the relative error reduction is greater than 5% and the splitting node has at least 30 instances. When there is no substantial improvement, NBTree stops the growth of the tree.

The classical decision tree predicts the same class for all test instances that reach a leaf. In NBTree, these instances are classified using a local NB in the leaf, which only considers those attributes that are not tested on the path to the leaf and those training instances that reach the leaf. Let B be the set of the splitting attributes on the path leading to the leaf, and let L be the set of the remaining attributes, we have

$$\begin{aligned} P(Y, \mathbf{X}) &= P(B)P(Y \,|\, B)P(L \,|\, Y, B) \\ &\propto P(Y \,|\, B)P(L \,|\, Y, B), \end{aligned}$$

where $P(Y \,|\, B)$ is the probability of the class in the leaf, in which the attributes in B have same values, and $P(L \,|\, Y, B)$ is the conditional probability of the remaining attributes given the class in the leaf. Therefore, NBTree estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y, b) \prod_{i \in l} \hat{P}(x_i \,|\, y, b),$$

where b is a value of B and l is the set of indices of attributes in L. NBTree is expected to have the effect of mitigating the harmful attribute interdependence problem for each local NB if B can be selected appropriately.

In the worst case, the height of the tree is $O(n)$. At the root, NBTree performs 5-fold cross validation on each attribute. The split on X_i results in v_i subtrees and the cost of performing cross validation is $t_1 k(n-1) + \cdots + t_{v_i} k(n-1) = tk(n-1)$, where t_i is the number of instances in the ith subtree. At level one, each instance is described by $n-1$ attributes and the cost of performing cross validation for each attribute is $tk(n-2)$. Therefore, the worst cost of building the tree is $\sum_{j=n}^{2} tkj(j-1) = O(tkn^3)$. In the worst case, the number of leaves is $O(t)$ and the height of the tree is $O(n)$. The space complexity is $O(kvt)$. Classification of a single example has time complexity $O(kn)$ and space complexity $O(kvt)$.

3.2.3 Lazy Bayesian Rules

Lazy Bayesian Rules (LBR) (Zheng and Webb, 2000) adopts a lazy approach and generates a Bayesian rule according to each test example. The antecedent of a Bayesian rule is a conjunction of attribute-value pairs, and the consequent of the rule is a local NB, which uses those attributes that do not appear in the antecedent. The utility of an attribute-value pair is assessed using leave-one-out cross validation in the local training samples, those examples that have the attribute values in the antecedent together with the attribute value being tested. As different attribute-value pairs cover different subsets of the training samples, it is necessary to be careful in assessing the relative effectiveness of the alternatives. For example, one attribute value might select a set of examples that are already all correctly classified by the existing antecedent whereas another might select only examples that are not. If the former made no errors and the latter made only 50% errors then it would be the latter that provided that greatest improvement even though it had the higher error. To measure each attribute-value pair on the whole local training sample, the errors of the existing local NB on the training samples that satisfy the attribute values in antecedent but not the attribute value being tested, are added to the errors of the NB on the local training examples. The attribute-value pair with the lowest error, out of these that are significantly lower than the error of the current local NB, is added to the antecedent. The difference is considered as significant if the outcome of a one-tailed pairwise sign test is better than 0.05. LBR stops adding attribute-value pairs into the antecedent if there is no significant improvement.

Let q be a value of the set of attributes in the antecedent, and s the set of indices of remaining attributes, LBR estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y, q) \prod_{i \in s} \hat{P}(x_i \,|\, y, q).$$

The Bayesian rule generated by LBR can be considered as a branch of a tree built by NBTree. LBR generates a rule for each test instance, while NBtree builds a single model according to all the examples in the training data. If examples are not evenly distributed among branches, NBTree may suffer from the small disjunct problem (Holte et al., 1989). As LBR uses lazy learning, it may mitigate the problem by avoiding splits on an attribute when the relevant value is infrequent. It is efficient when few examples are to be classified. However, the computational overhead of LBR may be excessive when large numbers of examples are to be classified.

At training time, the time and space complexity of LBR are $O(tn)$, as it only stores the training data. At classification time, LBR adds attribute-value pairs to the antecedent with time complexity of $O(tkn^3)$, as the selection of an attribute-value pair for the antecedent is order $O(tkn^2)$ and this selection is performed repeatedly until there is no significant improvement on accuracy. The space complexity is $O(tn + knv)$.

3.2.4 Averaged One-Dependence Estimators

To avoid model selection and attain the efficiency of 1-dependence classifiers, Averaged One-Dependence Estimators (AODE) (Webb et al., 2005) selects a restricted class of One-Dependence Estimators (ODEs) and aggregates the predictions of all qualified estimators within this class. A single attribute, called the *SuperParent* if we borrow the term from SP-TAN, is selected as the parent of all the other attributes in each ODE. This type of ODE is called a *SuperParent One-Dependence Estimator* (SPODE). In order to avoid unreliable base probability estimates, when classifying an instance $\mathbf{x}$ the original AODE excludes SPODEs with SuperParent x_i where the frequency of the value x_i is lower than limit $m = 30$, a widely used minimum on sample size for statistical inference purposes. However, subsequent research (Cerquides and Mántaras, 2005a) reveals that this constraint actually increases error and hence the current research uses $m = 1$.

For any attribute value x_i,

$$P(y, \mathbf{x}) = P(y, x_i) P(\mathbf{x} \mid y, x_i).$$

This equality holds for every x_i. Therefore,

$$P(y,\mathbf{x}) = \frac{\sum_{i\in N \wedge F(x_i)\geq m} P(y,x_i)P(\mathbf{x} \mid y,x_i)}{|\{i \in N \wedge F(x_i) \geq m\}|}, \tag{6}$$

where $F(x_i)$ is the frequency of attribute value x_i in the training sample.

AODE utilizes (6) and, for each ODE, an assumption that the attributes are independent given the class and the privileged attribute value x_i, estimating $P(y,\mathbf{x})$ by

$$\hat{P}(y,\mathbf{x}) = \frac{\sum_{i\in N \wedge F(x_i)\geq m} \hat{P}(y,x_i)\prod_{j\in N, j\neq i}\hat{P}(x_j \mid y,x_i)}{|\{i \in N \wedge F(x_i) \geq m\}|}. \tag{7}$$

It averages over estimates of the terms in (6), rather than the true values, which has the effect of reducing the variance of these estimates.

At training time AODE generates probability tables as TAN does, time complexity $O(tn^2)$. Classification requires the tables of probability estimates formed at training time of space complexity $O(k(nv)^2)$. The time complexity of classifying a single example is $O(kn^2)$ as we need to consider each pair of qualified parent and child attribute within each class. This process can be sped up by introducing a constant array to store estimates of $P(y,x_i)$ and $P(x_j \,|\, y,x_i)$ at training time. Therefore, at classification time, we only need to read, other than calculate, these estimates. Although this does not change the classification time complexity, in practice, it may result in substantial speed-up.

3.2.5 Maximum a Posteriori Linear Mixture of Generative Distributions

AODE classifies by uniformly aggregating all qualified ODEs. One natural extension to AODE is to use a linear mixture assigning a weight to each ODE. Maximum a Posteriori Linear Mixture of Generative Distributions (MAPLMG) (Cerquides and Mántaras, 2005a) assigns the weights $\mathbf{w} = \langle \mathrm{w}_1, \ldots, \mathrm{w}_n\rangle$, with which the supervised posteriori (defined in (8)) for Linear Mixture of Generative Distributions (LMG) is maximized, to the ODEs. This is an optimization problem under the constraint that $\forall i \in \{1,\ldots,n\}$, $\mathrm{w}_i \geq 0$ and $\sum_{i=1}^{n} \mathrm{w}_i = 1$. The Augmented Lagrangian method (Pedregal, 2004) is used to transform the constrained nonlinear optimization problem into a sequence of unconstrained nonlinear optimization problems, which are solved by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (Avriel, 2003).

The supervised posterior for LMG using Leave-One-Out cross validation is

$$\hat{P}_{LMG}(\mathrm{w}|T) = \prod_{\langle \mathbf{x},y\rangle \in T}\left(\frac{\sum_{i\in N}\mathrm{w}_i\hat{P}_i^{LOO}(\overline{\mathbf{x},y})}{\sum_{y\in Y}\sum_{i\in N}\mathrm{w}_i\hat{P}_i^{LOO}(\overline{\mathbf{x},y})}\prod_{i\in N}\mathrm{w}_i\right), \tag{8}$$

where

$$\hat{P}_i^{LOO}(\overline{\mathbf{x},y}) = \hat{P}(x_i,y)\prod_{j\in N, j\neq i}\hat{P}(x_j|x_i,y),$$

which is estimated by excluding instance $\langle \mathbf{x},y\rangle$ from training set T.

MAPLMG estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \sum_{i \in N} \mathrm{w}_i \hat{P}(y, x_i) \prod_{j \in N, j \neq i} \hat{P}(x_j \mid y, x_i).$$

At training time, MAPLMG first estimates the generative probabilities for the instances left out. It generates a three-dimensional table of probability estimates and stores the training data to perform Leave-One-Out cross validation, space complexity of $O(tn + k(nv)^2)$. Since we need to go through the training data and consider each parent and child attribute pair within each class, the estimation process has time complexity of $O(tkn^2)$. The second step is to maximize the supervised posterior for LMG, which has space complexity of $O(n^2)$ and time complexity of $O(tknI)$, where I is the upper limit of the number of iterations. Therefore, the overall time complexity is $O(tkn^2 + tknI)$. At classification time, MAPLMG has identical time and space complexity to AODE.

3.2.6 Subsumption Resolution (SR)

The specialization-generalization relationship is an extreme type of interdependence. For two attribute values x_i and x_j, if $P(x_j \mid x_i) = 1.0$ then x_j is a generalization of x_i and x_i a specialization of x_j. If x_j is a generalization of x_i, then $P(y|x_1, \ldots, x_n) = P(y|x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$. Therefore, deleting the generalization should not be harmful and in fact it removes a violation of the attribute independence assumption. Subsumption Resolution (SR) (Zheng and Webb, 2006) identifies at classification time pairs of attribute-values such that one appears to subsume (be a generalization of) the other and deletes the generalization. This requires a method for inferring from the training data whether one attribute-value is a generalization of another. SR uses the criterion

$$F(x_i) = F(x_i, x_j) \geq l$$

to infer that x_j is a generalization of x_i, where $F(x_i, x_j)$ is the frequency of x_i and x_j in the training sample and l is a user-specified minimum frequency. It is necessary to specify a minimum frequency to prevent unsound inferences that $P(x_i) = P(x_i, x_j)$ from small samples. It might be thought that a less arbitrary technique than use of the "magic number" l might be obtained using either statistical or information theoretic approaches, but each would still require an arbitrary critical value, such as α.

SR is suited to algorithms without model selection, such as NB and AODE, but not algorithms with model selection, such as TAN and NBTree, as the attribute value elimination can be directly applied to the former but it may affect the model structure built by the latter.

In the context of NB, SR deletes generalization attribute-values if a specialization is detected and applies NB to the resulting attribute value set. NB with SR (indicated as NB^{SR}) uses (4) to estimate $P(y, \mathbf{x})$ where the set of indices of the resulting attribute value set is substituted for N.

To identify the specialization-generalization relationship, NB^{SR} must generate at training time a two-dimensional table of probability estimates for each attribute-value, conditioned by each other attribute-value in addition to the two probability estimates tables generated by NB, space complexity $O(knv + (nv)^2)$. The time complexity of forming the additional two-dimensional probability estimates table is $O(tn^2)$. Classification of a single example requires considering each pair of attributes to detect dependency and is of time complexity $O(n^2 + kn)$. The space complexity is $O(knv + (nv)^2)$.

When SR is applied to AODE, the resulting classifier acts as AODE except that it deletes generalization attribute-values if a specialization is detected, and aggregates the predictions of all qualified classifiers using the remaining attribute-values. AODE with SR (indicated as AODE^{SR}) uses (7) to estimate $P(y, \mathbf{x})$ where the set of indices of the resulting attribute value set is substituted for N.

AODE^{SR} has identical time and space complexity to AODE. At training time it behaves identically to AODE. At classification time, it must check all attribute-value pairs for generalization relationships, an additional operation of time complexity $O(n^2)$. However, the time complexity of AODE at classification time is $O(kn^2)$ and so this additional computation does not increase the time complexity of AODE.

3.3 Applying NB to a Subset of the Training Set

Another effective approach to accommodating violations of the conditional independence assumption is to apply NB to a subset of the training set, as it is possible that the assumption, although violated in the whole training set, may hold or approximately hold in a subset of the training set. As has been discussed, this group and the second group that alters NB by allowing interdependencies between attributes are not mutually exclusive. NBTree and LBR use local NBs to classify an instance and can also be classified into this group.

3.3.1 Recursive Bayesian Classifiers

Recursive Bayesian Classifiers (RBC) (Langley, 1993) forms NB on the training data, then partitions the training data, placing each instance in a partition corresponding to the class that NB assigns it. This process is repeated recursively on each partition forming a tree, until each leaf partition contains only instances from one class. At classification time, NB is applied to the test instance to direct it down one branch of the tree. Then the NB formed at the appropriate partition is applied, and so on, until a leaf is reached, at which point the NB for the leaf is applied to obtain a classification. While results on artificial data were promising, the reported experimental results for RBC on natural data sets are disappointing. As a result the technique has received little attention.

3.3.2 Locally Weighted Naive Bayes

Inspired by locally weighted linear regression (Cleveland, 1979; Atkeson, Moore, and Schaal, 1997; Loader, 1999; Hastie, Tibshirani, and Friedman, 2001), Frank, Hall and Pfahringer incorporate locally weighted learning into NB (Frank, Hall, and Pfahringer, 2003). Locally Weighted Naive Bayes (LWNB), at classification time, assigns a weight to each instance in the training set and applies NB to the subset of the training set in which the weights of instances are greater than zero. The instance weights decrease linearly with the Euclidean distance to the test instance and the number of instances in the subset is determined by a user-specified parameter h. The Euclidean distance of the test instance to the ith nearest neighbor is denoted as d_i. Before the distance is calculated, quantitative attributes are assumed to be normalized ($0 \leq x_i \leq 1$) and qualitative attributes to be binarized. LWNB assigns the weight $\mathrm{w}_i = f(d_i/d_h)$ to the ith instance, where $f(x)$ is zero if $x > 1$ and $1 - x$ otherwise.

Let h' be the number of instances whose distances are less than d_h. LWNB applies NB to the training set with rescaled weights w'_i which is calculated by $\mathrm{w}'_i = \frac{\mathrm{w}_i \times h'}{\sum_{j=1}^{t} \mathrm{w}_j}$. With these rescaled weights, the total weight in the subset in which all weights are greater than zero is approximately h. That is, $\sum_{i=1}^{h'} \mathrm{w}'_i \approx h$. In this subset, the frequency of the ith class is:

$$F(c_i) = \sum_{l=1}^{h'} \mathrm{w}'_l E(c_i, c_l),$$

where $E(a, b)$ is one if $a = b$ and zero otherwise. The frequency of x_j (the value of the jth attribute in the test instance) given class c_i is:

$$F(x_j \,|\, c_i) = \sum_{l=1}^{h'} \mathrm{w}'_l E(c_i, c_l) E(x_j, x_j^l),$$

where x_j^l is the value of the jth attribute in the lth instance. LWNB uses Equation (4) to estimate $P(y, \mathbf{x})$.

At training time, as LWNB only stores the training data, it has time and space complexity of $O(tn)$. At classification time, the time complexity of computing distances is $O(tn)$ if a linear search is performed.[1] Estimating class probability and conditional attribute-value probability has time complexity of $O(h'n)$. To classify the test example requires time of order $O(kn)$. Hence, the overall classification time complexity is $O(tn + kn)$. The space complexity is $O(tn + knv)$.

3.4 Calibrating NB's Probability Estimates

Due to the attribute independence assumption, NB tends to output a bimodal posterior (Bennett, 2000). That is, for most instances, either the posterior estimates are close to 0 or close to 1. The distortion of probabilities that the independence assumption produces might be corrected by making adjustments to posterior probabilities, class probabilities or attribute conditional probabilities. This line of reasoning leads to the fourth group of methods, which modify the probability outcome of NB.

Hilden and Bjerregaard (1976) impose smoothing by raising the conditional probability estimates to a power B ($0 < B < 1$), in effect shrinking the estimates towards zero. Therefore, $\hat{P}(y, \mathbf{x})$ is estimated by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y) \prod_{i \in N} \hat{P}(x_i \,|\, y)^B.$$

They also use m-estimation ($m = 1$) to estimate base probabilities. In their experiments, using $B = 0.8$ provided the best performance. Note that this method can be applied to any semi-naive Bayesian method without additional computation. It has identical time and space complexity to NB.

1. When space-partitioning methods, such as KD-Tree, are performed, less time is required for computing distances.

3.4.1 Isotonic Regression

Platt (1999) proposed a parametric method to map the SVM outputs $o(\mathbf{x})$ into posterior probabilities by passing the outputs through a sigmoid:

$$\hat{P}(y \mid \mathbf{x}) = \frac{1}{1 + e^{Ao(\mathbf{x})+B}},$$

where the parameters A and B are found by minimizing the negative log-likelihood. Bennett (2000) applied this method to NB and found that the mean squared error of the estimates is often reduced, while the classification error is not. Zadrozny and Elkan (2001) used a non-parametric method, binning, to calibrate NB's posterior outputs $\hat{P}(y \mid \mathbf{x})$. It first sorts the training set according to $\hat{P}(y \mid \mathbf{x})$, and then divides the sorted set into b equal-sized subsets. Each subset, called a *bin*, has an upper bound and a lower bound value. Given a test instance $\mathbf{x}$, it is placed in the bin according to $\hat{P}(y \mid \mathbf{x})$. The calibrated probability estimate is the fraction of training examples in the bin that actually belong to the class y. In their experiments, the number of bins is set to 10.

Zadrozny and Elkan (2002) also used Isotonic Regression (IR) (Robertson et al., 1988), an intermediary approach between sigmoid fitting and binning, to calibrate predictions made by NB. The rationale behind this method is that if a classifier can correctly rank examples, the mapping from the outputs of the classifier into posterior probabilities is non-decreasing. Pair-adjacent violators (PAV) (Ayer et al., 1955) is a commonly used algorithm for the Isotonic Regression problem, finding a stepwise constant isotonic function to minimize mean squared error over the entire training set. Let $g(\mathbf{x})$ be the value of the function to be learned for instance $\mathbf{x}$ and g^* the isotonic regression. Given a training set ordered by $\hat{P}(y \mid \mathbf{x})$, PAV initializes $g(\mathbf{x})$ to one if $\mathbf{x}$ belongs to y and to zero otherwise. If g is not isotonic, it follows that there exists a subscript i such that $g(\mathbf{x}_{i-1}) > g(\mathbf{x}_i)$. These two instances $\mathbf{x}_{i-1}$ and $\mathbf{x}_{i-1}$ are called *pair-adjacent violators.* The values of $g(\mathbf{x}_{i-1})$ and $g(\mathbf{x}_i)$ are then replaced by their average $g^*(\mathbf{x}_{i-1}) = g^*(\mathbf{x}_i) = (g(\mathbf{x}_{i-1}) + g(\mathbf{x}_i))/2$. This process of replacement is repeated until there is no violators. PAV returns a set of intervals and an estimate $g^*(i)$ for each interval i such that $g^*(i-1) \leq g^*(i)$. To classify a test instance $\mathbf{x}$, IR first finds the interval i in which $\hat{P}(y \mid \mathbf{x})$ fits and assigns $g^*(i)$ as the calibrated posterior probability.

IR is a binary classifier algorithm. Zadrozny and Elkan used *one-against-all* approach to generalizing from two-class classification to multi-class classification and recommended using simple normalization to combine the binary estimates.

At training time, IR generates two probability tables as NB does, time complexity $O(tn)$. It also generates a two-dimensional table, indexed by instance and class, to store the outputs of NB. Each entry in the table is the estimate of the posterior probability that instance $\mathbf{x}$ belongs to class y. The resulting time complexity is $O(tkn)$ and space complexity $O(tk)$. For each class y, sorting the training instances according to $\hat{P}(y \mid \mathbf{x})$ is order $O(tlogt)$ and generating g^* using PAV is order $O(t)$. Therefore, for k classes, the time complexity is $O(tklogt)$. At classification time, to obtain NB's classification requires of order $O(kn)$ and find a interval for the instance requires of order $O(klogb)$, where b is the number of intervals. Since $logb$ is usually smaller than n, the classification time complexity can be summarized as $O(kn)$ and the space complexity as $O(knv)$.

