Expert Systems with Applications 41 (2014) 1937–1946

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks

Dewan Md. Farid [a], Li Zhang [a,*], Chowdhury Mofizur Rahman [b], M.A. Hossain [a], Rebecca Strachan [a]

[a] Computational Intelligence Group, Department of Computer Science and Digital Technology, Northumbria University, Newcastle upon Tyne, UK
[b] Department of Computer Science & Engineering, United International University, Bangladesh

ARTICLE INFO

Keywords:
Data mining
Classification
Hybrid
Decision tree
Naïve Bayes classifier

ABSTRACT

In this paper, we introduce two independent hybrid mining algorithms to improve the classification accuracy rates of decision tree (DT) and naïve Bayes (NB) classifiers for the classification of multi-class problems. Both DT and NB classifiers are useful, efficient and commonly used for solving classification problems in data mining. Since the presence of noisy contradictory instances in the training set may cause the generated decision tree suffers from overfitting and its accuracy may decrease, in our first proposed hybrid DT algorithm, we employ a naïve Bayes (NB) classifier to remove the noisy troublesome instances from the training set before the DT induction. Moreover, it is extremely computationally expensive for a NB classifier to compute class conditional independence for a dataset with high dimensional attributes. Thus, in the second proposed hybrid NB classifier, we employ a DT induction to select a comparatively more important subset of attributes for the production of naïve assumption of class conditional independence. We tested the performances of the two proposed hybrid algorithms against those of the existing DT and NB classifiers respectively using the classification accuracy, precision, sensitivity-specificity analysis, and 10-fold cross validation on 10 real benchmark datasets from UCI (University of California, Irvine) machine learning repository. The experimental results indicate that the proposed methods have produced impressive results in the classification of real life challenging multi-class problems. They are also able to automatically extract the most valuable training datasets and identify the most effective attributes for the description of instances from noisy complex training databases with large dimensions of attributes.

1. Introduction

During the past decade, a sufficient number of data mining algorithms have been proposed by the computational intelligence researchers for solving real world classification and clustering problems (Farid et al., 2013; Liao, Chu, & Hsiao, 2012; Ngai, Xiu, & Chau, 2009). Generally, classification is a data mining function that describes and distinguishes data classes or concepts. The goal of classification is to accurately predict class labels of instances whose attribute values are known, but class values are unknown. Clustering is the task of grouping a set of instances in such a way that instances within a cluster have high similarities in comparison to one another, but are very dissimilar to instances in other clusters. It analyzes instances without consulting a known class label. The instances are clustered based on the principle of maximizing the intraclass similarity and minimizing the interclass similarity. The performance of data mining algorithms in most cases depends on dataset quality, since low-quality training data may lead to the construction of overfitting or fragile classifiers. Thus, data preprocessing techniques are needed, where the data are prepared for mining. It can improve the quality of the data, thereby helping to improve the accuracy and efficiency of the mining process. There are a number of data preprocessing techniques available such as (a) data cleaning: removal of noisy data, (b) data integration: merging data from multiple sources, (c) data transformations: normalization of data, and (d) data reduction: reducing the data size by aggregating and eliminating redundant features.

This paper presents two independent hybrid algorithms for scaling up the classification accuracy of decision tree (DT) and naïve Bayes (NB) classifiers in multi-class classification problems. DT is a classification tool commonly used in data mining tasks such as ID3 (Quinlan, 1986), ID4 (Utgoff, 1989), ID5 (Utgoff, 1988), C4.5 (Quinlan, 1993), C5.0 (Bujlow, Riaz, & Pedersen, 2012), and CART (Breiman, Friedman, Stone, & Olshen, 1984). The goal of DT is to create a model that predicts the value of a target class for an unseen test instance based on several input features (Loh & Shih, 1997; Safavian & Landgrebe, 1991; Turney, 1995). Amongst other data mining methods, DTs have various advantages: (a) simple to understand, (b) easy to implement, (c) requiring little prior

* Corresponding author. Tel.: +44 191 243 7089.
E-mail address: li.zhang@northumbria.ac.uk (L. Zhang).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.08.089

knowledge, (d) able to handle both numerical and categorical data, (e) robust, and (f) dealing with large and noisy datasets. A naïve Bayes (NB) classifier is a simple probabilistic classifier based on: (a) Bayes theorem, (b) strong (naïve) independence assumptions, and (c) independent feature models (Farid, Rahman, & Rahman, 2011, 2010; Lee & Isa, 2010). It is also an important mining classifier for data mining and applied in many real world classification problems because of its high classification performance. Similar to DT, the NB classifier also has several advantages such as (a) easy to use, (b) only one scan of the training data required, (c) handling missing attribute values, and (d) continuous data.