3.4.2 Adjusted Probability Naive Bayesian Classification

Adjusted Probability Naive Bayesian Classification (APNB) (Webb and Pazzani, 1998) applies linear adjustments to the class probabilities. In the two class case, it only needs to find an adjustment for one of the classes. As the adjustment for one class will influence the adjustments for other classes in the multiple class case, APNB uses a simple hill-climbing search to find adjustments that maximize resubstitution accuracy.

If an instance $\mathbf{x}$ of class c_i is misclassified as class c_j by APNB with the current vector of adjustments A, there are two possible adjustments to correct that misclassification. One is an upward adjustment, which multiplies the original probability estimate for class c_i by an adjustment $> \frac{A_{c_j}\hat{P}(c_j|\mathbf{x})}{\hat{P}(c_i|\mathbf{x})}$, where $\hat{P}(c_j \mid \mathbf{x})$ and $\hat{P}(c_i \mid \mathbf{x})$ are the probability estimates produced by NB, and A_{c_z} is the adjustment for class c_z. Another is a downward adjustment, which multiplies the probability of class c_j with an adjustment $< \frac{A_{c_i}P(c_i|\mathbf{x})}{P(c_j|\mathbf{x})}$. APNB selects a value slightly above or below the bounds implied by these ranges, a small value (10^{-5}) being added to the relevant bound for upward adjustments and subtracted from the relevant bound for downward adjustments. If there is a significant accuracy improvement by using a upward or downward adjustment, another bound value for the adjustment is computed. For the upward adjustment, the upper bound value is the lowest value greater than the lower bound value, that has a higher resubstitution error than the lower bound value. For the downward adjustment, the lower bound value is the highest value less than the upper bound value, that has a higher resubstitution error than the upper bound value. The adjustment is replaced by the midpoint of the two bound values. This process repeated until there is no accuracy improvement that passes a binomial sign test for significance at the 0.05 level.

APNB estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = A_y \hat{P}(y) \prod_{i \in N} \hat{P}(x_i \,|\, y).$$

At training time APNB stores the training data to perform leave-one-out cross validation in addition to the two probability estimates tables generated by NB. Hence, the space complexity is $O(tn + knv)$. In the worse case, there are $O(t)$ misclassified instances, and each possible adjustment for the instance has time complexity of $O(tk)$. This process is repeated once for each class to find a single adjustment that maximizes resubsitution accuracy. Therefore, the time complexity is $O(t^2k^2)$. It has identical time and space complexity to NB at classification time.

3.4.3 Iterative Bayes

Iterative Bayes (IB) (Gama, 2003) starts with the conditional attribute-value frequency table generated by NB, indexed by class and attribute-value, and iteratively updates the frequency table by cycling through all the training examples.

At each iteration, all the examples in the training set are classified by NB using the current frequency tables. The conditional attribute-value frequency table is updated through each training example. The adjustment (indicated as A) for an example $\mathbf{x}$ is

$$\frac{1.0 - \hat{P}(y_i \,|\, \mathbf{x})}{k},$$

where y_i is the predicted class. If the example is correctly classified, A is added to the entries $\langle y_i, x_j \rangle$, $1 \leq j \leq n$, and $A/(k-1)$ is subtracted from the entries $\langle y_l, x_j \rangle$, $1 \leq l \leq k$, $l \neq i$ and $1 \leq j \leq n$. If the example is misclassified, A is subtracted from the entries $\langle y_i, x_j \rangle$, $1 \leq j \leq n$, and $A/(k-1)$ is added to the entries $\langle y_l, x_j \rangle$, $1 \leq l \leq k$, $l \neq i$ and $1 \leq j \leq n$. This process is terminated when the number of iterations exceeds 10 or the following evaluation function increases:

$$\frac{1}{t}\sum_{i=1}^{t}\left(1.0 - \operatorname*{argmax}_{y} \hat{P}(y \,|\, \mathbf{x}_i)\right),$$

where $\mathbf{x}_i$ is the ith example. It uses (4) to estimate $P(y, \mathbf{x})$.

At training time, IB generates two probability tables and stores the training data, space complexity $O(tn + knv)$. At each iteration, to update conditional frequency table requires time of order $O(tkn)$, as we need to adjust each entry for every combination of the classes and attribute-values for every example, and to perform classification for all examples also requires time of order $O(tkn)$. Therefore, the total training time complexity is $O(tkn)$. At classification time, it has identical time and space complexity to NB.

3.5 Introducing Hidden Variables to NB

Hidden variables, also called *latent* variables, are variables that are not observed. If two observed attributes are correlated, the introduction of a hidden variable which aggregates the information from these two attributes may mitigate the attribute interdependence problem. When dependencies between two attributes are detected, joining these two attributes or their values corresponds to introducing a hidden variable. The Semi-naive Bayesian Classifier (Kononenko, 1991) uses an exhaustive search to join attribute values iteratively based on a statistical method for identifying correlations between attributes. Unlike most subsequent semi-naive approaches, Kononenko joins attribute-values, rather than attributes. This means that different pairs of attributes may be joined when classifying different test instances. However, the reported experimental results were not compelling and the technique does not appear to have been utilized since it was first proposed.

3.5.1 Backward Sequential Elimination and Joining

Backward Sequential Elimination and Joining (BSEJ) (Pazzani, 1996) uses predictive accuracy on leave-one-out cross validation as a merging criterion to create new Cartesian product attributes. The value set of a new Cartesian product attribute is the Cartesian product of the value sets of the two original attributes. For instance, if attribute X_1 has two values: v_1^1 and v_1^2, and attribute X_2 has three values: v_2^1, v_2^2 and v_2^3, the new Cartesian product attribute will have six values: $v_1^1v_2^1$,$v_1^1v_2^2$, $v_1^1v_2^3$, $v_1^2v_2^1$, $v_1^2v_2^2$ and $v_1^2v_2^3$. In addition to creating new Cartesian product attributes, BSEJ deletes original attributes and also new Cartesian product attributes during a hill-climbing search. It repeatedly joins a pair of attributes or deletes an attribute such that the action most improves predictive accuracy on leave-one-out cross validation. This process terminates when there is no further accuracy improvement.

The resulting Cartesian product attribute set is denoted as $H = \{H_1, \cdots, H_q\}$. The set of indices of remaining original attributes that have not been either deleted or joined is indicated as R. Independence is assumed among the attributes in $\{X_i|i \in R\}$ and H given the class. Hence, BSEJ estimates $P(y, \mathbf{x})$ by

$$\hat{P}(y, \mathbf{x}) = \hat{P}(y) \prod_{i \in R} \hat{P}(x_i \,|\, y) \prod_{j=1}^{q} \hat{P}(h_j \,|\, y), \tag{9}$$

where h_j is a value of H_j.

At training time BSEJ generates tables of class and conditional attribute-value probability estimates as NB does. It also generates two-dimensional tables of conditional Cartesian product attribute-value probability estimates, indexed by class and compound attribute-value. In the worst case, the new Cartesian product attribute has $O(v')$ values, where v' is $\prod_{i=1}^{n} v_i$. Therefore, the space complexity is $O(tn + kv')$ and the time complexity of initializing the tables is $O(kv')$. BSEJ considers at most $O(n^3)$ Cartesian product attributes, each requiring a pass through the training data to generate joint probability estimates and performing leave-one-out cross validation. The time complexity of joining and deleting attributes is $O(tkn^3)$ and the overall training time complexity is $O(tkn^3 + kv')$. At classification time, to classify a single example has time complexity $O(kn)$ and space complexity $O(kv')$.

3.5.2 Hierarchical Naive Bayes

Zhang et al. (2004) explore the problem of discovering hidden variables in the NB model and call the model Hierarchical Naive Bayes (HierNB [2]), in which the root is the class, the leaves are attributes observed and the internal nodes are hidden variables. An HierNB model is *parsimonious* if there does not exist another HierNB model with the same class and attributes such that the latter has fewer parameters than the former and the probability distribution over the class and attributes are the same in the two models. In addition to parsimonious models, they introduce *regular* HierNB models. A hidden variable is *singly connected* if it has only two neighbors, one being its parent and another child. They call an HierNB model regular if it satisfies three conditions. First, no two singly connected hidden variables are neighbors. Second, for any singly connected hidden variable H, $v_h \le \frac{v_i v_j}{v_i + v_j - 1}$, where v_h is the number of values of H and v_i and v_j are the number of values of the two neighbors of H. Third, for any hidden node H, $v_h \le \prod_{i \in C} v_i$, where C is the set of indices of children of H. They prove that any parsimonious HierNB model is regular and the set of all regular HierNB models for a given set of class and attributes is finite. The search is then restricted to those regular models. A hill-climbing algorithm is employed to learn a regular model by maximizing the Bayesian Information Criterion (BIC) score (Schwarz, 1978). This method uses EM algorithm (Dempster, Laird, and Rubin, 1977) to estimate parameters for each candidate model, and hence it has very high time complexity.

Langseth and Nielsen (2006) also use a greedy search to learn a regular HierNB model, but use different criteria to find a candidate hidden variable and to specify cardinalities for the hidden variable. Under the assumption that X_i and X_j are independent given Y, $2tI(X_i, X_j \,|\, Y)$ converges towards $\chi^2_{k(v_i-1)(v_j-1)}$ in distribution, where $I(X_i, X_j \,|\, Y)$ is the

2. To distinguish Hierarchical Naive Bayes from Hidden Naive Bayes, we use HierNB to denote the former and HidNB the latter.

conditional mutual information. A new hidden variable is allocated as the parent of two attributes X_i and X_j with the highest probability $P(Z \leq 2tI(X_i, X_j \,|\, Y))$, where Z is χ^2 distributed with $k(v_i - 1)(v_j - 1)$ degrees of freedom. The value set of a hidden variable is initialized to the Cartesian product over all the value sets of its children. Using a greedy search, this method collapses values of the hidden variable by maximizing conditional log-likelihood via the Minimum Description Length (MDL) (Rissanen, 1978) scoring function. They report that HierNB models generated by this method have a significant accuracy advantage over those by Zhang et al. (2004). HierNB uses (9) to estimate $P(y, \mathbf{x})$, where the set of indices of original attributes without a hidden parent is substituted for R, the set of hidden variables without a hidden parent is substituted for H and h_j is a value of the jth hidden variable belonging to H.

There are two main differences between BSEJ and Langseth and Nielsen's HierNB. The first difference is the criterion to join attributes (or create hidden variables). The former uses predictive accuracy on leave-one-out cross validation, while the latter uses conditional mutual information. In addition to joining attributes, BSEJ also deletes attributes. Rather than using the full combination of values of children of a hidden variable, HierNB collapses those values.

For the detailed time complexity analysis of HierNB, we refer the reader to Langseth and Nielsen (2006). The training time complexity is $O(tn^2v'^2)$ and training space complexity $O(kv')$. At classification time, it has identical time and space complexity to BSEJ.

3.5.3 Hidden Naive Bayes

Hidden Naive Bayes (HidNB) (Zhang, Jiang, and Su, 2005) creates a hidden parent for each attribute. The probability of attribute X_i given the class and its hidden parent H_i is defined as follows.

$$P(X_i \,|\, Y, H_i) = \sum_{j=1, j\neq i}^{n} W_{ij} P(X_i \,|\, Y, X_j),$$

where

$$W_{ij} = \frac{I(X_i, X_j \mid Y)}{\sum_{j=1,j\neq i}^{n} I(X_i, X_j \mid Y)}.$$

It estimates $P(y, \mathbf{x})$ by

$$P(y, \mathbf{x}) = P(y) \prod_{i \in N} P(x_i \,|\, y, h_i), \tag{10}$$

where h_i is a value of H_i.

At training time, HidNB generates probability tables and forms conditional mutual information matrix as TAN does, space complexity $O(k(nv)^2)$. The time complexity of computing weights is $O(n^2)$ and overall time complexity is $O(tn^2 + k(nv)^2)$. At classification time, it has identical time and space complexity to AODE.

Table 1: Computational Complexity

	Training		**Classification**	
Algorithm	**Time**	**Space**	**Time**	**Space**
NB	$O(tn)$	$O(knv)$	$O(kn)$	$O(knv)$
BSE	$O(tkn^2)$	$O(tn+tk+knv)$	$O(kn)$	$O(knv)$
FSS	$O(tkn^2)$	$O(tn+tk+knv)$	$O(kn)$	$O(knv)$
BSEJ	$O(tkn^3+kv')$	$O(tn+kv')$	$O(kn)$	$O(kv')$
TAN	$O(tn^2+k(nv)^2+n^2logn)$	$O(k(nv)^2)$	$O(kn)$	$O(knv^2)$
SP-TAN	$O(tkn^3)$	$O(tn+k(nv)^2)$	$O(kn)$	$O(knv^2)$
NBTree	$O(tkn^3)$	$O(tkv)$	$O(kn)$	$O(tkv)$
LBR	$O(tn)$	$O(tn)$	$O(tkn^3)$	$O(tn+knv)$
AODE	$O(tn^2)$	$O(k(nv)^2)$	$O(kn^2)$	$O(k(nv)^2)$
MAPLMG	$O(tkn^2+tknI)$	$O(tn+k(nv)^2)$	$O(kn^2)$	$O(k(nv)^2)$
NB^{SR}	$O(tn^2)$	$O(knv+(nv)^2)$	$O(n^2+kn)$	$O(knv+(nv)^2)$
AODE^{SR}	$O(tn^2)$	$O(k(nv)^2)$	$O(kn^2)$	$O(k(nv)^2)$
LWNB	$O(tn)$	$O(tn)$	$O(tn+kn)$	$O(tn+knv)$
IR	$O(tkn+tklogt)$	$O(tk+knv)$	$O(kn)$	$O(knv)$
APNB	$O(t^2k^2)$	$O(tn+knv)$	$O(kn)$	$O(knv)$
IB	$O(tkn)$	$O(tn+knv)$	$O(kn)$	$O(knv)$
HierNB	$O(tn^2v'^2)$	$O(kv')$	$O(kn)$	$O(kv')$
HidNB	$O(tn^2+k(nv)^2)$	$O(k(nv)^2)$	$O(kn^2)$	$O(k(nv)^2)$

k is the number of classes

n is the number of attributes

t is the number of training examples

v is the mean number of values per attribute

v' is the number of combinations of attribute values

I is the upper limit of the number of iterations for BFGS

3.6 Complexity Summary

Table 1 summarizes the complexity of each of the algorithms discussed. We display the time complexity and the space complexity of each algorithm for each of training time and classification time.

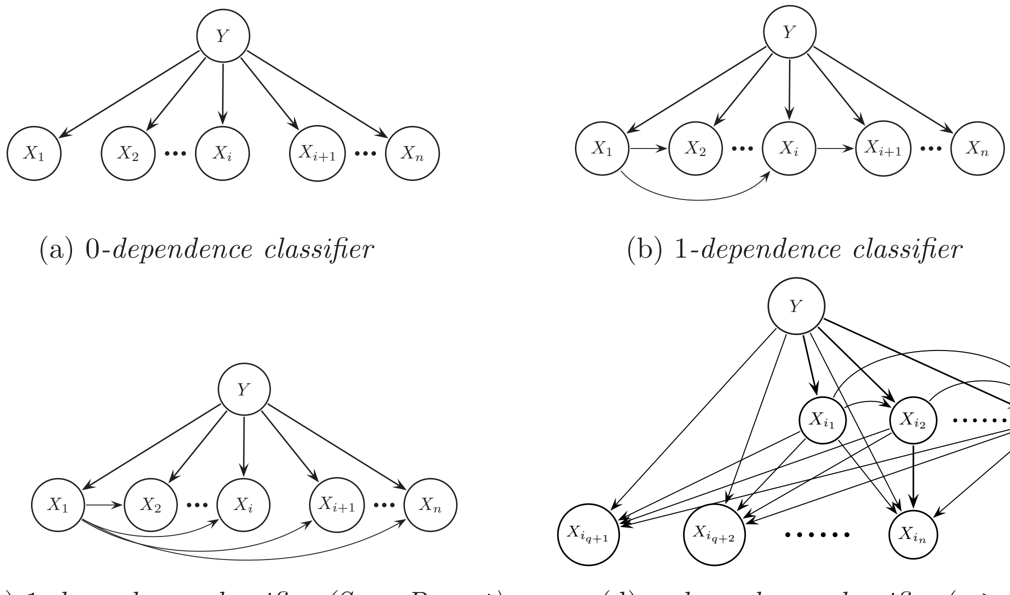
The training time complexity of SP-TAN and NBTree is cubic in the number of attributes and that of APNB is quadratic in the number of instances and classes. BSEJ has high training time complexity of $O(tkn^3+kv')$. Hence, BSEJ, SP-TAN and NBTree have very high training time if the number of attributes is large and APNB if the number of instances and classes are large. Since IR needs to sort training instances, it also has high training time if the number of instances is large. When the cardinalities of hidden variables are large, the training time cost of HierNB is high. The classification time complexity of these algorithms is linear in the number of classes and attributes. Therefore, once models are generated, these methods can classify test instances efficiently.

LBR and LWNB have identical training time complexity to NB, and hence they are highly efficient when few instances are to be classified. In other words, the relative efficiency is maximal when the number of classifications is low. However, the high classification time complexity of LBR, $O(tkn^3)$, hampers its application when large numbers of instances are to be classified. The classification time complexity of LWNB, $O(tn + kn)$, is higher than that of the other methods except LBR, NB^{SR}, AODE, MAPLMG, AODE^{SR} and HidNB. As t is usually considerably greater than kn, LWNB has substantially higher classification time relative to NB^{SR}, AODE, MAPLMG, AODE^{SR} and HidNB in most cases.

The training time complexity of MAPLMG, BSE and FSS is relatively higher than that of TAN, NB^{SR}, AODE, AODE^{SR}, IB and HidNB, whose training time complexities are moderate. NB^{SR}, AODE, AODE^{SR}, MAPLMG and HidNB have high classification time when the number of attributes are large, for example, in text classification. However, for many classification tasks with moderate or small number of attributes, their classification time complexity is modest. BSEJ and HierNB have very high space complexity.

3.7 Bayesian Network Perspective

From the Bayesian Network perspective, the methods discussed except BSEJ, HierNB and HidNB can be classified into three groups, as depicted in Figure 2. The first group (Figure 2



(a) 0-*dependence classifier*

(b) 1-*dependence classifier*

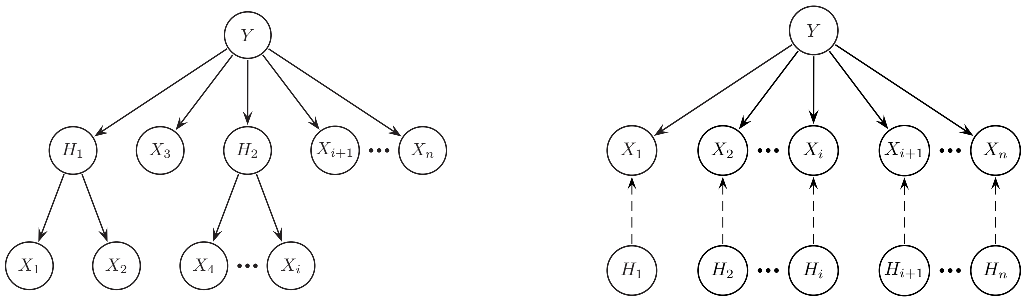
(c) 1-*dependence classifier (SuperParent)*

(d) z-*dependence classifier* $(z \geq 0)$

Figure 2: Bayesian Network

(a)) only allows each attribute to depend on the class. Examples include NB, BSE, FSS, NB^{SR}, LWNB, IR, APNB and IB. TAN, SP-TAN, AODE, MAPLMG and AODE^{SR} belong to the second group (Figure 2 (b) and (c)), in which each attribute depends on the class and at most one other attribute. For instance, in graph (b), attribute X_2 and X_i depend on attribute X_1, X_{i+1} depends on X_i and so forth. AODE, MAPLMG and AODE^{SR} are special types of 1-dependence classifiers, in which all attributes depend on the class and SuperParent, such as attribute X_1 in Figure 2 (c). The third group assumes independence among fewer attributes by allowing each attribute to depend on the class and at most z ($z \geq 0$) other attributes. In Figure 2 (d), independence is assumed among attributes in $O = \{X_{i_{q+1}}, \ldots, X_{i_n}\}$ given the class, and these attributes depend on all the attributes in $P = \{X_{i_1}, \ldots, X_{i_q}\}$. For NBTree, P is the set of the splitting attributes on the path leading to the leaf, and O is the set of leaf attributes. For LBR, P is the set of attributes in the antecedent, and O is the set of attributes in the consequent.