In this paper, we propose two hybrid algorithms respectively for a DT classifier and a NB classifier for multi-class classification tasks. The first proposed hybrid DT algorithm finds the troublesome instances in the training data using a NB classifier and removes these instances from the training set before constructing the learning tree for decision making. Otherwise, DT may suffer from overfitting due to the presence of such noisy instances and its accuracy may decrease. Moreover, it is also noted that to compute class conditional independence using a NB classifier is extremely computationally expensive for a dataset with many attributes. Our second proposed hybrid NB algorithm finds the most crucial subset of attributes using a DT induction. The weights of the selected attributes by DT are also calculated. Then only these most important attributes selected by DT with their corresponding weights are employed for the calculation of the naïve assumption of class conditional independence. We evaluate the performances of the proposed hybrid algorithms against those of existing DT and NB classifiers using the classification accuracy, precision, sensitivity–specificity analysis, and 10-fold cross validation on 10 real benchmark datasets from UCI (University of California, Irvine) machine learning repository (Frank & Asuncion, 2010). The experimental results prove that the proposed methods have produced very promising results in the classification of real world challenging multi-class problems. These methods also allow us to automatically extract the most representative high quality training datasets and identify the most important attributes for the characterization of instances from a large amount of noisy training data with high dimensional attributes.

The rest of the paper is organized as follows. Section 2 gives an overview of the work related to DT and NB classifiers. Section 3 introduces the basic DT and NB classification techniques. Section 4 presents our proposed two hybrid algorithms for the multi-class classification problems respectively based on DT and NB classifiers. Section 5 provides experimental results and a comparison against existing DT and NB algorithms using 10 real benchmark datasets from UCI machine learning repository. Finally, Section 6 concludes the findings and proposes directions for future work.

2. Related work

In this section, we review recent research on decision trees and naïve Bayes classifiers for various real world multi-class classification problems.

2.1. Decision trees

Decision tree classification provides a rapid and useful solution for classifying instances in large datasets with a large number of variables. There are two common issues for the construction of decision trees: (a) the growth of the tree to enable it to accurately categorize the training dataset, and (b) the pruning stage, whereby superfluous nodes and branches are removed in order to improve classification accuracy. Franco-Arcega, Carrasco-Ochoa, Sanchez-Diaz, and Martinez-Trinidad (2011) presented decision trees using fast splitting attribute selection (DTFS), an algorithm for building DTs for large datasets. DTFS used this attribute selection technique to expand nodes and process all the training instances in an incremental way. In order to avoid storing all the instances into the DT, DTFS stored at most N number of instances in a leaf node. When the number of instances stored in a leaf node reached its limit, DTFS expended or updated the leaf node according to the class labels of the instances stored in it. If the leaf node of DT contained training instances from only one class, then DTFS updated the value of the input tree branch of this leaf node. Otherwise, DTFS expanded the leaf node by choosing a splitting attribute and created one branch for each class of the stored instances. In this approach, DTFS always considered a small number of instances in the main memory for the building of a DT. Aviad and Roy (2011) also introduced a decision tree construction method based on adjusted cluster analysis classification called classification by clustering (CbC). It found similarities between instances using clustering algorithms and also selected target attributes. Then it calculated the target attributes distribution for each cluster. When a threshold for the number of instances stored in a cluster was reached, all the instances in each cluster were classified with respect to the appropriate value of the target attribute.

Polat and Gunes (2009) proposed a hybrid classification system based on a C4.5 decision tree classifier and a one-against-all method to improve the classification accuracy for multi-class classification problems. Their one-against-all method constructed M number of binary C4.5 decision tree classifiers, each of which separated one class from all of the rest. The ith C4.5 decision tree classifier was trained with all the training instances of the ith class with positive labels and all the others with negative labels. The performance of this hybrid classifier was tested using the classification accuracy, sensitivity-specificity analysis, and 10-fold cross validation on three datasets taken from the UCI machine learning repository (Frank & Asuncion, 2010). Balamurugan and Rajaram (2009) proposed a method to resolve one of the exceptions in basic decision tree induction algorithms when the class prediction at a leaf node cannot be determined by majority voting. The influential factor of attributes was found in their work, which gave the dependability of the attribute value on the class label. The DT was pruned based on this factor. When classifying new instances using this pruned tree, the class labels can be assigned more accurately than the basic assignment by traditional DT algorithms.