BSEJ, HierNB and HidNB create hidden variables and do not fall into the above groups. BSEJ and HierNB join attributes, including original attributes and also new compound attributes. In Figure 3 (a), two hidden variables are introduced into NB. H_1 combines X_1 and X_2. H_2 combines several attributes (from X_4 to X_i). HidNB creates a hidden parent for each attribute (Figure 3 (b)).



(a) *BSEJ and HierNB* (b) *HidNB*

Figure 3: Hidden Variables

4. Logistic Regression and Its Extension

Discriminative classifiers directly estimate the parameters of the conditional distribution $P(Y \mid X)$, without considering marginal distribution of $P(X)$. Generally, discriminative classifiers have lower bias and higher variance than their generative counterparts. A comparison of generative and discriminative learning can be found in (Rubinstein and Hastie, 1997; Ng and Jordan, 2001; Klein and Manning, 2002).

Logistic Regression forms linear models for classification (McLachlan, 1992; Mitchell, 2005). It directly estimates the posterior class probability by fitting the training data to a logistic curve and assigns the class with the highest posterior probability to the test instance. It also assumes that attributes are independent given the class, and hence it is the discriminative counterpart of NB. In the case where Y is a boolean variable, the Logistic Regression model is defined as

$$P(Y = 1 \mid X) = \frac{1}{1 + \exp(\mathrm{w}_0 + \sum_{i=1}^{n} \mathrm{w}_i X_i)}$$

and

$$P(Y = 0 \mid X) = \frac{\exp(\mathrm{w}_0 + \sum_{i=1}^{n} \mathrm{w}_i X_i)}{1 + \exp(\mathrm{w}_0 + \sum_{i=1}^{n} \mathrm{w}_i X_i)},$$

where w_i is the parameter to be estimated $(0 \leq i \leq n)$. The vector of parameters $W = \langle \mathrm{w}_0, \ldots, \mathrm{w}_n \rangle$ is usually fit to maximize the conditional log-likelihood:

$$W \leftarrow \underset{W}{\mathrm{argmax}} \sum_{o=1}^{t} \ln P(Y^o \mid X^o, W),$$

where Y^o and X^o are respectively the observed Y value and X value in the oth training instance. A commonly used method for this maximization problem is *gradient ascent* (Avriel, 2003; Pedregal, 2004). The parameters are initialized to zero and continually updated in the direction of the gradient until a global maximum is reached:

$$W \leftarrow W + \eta \frac{\partial \sum_{o=1}^{t} \ln P(Y^o \mid X^o, W)}{\partial W}, \tag{11}$$

where η is the learning rate.

Roos et al. (2005) prove that for Bayesian network structures satisfying a certain graph-theoretic property the conditional log-likelihood is a concave function of the parameters. Therefore, the globally optimal parameters of the conditional log-likelihood can be obtained by simple local optimization methods. For an arbitrary Bayesian network structure B, they define a corresponding canonical form B^* which is the Markov blanket of Y in B, with arcs added to make all parents of Y fully connected. They show that if, in B^*, all nodes having a common child are connected, the problem of seeking the parameters of B is equivalent to a Logistic Regression problem.

Greiner and Zhou (2002) propose a general discriminative learning algorithm, Extended Logistic Regression (ELR), that can be applied to arbitrary Bayesian network structures. It uses gradient descent and line search to directly maximize the conditional log-likelihood. ELR initializes the parameters to frequency estimates in the training data and uses cross-validation to estimate the optimal number of iterations for the gradient descent. Their experimental results show that ELR is effective and often outperforms its generative counterpart. Grossman and Domingos (2004) introduce BNC, an algorithm that learns the structure of a Bayesian network classifier by maximizing conditional likelihood. For computational efficiency, it trains the parameters by maximizing joint likelihood. Jing et al. (2005) propose an efficient approach to discriminative training of Bayesian networks, called Boosted Augmented Naive Bayes (BAN). It operates by greedily adding edges into NB until the addition of edges does not improve conditional log-likelihood. Starting from NB, at iteration u, it adds u edges with the highest conditional mutual information into NB. AdaBoost is used to boost the resulting classifier, where the parameters are estimated by maximizing joint likelihood and an upper bound on the negative conditional log-likelihood is minimized. They report that BAN has competitive classification accuracy with ELR and BNC.

There are many other approaches to estimating the parameters. Ridgeway et al. (1998) introduce a weight of evidence formulation for Boosted NB (Elkan, 1997) to modify the probability estimate of $P(X_i \,|\, Y)$. Nigam et al. (1999) use maximum entropy to estimate probability distributions in context of text classification. Wettig et al. (2002) employ a parameter transformation to make the conditional likelihood a concave function of the parameters, and use local optimization methods to find the global maximum conditional likelihood parameters. Jiang et al. (2004) estimate conditional probabilities by minimizing a smoothed classification error in the training data.

5. Comparison of Fifteen Methods

In this section, the performance of NB and twelve semi-naive Bayesian algorithms will be analyzed in terms of classification error, bias, variance, root mean squared error (RMSE), training time and classification time on sixty natural domains from the UCI Repository of machine learning (Newman et al., 1998). These semi-naive Bayesian algorithms are BSE, TAN, SP-TAN, NBTree, LBR, AODE, MAPLMG, AODE^{SR}, LWNB, IR, BSEJ and HidNB. We do not compare all the semi-naive Bayesian algorithms introduced in Section 3, as the power of the Friedman and Nemenyi tests we used is low with a large number of algorithms. In our data collection, BSE outperforms FSS in classification accuracy (refer to Section 5.3), therefore, FSS is not included in these comparisons. We select IR as a representative method for the fourth group. Because we cannot obtain the code of HierNB, this method is not included. To provide a baseline for comparison, we also compare these twelve methods to Logistic Regression and LibSVM with parameter search. Table 2 summarizes the characteristics of each data set used in this research, including the number of the instances, attributes and classes.

5.1 Experimental Methodology

These experiments compare algorithms implemented in the Weka workbench (version 3-5-7) (Witten and Frank, 2005) on the data sets described in Table 2. Each algorithm is tested on each data set using a 50-run 2-fold cross validation. The Friedman test and Nemenyi test with 0.05 level of significance are employed to evaluate the performance of algorithms, including classification error, bias, variance, RMSE, training time and classification time. Experiments on the algorithms except LBR and LibSVM were performed on a dual-processor 1.7 GHz Pentium 4 Linux computer with 2 Gb RAM. LBR and LibSVM were executed on a Linux Cluster based on Xeon 2.8 GHz CPUs.

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 (Kong and Dietterich, 1995; Breiman, 1996; Kohavi and Wolpert, 1996; Friedman, 1997; Webb, 2000). Unfortunately, we cannot in general minimize both simultaneously. There is a bias-variance tradeoff such that bias typically increases when variance decreases and vice versa. Algorithms that form models with few parameters, such as NB, usually have low variance in that they are insensitive

Table 2: Data sets

No.	Domain	Case	Att	Class	No.	Domain	Case	Att	Class
1	Abalone	4177	8	3	31	Lung Cancer	32	56	3
2	Adult	48842	14	2	32	Lymphography	148	18	4
3	Annealing	898	38	6	33	MAGIC Gamma Telescope	19020	10	2
4	Audiology	226	69	24	34	Mushrooms	8124	22	2
5	Auto Imports	205	25	7	35	Nettalk(Phoneme)	5438	7	52
6	Balance Scale	625	4	3	36	New-Thyroid	215	5	3
7	Breast Cancer (Wisconsin)	699	9	2	37	Nursery	12960	8	5
8	Car Evaluation	1728	7	4	38	Optical Digits	5620	48	10
9	Contact-lenses	24	4	3	39	Page Blocks	5473	10	5
10	Contraceptive Method Choice	1473	9	3	40	Pen Digits	10992	16	10
11	Credit Screening	690	15	2	41	Pima Indians Diabetes	768	8	2
12	Cylinder Bands	540	39	2	42	Postoperative Patient	90	8	3
13	Dermatology	366	34	6	43	Primary Tumor	339	17	22
14	Echocardiogram	131	6	2	44	Promoter Gene Sequences	106	57	2
15	German	1000	20	2	45	Segment	2310	19	7
16	Glass Identification	214	9	3	46	Sick-euthyroid	3772	29	2
17	Haberman's Survival	306	3	2	47	Sign	12546	8	3
18	Heart Disease (Cleveland)	303	13	2	48	Solar Flare	1389	9	2
19	Hepatitis	155	19	2	49	Sonar Classification	208	60	2
20	Horse Colic	368	21	2	50	SPAM E-mail	4601	57	2
21	House Votes 84	435	16	2	51	Splice-junction Gene Sequences	3190	61	3
22	Hungarian	294	13	2	52	Syncon	600	60	6
23	Hypothyroid(Garavan)	3772	29	4	53	Teaching Assistant Evaluation	151	5	3
24	Ionosphere	351	34	2	54	Tic-Tac-Toe Endgame	958	9	2
25	Iris Claasification	150	4	3	55	Vehicle	846	18	4
26	King-rook-vs-king-pawn	3196	36	2	56	Volcanoes	1520	3	4
27	Labor negotiations	57	16	2	57	Vowel	990	13	11
28	LED	1000	7	10	58	Waveform-5000	5000	40	3
29	Letter Recognition	20000	16	26	59	Wine Recognition	178	13	3
30	Liver Disorders (Bupa)	345	6	2	60	Zoo	101	16	7

to data variations, and high bias because the simple models generated generally underfit the data. In contrast, algorithms that learn models that are highly parameterized, such as NBTree, usually have low bias because they can generate complex models to fit the training data closely, and high variance for the reason that the models they create differ substantially between different training samples. In general, the better the learner is able to fit the training data, the lower the bias. However, closely fitting the training data may result in greater changes in the models formed from sample to sample, and hence higher variance.

There are a number of different bias-variance decomposition definitions (Kong and Dietterich, 1995; Breiman, 1996; Kohavi and Wolpert, 1996; Friedman, 1997; Webb, 2000). 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:

$$bias^2 = \frac{1}{2}\sum_{y\in Y}(P(Y = y \mid \mathbf{X} = \mathbf{x}) - P(\mathcal{C}(\mathbf{x}) = y))^2$$

and

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

where $\mathcal{C}$ is a classifier.

In order to maximize the variation in the training data from trial to trial we use two-fold cross validation. The training data is first randomized. Then, it is randomly divided into two folds, $fold_1^i$ and $fold_2^i$. $Classifier_1^i$ is generated from $fold_1^i$ and $Classifier_2^i$ is generated from $fold_2^i$. $fold_1^i$ and $fold_2^i$ are respectively used as a test set for $Classifier_2^i$ and $Classifier_1^i$. In this manner, each available instance is classified once for each two-fold cross-validation. In order to give a more accurate estimation of the average performance of an algorithm, this process is repeated 50 times which produces 100 folds. Bias, variance, error and RMSE are estimated by evaluation of the predictions of $Classifier_1^i$ and $Classifier_2^i$ when applied to $fold_2^i$ and $fold_1^i$ for $1 \leq i \leq 50$.

In Kohavi and Wolpert's method (Kohavi and Wolpert, 1996), the default bias-variance estimation method in Weka, the randomized training data are divided into a training pool and a test pool randomly. 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.

The repeated cross-validation bias-variance estimation method is preferred to Kohavi and Wolpert's method as it results in the use of substantially larger training sets. If more than two folds are used, the multiple classifiers are trained from training sets with large overlap, and hence the generation of variance is compromised. In addition, every case in the training data is used the same number of times for both training and testing.

5.1.2 Statistics Employed

We use the Friedman test and Nemenyi test to compare the performance of multiple algorithms and Win/Draw/Loss record to compare the performance of two algorithms. Mean metrics across all data sets are also employed to provide a simplistic overall measure of relative performance. In addition, we use the Spearman's rank correlation test to examine whether the bias proportion of error on large data sets is higher than that on small data sets.

Friedman Test. Demšar (2006) recommends the Friedman test (Friedman, 1937, 1940) for comparisons of multiple algorithms over multiple data sets. It first calculates the ranks of algorithms for each data set separately (average ranks are assigned if there are tied values), and then compares the average ranks of algorithms over data sets. The null-hypothesis is that there is no difference in average ranks. If the null-hypothesis is rejected then it is probable that there is a true difference in the average ranks of at least two algorithms. Post-hoc tests are used to determine which pairs of algorithms have significant differences. We reject the null-hypothesis if the Friedman statistic derived by Iman and Davenport (1980) is lager than the critical value of the F distribution with $a - 1$ and $(a - 1)(D - 1)$ degrees of freedom for $\alpha = 0.05$.

Nemenyi Test. If the Friedman test rejects the null-hypothesis, the Nemenyi test is used to further analyze which pairs of algorithms are significant different. Let d_i^j be the difference between *ith* algorithm and *jth* algorithm. We assess a difference between *ith* algorithm and *jth* algorithm as significant if $d_i^j >$ *critical difference* (CD):

$$CD = q_{0.05}\sqrt{\frac{a(a+1)}{6D}},$$

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 $\sqrt{2}$.

Win/Draw/Loss Record. When two algorithms are compared, we count the number of data sets for which one algorithm performs better, equally or worse to 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.

Mean. The arithmetic mean across all data sets provides a gross indication of relative performance and adjunct to other statistics.

Spearman's Rank Correlation Test. We use this test to assess whether there is a relationship between the mean bias proportion of error across large data sets and small data sets over a algorithms. Mean values across large and small data sets are separately converted to ranks (average rank is used if two or more values are equal). Let d_i be the difference between the ranks on the ith out of a algorithms. The Spearman's rank correlation coefficient is

$$r_s = 1 - \frac{6\sum_{i=1}^{a} d_i^2}{a(a^2-1)}.$$

We reject the null-hypothesis that there is no relationship between the mean values across large and small data sets if r_s is greater than the critical value for a two-tailed Spearman's rank correlation test at $\alpha = 0.05$.

5.1.3 Other Issues

This section explains other issues related to the experiments.

Probability Estimates. The base probabilities of each algorithm, except Logistic Regression and LibSVM, are estimated using m-estimation, as it often appears to lead to more accurate probabilities than Laplace estimation for NB and its variants we tested. Kohavi et al. (1997) report that using $m \ll 1$ usually has higher classification accuracy than using $m = 1$. In this comparison, we use $m = 0.1$.

Numeric Values and Missing Values. For all algorithms except Logistic Regression and LibSVM, quantitative attributes are discretized using MDL discretization (Fayyad and Irani, 1993) within each cross-validation fold. In keeping with Weka's Logistic Regression, missing values for qualitative attributes are replaced with modes and those for quantitative attributes are replaced with means from the training data.

LWNB and AODE^{SR}. We use Weka's implementation of LWNB and set the number of neighbors to 50, as this number is favorable to LWNB (Frank et al., 2003). The original AODE^{SR} uses 30 as the minimum frequency l of accepting the generalization relationship. An empirical selection of l ($l = 10, 20, \cdots, 150$) reveals that the error of AODE can be significantly reduced by the addition of SR at all settings of l except 10 and 20 when Laplace-estimation is used, while at all settings of l except 10, 20, 30, 40 and 50 when m-estimation is employed. Therefore, in this study, we set the minimum frequency to 100.

Logistic Regression and LibSVM. We use Weka's implementation and default setting of Logistic Regression. The results of LibSVM using Weka's implementation and default setting with the exception of turning on normalization of data (recommended in (Hsu et al., 2003a)) are poor on many data sets. For instance, the errors on 2 data sets, Pen Digits and Syncon, are greater than 0.8, and errors on 13 data sets are greater than 0.5. This necessitates the use of parameter search. We perform a "grid-search" on C and γ for the RBF kernel using 5-fold cross-validation (Hsu et al., 2003a). Each pair of (C, γ) is tried ($C = 2^{-5}, 2^{-3}, \ldots, 2^{15}, \gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$), and the one with the lowest cross-validation error is selected. However, this process has very high time complexity. For this reason, the results of LibSVM on Adult, Letter Recognition, Optical Digits and SPAM E-mail are obtained from five runs of two-fold cross-validation.

LBR and BSEJ. As LBR has very high classification time complexity, the results of LBR on the two largest data sets (Adult and Letter Recognition) are obtained from five runs of two-fold cross-validation. BSEJ has a memory shortage problem on Letter Recognition and Cylinder-bands. To estimate the results of BSEJ, we average the results of $i - 1$ iterations if the ith run is the first iteration that has the problem.

5.2 Experimental Results

Following the graphical presentation of Demšar, we show the comparison of algorithms against each other with the Nemenyi test on each metric. We plot the algorithms on the left line according to their average ranks, which are indicated on the parallel right line. Critical difference (CD) is also presented in the graphs. The lower the position of algorithms, the lower the ranks they obtain, and hence the better the performance. Algorithms are connected by a line if their differences are not significant. Since the comparison involves 15 algorithms, the power of the Nemenyi test is low and so only large effects are likely to be apparent.

Section 5.2.1 presents the results of error, bias, variance and RMSE. Section 5.2.2 discusses how data set size interacts with bias and variance, which in turn affects error. The training and test time results are presented in Section 5.2.3. Detailed error, bias, variance and RMSE by data sets are presented in the Appendix A.

5.2.1 Error, Bias, Variance and RMSE

With 15 algorithms and 60 data sets, the Friedman statistic is distributed according to the F distribution with $a - 1 = 15 - 1 = 14$ and $(a - 1)(D - 1) = (15 - 1) * (60 - 1) = 826$ degrees of freedom. The critical value of $F(14, 826)$ for $\alpha = 0.05$ is 1.7037. The Friedman statistics for error, bias and variance in our experiments are 7.1621, 10.2482 and 13.6646 respectively, and hence we reject all the null-hypotheses. The critical difference for $\alpha = 0.05$ is CD $= 3.392 * \sqrt{a(a+1)/(6*D)} = 3.392 * \sqrt{15(15+1)/(6*60)} = 2.7696$. We show the comparison of the 15 algorithms against each other using the Nemenyi test on error, bias and variance in Figures 4 (a), 5 (a) and 5 (b) respectively.

As LibSVM does not provide probability estimates, we present RMSE results (Figures 4 (b)) for all the algorithms except LibSVM. The null-hypothesis that there is no difference in average RMSE ranks is rejected because the Friedman statistic for RMSE (12.2775) is greater than $F(13, 767)$ for $\alpha = 0.05$ (1.7329). The critical difference of 14 algorithms and 60 data sets for $\alpha = 0.05$ is 2.5617. We use the abbreviation "Logistic" to denote Logistic Regression in each graph.

Error

MAPLMG achieves the lowest mean error rank (5.3917). AODE^{SR} comes next (6.1500). AODE and LBR obtain the third and fourth lowest mean rank of error (rank being 6.2500 and 6.3083 respectively). IR and TAN have the highest and second highest mean rank of error (rank being 10.3583 and 10.0417 respectively).

Four semi-naive Bayesian algorithms, MAPLMG, AODE^{SR}, AODE and LBR outperform NB on the Nemenyi test. MAPLMG enjoys a significant error advantage over NBTree, HidNB, Logistic Regression, BSE, NB, TAN and IR and shares a similar level of error with the rest of algorithms. AODE^{SR} has lower mean error rank than AODE, LBR, LibSVM, SP-TAN, LWNB, BSEJ, NBTree and HidNB and significantly lower mean error rank than Logistic Regression, BSE, NB, TAN and IR. The mean error rank differences between AODE and BSE, NB, TAN and IR are statistically significant, and those between AODE and the remaining algorithms are not. LBR has a significant error advantage over NB, TAN and IR, and does not have a significant disadvantage relative to any algorithm.