Chen and Hung (2009) presented an associative classification tree (ACT) that combined the advantages of both associative classification and decision trees. The ACT tree was built using a set of associative classification rules with high classification predictive accuracy. ACT followed a simple heuristic which selected the attribute with the highest gain measure as the splitting attribute. Chandra and Varghese (2009) proposed a fuzzy decision tree Gini Index based (G-FDT) algorithm to fuzzify the decision boundary without converting the numeric attributes into fuzzy linguistic terms. The G-FDT tree used the Gini Index as the split measure to choose the most appropriate splitting attribute for each node in the decision tree. For the construction of the decision tree, the Gini Index was computed using the fuzzy-membership values of the attribute corresponding to a split value and fuzzy-membership values of the instances. The split-points were chosen as the midpoints of attribute values where the class information changed. Aitkenhead (2008) presented a co-evolving decision tree method, where a large number of variables in datasets were being considered. They proposed a novel combination of DTs and evolutionary methods, such as the bagging approach of a DT classifier and a back-propagation neural network method, to improve the classification accuracy. Such methods evolved the structure of a decision tree and also handled comparatively a wider range of values and data types.

2.2. Naïve Bayes classifiers

The naïve Bayes classifier is also widely used for classification problems in data mining and machine learning fields because of its simplicity and impressive classification accuracy. Koc, Mazzuchi, and Sarkani (2012) applied a hidden naïve Bayes (HNB) classifier to a network intrusion detection system (NIDS) to classify network attacks. It especially significantly improved the accuracy for the detection of denial-of-services (DoS) attacks. The HNB classifier was an extended version of a basic NB classier. It relaxed the conditional independence assumption imposed on the basic NB classifier. The HNB method was based on the idea of creating another layer that represented a hidden parent of each attribute. The influences from all of the other attributes can thus be easily combined through conditional probabilities by estimating the attributes from the training dataset. The HNB multiclass classification model exhibited a superior overall performance in terms of accuracy, error rate and misclassification cost compared with the traditional NB classifier on the KDD 99 dataset (McHugh, 2000; Tavallaee, Bagheri, Lu, & Ghorbani, 2009). Valle, Varas, and Ruz (2012) also presented an approach to predict the performance of sales agents of a call centre dedicated exclusively to sales and telemarketing based on a NB classifier. This model was tested using socio-demographic (age, gender, marital status, socioeconomic status and experience) and performance (logged hours, talked hours, effective contacts and finished records) information as attributes of individuals. The results showed that the socio-demographic attributes were not suitable for predicting sale performances, but operational records proved to be useful for the prediction of the performances of sales agents.

Chandra and Gupta (2011) proposed a robust naïve Bayes classifier (R-NBC) to overcome two major limitations i.e., underflow and over-fitting for the classification of gene expression datasets. R-NBC used logarithms of probabilities rather than multiplying probabilities to handle the underflow problem and employed an estimate approach for providing solutions to over-fitting problems. It did not require any prior feature selection approaches in the field of microarray data analysis where a large number of attributes were considered. Fan, Poh, and Zhou (2010) proposed a partition-conditional independent component analysis (PC-ICA) method for naïve Bayes classification in microarray data analysis. It further extended the class-conditional independent component analysis (CC-ICA) method. PC-ICA spited the small-size data samples into different partitions so that independent component analysis (ICA) can be done within each partition. PC-ICA also attempted to do ICA-based feature extraction within each partition that may consist of several classes. Hsu, Huang, and Chang (2008) presented a classification method called extended naïve Bayes (ENB) for the classification of mixed types of data. The mixed types of data included categorical and numeric data. ENB used a normal NB algorithm to calculate the probabilities of categorical attributes. When handling numeric attributes, it adopted the statistical theory to discrete the numeric attributes into symbols by considering the average and variance of numeric values.

3. Supervised classification

Classification is one of the most popular data mining techniques that can be used for intelligent decision making. In this section, we discuss some basic techniques for data classification using decision tree and naïve Bayes classifiers. Table 1 summarizes the most commonly used symbols and terms throughout the paper.

3.1. Decision tree induction

A decision tree classifier is typically a top-down greedy approach, which provides a rapid and effective method for classifying data instances (Chandra & Paul Varghese, 2009; Jamain & Hand, 2008). Generally, each DT is a rule set. DT recursively partitions the training datasets into smaller subsets until all the subsets belong to a single class. The common DT algorithm is ID3 (Iterative Dichotomiser), which uses information theory as its attribute selection measure (Quinlan, 1986). The root node of DT is chosen based on the highest information gain of the attribute. Given a training dataset, D, the expected information needed to correctly classify an instance, $x_i \in D$, is given in Eq. (1), where p_i is the probability that $x_i \in D$, belongs to a class, C_i, and is estimated by $|C_i,D|/|D|$.

Table 1
Commonly used symbols and terms.

Symbol	Term
x_i	A data point or instance
X	A set of instances
A_i	An attribute
A_{ij}	An attribute's value
W_i	The weight of attribute A_i
C	Total number of classes in a training dataset
C_i	A class label
D	A training dataset
D_i	A subset of a training dataset
T	A decision tree

$$Info(D) = -\sum_{i=1}^{m} p_i log_2(p_i) \tag{1}$$

In Eq. (1), $Info(D)$ is the average amount of information needed to identify C_i of an instance, $x_i \in D$. The goal of DT is to iteratively partition, D, into subsets, $\{D_1,D_2,\ldots,D_n\}$, where all instances in each D_i belong to the same class, C_i. $Info_A(D)$ is the expected information required to correctly classify an instance, x_i, from D based on the partitioning by attributes, A. Eq. (2) shows $Info_A(D)$ calculation, where $\frac{|D_j|}{|D|}$ acts as the weight of the jth partition.

$$Info_A(D) = \sum_{j=1}^{n} \frac{|D_j|}{|D|} \times Info(D_j) \tag{2}$$

Information gain is defined as the difference between the original information requirement and the new requirement that is shown in Eq. (3).

$$Gain(A) = Info(D) - Info_A(D) \tag{3}$$

The Gain Ratio is an extension to the information gain approach, also used in DT such as C4.5. A C4.5 classifier is a successor of ID3 classifier (Quinlan, 1993). It applies a kind of normalization to information gain using a "split information" value defined analogously with $Info(D)$ as shown in Eq. (4).

$$SplitInfo_A(D) = -\sum_{j=1}^{n} \frac{|D_j|}{|D|} \times log_2\left(\frac{|D_j|}{|D|}\right) \tag{4}$$

The attribute with the maximum Gain Ratio is selected as the splitting attribute, which is defined in Eq. (5).

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)} \tag{5}$$

Eq. (6) defines the gini for a dataset, D, where, p_j, is the frequency of class $C_j \in D$.

$$Gini(D) = 1 - \sum_{j=1}^{m} p_j^2 \tag{6}$$

The goodness of a split of D into subsets D_1 and D_2 is defined by Eq. (7).

$$gini_{split}(D) = \frac{n_1}{n(gini(D_1))} + \frac{n_2}{n(gini(D_2))} \quad (7)$$

In this way, the split with the best gini value is chosen. To illustrate the operation of DT, we consider a small dataset in Table 2 described by four attributes namely Outlook, Temperature, Humidity, and Wind, which represent the weather condition of a particular day. Each attribute has several unique attribute values. The Play column in Table 2 represents the class category of each instance. It indicates whether a particular weather condition is suitable or not for playing tennis. Fig. 1 shows the decision tree model constructed using the playing tennis dataset shown in Table 2.

3.2. Naïve Bayes classification

A naïve Bayes classifier is a simple probabilistic based method, which can predict the class membership probabilities (Chen, Huang, Tian, & Qu, 2009; Farid & Rahman, 2010). It has several advantages: (a) easy to use, and (b) only one scan of the training data required for probability generation. A NB classifier can easily handle missing attribute values by simply omitting the corresponding probabilities for those attributes when calculating the likelihood of membership for each class. The NB classifier also requires the class conditional independence, i.e., the effect of an attribute on a given class is independent of those of other attributes.

Given a training dataset, $D = \{X_1, X_2, \ldots, X_n\}$, each data record is represented as, $X_i = \{x_1, x_2, \ldots, x_n\}$. D contains the following attributes $\{A_1, A_2, \ldots, A_n\}$ and each attribute A_i contains the following attribute values $\{A_{i1}, A_{i2}, \ldots, A_{ih}\}$. The attribute values can be discrete or continuous. D also contains a set of classes $C = \{C_1, C_2, \ldots, C_m\}$. Each training instance, $X \in D$, has a particular class label C_i. For a test instance, X, the classifier will predict that X belongs to the class with the highest posterior probability, conditioned on X. That is, the NB classifier predicts that the instance X belongs to the class C_i, if and only if $P(C_i|X) > P(C_j|X)$ for $1 \leqslant j \leqslant m, j \neq i$. The class C_i for which $P(C_i|X)$ is maximized is called the Maximum Posteriori Hypothesis.

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \quad (8)$$

In Bayes theorem shown in Eq. (8), as $P(X)$ is a constant for all classes, only $P(X|C_i)P(C_i)$ needs to be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, $P(C_1) = P(C_2) = \cdots = P(C_m)$, and therefore maximize $P(X|C_i)$. Otherwise, maximize $P(X|C_i)P(C_i)$. The class prior probabilities are calculated by $P(C_i) = |C_{i,D}|/|D|$, where $|C_{i,D}|$ is the number of training instances belonging to the class C_i in D. To compute $P(X|C_i)$ in a dataset with many attributes is extremely computationally expensive. Thus, the naïve assumption of class conditional independence is made in order to reduce computation in evaluating $P(X|C_i)$. The attributes are conditionally independent of one another, given the class label of the instance. Thus, Eqs. (9) and (10) are used to produce $P(X|C_i)$.

Table 2
The playing tennis dataset.