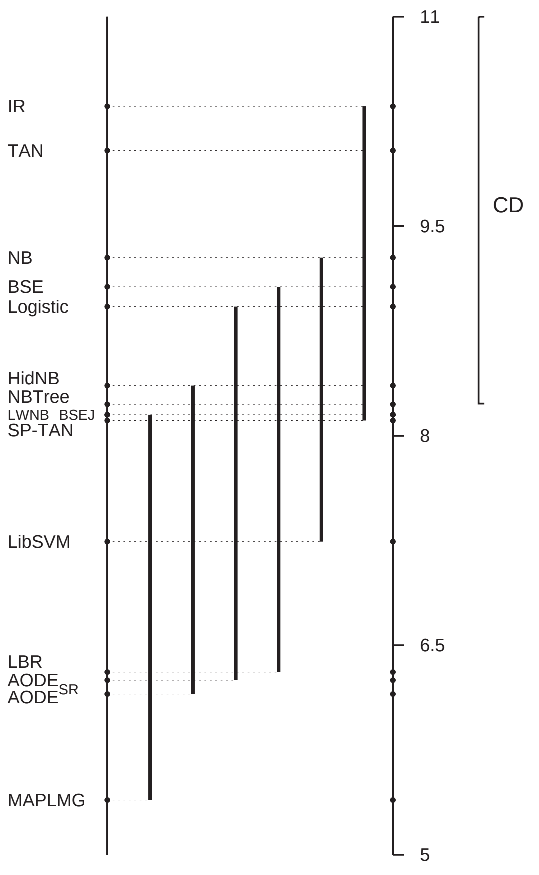
LibSVM has lower mean error rank compared to all the other algorithms except MAPLMG, AODE^{SR}, AODE and LBR, but only has a significant error advantage relative to TAN and IR. SP-TAN, BSEJ and LWNB have higher mean error ranks than MAPLMG, AODE^{SR}, AODE, LBR and LibSVM and lower mean error ranks than all the remaining algorithms. However, the Nemenyi test does not reveal any of these differences as statistically significant. LWNB and BSEJ have identical mean error rank (8.1500). The only significant difference in error performance between either NBTree or HidNB and another algorithm is with MAPLMG.

IR and TAN have a significant error disadvantage relative to LibSVM, LBR, AODE, AODE^{SR} and MAPLMG, and share a similar level of error with all the other methods. BSE has significantly higher mean error rank than MAPLMG, AODE^{SR} and AODE and Logistic Regression has significantly higher mean error rank than MAPLMG and AODE^{SR}.

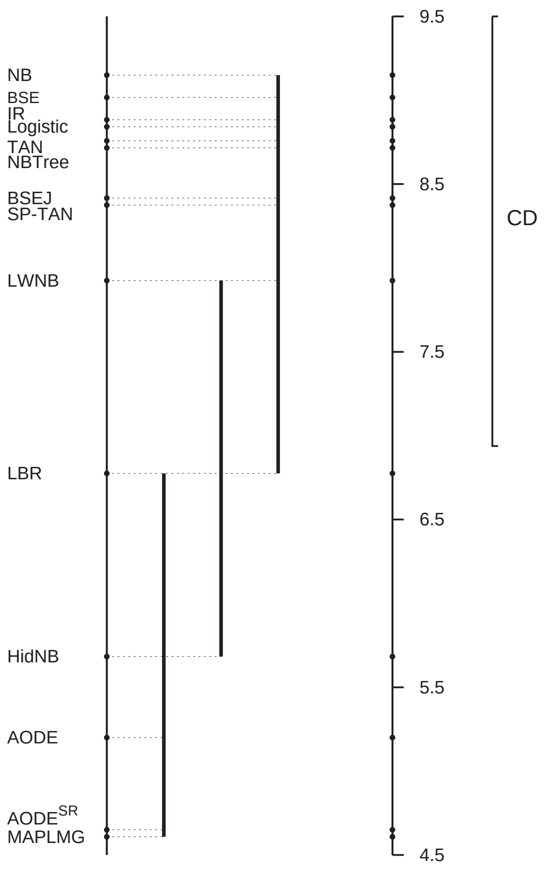
RMSE

MAPLMG delivers the lowest mean rank of RMSE (4.6083). AODE^{SR} is a close second (4.6500), followed by AODE (5.2000). The Nemenyi test indicates that all three enjoy a significant RMSE advantage relative to all the rest of algorithms except HidNB and LBR.

The differences between HidNB and SP-TAN, BSEJ, NBTree, TAN, Logistic Regression, IR, BSE and NB are statistically significant. LBR has lower mean RMSE rank than all the



(a) *Error*



(b) *RMSE*

Figure 4: (a) Error comparison of the 15 algorithms with the Nemenyi test on 60 data sets. CD = 2.7696. (b) RMSE comparison of the 14 algorithms with the Nemenyi test on 60 data sets. CD = 2.5617.

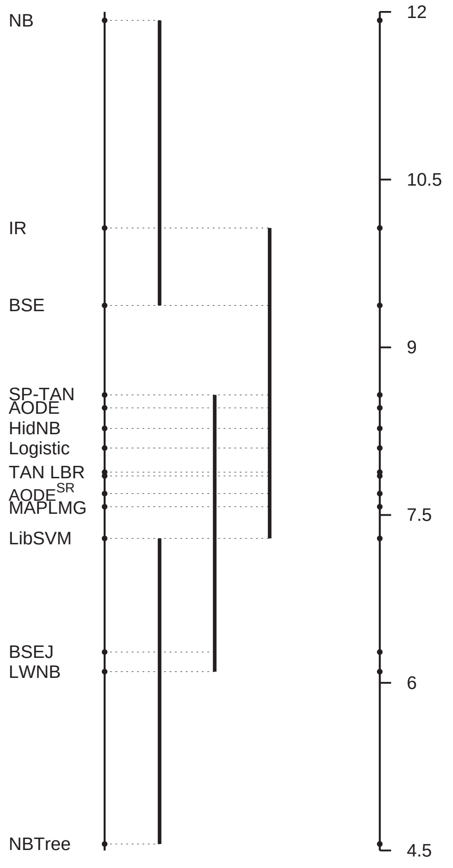
other methods but HidNB, AODE, AODE^{SR} and MAPLMG. However, all these differences are not significant.

All semi-naive Bayesian methods improve upon the probability estimation accuracy of NB, but these improvements are statistically significant for only four methods, MAPLMG, AODE^{SR}, AODE and HidNB.

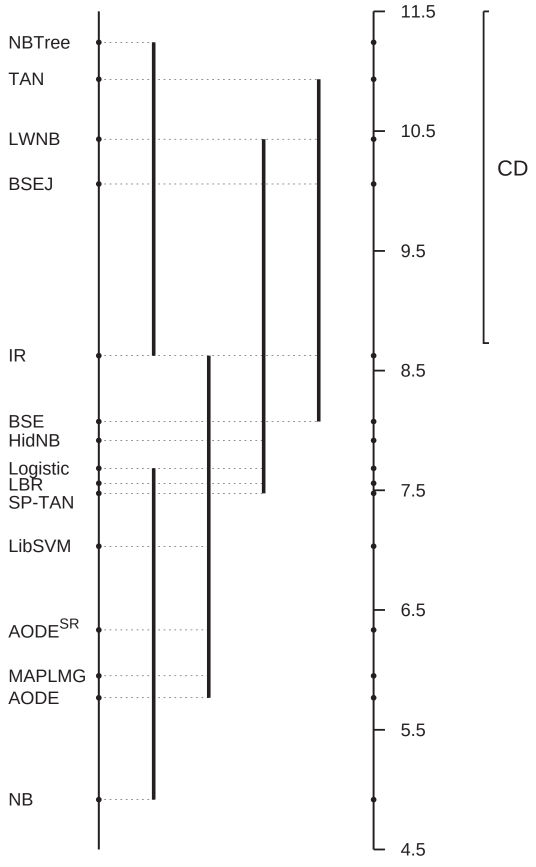
Bias

NBTree comes out ahead when mean bias ranks are compared (rank being 4.5583). It enjoys a significant advantage over all the other algorithms except LWNB, BSEJ and LibSVM.

LWNB and BSEJ achieve the second (6.1000) and third (6.2750) lowest mean bias ranks, which are significantly lower than that of BSE, IR and NB. LibSVM shares a similar level



(a) *Bias*



(b) *Variance*

Figure 5: Bias and Variance comparison of the 15 algorithms with the Nemenyi test on 60 data sets. CD = 2.7696.

of bias with all the other methods but NB. MAPLMG, AODE^{SR}, LBR, TAN, Logistic Regression, HidNB, AODE and SP-TAN have significantly lower mean bias ranks than NB and significantly higher mean bias ranks than NBTree. The mean bias rank of BSE and IR are lower than that of NB, higher than that of all the other methods, and significantly higher than that of NBTree, LWNB and BSEJ.

All semi-naive Bayesian methods reduce the bias of NB. Ten semi-naive Bayesian methods (all those except IR and BSE), provide significant bias reduction in NB.

Variance

NB and AODE achieve the lowest and second lowest mean rank of variance (rank being 4.9167 and 5.7667 respectively). NB enjoys a significant advantage over HidNB, BSE, IR, BSEJ, LWNB, TAN and NBTree. The differences between NB and AODE, MAPLMG, AODE^{SR}, LibSVM, SP-TAN, LBR and Logistic Regression are not significant. AODE, MAPLMG, AODE^{SR} and LibSVM have significantly lower mean rank than BSEJ, LWNB, TAN and NBTree.

The Nemenyi test differentiates SP-TAN, LBR and Logistic Regression from TAN and NBTree, but not from the rest of the algorithms. HidNB has a significant disadvantage relative to NB and a significant advantage over TAN and NBTree. BSE has a significantly lower mean variance rank than NBTree and a significantly higher rank than NB.

NBTree has the highest mean rank of variance (11.2417). It has a significant variance disadvantage relative to all the rest of algorithms except TAN, LWNB, BSEJ and IR. TAN has the second highest mean rank of variance (10.9333). The differences between TAN and all the remaining algorithms but NBTree, LWNB, BSEJ, IR and BSE are significant. LWNB and BSEJ have significantly higher mean variance ranks than NB, AODE, MAPLMG, AODE^{SR} and LibSVM. The Nemenyi test differentiates IR from NB, but does not differentiate it from any other methods.

5.2.2 Bias and Variance: in Relation to the Size of Data Sets

In this section, we discuss how data set size interacts with bias and variance, which in turn affects error. It is quite likely that differences between small samples are greater than those between large samples. In other words, differences between samples are expected to decrease with increasing sample size. It follows that differences between models formed from those samples are expected to decrease and hence variance is expected to decrease (Brain and Webb, 2002). Geurts (2002) reported that the behaviors of bias with respect to the sample size is algorithm dependent. In his study, the bias of linear regression is independent of sample size, while decision trees decrease in bias with increasing sample size. Brain and Webb (2002) observed that the bias of C4.5 and its variants tends to decrease, while that of NB increases on all data sets tested except Waveform.

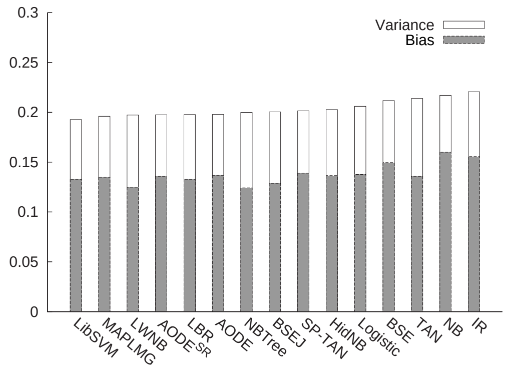
If variance decreases as training set size increases, the bias proportion of error may be higher on large data sets than on small data sets and the variance proportion of error may be higher on small data sets than on large data sets. In consequence, low bias algorithms may have advantage in error on large data sets and low variance algorithms on small data sets (Brain and Webb, 2002).

To assess whether the bias proportion of error is higher on large data sets than small data sets, we compare the results of bias as a proportion of error on the twenty-four largest data sets, each with 1000 or more cases, and on the thirty-six smallest data sets, each with less than 1000 cases. Spearman's rank test is used to test whether the difference between the results on large data sets and small data sets over the 15 algorithms is non-random. We assess a difference as significant if $r_s = 1 - 6\sum d_i^2/a(a^2-1) > 0.525$, where a is 15, d_i is the difference between the rank of mean value on the two sets of data sets on the ith out of 15 algorithms, and 0.525 is the critical value for a two-tailed test at a significance level of 0.05.

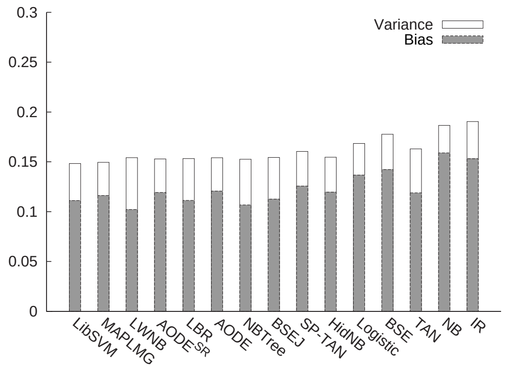
Table 3 presents the mean bias proportion of error on the 24 largest data sets and the 36 remaining data sets over 15 algorithms. Figure 6 presents the mean bias proportion on all, the large and the small data sets respectively. From Table 3 and subgraph (b) and (c) of Figure 6 we can see that each mean bias proportion on the large data sets is higher than

Table 3: Mean bias proportion of error on 24 largest and 36 smaller data sets

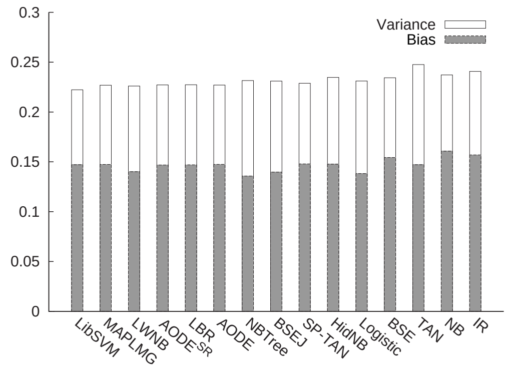
Bias/Error	NB	BSE	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	AODE^{SR}	LWNB	IR	BSEJ	HidNB	Logistic	LibSVM
large data	0.836	0.776	0.705	0.729	0.571	0.639	0.750	0.750	0.747	0.596	0.790	0.640	0.769	0.742	0.631
Rank	15	13	6	7	1	4	10	11	9	2	14	5	12	8	3
small data	0.671	0.645	0.572	0.637	0.555	0.638	0.636	0.635	0.632	0.583	0.635	0.592	0.606	0.576	0.622
Rank	15	14	2	12	1	13	11	9	8	4	10	5	6	3	7



(a) *Mean error on 60 data sets*



(b) *Mean error on 24 largest data sets*



(c) *Mean error on 36 smallest data sets*

Figure 6: Mean error. (Algorithms are sorted in ascending order on the mean error on 60 data sets.)

that on small data sets. The r_s value for the Spearman's rank test is $0.596 > 0.525$, which suggests that bias accounts for a larger proportion of error on large data sets.

Figure 7 presents a comparison of the error of the 15 algorithms against each other on the 24 largest data sets using an Nemenyi test (CD = 4.3791). NBTree, LWNB, BSEJ, LibSVM (the four methods have the lowest bias in Figure 5 (a)) share a similar level of error with NB on all 60 data sets, while they achieve significantly lower mean error ranks than NB on the 24 largest data sets, despite the power of the test being substantially lower

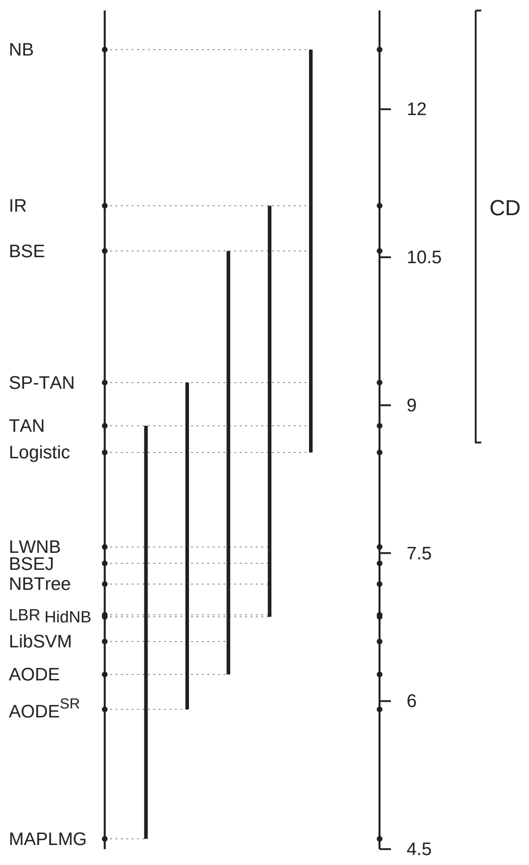


Figure 7: Error comparison of the 15 algorithms with the Nemenyi test on 24 largest data sets, CD = 4.3791.

due to its use of fewer data points. Compared with NB, IR and TAN have lower mean bias ranks and higher mean error tanks on all 60 data sets, while they deliver lower mean error ranks on the 24 largest data sets. These results correspond well to the expectation that algorithms with a low bias profile, such as NBTree, tend to have lower relative error on large training sets and algorithms with a low variance profile, such as NB, tend to have lower relative error on small training sets.

When AODE, LBR and LibSVM are compared with TAN, MAPLMG is compared with NBTree, Logistic Regression, HidNB and TAN, AODE^{SR} is compared with Logistic Regression and TAN and LBR is compared with IR and on all 60 data sets, the former of each pair has a significant advantage in error relative to the latter. Nonetheless, the differences between the former of each pair and the latter are not significant on the 24 largest data sets. One possible reason is that fewer data sets reduce the power of the Nemenyi test to differentiate these methods. For low bias algorithms, another possible reason is that they appear to have lower error on large data sets and hence the comparison on the 24 largest data sets is favorable to them. Note that eight semi-naive Bayesian algorithms, MAPLMG, AODE^{SR}, AODE, LBR, HidNB, NBTree, BSEJ and LWNB, provide a significant error reduction relative to NB on the 24 largest data sets.

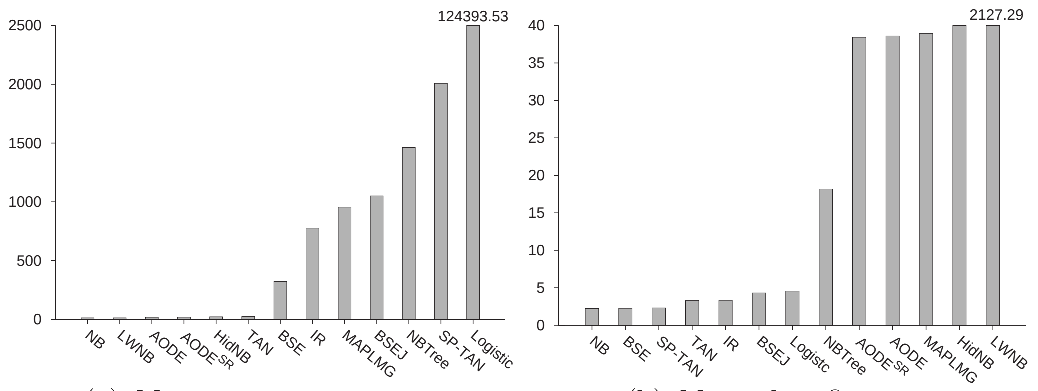
In an earlier study (Zheng and Webb, 2005) which compared NB, AODE, NBTree, LBR, TAN, SP-TAN, BSEJ, BSE and FSS, AODE was the only algorithm to have a significant advantage in error over NB. However, four semi-naive Bayesian algorithms, MAPLMG, AODE^{SR}, AODE and LBR, achieve a significant advantage in error over NB in the current study. AODE, LBR and SP-TAN achieve statistically significant win-draw-loss error records over NBTree in the previous study, while they share similar levels of error with NBTree in this study. The advantage of AODE over BSEJ is significant in the earlier study, but it is not in the current study. Although different comparison methods (the binomial sign test for the earlier study and the Nemenyi test for this study) are employed, the main reasons for the different outcomes might be that the previous study compared the algorithms on 36 data sets with a smaller average size and used Kohavi and Wolpert's bias-variance estimation technique, which produces smaller training sets than the technique employed by the current study. Hence, the earlier experiments might put algorithms with a low bias profile at a disadvantage.

5.2.3 Compute Time Results

Some caution is required in comparing compute times for execution of implementations of alternative learning algorithms, as there is always room for uncertainty to what extent differences in performance can be attributed to the relative efficiency with which the algorithms have been implemented as opposed to fundamental efficiencies in the actual algorithms. Nonetheless, empirical comparison of real world running time can provide a valuable adjunct to computational complexity analysis. We here present the results of such an empirical evaluation with the qualification that specific outcomes should be treated with caution.

As has been explained, LibSVM has very high training time and LBR has very high classification time. They were executed on a different machine, therefore, they are excluded from this compute time comparison. Because BSEJ has a memory shortage problem on two data sets (Letter Recognition and Cylinder-bands), we estimate training time for each data set by multiplying the total training time on $i-1$ iterations by $50/(i-1)$ if the ith iteration is the first iteration that has the problem. Classification time of BSEJ is estimated by the same approach.