Outlook	Temperature	Humidity	Wind	Play
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

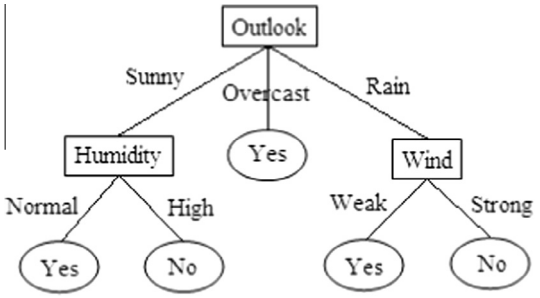

Fig. 1. A decision tree generated using the playing tennis dataset.

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \quad (9)$$

$$P(X|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i) \quad (10)$$

In Eq. (9), x_k refers to the value of attribute A_k for instance X. Therefore, these probabilities $P(x_1|C_i), P(x_2|C_i), \ldots, P(x_n|C_i)$ can be easily estimated from the training instances. Moreover, the attributes in training datasets can be categorical or continuous-valued. If the attribute value, A_k, is categorical, then $P(x_k|C_i)$ is the number of instances in the class $C_i \in D$ with the value x_k for A_k, divided by $|C_{i,D}|$, i.e., the number of instances belonging to the class $C_i \in D$.

If A_k is a continuous-valued attribute, then A_k is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ, defined respectively by the following two equations:

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) \quad (11)$$

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-p)^2}{2\sigma^2}} \quad (12)$$

In Eq. (11), μ_{C_i} is the mean and σ_{C_i} is the standard deviation of the values of the attribute A_k for all training instances in the class C_i. Now we can bring these two quantities to Eq. (12), together with x_k, in order to estimate $P(x_k|C_i)$. To predict the class label of instance X, $P(X|C_i)P(C_i)$ is evaluated for each class $C_i \in D$. The NB classifier predicts that the class label of instance X is the class C_i, if and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad (13)$$

In Eq. (13), $1 \leqslant j \leqslant m$ and $j \neq i$. That is the predicted class label is the class C_i for which $P(X|C_i)P(C_i)$ is the maximum probability. Tables 3 and 4 respectively tabulate the prior probabilities for each class and conditional probabilities for each attribute value generated using the playing tennis dataset shown in Table 2.

4. The proposed hybrid learning algorithms

In this paper, we have proposed two independent hybrid algorithms respectively for decision tree and naïve Bayes classifiers to improve the classification accuracy in multi-class classification tasks. These proposed algorithms are described in the following

Table 3
Prior probabilities for each class generated using the playing tennis dataset.

Probability	Value
$P(Play = Yes)$	9/14 = 0.642
$P(Play = No)$	5/14 = 0.375

Table 4
Conditional probabilities for each attribute value calculated using the playing tennis dataset.

Probability	Value
$P(Outlook = Sunny\|Play = Yes)$	2/9 = 0.222
$P(Outlook = Sunny\|Play = No)$	3/5 = 0.6
$P(Outlook = Overcast\|Play = Yes)$	4/9 = 0.444
$P(Outlook = Overcast\|Play = No)$	0/5 = 0.0
$P(Outlook = Rain\|Play = Yes)$	3/9 = 0.3
$P(Outlook = Rain\|Play = No)$	2/5 = 0.4
$P(Temperature = Hot\|Play = Yes)$	2/9 = 0.222
$P(Temperature = Hot\|Play = No)$	2/5 = 0.4
$P(Temperature = Mild\|Play = Yes)$	4/9 = 0.444
$P(Temperature = Mild\|Play = No)$	2/5 = 0.4
$P(Temperature = Cool\|Play = Yes)$	3/9 = 0.333
$P(Temperature = Cool\|Play = No)$	1/5 = 0.2
$P(Humidity = High\|Play = Yes)$	3/9 = 0.333
$P(Humidity = High\|Play = No)$	4/5 = 0.8
$P(Humidity = Normal\|Play = Yes)$	6/9 = 0.666
$P(Humidity = Normal\|Play = No)$	1/5 = 0.2
$P(Wind = Weak\|Play = Yes)$	6/9 = 0.666
$P(Wind = Weak\|Play = No)$	2/5 = 0.4
$P(Wind = Strong\|Play = Yes)$	3/9 = 0.333
$P(Wind = Strong\|Play = No)$	3/5 = 0.6

Sections 4.1 and 4.2. Algorithm 1 is used to describe the proposed hybrid DT induction, which employs a NB classifier to remove any noisy training data at an initial stage to avoid overfitting. Algorithm 2 is used for the construction of a hybrid NB classifier. It embeds a DT classifier to identify a subset of most important attributes to improve efficiency.