The mean training and classification time across 60 data sets are presented in Figures 8 (a) and (b). These algorithms are sorted in ascending order on the mean metrics. To scale these graphs appropriately, we cut short the bars of Logistic Regression in (a) and LWNB in (b). The mean training time of Logistic Regression is 124393.53 seconds and the mean classification time of LWNB is 2127.29 seconds. Logistic Regression has very high training time on Nettalk (7327604.69 seconds), which dominates the mean training time. The training time comparison of 13 algorithms against each other using the Nemenyi test is shown in Figures 9 (a) and the classification time comparison is in Figures 9 (b). The critical difference of 13 algorithms and 60 data sets for $\alpha = 0.05$ is 2.3556.



(a) *Mean training time* (b) *Mean classification time*

Figure 8: Mean time across 60 data sets (seconds).

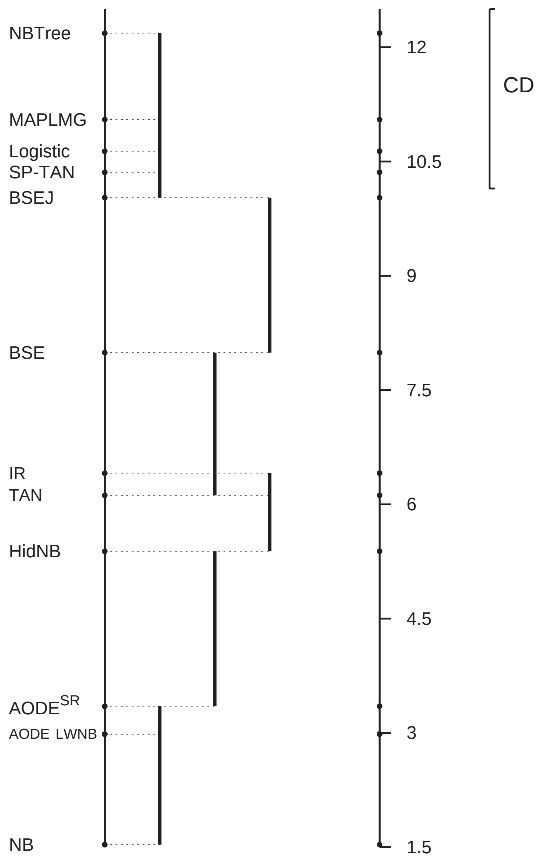
Training Time

NB has the lowest mean training time (13.53 seconds) and rank (1.5333). The Nemenyi test shows that it enjoys a significant advantage in training time over all the rest of algorithms except LWNB, AODE and AODE^{SR}. AODE and LWNB have identical rank (2.9833). They are significantly more efficient than all the other methods except NB and AODE^{SR}. The differences between AODE^{SR} and all other algorithms but NB, AODE, LWNB and HidNB are statistically significant. The differences between HidNB and AODE^{SR}, TAN and IR are not statistically significant. HidNB has significantly faster training than BSE, BSEJ, SP-TAN, Logistic Regression, MAPLMG and NBTree. TAN and IR have significantly slower training than NB, AODE, LWNB and AODE^{SR} and significantly faster training than BSEJ, SP-TAN, Logistic Regression, MAPLMG and NBTree. Although the Nemenyi test does not reveal the difference between IR and HidNB and between IR and TAN are significant, the mean training time differences are large. The mean training time of HidNB and TAN are 22.24 and 24.13 seconds respectively, while that of IR is 776.94 seconds. BSE is significantly faster than NBTree, MAPLMG, Logistic Regression and SP-TAN and significantly slower than NB, AODE, LWNB, AODE^{SR}, HidNB.

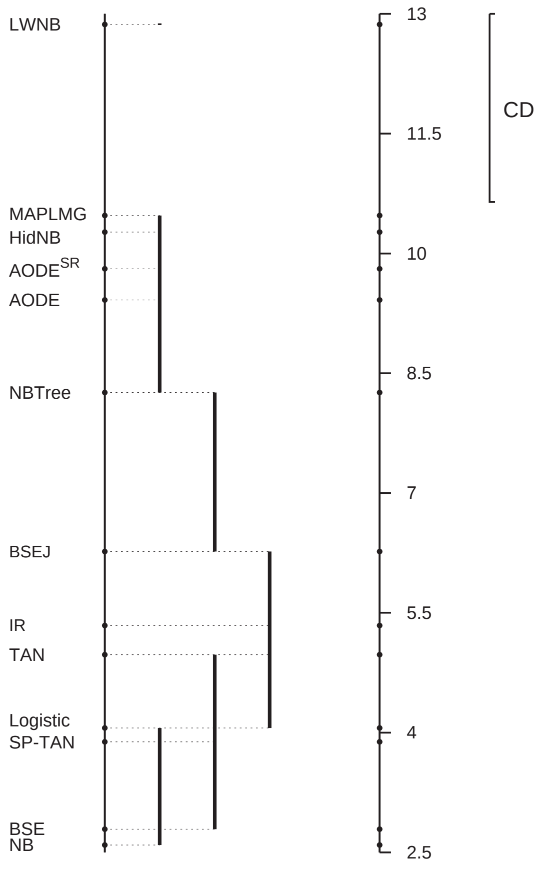
NBTree, MAPLMG, Logistic Regression and SP-TAN have high mean training time and ranks. The training time of these methods is significantly higher than all the other methods included in the comparison, other than BSEJ. Note, however, that LibSVM was excluded from the comparison due to extremely high training time. NBTree, SP-TAN and Logistic Regression have very high training time on data sets with large number of instances and attributes. On the 24 largest data sets, NBTree has the highest training time on 11 data sets, Logistic Regression on 7 data sets and SP-TAN on 4 data sets.

Classification Time

NB and BSE have the lowest mean classification time and ranks. NB has a clear and consistent advantage over all the other methods but BSE, SP-TAN and Logistic Regression. BSE and SP-TAN have lower mean classification time ranks than Logistic Regression and TAN, and significantly lower ranks than all the remaining methods except NB. Logistic



(a) *Training time*



(b) *Classification time*

Figure 9: Training and classification time comparison of 13 algorithms (excluding LBR and LibSVM) with the Nemenyi test on 60 data sets. CD = 2.3556.

Regression, TAN and IR provide significantly faster classification than NBTree, AODE, AODE^{SR}, HidNB, MAPLMG and LWNB. BSEJ is faster than NBTree and significantly faster than AODE, AODE^{SR}, HidNB, MAPLMG and LWNB.

LWNB has the highest mean classification time and rank. It is significantly slower than all the rest of algorithms included in the comparison, although it should be recalled that LBR was excluded because of its unduly high classification time. AODE, AODE^{SR}, HidNB and MAPLMG have significantly slower classification than all the remaining methods but LWNB and NBTree.

5.3 Discussion

NB does not perform model selection, using a fixed formula to classify test instances. This results in relatively low variance. The relaxation of the attribute independence assumption may make semi-naive Bayesian methods fit the training sample closer. Consequently, they may have lower bias, but higher variance compared with NB. If a semi-naive Bayesian method can find the right balance between bias and variance, it may deliver higher classification accuracy than NB.

NBTree and LBR

NBTree, which combines decision trees with NB, is a relatively highly parameterized method. It achieves the lowest mean bias rank, being significantly lower than all the other methods except LWNB, BSEJ and LibSVM. LBR establishes a rule in a lazy manner. This rule can be seen as a branch of the tree produced by NBTree. Thus, both approaches can produce models with as high a level of dependence as there are attributes. Because LBR uses lazy learning and hence can adjust its model to each test case, it is expected to have lower bias compared with NBTree. However, Figure 5 (a) shows that NBTree has significantly lower bias than LBR.

There are two main differences between these two methods. First, NBTree uses 5-fold cross validation accuracy estimation as the splitting criterion, while LBR uses Leave-One-Out cross validation accuracy estimation. Another difference between NBTree and LBR is the stopping criterion. The former stops the growth of the tree when the relative error reduction is less than 5% or the number of the instances in a splitting node is less than 30, and the latter stops the growth of the rule when there is no significant accuracy improvement as assessed by a sign test. It is quite likely that the sign test at a significance level of 0.05 is stricter than the criterion that the relative error reduction is greater than 5%. Hence, the latter might result in closer fitting compared to the former. For instance, if the best error so far is 4 and the current error is 0, the relative error reduction is 100% $>$ 5%, but this improvement fails the significance test.

To assess this expectation we compare LBR with original NBTree (NBTree$_{ns}^{5fold}$) and three variants of NBTree. The first variant, called NBTree$_{ns}^{LOO}$ uses the same stopping criterion as original NBTree and Leave-One-Out cross validation as the evaluation function. The second variant, called NBTree$_{s}^{5fold}$, uses the same splitting criterion as original NBTree and a sign test as the stopping criterion. NBTree$_{s}^{LOO}$ uses Leave-One-Out cross validation accuracy estimate as the splitting criterion and a sign test as the stopping criterion. Figure 10 presents the bias comparison of the five algorithms using the Nemenyi test (CD = 0.7875).

Both NBTree$_{ns}^{5fold}$ and NBTree$_{ns}^{LOO}$ enjoy lower mean bias rank compared with LBR. This result suggests that the splitting criterion is not the cause of the discrepancy from our expectation that LBR has lower bias compared with NBTree. In contrast, NBTree$_{s}^{5fold}$ and NBTree$_{s}^{LOO}$ have higher mean bias rank compared with LBR. Furthermore, NBTree$_{s}^{LOO}$, which uses the same splitting and stopping criteria as LBR, has a significant disadvantage relative to LBR. This result is consistent with the expectation that the relative error reduction constraint results in closer fitting to the data than does the sign test.

Since NBTree and LBR are z-dependence classifiers, they may have great potential to have lower bias on large data sets as these may have sufficient data to obtain accurate higher order probability estimates and hence to have appropriate model selection. As a consequence, these two algorithms may have an advantage in error on large data sets,

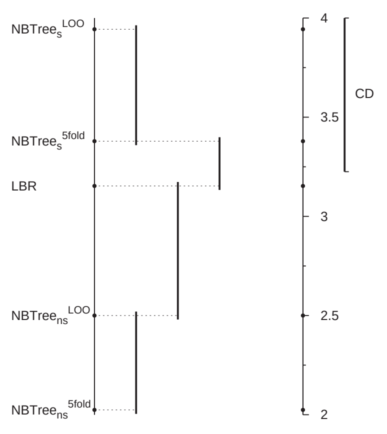


Figure 10: Bias comparison of NBTree, NBTree's variants and LBR with the Nemenyi test on 60 data sets. CD = 0.7875

however, they may not scale up well due to the high training time complexity of NBTree and high classification time complexity of LBR.

LWNB and LBR

Both LWNB and LBR can be considered as techniques that identify a relevant subset of the training set and learn a local NB therefrom. LWNB finds those training instances closest in Euclidean distance to the given test instance. The closer the distance between a training instance and the test instance, the higher weight the training instance obtains. LBR selects a subset of the training set using a rule in which all attributes in the antecedent have the same values as those of the test instance. An attribute is added to the antecedent if its removal from the consequent can mitigate the attribute interdependencies detected by leave-one-out cross validation on the current local NB.

Both of them use lazy learning. In consequence, they may have relatively low bias at the cost of considerably increased classification time and space. The Nemenyi test shows that LBR has substantially higher bias, lower variance, lower error and lower RMSE than LWNB on the full suite of 60 data sets. When LBR is compared with LWNB on the 36 smallest data sets, it outperforms LWNB in error, variance and RMSE on the win/draw/loss records. Table 4 and 5 present the win/draw/loss records for LBR against LWNB on the 36 smallest and 24 largest data sets respectively. Each win/draw/loss record is the number of data sets for which LBR obtains lower, the same and higher value on a corresponding metric than LWNB. On the 24 largest and 36 smallest data sets, the bias advantage of LWNB is clear. The error, variance and RMSE differences are small on the 24 largest data sets. These results suggest that LBR may be a more desirable option when small number of examples are to be classified, while LWNB may be a more appealing option for large data sets due to its low bias profile and considerably faster classification.

Table 4: Win/Draw/Loss: LBR vs. LWNB on the 36 smallest data sets

	LBR vs. LWNB	
	W/D/L	p
Error	23/1/12	**0.0448**
Bias	12/1/23	**0.0448**
Variance	25/1/10	**0.0083**
RMSE	23/1/12	**0.0448**

Table 5: Win/Draw/Loss: LBR vs. LWNB on the 24 largest data sets

	LBR vs. LWNB	
	W/D/L	p
Error	11/1/12	0.5000
Bias	3/1/19	**0.0004**
Variance	15/1/8	0.1050
RMSE	12/0/12	0.5806

FSS, BSE and BSEJ

FSS and BSE form a subset of attributes by deleting attributes to remove harmful interdependencies and applies conventional NB to this subset. Both of them substantially reduce the bias and increase the variance of NB. However, the increase in variance provided by FSS outweighs the reduction in bias and results in an overall increase in error on the 60 data sets. Table 4 present the win/draw/loss records for FSS against NB and BSE on the 60 data sets. FSS has a significant bias advantage and error and variance disadvantages relative to BSE. The reason for FSS's high variance might be that FSS employs forward selection, which appears to produce an attribute subset with a small number of attributes. This attribute subset tends to change greatly from sample to sample. In contrast, BSE uses backward selection and usually uses most of the attributes to classify instances. Consequently, there is often less variation between the models created by BSE than there is between those created by FSS. In addition, FSS might be susceptible to getting trapped into poor selections by local minima.

Table 6: Win/Draw/Loss: FSS vs. NB and BSE on the 60 data sets

	FSS vs. NB		FSS vs. BSE	
	W/D/L	p	W/D/L	p
Error	26/2/32	0.2559	20/5/35	**0.0290**
Bias	50/3/7	**<0.0001**	40/4/16	**0.0009**
Variance	11/2/47	**<0.0001**	14/4/42	**0.0001**
RMSE	32/2/26	0.2559	26/5/29	0.3939

In addition to deleting attributes, BSEJ creates hidden variables, which process further reduces bias of BSE, and achieves the third lowest mean bias rank in our comparison. Table 7 and 8 present the win/draw/loss records for BSEJ against BSE on the 60 data sets and 24 largest data sets respectively. Both tables show that the bias advantage and variance disadvantage of BSEJ are clear. It has a marginal error advantage on the 60 data sets and significant error and RMSE advantages on the 24 largest data sets. Like NBTree and LWNB, BSEJ may also have the potential to deliver low error on large data sets. The success of relaxing attribute independence assumption by adding new compound attributes is evident in the Coil Challenge 2000 data mining contest (Elkan, 2001). However, due to

Table 7: Win/Draw/Loss: BSEJ vs. BSE on the 60 data sets

	BSEJ vs. BSE	
	W/D/L	p
Error	34/4/ 22	0.0704
Bias	50/2/8	**<0.0001**
Variance	15/2/43	**0.0002**
RMSE	31/3/26	0.2983

Table 8: Win/Draw/Loss: BSEJ vs. BSE on the 24 largest data sets

	BSEJ vs. BSE	
	W/D/L	p
Error	18/1/5	**0.0053**
Bias	23/1/0	**<0.0001**
Variance	5/1/18	**0.0053**
RMSE	18/1/5	**0.0053**

the high space complexity, its Weka implementation suffers from memory shortages when large numbers of attributes are joined.

TAN and SP-TAN

In the original TAN (Friedman et al., 1997), a smoothing operation was introduced and the conditional probability of x_i given its parents, $\pi(x_i)$, is calculated by

$$\hat{P}^s(x_i|\pi(x_i)) = \alpha F(x_i|\pi(x_i)) + (1-\alpha)F(x_i),$$

where $\alpha = \frac{tF(\pi(x_i))}{tF(\pi(x_i))+s}$ and $s = 5$. We compare TAN with the smoothing operation, denoted as TAN^s, and with m-estimation ($m = 0.1$). Table 9 presents the win/draw/loss records on 60 data sets. TAN^s has a significant bias disadvantage and variance and RMSE advantages relative to TAN. Both win 29 times against each other in error. When TAN^s is substituted for TAN in the comparison of 15 methods using the Friedman and Nemenyi test, TAN^s moves up four positions in bias (rank being higher than SP-TAN), down two positions in variance (rank being lower than BSEJ) and down five positions in RMSE (rank being lower than LBR). This substitution does not change its position in error. Those differences in bias and variance do not affect the outcomes of the Nemenyi test. However, TAN^s significantly improves the probability prediction of NB in this context.

Table 9: Win/Draw/Loss: TAN vs. TAN^s on 60 data sets

	TAN vs. TAN^s	
	W/D/L	p
Error	29/2/29	0.5522
Bias	46/2/12	**<0.0001**
Variance	14/2/44	**<0.0001**
RMSE	19/0/41	**0.0031**

Table 10: Win/Draw/Loss: TAN vs. SP-TAN on the 24 largest data sets

	TAN vs. SP-TAN	
	W/D/L	p
Error	15/1/8	0.1050
Bias	17/1/6	**0.0173**
Variance	10/2/12	0.4159
RMSE	17/1/6	**0.0173**

Comparing SP-TAN with TAN, the former has significantly lower variance and substantially lower error than the latter on the Nemenyi test. One factor that may contribute to SP-TAN's lower variance is that SP-TAN stops adding arcs when there is no accuracy improvement and hence usually produces a forest, while TAN tends to establish a tree with $n-1$ arcs. Since variance is expected to decrease on large data, TAN may deliver a similar level of classification accuracy relative to SP-TAN on the data. Table 10 presents the win/draw/loss records of TAN against SP-TAN on the 24 largest data sets. The variance and error differences between TAN and SP-TAN are small, but the bias and RMSE differences are significant. Due to the relative bias/variance profile, TAN is likely to achieve lower error than SP-TAN on large data sets. In addition, TAN provides substantially faster training than SP-TAN and hence might be superior, especially on large data, when training time and probability estimation accuracy are concerned.

IR and Hilden and Bjerregaard's Smoothing Method

The comparative study of Caruana and Niculescu-Mizil (2006) reveals that calibration with isotonic regression results in substantial performance improvements. At first glance, their results appear to be inconsistent with this study indicating that IR has higher mean error rank than NB on the 60 data sets. However, when IR is compared with NB on the 24 largest data sets, it significantly improves the classification accuracy and probability prediction of NB (Table 12). In fact, these observations correspondent well to Caruana and Niculescu-Mizil's observations, as all the 11 training sets in their study are of size 5000. On all the data sets we collected, IR obtains lower RMSE on 37 data sets and higher on 23 data sets (Table 11). This RMSE difference is also statistically significant.

Table 11: Win/Draw/Loss: IR vs. NB on 60 data sets

	IR vs. NB	
	W/D/L	p
Error	26/1/33	0.2175
Bias	47/1/12	**<0.0001**
Variance	10/2/48	**<0.0001**
RMSE	37/0/23	**0.0462**

Table 12: Win/Draw/Loss: IR vs. NB on the 24 largest data sets

	IR vs. NB	
	W/D/L	p
Error	17/0/7	**0.0320**
Bias	23/0/1	**<0.0001**
Variance	4/0/20	**0.0008**
RMSE	20/0/4	**0.0008**

Hilden and Bjerregaard's Smoothing Method does not improve upon the classification accuracy of NB in our data collection. NB with the smoothing method has lower error on 23 data sets and higher on 26 data sets. However, this smoothing method is very effective in improving the precision of conditional probability estimates without additional computation. It reduces RMSE on 53 data sets and increases RMSE on 7 data sets. It could potentially be applied with any of the semi-naive Bayesian methods examined herein, and is worthy of consideration in any practical application of such methods.

AODE, MAPLMG and AODE^{SR}

AODE achieves competitive variance with NB, reducing variance successfully by aggregating all qualified 1-dependence classifiers. It maintains the robustness and much of the efficiency of NB, and at the same time exhibits significantly higher classification accuracy and probabilistic prediction for many data sets. In our comparison, it shares a similar level of error with MAPLMG, LBR, AODE^{SR}, LibSVM, SP-TAN, BSEJ, LWNB, NBTree, Logistic Regression and HidNB, while having considerably lower training time relative to MAPLMG, LibSVM, SP-TAN, BSEJ, NBTree and Logistic Regression and classification time relative to LWNB in most cases and LBR.

MAPLMG and AODE^{SR} further reduce the bias, error and RMSE of their base learner at the cost of an increase in variance. AODE^{SR} demonstrates competitive error and RMSE with MAPLMG whose mean error and RMSE ranks are the lowest in our comparison. However, MAPLMG imposes very high training time overheads on AODE, while AODE^{SR} imposes no extra training time overheads and only modest test time overheads on AODE. In our data collection, AODE^{SR} deletes more than 10% of attribute values for more than 50% of data sets, resulting in substantial classification time speed-up. It has lower classification time on 28 and 41 data sets compared to AODE and MAPLMG respectively. The classification time complexity of these three methods is higher than all the other methods except LBR and LWNB. Nonetheless, it is modest for many classification tasks with moderate or small numbers of attributes.