4.1. The proposed hybrid decision tree algorithm

In this section, we discuss the proposed Algorithm 1 of the hybrid DT induction. It is developed based on a basic C4.5 algorithm. Given a training dataset, $D = \{x_1, x_2, \ldots, x_n\}$, each training instance is represented as $x_i = \{x_{i1}, x_{i2}, \ldots, x_{ih}\}$ and D contains the following attributes $\{A_1, A_2, \ldots, A_n\}$. Each attribute, A_i, contains the following attribute values $\{A_{i1}, A_{i2}, \ldots, A_{ih}\}$. The training data also belong to a set of classes $C = \{C_1, C_2, \ldots, C_m\}$. A decision tree is a classification tree associated with D that has the following properties: (a) each internal node labeled with an attribute, A_i, (b) each arc labeled with a predicate that can be applied to the attribute associated with the parent, and (c) each leaf node labeled with a class, C_i. Once the tree is built, it is used to classify each test instance, $x_i \in D$. The result is a classification for that instance, x_i. There are two basic steps for the development of a DT based application: (a) building the DT from a training dataset, and (b) applying the DT to a test dataset, D.

For the training dataset, D, we first apply a basic NB classifier to classify each training instance, $x_i \in D$. We calculate the prior probability, $P(C_i)$, for each class, $C_i \in D$ and the class conditional probability, $P(A_{ij}|C_i)$, for each attribute value (even if it is numeric) in D. Then we classify each training instance, $x_i \in D$, using these probabilities. The class, C_i, with the highest posterior probability, $P(C_i|x_i)$, is selected as the final classification for the instance, x_i. Then we remove all the misclassified training instances from the dataset D. In our experiments, these misclassified instances tend to be the troublesome training examples. For example, some of these examples either contain contradictory characteristics, or carry exceptional features. Suppose there is a training dataset with two classes. We calculate the prior and class conditional probabilities using this example training dataset. Then we calculate the $P(Class|D)$ for each instance based on these probabilities. We have found some instances where the probabilities calculated using the NB classifier indicate that they belong to "Class = yes". However in the training dataset they are labeled as "Class = no". It seems that there is some noise within these data, which leads to contradictory results in comparison with the original labels. Thus these misclassified instances are regarded as troublesome examples. The presence of such noisy training instances is more likely to cause a DT classifier to become overfitting, and thus decrease its accuracy.

After removing those misclassified/troublesome instances from the training dataset, D, we subsequently build a DT for decision making using the updated training dataset D with those purely noise free data. For the decision tree generation, we select the best splitting attribute with the maximum information gain value as the root node of the tree. Once the root node of DT has been determined, the child nodes and its arcs are created and added to the DT. The algorithm continues recursively by adding new subtrees to each branching arc. The algorithm terminates when the instances in the reduced training set all belong to the same class. This class is then used to label the corresponding leaf node. Algorithm 2 outlines the proposed DT algorithm. The time and space complexity of a DT algorithm depends on the size of training dataset, the number of attributes, and the size of the generated tree.

Algorithm 1. Decision tree induction

Input: $D = \{x_1, x_2, \ldots, x_n\}$ // Training dataset, D, which contains a set of training instances and their associated class labels.
Output: T, Decision tree.
Method: 1: **for** each class, $C_i \in D$, **do**
2: Find the prior probabilities, $P(C_i)$.
3: **end for**
4: **for** each attribute value, $A_{ij} \in D$, **do**
5: Find the class conditional probabilities, $P(A_{ij}|C_i)$.
6: **end for**
7: **for** each training instance, $x_i \in D$, **do**
8: Find the posterior probability, $P(C_i|x_i)$
9: **if** x_i is misclassified, **do**
10: Remove x_i from D;
11: **end if**
12: **end for**
13: $T = \emptyset$;
14: Determine best splitting attribute;
15: T = Create the root node and label it with the splitting attribute;
16: T = Add arc to the root node for each split predicate and label;
17: **for** each arc **do**
18: D = Dataset created by applying splitting predicate to D;
19: **if** stopping point reached for this path,
20: T' = Create a leaf node and label it with an appropriate class;
21: **else**
22: T' = DTBuild(D);
23: **end if**
24: T = Add T' to arc;
25: **end for**