NB and Logistic Regression

Logistic Regression has a significant bias advantage and variance disadvantage relative to NB. As discussed in Section 5.2.2, low bias algorithms appear to have an advantage in error with larger training sets, while low variance algorithms appear to have an advantage with small training sets. It follows that NB may deliver lower error than Logistic Regression at small data set sizes and Logistic Regression may achieve lower error than NB at larger data set sizes. This is consistent with Ng and Jordan's finding that NB has higher asymptotic error than Logistic Regression, but it can approach its asymptotic error much faster than Logistic Regression (Ng and Jordan, 2001). At training time, Logistic Regression is considerably slower than NB. The mean training time on 60 data sets for Logistic Regression is 124393.53 seconds, while that for NB is 13.53 seconds. It has higher training time than NB on every data set of our collection.

Selection Between Semi-naive Bayesian Methods

All semi-naive Bayesian methods provide substantial reduction in bias and increase in variance. Figure 7 does not reveal error differences between NB and TAN, SP-TAN, BSE and IR are statistically significant, as 15 algorithms are compared on the 24 largest data sets and hence the power of the Nemenyi test is low. Nonetheless, the win/draw/loss records show that all of them enjoy a significant advantage in error relative to NB on the 24 data sets. We discuss the selection between these methods based on the characteristics of the application to which they are applied.

As has been discussed, bias appears to dominate error when more data is available. Therefore, when predictive error is of major concern, algorithms with low bias may have an advantage on large data sets. However, such algorithms usually have very high time complexity, and may not be attractive when efficiency is an important issue. Generally, for large training data we recommend use of the lowest bias semi-naive Bayesian method whose computational complexity satisfies the computational constraints of the application context. For small training data we recommend the lowest variance semi-naive Bayesian method that has suitable computational complexity. For intermediate size training samples, an appropriate trade-off between bias and variance should be sought within the prevailing computational complexity constraints.

For extremely small data NB may prove best and for large data NBTree, LWNB and BSEJ may have an advantage if their computational profiles are appropriate to the task. AODE achieves very low variance, relatively low bias, low RMSE, and low training time and space complexity. AODE^{SR} and MAPLMG further enhance AODE by substantially reducing bias and error and improving probability prediction with modest time complexity. In consequence, they may prove competitive over a considerable range of classification tasks. Furthermore, MAPLMG may have an advantage if the primary consideration is attaining the highest possible classification accuracy and AODE^{SR} may excel if one wishes efficient classification.

6. Conclusions

The elegant simplicity, computational efficiency and classification efficacy of NB fosters ongoing interest in exploring semi-naive Bayesian algorithms that improve NB's accuracy by alleviating the attribute interdependence problem. This paper examines these techniques and provides empirical studies of twelve representative semi-naive Bayesian algorithms. We describe each algorithm and provide details of their time and space complexity. NBTree, SP-TAN, BSEJ and APNB have relatively high training time complexity, while LBR and LWNB have high classification time complexity. BSEJ and HierNB have very high space complexity.

In the empirical part of paper, we compare the algorithms using the bias-variance decomposition and quadratic loss function on sixty natural domains from the UCI Machine Learning Repository. The study reveals the outstanding performance of MAPLMG, AODE^{SR}, AODE and LBR on our data collection. They achieve a considerable advantage in error and probabilistic estimation relative to the other semi-naive Bayesian methods. NBTree provides substantial decrease in bias, and significantly wins against the remaining algorithms other than LWNB and BSEJ. NB, AODE, MAPLMG and AODE^{SR} have the lowest mean variance ranks. Based on the observation that variance tends to decrease and bias tends to be a larger portion of error when training set size increases, we suggest using low bias methods for large data sets, and low variance methods for small data sets to obtain better accuracy performance, within the further constraints on applicable algorithms implied by the computational requirements of the given application. Computation cost and the trade-off between bias and variance should be considered for intermediate size data.

Acknowledgments

The authors are grateful to Jesús Cerquides for kindly providing the executable file of MAPLMG. Thanks to Janez Demšar for his help with the Nemenyi test.

APPENDIX

This appendix presents the detailed results for Error (Table 13), Bias (Table 14), Variance (Table 15) and RMSE (Table 16). The data sets are in the number sequence of Table 2. Each table presents the mean and mean rank of each metric.

Table 13: Error (the data sets are in the number sequence of Table 2)

No.	NB	BSE	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	AODE^{SR}	LWNB	IR	BSEJ	HidNB	Logistic	LibSVM
1	0.4821	0.4797	0.4718	0.4776	0.4806	0.4741	0.4625	0.4623	0.4625	0.4733	0.4752	0.4757	0.4763	0.4445	0.4476
2	0.1603	0.1403	0.1420	0.1525	0.1417	0.1337	0.1470	0.1375	0.1421	0.1533	0.1382	0.1391	0.1545	0.1491	0.2325
3	0.0862	0.0900	0.0779	0.0817	0.0813	0.0841	0.0833	0.0834	0.0814	0.0789	0.0874	0.0824	0.0863	0.1362	0.1602
4	0.2858	0.2812	0.3043	0.2850	0.2829	0.2840	0.2833	0.2831	0.2845	0.2807	0.2863	0.2882	0.2936	0.3059	0.2942
5	0.3749	0.3707	0.2898	0.3379	0.3169	0.3489	0.2960	0.2940	0.2939	0.2968	0.3648	0.3175	0.2967	0.4003	0.6087
6	0.2463	0.2496	0.2505	0.2463	0.2481	0.2466	0.2493	0.2499	0.2493	0.2500	0.2622	0.2476	0.2503	0.1096	0.0225
7	0.0281	0.0283	0.0817	0.0281	0.0281	0.0281	0.0357	0.0357	0.0357	0.0345	0.0347	0.0317	0.0480	0.0819	0.0366
8	0.1557	0.1556	0.1028	0.0732	0.0767	0.0961	0.1027	0.0922	0.1027	0.1077	0.1367	0.0701	0.0769	0.0728	0.0261
9	0.3575	0.2875	0.4458	0.3667	0.3575	0.3567	0.3458	0.3392	0.3458	0.3625	0.3583	0.3175	0.5092	0.3475	0.3600
10	0.4995	0.5016	0.5078	0.4994	0.5016	0.5007	0.4962	0.4963	0.4963	0.5252	0.4334	0.5032	0.4977	0.4923	0.4556
11	0.1457	0.1472	0.1681	0.1477	0.1571	0.1459	0.1466	0.1453	0.1462	0.1713	0.1506	0.1519	0.1600	0.1608	0.2846
12	0.2478	0.2532	0.2878	0.2480	0.2618	0.2494	0.2503	0.2647	0.2537	0.2610	0.2631	0.2630	0.2627	0.2678	0.3143
13	0.0208	0.0217	0.0560	0.0213	0.0368	0.0208	0.0223	0.0222	0.0220	0.0257	0.0196	0.0261	0.0209	0.0340	0.0445
14	0.3478	0.3441	0.3475	0.3449	0.3455	0.3446	0.3456	0.3489	0.3456	0.3426	0.3307	0.3414	0.3522	0.2811	0.3273
15	0.2626	0.2647	0.2882	0.2651	0.2776	0.2647	0.2590	0.2590	0.2602	0.2840	0.2639	0.2749	0.2592	0.2569	0.3062
16	0.3072	0.3049	0.3016	0.3056	0.3046	0.3053	0.3032	0.3038	0.3030	0.2964	0.3138	0.3070	0.3067	0.3194	0.2351
17	0.2821	0.2795	0.2865	0.2827	0.2846	0.2817	0.2842	0.2839	0.2861	0.2833	0.2816	0.2795	0.3024	0.2667	0.2712
18	0.1789	0.1833	0.1966	0.1834	0.1960	0.1792	0.1771	0.1766	0.1774	0.2091	0.1878	0.1939	0.1820	0.1859	0.2226
19	0.1632	0.1667	0.1668	0.1694	0.1870	0.1655	0.1663	0.1672	0.1663	0.1845	0.1681	0.1761	0.1655	0.2234	0.2279
20	0.1934	0.1883	0.2198	0.1932	0.2086	0.1920	0.1915	0.1896	0.1908	0.2248	0.1914	0.2046	0.1932	0.2455	0.3717
21	0.0964	0.0847	0.0644	0.0730	0.0564	0.0641	0.0545	0.0498	0.0545	0.0491	0.1025	0.0693	0.0592	0.0856	0.0499
22	0.1586	0.1701	0.1755	0.1665	0.1747	0.1600	0.1584	0.1586	0.1588	0.1822	0.1671	0.1766	0.1720	0.1791	0.2205
23	0.0151	0.0135	0.0113	0.0124	0.0127	0.0131	0.0130	0.0122	0.0132	0.0159	0.0123	0.0120	0.0122	0.0342	0.0263
24	0.1038	0.1021	0.0905	0.1025	0.1084	0.1018	0.0830	0.0830	0.0828	0.0924	0.1060	0.1004	0.0832	0.1608	0.0659
25	0.0612	0.0611	0.0623	0.0603	0.0612	0.0612	0.0575	0.0571	0.0575	0.0620	0.0632	0.0605	0.0827	0.0505	0.0396
26	0.1270	0.0713	0.0650	0.0736	0.0197	0.0318	0.0893	0.0574	0.0753	0.0337	0.1237	0.0324	0.0789	0.0286	0.0109
27	0.1347	0.1389	0.1702	0.1393	0.1347	0.1347	0.1421	0.1389	0.1421	0.1323	0.1400	0.1418	0.1446	0.0972	0.0954
28	0.2636	0.2645	0.2668	0.2649	0.2669	0.2646	0.2662	0.2668	0.2662	0.2771	0.2671	0.2674	0.2682	0.2691	0.2714
29	0.2593	0.2551	0.1691	0.1465	0.1513	0.1531	0.1065	0.1011	0.1057	0.0820	0.2546	0.1604	0.1176	0.2281	0.0308
30	0.4222	0.4222	0.4221	0.4222	0.4222	0.4222	0.4222	0.4222	0.4222	0.4222	0.4255	0.4222	0.4221	0.3268	0.2999
31	0.5337	0.5344	0.5981	0.5288	0.5337	0.5356	0.5500	0.5488	0.5500	0.5731	0.5419	0.5344	0.5450	0.5525	0.5869
32	0.1815	0.1851	0.1876	0.1843	0.2139	0.1823	0.1845	0.1835	0.1845	0.1999	0.1988	0.1955	0.1834	0.2530	0.2104
33	0.2273	0.1903	0.1724	0.1854	0.1681	0.1654	0.1833	0.1744	0.1801	0.1686	0.2011	0.1708	0.1762	0.2091	0.1451
34	0.0109	0.0013	0.0013	0.0016	0.0001	0.0008	0.0004	0.0002	0.0004	0.0000	0.0110	0.0002	0.0004	0.0016	0.0000
35	0.3111	0.2962	0.3120	0.2364	0.2164	0.2493	0.2786	0.2492	0.2798	0.3163	0.3122	0.2432	0.3010	0.2940	0.3914
36	0.0540	0.0595	0.0638	0.0572	0.0592	0.0540	0.0565	0.0569	0.0565	0.0640	0.0567	0.0593	0.0805	0.0499	0.0522
37	0.0978	0.0978	0.0740	0.0889	0.0345	0.0442	0.0740	0.0776	0.0740	0.0362	0.0857	0.0427	0.0603	0.0751	0.0019
38	0.0793	0.0775	0.0449	0.0714	0.0714	0.0609	0.0336	0.0337	0.0335	0.0307	0.0787	0.0594	0.0412	0.0621	0.0109
39	0.0620	0.0456	0.0450	0.0470	0.0380	0.0390	0.0362	0.0374	0.0363	0.0358	0.0464	0.0417	0.0355	0.0359	0.0443
40	0.1201	0.1152	0.0494	0.0376	0.0494	0.0485	0.0241	0.0238	0.0241	0.0210	0.1060	0.0493	0.0301	0.0467	0.0047
41	0.2578	0.2543	0.2535	0.2539	0.2568	0.2567	0.2544	0.2531	0.2540	0.2569	0.2584	0.2534	0.2582	0.2327	0.2533
42	0.3760	0.3587	0.4387	0.3842	0.4000	0.3780	0.3820	0.3813	0.3820	0.3789	0.3736	0.3824	0.4042	0.4342	0.3013
43	0.5740	0.5789	0.5902	0.5756	0.5991	0.5752	0.5731	0.5732	0.5742	0.6112	0.5827	0.5916	0.5789	0.6865	0.5858
44	0.1487	0.1506	0.2783	0.1504	0.2183	0.1496	0.2711	0.2704	0.2711	0.1623	0.1830	0.1860	0.1353	0.1277	0.2794
45	0.0848	0.0747	0.0521	0.0657	0.0619	0.0704	0.0433	0.0426	0.0434	0.0461	0.0834	0.0583	0.0437	0.0511	0.0380
46	0.0315	0.0280	0.0269	0.0286	0.0258	0.0287	0.0287	0.0262	0.0275	0.0276	0.0301	0.0271	0.0261	0.0338	0.0293
47	0.3609	0.3601	0.2858	0.2888	0.2505	0.2521	0.2921	0.2834	0.2880	0.2307	0.3531	0.2633	0.2804	0.4078	0.1635
48	0.1841	0.1807	0.1663	0.1759	0.1740	0.1702	0.1615	0.1610	0.1607	0.1706	0.4992	0.1713	0.1619	0.1617	0.1639
49	0.2637	0.2750	0.2905	0.2648	0.2810	0.2643	0.2644	0.2649	0.2706	0.2879	0.2715	0.2697	0.2868	0.3053	0.1728
50	0.1019	0.0817	0.0817	0.0935	0.0763	0.0652	0.0701	0.0627	0.0696	0.0604	0.1023	0.0830	0.0807	0.0777	0.0842
51	0.0462	0.0444	0.0566	0.0460	0.0463	0.0451	0.0384	0.0383	0.0384	0.0773	0.0460	0.0468	0.0405	0.1324	0.1498
52	0.0349	0.0358	0.0223	0.0346	0.0429	0.0349	0.0152	0.0151	0.0152	0.0172	0.0350	0.0343	0.0307	0.2497	0.0051
53	0.5722	0.5837	0.5764	0.5730	0.5736	0.5713	0.5719	0.5719	0.5719	0.5774	0.6019	0.5807	0.6036	0.5090	0.5275
54	0.2917	0.2873	0.2659	0.2853	0.2034	0.2307	0.2523	0.2505	0.2523	0.0824	0.2716	0.2270	0.2367	0.0244	0.0170
55	0.4173	0.4136	0.3495	0.3784	0.3408	0.3421	0.3313	0.3312	0.3327	0.3298	0.4051	0.3556	0.3429	0.2217	0.1966
56	0.3338	0.3338	0.3338	0.3338	0.3338	0.3338	0.3338	0.3338	0.3338	0.3338	0.3374	0.3338	0.3339	0.3401	0.3900
57	0.4963	0.4428	0.4003	0.2648	0.2503	0.3335	0.2753	0.2803	0.2719	0.2444	0.4704	0.3457	0.2554	0.2841	0.0471
58	0.2010	0.1917	0.1861	0.1846	0.1879	0.1731	0.1576	0.1584	0.1553	0.1932	0.1715	0.1805	0.1572	0.1370	0.1353
59	0.0329	0.0338	0.0538	0.0331	0.0398	0.0329	0.0333	0.0334	0.0333	0.0446	0.0452	0.0345	0.0530	0.0393	0.1398
60	0.0634	0.0669	0.0800	0.0636	0.0655	0.0634	0.0582	0.0582	0.0582	0.0642	0.0634	0.0661	0.0608	0.0846	0.0743
ME	0.2170	0.2117	0.2138	0.2015	0.1999	0.1977	0.1978	0.1960	0.1975	0.1973	0.2206	0.2004	0.2027	0.2060	**0.1927**
MER	9.2750	9.0667	10.0417	8.1083	8.2250	6.3083	6.2500	**5.3917**	6.1500	8.1500	10.3583	8.1500	8.3583	8.9250	7.2417

Mean Error and Mean Error Rank are abbreviated as ME and MER

Table 14: Bias (the data sets are in the number sequence of Table 2)