4.2. The proposed hybrid algorithm for a naïve Bayes classifier

In this section, we present a hybrid naïve Bayes classifier with the integration of a decision tree in order to find a subset of attributes with attribute weighting, which play more important roles in class determination. In a given training dataset, each instance, x_i, contains values $\{x_{i1}, x_i, \ldots, x_{ih}\}$. There is a set of attributes used to describe the training data, $D = \{A_1, A_2, \ldots, A_n\}$. Each attribute contains attribute values $A_i = \{A_{i1}, A_{i2}, \ldots, A_{ik}\}$. A set of classes $C = \{C_1, C_2, \ldots, C_n\}$ is also used to label the training instances, where each

class $C_i = \{C_{i1}, C_{i2}, \ldots, C_{ik}\}$ also has some values. First of all, in this proposed hybrid NB algorithm, we generate a basic decision tree, T, from the training dataset, D, and collect the attributes appearing in the tree. In this approach, DT is applied as an attribute selection and attribute weighting method. That is we use the DT classifier to find the subset of the attributes in the training dataset which play crucial roles in the final classification. After the tree construction, we initialize the weight, W_i, for each attribute, $A_i \in D$. If the attribute, $A_i \in D$, is not tested in the DT then the weight, W_i, of the attribute, A_i, is initialized to zero. Otherwise, we calculate the minimum depth, d, where the attribute, $A_i \in T$, is tested in the DT and initialize the weight, W_i, of the attribute, A_i, with the value of $\frac{1}{\sqrt{d}}$. In this way, the importance of the attributes is measured by their corresponding weights. For example, the root node of the tree has a higher weight value in comparison with those of its child nodes. Subsequently, we calculate the class conditional probabilities using only those attributes selected by DT (i.e., $W_i \neq 0$) and classify each instance $x_i \in D$ using these probabilities. The weights of the selected attributes by the DT are also used as exponential parameters (see Eq. (14)) for class conditional probability calculation. Other attributes which are not selected by the DT (i.e., $W_i = 0$) will not be considered in the final result probability calculation. I.e., the class conditional probabilities of those unselected attributes by DT will not be generated and employed in the classification result production.

Moreover, we calculate the prior probability, $P(C_i)$, for each class, C_i, by counting how often C_i occurs in D. For each attribute, A_i, the number of occurrences of each attribute value, A_{ij}, can be counted to determine $P(A_i)$. Similarly, the probability $P(A_{ij}|C_i)$ also can be estimated by counting how often each A_{ij} occurs in $C_i \in D$. This is done only for those attributes that appear in the DT, T. To classify an instance, $x_i \in D$,$P(C_i)$ and $P(A_{ij}|C_i)$ from D are used to make the prediction. This is conducted by combining the effects of different attribute values, $A_{ij} \in x_i$. As mentioned earlier, the weights of the selected attributes also influence their class conditional probability calculation as exponential parameters. We estimate $P(x_i|C_i)$ by Eq. (14).

$$P(x_i|C_i) = \prod_{j=1}^{n} P(A_{ij}|C_i)^{W_i} \tag{14}$$

In Eq. (14), W_i refers to the weight of the attribute, A_i, which effects on class conditional probability calculation as an exponential parameter. To calculate $P(C_i|x_i)$, we need $P(C_i)$ for each C_i, and $P(x_i|C_i)$, and estimate the likelihood that x_i is in each C_i. The posterior probability, $P(C_i|x_i)$, is then found for C_i. The class, C_i, with the highest probability is used to label the instance, x_i. Algorithm 2 outlines the proposed hybrid algorithm for the naïve Bayes classifier.

Algorithm 2. Naïve Bayes classifier

Input: $D = \{x_1, x_2, \ldots, x_n\}$ // Training data.
Output: A classification Model.
Method: 1: $T = \emptyset$;
2: Determine the best splitting attribute;
3: T = Create the root node and label it with the splitting attribute;
4: T = Add arc to the root node for each split predicate and label;
5: **for** each arc **do**
6: D = Dataset created by applying splitting predicate to D;
7: **if**stopping point reached for this path, **then**
8: T' = Create a leaf node and label it with an appropriate class;
9: **else**
10: T' = DTBuild(D);
11: **end if**
12: T = Add T' to arc;
13: **end for**
14: **for** each attribute, $A_i \in D$, **do**
15: **if** A_i is not tested in T, **then**
16: $W_i = 0$;
17: **else**
18: d as the minimum depth of $A_i \in T$, and $W_i = \frac{1}{\sqrt{d}}$;
19: **end if**
20: **end for**
21: **for** each class, $C_i \in D$, **do**
22: Find the prior probabilities, $P(C_i)$.
23: **end for**
24: **for** each attribute, $A_i \in D$ and $W_i \neq 0$, **do**
25: **for** each attribute value, $A_{ij} \in A_i$, **do**
26: Find the class conditional probabilities, $P(A_{ij} \mid C_i)^{W_i}$.
27: **end for**
28: **end for**
29: **for** each instance, $x_i \in D$, **do**
30: Find the posterior probability, $P(C_i|x_i)$;
31: **end for**

5. Experiments

In this section, we describe the test datasets and experimental environments, and present the evaluation results for both of the proposed hybrid decision tree and naïve Bayes classifiers.