No.	NB	BSE	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	AODE^{SR}	LWNB	IR	BSEJ	HidNB	Logistic	LibSVM
1	0.4147	0.3937	0.2995	0.3834	0.3538	0.3379	0.3139	0.3130	0.3140	0.2837	0.3878	0.3522	0.3190	0.3849	0.3314
2	0.1500	0.1128	0.1251	0.1291	0.1180	0.1183	0.1342	0.1251	0.1309	0.1092	0.1294	0.1112	0.1383	0.1408	0.2145
3	0.0657	0.0626	0.0560	0.0620	0.0505	0.0604	0.0648	0.0646	0.0641	0.0451	0.0509	0.0593	0.0623	0.0931	0.0993
4	0.1627	0.1547	0.1728	0.1594	0.1383	0.1593	0.1624	0.1622	0.1593	0.1549	0.1632	0.1469	0.1731	0.1482	0.1778
5	0.2262	0.2205	0.1400	0.1814	0.1346	0.1864	0.1493	0.1470	0.1486	0.1468	0.2160	0.1537	0.1548	0.1733	0.5140
6	0.1273	0.1295	0.1316	0.1284	0.1280	0.1275	0.1295	0.1301	0.1295	0.1295	0.1248	0.1293	0.1300	0.0689	0.0068
7	0.0256	0.0249	0.0414	0.0256	0.0256	0.0256	0.0275	0.0272	0.0275	0.0262	0.0251	0.0247	0.0303	0.0338	0.0289
8	0.1048	0.1047	0.0595	0.0459	0.0285	0.0434	0.0554	0.0489	0.0554	0.0462	0.0830	0.0437	0.0549	0.0512	0.0083
9	0.2410	0.1589	0.2202	0.1973	0.2410	0.2396	0.2026	0.2010	0.2026	0.2248	0.2436	0.1557	0.2554	0.1709	0.2137
10	0.4329	0.3869	0.3625	0.4081	0.4088	0.4100	0.4067	0.3971	0.4067	0.3468	0.3352	0.3823	0.3699	0.4019	0.3394
11	0.1198	0.1138	0.1119	0.1183	0.1058	0.1188	0.1149	0.1123	0.1127	0.1141	0.1164	0.1116	0.1213	0.1146	0.1691
12	0.1460	0.1472	0.1628	0.1358	0.1249	0.1381	0.1358	0.1435	0.1365	0.1283	0.1584	0.1373	0.1603	0.1416	0.1926
13	0.0098	0.0089	0.0191	0.0094	0.0104	0.0098	0.0107	0.0107	0.0103	0.0093	0.0082	0.0098	0.0092	0.0169	0.0238
14	0.2259	0.2302	0.2307	0.2303	0.2289	0.2301	0.2284	0.2370	0.2284	0.2357	0.2537	0.2353	0.2354	0.2247	0.2241
15	0.2028	0.1961	0.1852	0.1980	0.1848	0.2006	0.1953	0.1944	0.1942	0.1901	0.2011	0.1881	0.1911	0.1924	0.2844
16	0.1914	0.1813	0.1747	0.1849	0.1767	0.1872	0.1810	0.1805	0.1803	0.1692	0.1876	0.1811	0.1724	0.2242	0.1468
17	0.2139	0.2126	0.2174	0.2144	0.2136	0.2137	0.2141	0.2143	0.2141	0.2218	0.2154	0.2152	0.2096	0.2232	0.2171
18	0.1439	0.1366	0.1337	0.1412	0.1240	0.1432	0.1374	0.1354	0.1359	0.1380	0.1420	0.1345	0.1344	0.1308	0.1592
19	0.1253	0.1193	0.1055	0.1243	0.1166	0.1255	0.1210	0.1251	0.1210	0.1221	0.1174	0.1172	0.1133	0.1143	0.1664
20	0.1660	0.1555	0.1430	0.1551	0.1317	0.1556	0.1519	0.1486	0.1476	0.1496	0.1557	0.1399	0.1454	0.1385	0.2991
21	0.0892	0.0701	0.0424	0.0571	0.0284	0.0422	0.0430	0.0375	0.0430	0.0327	0.0751	0.0459	0.0485	0.0338	0.0348
22	0.1387	0.1333	0.1177	0.1345	0.1257	0.1375	0.1362	0.1349	0.1342	0.1262	0.1312	0.1300	0.1228	0.1292	0.1441
23	0.0094	0.0077	0.0066	0.0077	0.0056	0.0078	0.0074	0.0069	0.0073	0.0076	0.0075	0.0064	0.0065	0.0240	0.0145
24	0.0810	0.0792	0.0601	0.0802	0.0629	0.0787	0.0605	0.0606	0.0598	0.0609	0.0795	0.0764	0.0612	0.0816	0.0418
25	0.0385	0.0383	0.0385	0.0375	0.0372	0.0385	0.0375	0.0370	0.0375	0.0390	0.0396	0.0381	0.0380	0.0270	0.0247
26	0.1072	0.0457	0.0534	0.0537	0.0067	0.0152	0.0693	0.0457	0.0576	0.0150	0.1023	0.0188	0.0637	0.0199	0.0043
27	0.0529	0.0534	0.0703	0.0524	0.0529	0.0529	0.0566	0.0601	0.0566	0.0530	0.0528	0.0578	0.0566	0.0358	0.0417
28	0.2265	0.2265	0.2257	0.2258	0.2247	0.2252	0.2269	0.2271	0.2269	0.2275	0.2249	0.2246	0.2263	0.2270	0.2251
29	0.2163	0.2114	0.0962	0.0700	0.0573	0.0565	0.0668	0.0616	0.0660	0.0353	0.2065	0.0694	0.0809	0.1981	0.0152
30	0.3553	0.3553	0.3552	0.3553	0.3553	0.3553	0.3553	0.3553	0.3553	0.3553	0.3458	0.3553	0.3552	0.2614	0.2219
31	0.3355	0.3360	0.3407	0.3264	0.3355	0.3352	0.3432	0.3422	0.3432	0.3608	0.3380	0.3278	0.3500	0.3604	0.3514
32	0.1347	0.1280	0.0993	0.1218	0.1042	0.1322	0.1301	0.1299	0.1301	0.1279	0.1359	0.1027	0.1235	0.1227	0.1308
33	0.2087	0.1562	0.1271	0.1456	0.1193	0.1183	0.1530	0.1401	0.1501	0.1158	0.1625	0.1236	0.1445	0.2045	0.1194
34	0.0095	0.0010	0.0008	0.0006	0.0000	0.0002	0.0003	0.0002	0.0003	0.0000	0.0078	0.0000	0.0004	0.0001	0.0000
35	0.2004	0.1812	0.1777	0.1368	0.1168	0.1383	0.1555	0.1413	0.1535	0.1658	0.1941	0.1516	0.1642	0.1332	0.2375
36	0.0242	0.0264	0.0281	0.0256	0.0261	0.0242	0.0256	0.0254	0.0256	0.0328	0.0235	0.0260	0.0319	0.0219	0.0252
37	0.0901	0.0901	0.0606	0.0793	0.0115	0.0241	0.0651	0.0658	0.0651	0.0128	0.0737	0.0224	0.0509	0.0686	0.0004
38	0.0648	0.0589	0.0249	0.0524	0.0294	0.0278	0.0220	0.0220	0.0218	0.0168	0.0626	0.0331	0.0302	0.0236	0.0066
39	0.0467	0.0299	0.0290	0.0308	0.0211	0.0242	0.0239	0.0231	0.0238	0.0222	0.0356	0.0245	0.0243	0.0276	0.0363
40	0.1031	0.0919	0.0285	0.0160	0.0160	0.0185	0.0144	0.0137	0.0144	0.0092	0.0867	0.0188	0.0189	0.0361	0.0029
41	0.1799	0.1782	0.1764	0.1768	0.1780	0.1796	0.1794	0.1785	0.1790	0.1753	0.1801	0.1770	0.1787	0.2064	0.2032
42	0.2748	0.2730	0.2804	0.2683	0.2649	0.2747	0.2912	0.2897	0.2912	0.2970	0.2721	0.2637	0.2952	0.2590	0.2847
43	0.3920	0.3805	0.3717	0.3869	0.3690	0.3889	0.3894	0.3896	0.3898	0.4109	0.3803	0.3755	0.3888	0.3809	0.3854
44	0.0772	0.0793	0.1105	0.0760	0.0799	0.0772	0.1293	0.1291	0.1293	0.0657	0.0966	0.0795	0.0643	0.0625	0.1474
45	0.0612	0.0492	0.0240	0.0393	0.0236	0.0323	0.0231	0.0225	0.0229	0.0214	0.0591	0.0289	0.0240	0.0331	0.0177
46	0.0269	0.0235	0.0221	0.0236	0.0181	0.0225	0.0247	0.0222	0.0237	0.0207	0.0257	0.0211	0.0220	0.0286	0.0176
47	0.3304	0.3155	0.2455	0.2273	0.1672	0.1824	0.2553	0.2429	0.2493	0.1614	0.3118	0.1789	0.2403	0.3904	0.1171
48	0.1730	0.1587	0.1475	0.1529	0.1425	0.1497	0.1496	0.1498	0.1490	0.1451	0.3982	0.1476	0.1510	0.1513	0.1492
49	0.1705	0.1686	0.1584	0.1675	0.1507	0.1691	0.1670	0.1672	0.1632	0.1505	0.1651	0.1616	0.1608	0.1547	0.0863
50	0.0928	0.0647	0.0674	0.0816	0.0369	0.0429	0.0591	0.0513	0.0588	0.0341	0.0847	0.0624	0.0717	0.0612	0.0569
51	0.0379	0.0349	0.0361	0.0373	0.0378	0.0341	0.0292	0.0291	0.0292	0.0353	0.0362	0.0331	0.0313	0.0441	0.0999
52	0.0206	0.0210	0.0086	0.0204	0.0169	0.0204	0.0066	0.0066	0.0066	0.0069	0.0195	0.0200	0.0182	0.0884	0.0020
53	0.3949	0.3853	0.4001	0.3959	0.4004	0.3982	0.3994	0.3994	0.3994	0.4046	0.3489	0.3900	0.4167	0.3955	0.2891
54	0.2511	0.2352	0.1748	0.2076	0.0654	0.1222	0.2015	0.1963	0.2015	0.0339	0.2385	0.0877	0.1774	0.0168	0.0162
55	0.2949	0.2872	0.1895	0.2460	0.1748	0.1958	0.1956	0.1942	0.1946	0.1772	0.2840	0.2086	0.2017	0.1513	0.1077
56	0.3293	0.3293	0.3293	0.3293	0.3293	0.3293	0.3293	0.3293	0.3293	0.3293	0.3247	0.3293	0.3293	0.3232	0.2588
57	0.2467	0.2307	0.1607	0.0779	0.0700	0.1068	0.0896	0.0931	0.0879	0.0720	0.2205	0.1093	0.0833	0.1204	0.0074
58	0.1755	0.1441	0.1203	0.1434	0.1047	0.1128	0.1165	0.1145	0.1133	0.1013	0.1337	0.1294	0.1170	0.1143	0.1118
59	0.0141	0.0130	0.0185	0.0141	0.0149	0.0141	0.0142	0.0143	0.0142	0.0174	0.0180	0.0142	0.0193	0.0203	0.0777
60	0.0288	0.0278	0.0337	0.0282	0.0249	0.0288	0.0267	0.0265	0.0267	0.0280	0.0288	0.0294	0.0191	0.0295	0.0348
MB	0.1600	0.1495	0.1358	0.1390	**0.1242**	0.1328	0.1368	0.1349	0.1358	0.1249	0.1555	0.1288	0.1365	0.1376	0.1328
MBR	11.9259	9.3750	7.8500	8.5750	**4.5583**	7.8833	8.4583	7.5750	7.6917	6.1000	10.0667	6.2750	8.2750	8.1000	7.2917

Mean Bias and Mean Bias Rank are abbreviated as MB and MBR

Table 15: Variance (the data sets are in the number sequence of Table 2)

No.	NB	BSE	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	AODE^{SR}	LWNB	IR	BSEJ	HidNB	Logistic	LibSVM
1	0.0675	0.0861	0.1723	0.0942	0.1268	0.1362	0.1487	0.1493	0.1486	0.1896	0.0874	0.1235	0.1573	0.0596	0.1162
2	0.0102	0.0275	0.0169	0.0234	0.0237	0.0154	0.0128	0.0124	0.0111	0.0441	0.0089	0.0278	0.0162	0.0083	0.0180
3	0.0206	0.0274	0.0219	0.0198	0.0308	0.0237	0.0186	0.0188	0.0172	0.0338	0.0365	0.0231	0.0239	0.0431	0.0609
4	0.1232	0.1265	0.1316	0.1255	0.1446	0.1247	0.1208	0.1209	0.1252	0.1258	0.1231	0.1414	0.1205	0.1577	0.1164
5	0.1487	0.1502	0.1498	0.1565	0.1822	0.1625	0.1467	0.1469	0.1452	0.1500	0.1487	0.1637	0.1419	0.2270	0.0947
6	0.1190	0.1201	0.1189	0.1179	0.1201	0.1191	0.1199	0.1198	0.1199	0.1206	0.1374	0.1184	0.1203	0.0407	0.0157
7	0.0025	0.0034	0.0403	0.0025	0.0025	0.0025	0.0082	0.0086	0.0082	0.0083	0.0097	0.0070	0.0177	0.0481	0.0077
8	0.0509	0.0509	0.0433	0.0273	0.0482	0.0527	0.0473	0.0434	0.0473	0.0615	0.0538	0.0263	0.0220	0.0216	0.0179
9	0.1165	0.1286	0.2256	0.1694	0.1165	0.1171	0.1432	0.1382	0.1432	0.1377	0.1147	0.1618	0.2538	0.1766	0.1463
10	0.0667	0.1148	0.1454	0.0913	0.0927	0.0907	0.0895	0.0991	0.0896	0.1784	0.0982	0.1209	0.1278	0.0904	0.1163
11	0.0259	0.0334	0.0561	0.0293	0.0513	0.0272	0.0317	0.0330	0.0335	0.0572	0.0342	0.0403	0.0387	0.0461	0.1156
12	0.1018	0.1060	0.1251	0.1122	0.1370	0.1113	0.1144	0.1212	0.1173	0.1327	0.1048	0.1256	0.1024	0.1263	0.1217
13	0.0110	0.0129	0.0369	0.0119	0.0264	0.0110	0.0116	0.0115	0.0117	0.0164	0.0114	0.0163	0.0117	0.0171	0.0208
14	0.1219	0.1139	0.1167	0.1146	0.1166	0.1144	0.1172	0.1119	0.1172	0.1069	0.0770	0.1061	0.1168	0.0563	0.1032
15	0.0598	0.0686	0.1030	0.0671	0.0928	0.0640	0.0637	0.0647	0.0660	0.0938	0.0628	0.0868	0.0680	0.0645	0.0218
16	0.1158	0.1235	0.1269	0.1208	0.1279	0.1181	0.1222	0.1233	0.1227	0.1272	0.1262	0.1259	0.1344	0.0952	0.0884
17	0.0682	0.0669	0.0691	0.0683	0.0710	0.0680	0.0702	0.0696	0.0720	0.0615	0.0662	0.0642	0.0928	0.0434	0.0541
18	0.0351	0.0467	0.0629	0.0422	0.0720	0.0360	0.0397	0.0412	0.0415	0.0711	0.0458	0.0593	0.0476	0.0550	0.0634
19	0.0379	0.0474	0.0614	0.0451	0.0703	0.0400	0.0453	0.0421	0.0453	0.0624	0.0507	0.0590	0.0523	0.1090	0.0615
20	0.0274	0.0328	0.0768	0.0381	0.0769	0.0364	0.0395	0.0410	0.0432	0.0752	0.0358	0.0647	0.0478	0.1070	0.0726
21	0.0073	0.0146	0.0220	0.0159	0.0280	0.0219	0.0114	0.0122	0.0114	0.0164	0.0274	0.0234	0.0107	0.0517	0.0151
22	0.0199	0.0368	0.0578	0.0320	0.0490	0.0225	0.0222	0.0238	0.0246	0.0561	0.0359	0.0466	0.0493	0.0499	0.0764
23	0.0057	0.0058	0.0047	0.0047	0.0071	0.0053	0.0056	0.0053	0.0059	0.0083	0.0048	0.0056	0.0056	0.0102	0.0118
24	0.0228	0.0229	0.0304	0.0223	0.0455	0.0231	0.0226	0.0224	0.0230	0.0314	0.0265	0.0240	0.0220	0.0792	0.0241
25	0.0227	0.0228	0.0238	0.0228	0.0240	0.0227	0.0200	0.0201	0.0200	0.0230	0.0236	0.0225	0.0447	0.0235	0.0149
26	0.0198	0.0257	0.0115	0.0199	0.0130	0.0166	0.0200	0.0117	0.0177	0.0187	0.0215	0.0137	0.0152	0.0087	0.0066
27	0.0818	0.0856	0.0999	0.0869	0.0818	0.0818	0.0855	0.0788	0.0855	0.0792	0.0872	0.0840	0.0879	0.0614	0.0538
28	0.0371	0.0380	0.0411	0.0391	0.0421	0.0393	0.0393	0.0397	0.0393	0.0496	0.0422	0.0428	0.0418	0.0421	0.0464
29	0.0431	0.0438	0.0729	0.0765	0.0940	0.0965	0.0397	0.0395	0.0397	0.0467	0.0481	0.0910	0.0367	0.0300	0.0156
30	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669	0.0669	0.0797	0.0669	0.0669	0.0654	0.0780
31	0.1982	0.1984	0.2575	0.2024	0.1982	0.2004	0.2068	0.2065	0.2068	0.2123	0.2038	0.2066	0.1950	0.1921	0.2355
32	0.0468	0.0572	0.0883	0.0625	0.1097	0.0501	0.0544	0.0536	0.0544	0.0719	0.0629	0.0929	0.0599	0.1303	0.0796
33	0.0186	0.0341	0.0453	0.0398	0.0488	0.0471	0.0303	0.0343	0.0301	0.0528	0.0386	0.0472	0.0317	0.0046	0.0256
34	0.0014	0.0003	0.0005	0.0009	0.0001	0.0006	0.0001	0.0001	0.0001	0.0000	0.0031	0.0002	0.0001	0.0015	0.0000
35	0.1107	0.1150	0.1343	0.0996	0.0996	0.1110	0.1231	0.1080	0.1263	0.1504	0.1182	0.0916	0.1368	0.1608	0.1539
36	0.0298	0.0332	0.0357	0.0316	0.0331	0.0298	0.0308	0.0315	0.0308	0.0312	0.0332	0.0333	0.0486	0.0279	0.0270
37	0.0077	0.0077	0.0134	0.0096	0.0230	0.0201	0.0089	0.0118	0.0089	0.0235	0.0120	0.0203	0.0094	0.0065	0.0015
38	0.0146	0.0186	0.0200	0.0190	0.0419	0.0331	0.0116	0.0118	0.0117	0.0139	0.0161	0.0263	0.0110	0.0385	0.0043
39	0.0154	0.0157	0.0160	0.0162	0.0169	0.0148	0.0124	0.0143	0.0125	0.0137	0.0108	0.0172	0.0112	0.0083	0.0080
40	0.0170	0.0232	0.0209	0.0216	0.0334	0.0300	0.0098	0.0101	0.0098	0.0118	0.0194	0.0305	0.0112	0.0106	0.0018
41	0.0779	0.0762	0.0771	0.0771	0.0788	0.0771	0.0750	0.0746	0.0751	0.0815	0.0783	0.0764	0.0795	0.0263	0.0501
42	0.1012	0.0857	0.1583	0.1160	0.1351	0.1033	0.0908	0.0917	0.0908	0.0819	0.1014	0.1187	0.1091	0.1752	0.0167
43	0.1820	0.1984	0.2185	0.1888	0.2301	0.1864	0.1837	0.1836	0.1844	0.2003	0.2024	0.2161	0.1901	0.3056	0.2003
44	0.0715	0.0713	0.1678	0.0744	0.1384	0.0724	0.1418	0.1413	0.1418	0.0966	0.0864	0.1066	0.0710	0.0653	0.1320
45	0.0236	0.0255	0.0281	0.0263	0.0383	0.0381	0.0202	0.0201	0.0205	0.0247	0.0243	0.0295	0.0196	0.0180	0.0203
46	0.0045	0.0045	0.0048	0.0050	0.0077	0.0062	0.0040	0.0040	0.0038	0.0069	0.0044	0.0060	0.0040	0.0052	0.0116
47	0.0305	0.0446	0.0403	0.0614	0.0833	0.0697	0.0369	0.0405	0.0388	0.0693	0.0413	0.0843	0.0401	0.0174	0.0465
48	0.0111	0.0220	0.0188	0.0230	0.0315	0.0205	0.0119	0.0112	0.0117	0.0255	0.1010	0.0237	0.0109	0.0104	0.0147
49	0.0931	0.1064	0.1321	0.0974	0.1303	0.0953	0.0974	0.0977	0.1074	0.1374	0.1065	0.1081	0.1260	0.1506	0.0865
50	0.0091	0.0170	0.0143	0.0119	0.0394	0.0223	0.0109	0.0114	0.0108	0.0263	0.0176	0.0205	0.0090	0.0165	0.0273
51	0.0083	0.0095	0.0205	0.0087	0.0085	0.0110	0.0092	0.0093	0.0092	0.0419	0.0099	0.0137	0.0092	0.0883	0.0499
52	0.0143	0.0148	0.0137	0.0142	0.0260	0.0145	0.0086	0.0085	0.0086	0.0102	0.0155	0.0144	0.0125	0.1613	0.0031
53	0.1773	0.1984	0.1763	0.1770	0.1732	0.1731	0.1726	0.1726	0.1726	0.1727	0.2530	0.1906	0.1868	0.1135	0.2384
54	0.0406	0.0520	0.0911	0.0777	0.1380	0.1085	0.0508	0.0542	0.0508	0.0484	0.0331	0.1393	0.0593	0.0076	0.0008
55	0.1224	0.1264	0.1600	0.1324	0.1659	0.1463	0.1357	0.1370	0.1381	0.1526	0.1211	0.1470	0.1412	0.0704	0.0890
56	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045	0.0127	0.0045	0.0046	0.0169	0.1311
57	0.2496	0.2120	0.2396	0.1870	0.1804	0.2267	0.1856	0.1872	0.1840	0.1724	0.2499	0.2364	0.1721	0.1637	0.0397
58	0.0255	0.0476	0.0658	0.0412	0.0832	0.0603	0.0410	0.0439	0.0420	0.0919	0.0378	0.0512	0.0403	0.0227	0.0235
59	0.0188	0.0209	0.0354	0.0190	0.0249	0.0188	0.0191	0.0191	0.0191	0.0272	0.0272	0.0203	0.0337	0.0190	0.0621
60	0.0346	0.0392	0.0463	0.0354	0.0406	0.0346	0.0315	0.0317	0.0315	0.0362	0.0346	0.0367	0.0416	0.0550	0.0395
MV	**0.0570**	0.0622	0.0780	0.0624	0.0757	0.0649	0.0611	0.0611	0.0616	0.0723	0.0651	0.0715	0.0662	0.0684	0.0599
MVR	**4.9167**	8.0750	10.9333	7.4750	11.2417	7.5583	5.7667	5.9500	6.3333	10.4333	8.6259	10.0583	7.9167	7.6833	7.0333

Mean Variance and Mean Variance Rank are abbreviated as MV and MVR

Table 16: RMSE (the data sets are in the number sequence of Table 2)

No.	NB	BSE	TAN	SP-TAN	NBTree	LBR	AODE	MAPLMG	AODE^{SR}	LWNB	IR	BSEJ	HidNB	Logistic
1	0.4647	0.4547	0.4284	0.4552	0.4590	0.4524	0.4219	0.4215	0.4219	0.4427	0.4257	0.4488	0.4326	0.4176
2	0.3412	0.3141	0.3107	0.3323	0.3247	0.3143	0.3224	0.3131	0.3145	0.3366	0.3088	0.3129	0.3215	0.3207
3	0.1471	0.1492	0.1316	0.1434	0.1413	0.1446	0.1441	0.1441	0.1365	0.1343	0.1398	0.1436	0.1363	0.1856
4	0.1400	0.1385	0.1472	0.1400	0.1441	0.1399	0.1386	0.1385	0.1381	0.1411	0.1472	0.1420	0.1388	0.1551
5	0.3040	0.3027	0.2677	0.2907	0.2807	0.2937	0.2697	0.2692	0.2691	0.2736	0.2902	0.2808	0.2673	0.3329
6	0.3398	0.3473	0.3389	0.3393	0.3400	0.3398	0.3370	0.3372	0.3370	0.3373	0.3420	0.3414	0.3367	0.2124
7	0.1619	0.1642	0.2693	0.1619	0.1619	0.1619	0.1802	0.1801	0.1803	0.1796	0.1753	0.1728	0.2004	0.2805
8	0.2286	0.2285	0.1904	0.1800	0.1741	0.1906	0.2060	0.1950	0.2060	0.1980	0.2156	0.1789	0.1768	0.1643
9	0.4233	0.3877	0.4666	0.4325	0.4233	0.4234	0.4173	0.4149	0.4173	0.4409	0.4529	0.4028	0.4860	0.4744
10	0.4505	0.4450	0.4479	0.4504	0.4526	0.4508	0.4412	0.4407	0.4412	0.4660	0.2424	0.4485	0.4405	0.4419
11	0.3434	0.3417	0.3628	0.3454	0.3609	0.3441	0.3395	0.3355	0.3376	0.3685	0.3396	0.3505	0.3480	0.3494
12	0.4658	0.4678	0.4900	0.4623	0.4799	0.4670	0.4607	0.4717	0.4626	0.4605	0.5056	0.4723	0.4610	0.5049
13	0.0736	0.0739	0.1230	0.0743	0.0991	0.0736	0.0756	0.0756	0.0752	0.0833	0.0758	0.0845	0.0728	0.1042
14	0.4839	0.4843	0.4847	0.4851	0.4855	0.4843	0.4839	0.4866	0.4839	0.4844	0.4850	0.4849	0.4875	0.4365
15	0.4254	0.4255	0.4631	0.4302	0.4531	0.4282	0.4227	0.4232	0.4243	0.4701	0.4228	0.4434	0.4247	0.4214
16	0.3942	0.3915	0.3877	0.3924	0.3944	0.3936	0.3857	0.3856	0.3848	0.3877	0.3890	0.3935	0.3854	0.3981
17	0.4506	0.4457	0.4605	0.4578	0.4594	0.4516	0.4572	0.4553	0.4585	0.4621	0.4548	0.4498	0.4770	0.4414
18	0.3740	0.3769	0.3884	0.3793	0.4003	0.3744	0.3663	0.3652	0.3660	0.3948	0.3685	0.3910	0.3692	0.3763
19	0.3726	0.3735	0.3726	0.3782	0.4013	0.3746	0.3683	0.3690	0.3683	0.3836	0.3559	0.3891	0.3633	0.4635
20	0.4056	0.3978	0.4301	0.4057	0.4277	0.4047	0.3951	0.3935	0.3940	0.4335	0.3857	0.4213	0.3927	0.4610
21	0.2956	0.2747	0.2333	0.2563	0.2242	0.2360	0.2083	0.2003	0.2083	0.2058	0.2579	0.2489	0.2176	0.2898
22	0.3546	0.3620	0.3588	0.3614	0.3704	0.3552	0.3490	0.3487	0.3478	0.3664	0.3469	0.3720	0.3480	0.3675
23	0.0788	0.0766	0.0678	0.0723	0.0743	0.0749	0.0733	0.0719	0.0743	0.0815	0.0734	0.0723	0.0718	0.1158
24	0.3129	0.3099	0.2905	0.3112	0.3176	0.3104	0.2748	0.2749	0.2739	0.2939	0.2924	0.3077	0.2715	0.3990
25	0.1824	0.1856	0.1850	0.1831	0.1841	0.1824	0.1773	0.1776	0.1773	0.1835	0.1891	0.1854	0.2107	0.1773
26	0.3045	0.2763	0.2293	0.2403	0.1314	0.1675	0.2682	0.2314	0.2580	0.1763	0.2999	0.1680	0.2525	0.1544
27	0.3239	0.3353	0.3732	0.3377	0.3239	0.3239	0.3288	0.3280	0.3288	0.3283	0.3637	0.3435	0.3342	0.2992
28	0.1995	0.1999	0.2027	0.2000	0.2017	0.2001	0.2004	0.2005	0.2004	0.2089	0.2024	0.2016	0.2013	0.2023
29	0.1200	0.1187	0.0999	0.0940	0.0990	0.0976	0.0769	0.0753	0.0766	0.0741	0.1150	0.1005	0.0800	0.1132
30	0.4983	0.4983	0.4982	0.4983	0.4983	0.4983	0.4983	0.4983	0.4983	0.4983	0.4978	0.4983	0.4982	0.4664
31	0.5796	0.5801	0.6100	0.5788	0.5796	0.5808	0.5845	0.5839	0.5845	0.6045	0.5871	0.5812	0.6438	0.5967
32	0.2753	0.2755	0.2809	0.2771	0.3006	0.2764	0.2686	0.2688	0.2686	0.2883	0.2866	0.2892	0.2670	0.3478
33	0.4001	0.3708	0.3510	0.3660	0.3553	0.3506	0.3606	0.3543	0.3590	0.3572	0.3766	0.3565	0.3560	0.3840
34	0.0946	0.0428	0.0326	0.0370	0.0114	0.0264	0.0162	0.0133	0.0190	0.0067	0.0864	0.0182	0.0200	0.0359
35	0.0991	0.0957	0.1003	0.0875	0.0858	0.0900	0.0934	0.0895	0.0938	0.1012	0.1003	0.0853	0.0950	0.1069
36	0.1704	0.1814	0.1870	0.1753	0.1807	0.1704	0.1740	0.1744	0.1740	0.1899	0.1839	0.1810	0.2022	0.1789
37	0.1769	0.1769	0.1441	0.1608	0.1074	0.1356	0.1578	0.1479	0.1578	0.1224	0.1590	0.1283	0.1406	0.1468
38	0.1183	0.1162	0.0883	0.1122	0.1106	0.1035	0.0760	0.0760	0.0758	0.0754	0.1104	0.1026	0.0828	0.1095
39	0.1445	0.1228	0.1236	0.1258	0.1149	0.1150	0.1066	0.1078	0.1065	0.1091	0.1182	0.1201	0.1053	0.1054
40	0.1437	0.1394	0.0914	0.0798	0.0887	0.0888	0.0623	0.0617	0.0623	0.0606	0.1279	0.0932	0.0683	0.0857
41	0.4237	0.4218	0.4177	0.4224	0.4228	0.4225	0.4189	0.4181	0.4171	0.4217	0.4208	0.4223	0.4164	0.3996
42	0.4297	0.4223	0.4750	0.4408	0.4563	0.4320	0.4402	0.4407	0.4402	0.4460	0.4556	0.4467	0.4474	0.4841
43	0.1899	0.1897	0.1956	0.1909	0.2001	0.1905	0.1895	0.1894	0.1895	0.2055	0.1942	0.1947	0.1884	0.2345
44	0.3541	0.3581	0.5110	0.3569	0.4485	0.3556	0.4784	0.4804	0.4784	0.3823	0.4266	0.3996	0.3324	0.3542
45	0.1454	0.1342	0.1124	0.1288	0.1213	0.1307	0.1007	0.1001	0.1008	0.1059	0.1368	0.1217	0.0991	0.1075
46	0.1671	0.1598	0.1529	0.1597	0.1495	0.1594	0.1580	0.1512	0.1541	0.1531	0.1572	0.1561	0.1484	0.1671
47	0.3997	0.3999	0.3577	0.3629	0.3426	0.3443	0.3577	0.3547	0.3561	0.3271	0.3875	0.3475	0.3536	0.4211
48	0.3845	0.3813	0.3688	0.3783	0.3794	0.3749	0.3621	0.3621	0.3606	0.3771	0.4390	0.3762	0.3645	0.3626
49	0.4729	0.4764	0.4736	0.4730	0.4870	0.4729	0.4673	0.4672	0.4618	0.4742	0.4485	0.4758	0.4627	0.5469
50	0.3004	0.2642	0.2615	0.2858	0.2593	0.2343	0.2389	0.2267	0.2380	0.2262	0.2674	0.2710	0.2590	0.2462
51	0.1537	0.1513	0.1712	0.1534	0.1539	0.1523	0.1406	0.1403	0.1406	0.2175	0.1530	0.1556	0.1438	0.2946
52	0.1032	0.1043	0.0821	0.1028	0.1107	0.1030	0.0659	0.0656	0.0658	0.0715	0.1046	0.1023	0.0948	0.2811
53	0.4648	0.4662	0.4671	0.4655	0.4666	0.4649	0.4649	0.4650	0.4649	0.4667	0.4696	0.4662	0.4764	0.4680
54	0.4337	0.4329	0.4209	0.4322	0.4050	0.4021	0.4044	0.4030	0.4044	0.2821	0.4287	0.4059	0.3956	0.1381
55	0.4001	0.3955	0.3449	0.3791	0.3596	0.3563	0.3292	0.3295	0.3286	0.3388	0.3579	0.3681	0.3294	0.2835
56	0.3275	0.3275	0.3275	0.3275	0.3275	0.3275	0.3275	0.3275	0.3275	0.3276	0.3302	0.3275	0.3276	0.3277
57	0.2476	0.2320	0.2289	0.1916	0.1888	0.2165	0.1904	0.1929	0.1892	0.1865	0.2443	0.2150	0.1840	0.2235
58	0.3364	0.3176	0.2993	0.3144	0.3123	0.2956	0.2726	0.2716	0.2701	0.3142	0.2847	0.3080	0.2730	0.2554
59	0.1354	0.1364	0.1718	0.1362	0.1483	0.1354	0.1341	0.1342	0.1341	0.1574	0.1685	0.1394	0.1616	0.1575
60	0.1214	0.1247	0.1368	0.1207	0.1243	0.1214	0.1145	0.1144	0.1145	0.1218	0.1211	0.1231	0.1169	0.1483
MR	0.2942	0.2891	0.2914	0.2852	0.2848	0.2797	0.2774	**0.2756**	0.2766	0.2802	0.2882	0.2838	0.2793	0.2921
MRR	9.1500	9.0167	8.7583	8.3750	8.7167	6.7750	5.2000	**4.6083**	4.6500	7.9250	8.8833	8.4167	5.6833	8.8417

Mean RMSE and Mean RMSE Rank are abbreviated as MR and MRR

References

Slivia Acid, Luis M. De Campos, and Javier G. Castellano. Learning Bayesian network classifiers: Searching in a space of partially directed acyclic graphs. *Machine Learning*, 59(3):213–235, 2005.

John Aitchison and I. R. Dunsmore. *Statistical Prediction Analysis.* Cambridge University Press, 1975.

Chris Atkeson, Andrew Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

Mordecai Avriel. *Nonlinear Programming: Analysis and Methods.* Courier Dover Publications, 2003.

Miriam Ayer, H. D. Brunk, G. M. Ewing, W. T. Reid, and Edward Silverman. An empirical distribution function for sampling with incomplete information. *The Annals of Mathematical Statistics*, 26(4):641–647, 1955.

Paul N. Bennett. Assessing the calibration of naive Bayes' posterior estimates. Technical Report CMU-CS-00-155, School of Computer Science, Carnegie Mellon University, 2000.

Damien Brain and Geoffrey I. Webb. The need for low bias algorithms in classification learning from large data sets. In *Proceedings of the Sixteenth European Conference on Principles of Data Mining and Knowledge Discovery*, pages 62–73. Berlin:Springer-Verlag, 2002.

Leo Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California, Berkeley, CA, 1996.

Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the Twenty-third International Conference on Machine Learning*, pages 161–168. ACM, 2006.

Jesús Cerquides and Ramon López De Mántaras. Robust Bayesian linear classifier ensembles. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 70–81, 2005a.

Jesús Cerquides and Ramon López De Mántaras. TAN classifiers based on decomposable distributions. *Machine Learning*, 59(3):323–354, 2005b.

Bojan Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, pages 147–149. London: Pitman, 1990.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

William S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:859–836, 1979.

A. Philip David. Properties of diagnostic data distributions. *Biometrics*, 32(3):647–658, 1976.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

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

Pedro Domingos and Michael 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.

James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 194–202, 1995.

Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, New York, 1973.

Charles Elkan. Boosting and naive Bayesian learning. Technical Report CS97-557, Department of Computer Science and Engineering, University of California, San Diego, 1997.

Charles Elkan. Magical thinking in data mining: Lessons from coil challenge 2000. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, pages 426–431. ACM Press, 2001.

Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1029. Morgan Kaufmann, 1993.

Eibe Frank, Mark Hall, and Bernhard Pfahringer. Locally weighted naive Bayes. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 249–256. Morgan Kaufmann, 2003.

Jerome H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, 1997.

Milton 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.

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

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2):131–163, 1997.

João Gama. Iterative Bayes. *Theoretical Computer Science*, 292(2):417–430, 2003.

Pierre Geurts. *Contributions to Decision Tree Induction: Bias/Variance Tradeoff and Time Series Classification.* PhD thesis, University of Liège, Belgium, May 2002.

Russell Greiner and Wei Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. In *Eighteenth National Conference on Artificial Intelligence*, pages 167–173. American Association for Artificial Intelligence, 2002.

Daniel Grossman and Pedro Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proceedings of the Twenty-first International Conference on Machine Learning*, pages 361–368. ACM press, 2004.

David J. Hand and Keming Yu. Idiot's bayes: Not so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer, New York, 2001.

J Hilden and B. Bjerregaard. Computer-aided diagnosis and the atypical case. In F. T. de Dombal and F. Gremy, editors, *Decision Making and Medical Care: Can Information Science Help*, pages 365–378. Amsterdam: North-Holland, 1976.

Robert C. Holte, Liane Acker, and Bruce W. Porter. Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 813–818. San Mateo, CA: Morgan Kaufmann, 1989.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003a.

Chun-Nan Hsu, Hung-Ju Huang, and Tzu-Tsung Wong. Why discretization works for naive Bayesian classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 399–406. Morgan Kaufmann, 2000.

Chun-Nan Hsu, Hung-Ju Huang, and Tzu-Tsung Wong. Implications of the Dirichlet assumption for discretization of continuous variables in naive Bayesian classifiers. *Machine Learning*, 53(3):235–263, 2003b.

Ronald L. Iman and J. M. Davenport. Approximations of the critical region of the Friedman statistic. *Communications in Statistics*, pages 571–595, 1980.

Hui Jiang, Pengfei Liu, and Imed Zitouni. Discriminative training of naive Bayes classifiers for natural language call routing. In *Proceedings of the Eighth International Conference on Spoken Language Processing*, pages 1589–1592, 2004.

Yushi Jing, Vladimir Pavlović, and James M. Rehg. Efficient discriminative learning of Bayesian network classifiers via boosted augmented naive Bayes. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 369–376, 2005.

Eamonn J. Keogh and Michael J. Pazzani. Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 225–230, 1999.

Josef Kittler. Feature selection and extraction. In Tzay Y. Young and King-Sun Fu, editors, *Handbook of Pattern Recognition and Image Processing*. Academic Press, New York, 1986.

Dan Klein and Christopher D. Manning. Conditional structure versus conditional estimation in NLP models. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 9–16. Association for Computational Linguistics, 2002.

Ron Kohavi. Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, 1996.

Ron Kohavi, Barry Becker, and Dan Sommerfield. Improving simple Bayes. In *Proceedings of the Ninth European Conference on Machine Learning*, 1997.

Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

Ron Kohavi and David 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.

Eun Bae Kong and Thomas G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 313–321, 1995.

Igor Kononenko. Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition. In B. Wielinga, J. Boose, B.Gaines, G. Schreiber, and M. van Someren, editors, *Current Trends in Knowledge Acquisition*. Amsterdam: IOS Press, 1990.

Igor Kononenko. Semi-naive Bayesian classifier. In *Proceedings of the Sixth European Working Session on Machine learning*, pages 206–219. Berlin: Springer-Verlag, 1991.

Peter A. Lachenbruch. *Discriminant Analysis*. Macmillan, 1975.

Pat Langley. Induction of recursive Bayesian classifiers. In *Proceedings of the 1993 European Conference on Machine Learning*, pages 153–164. Berlin: Springer-Verlag, 1993.

Pat Langley, Wayne Iba, and Kevin 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.

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

Helge Langseth and Thomas D. Nielsen. Classification using hierarchical naive Bayes models. *Machine Learning*, 63(2):135 – 159, 2006.

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

Clive Loader. *Local Regression and Likelihood.* Springer, 1999.

Geoffrey J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition.* John Wiley & Sons, New York, 1992.

Tom Mitchell. *Machine Learning.* McGraw Hill, 1997.

Tom Mitchell. Generative and discriminative classifiers: Naive Bayes and logistic regression. Draft book chapter of september, 2005. URL `http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf`.

D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems*, pages 841–848. MIT Press, 2001.

Kamal Nigam, John Lafferty, and Andrew Mccallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

C. Ohmann, Q. Yang, M. Knneke, H. Stöltzing, and K. Thonand W. Lorenz. Bayes theorem and conditional dependence of symptoms: Different models applied to data of upper gastrointestinal bleeding. *Methods of Information in Medicine*, 27(2):73–83, 1988.

Michael J. Pazzani. Constructive induction of Cartesian product attributes. *ISIS: Information, Statistics and Induction in Science*, pages 66–77, 1996.

Pablo Pedregal. *Introduction to Optimization*, volume 46 of *Texts in Applied Mathematics.* Springer, 2004.

John C. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers.* MIT Press, 1999.

Greg Ridgeway, David Madigan, Thomas Richardson, and John O'Kane. Interpretable boosted naive Bayes classification. In *Proceedings of the Fourth International Conference Knowledge Discovery and Data Mining*, pages 101–104, 1998.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

Tim Robertson, F. T. Wright, and R. L. Dykstra. *Order Restricted Statistical Inference.* John Wiley & Sons, 1988.

Teemu Roos, Hannes Wettig, Peter Grünwald, Petri Myllymäki, and Henry Tirri. On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, 59 (3):267–296, 2005.

Y. Dan Rubinstein and Trevor Hastie. Discriminative vs informative learning. In *Proceedings of the Third International Conference Knowledge Discovery and Data Mining*, pages 49–53. AAAI Press, 1997.

Mehran Sahami. Learning limited dependence Bayesian classifiers. In *Proceedings of the Second International Conference on Knowledge Discovery in Databases*, pages 334–338. Menlo Park, CA: AAAI Press, 1996.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2): 461–464, 1978.

Moninder Singh and Gregory M. Provan. Efficient learning of selective Bayesian network classifiers. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 453–461. Morgan Kaufmann, 1996.

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

Geoffrey I. Webb. Candidate elimination criteria for lazy Bayesian rules. In *Proceedings of the Fourteenth Australian Joint Conference on Artificial Intelligence*, volume 2256, pages 545–556. Berlin:Springer, 2001.

Geoffrey I. Webb, Janice Boughton, and Zhihai Wang. Not so naive Bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005.

Geoffrey I. Webb and Michael J. Pazzani. Adjusted probability naive Bayesian induction. In *Proceedings of the Eleventh Australian Joint Conference on Artificial Intelligence*, pages 285–295. Berlin:Springer, 1998.

Hannes Wettig, Peter Grünwald, and Teemu Roos. Supervised naive Bayes parameters. In *Proceedings of the Tenth Finnish Artificial Intelligence Conference*, pages 72–83. Finnish Artificial Intelligence Society, 2002.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

Zhipeng Xie, Wynne Hsu, Zongtian Liu, and Mong Li Lee. Snnb: A selective neighborhood based naive Bayes for lazy learning. In *Advances in Knowledge Discovery and Data Mining, Proceedings of the Pacific-Asia Conference*, pages 104–114. Berlin:Springer, 2002.

Ying Yang and Geoff Webb. On why discretization works for naive-Bayes classifiers. In *Proceedings of the Sixteenth Australian Joint Conference on Artificial Intelligence*, pages 440–452. Berlin/Heidelberg: Springer, 2003.

Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 609–616. Morgan Kaufmann, San Francisco, CA, 2001.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining*, pages 694–699. ACM Press, 2002.

Harry Zhang, Liangxiao Jiang, and Jiang Su. Hidden naive Bayes. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 919–924. AAAI Press, 2005.

Nevin Lianwen Zhang, Thomas D. Nielsen, and Finn Verner Jensen. Latent variable discovery in classification models. *Artificial Intelligence in Medicine*, 30(3):283–299, 2004.

Fei Zheng and Geoffrey I. Webb. A comparative study of semi-naive Bayes methods in classification learning. In *Proceedings of the Fourth Australasian Data Mining Conference*, pages 141–156, 2005.

Fei Zheng and Geoffrey I. Webb. Efficient lazy elimination for averaged-one dependence estimators. In *Proceedings of the Twenty-third International Conference on Machine Learning*, pages 1113–1120. ACM Press, 2006.

Fei Zheng and Geoffrey I. Webb. Finding the right family: Parent and child selection for averaged one-dependence estimators. In *Proceedings of the Eighteenth European Conference on Machine Learning*, pages 490–501. Springer Berlin / Heidelberg, 2007.

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