5.1. Datasets

The performances of both of the proposed hybrid decision tree and naïve Bayes algorithms are tested on 10 real benchmark datasets from UCI machine learning repository (Frank & Asuncion, 2010). Table 5 describes the datasets used in experimental analysis. Each dataset is roughly equivalent to a two-dimensional spreadsheet or a database table. The 10 datasets are:

1. Breast Cancer Data (Breast cancer).
2. Fitting Contact Lenses Database (Contact lenses).
3. Pima Indians Diabetes Database (Diabetes).
4. Glass Identification Database (Glass).
5. Iris Plants Database (Iris plants).
6. Large Soybean Database (Soybean).
7. 1984 United States Congressional Voting Records Database (Vote).
8. Image Segmentation Data (Image seg.)
9. Tic-Tac-Toe Endgame Data (Tic-Tac-Toe).
10. NSL-KDD Dataset (NSL-KDD).

Table 5
Dataset descriptions.

Datasets	No of Att.	Att. Types	Instances	Classes
Breast cancer	9	Nominal	286	2
Contact lenses	4	Nominal	24	3
Diabetes	8	Real	768	2
Glass	9	Real	214	7
Iris plants	4	Real	150	3
Soybean	35	Nominal	683	19
Vote	16	Nominal	435	2
Image seg.	19	Real	1500	7
Tic-Tac-Toe	9	Nominal	958	2
NSL-KDD	41	Real & Nominal	25192	23

5.2. Experimental setup

The experiments were conducted using a machine with an Intel Core 2 Duo Processor 2.0 GHz processor (2 MB Cache, 800 MHz FSB) and 1 GB of RAM. We implement both of the proposed algorithms (Algorithms 1 and 2) in Java. We use NetBeans IDE 7.1 in Redhat enterprise Linux 5 for Java coding. NetBeans IDE is the first IDE providing support for JDK 7 and Java EE 6 (http://netbeans.org/index.html). The code for the basic versions of the DT and NB classifiers is adopted from Weka3, which is open source data mining software (Hall et al., 2009). It is a collection of machine learning algorithms for data mining tasks. Weka3 contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. The mining algorithms in Weka3 can be either applied directly to a dataset or called from our own coding.

To test the proposed hybrid methods, we have used the classification accuracy, precision, sensitivity-specificity analysis, and 10-fold cross validation. The classification accuracy is measured either by Eq. (15) or by Eq. (16).

$$accuracy = \frac{\sum_{i=1}^{|X|} assess(x_i)}{|X|}, \quad x_i \in X \tag{15}$$

If $classify(x) = x \cdot c$ then $assess(x) = 1$ else $assess(x) = 0$, where X is the set of instances to be classified, $x \in X$ and $x \cdot c$ is the class of instance, x. Also, $classify(x)$ returns the classification of x. Eqs. (17)–(19) are used for the calculations of precision, sensitivity (also called the true positive rate, or the recall rate), and specificity (also called the true negative rate). A perfect classifier would be described as 100% sensitivity and 100% specificity. Table 6 summarizes the symbols and terms used throughout in Eqs. (16)–(19).

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{16}$$

$$precision = \frac{TP}{TP + FP} \tag{17}$$

$$sensitivity = \frac{TP}{TP + FN} \tag{18}$$

$$specificity = \frac{TN}{TN + FP} \tag{19}$$

where, TP, TN, FP and FN denote true positives, true negatives, false positives, and false negatives, respectively.

In k-fold cross-validation, the initial data are randomly partitioned into k mutually exclusive subsets or "folds", $D_1, D_2, \ldots, D_k$, each of which has an approximately equal size. Training and testing are performed k times. In iteration i, the partition D_i is reserved as the test set, and the remaining partitions are collectively used to train the classifier. 10-fold cross validation breaks data into 10 sets of size N/10. It trains the classifier on 9 datasets and tests it using the remaining one dataset. This repeats 10 times and we take a mean accuracy rate. For classification, the accuracy estimate is the overall number of correct classifications from the k iterations, divided by the total number of instances in the initial dataset.

Table 6
Symbols used in Eqs. (16)–(19) and their meanings.

Symbol	Intuitive Meaning
TP	x_i predicted to be in C_i and is actually in it
TN	x_i not predicted to be in C_i and is not actually in it
FP	x_i predicted to be in C_i but is not actually in it
FN	x_i not predicted to be in C_i but is actually in it
accuracy	"%" of predictions that are correct
precision	"%" of positive predictions that are correct
sensitivity	"%" of positive instances that are predicted as positive
specificity	"%" of negative instances that are predicted as negative

5.3. Results and discussion

Firstly, we evaluated the performances of proposed algorithms (Algorithms 1 and 2) against existing DT and NB classifiers using the classification accuracy on the training sets of the 10 benchmark datasets shown in Table 5. Table 7 summarizes the classification accuracy rates of the basic C4.5 and NB classifiers, and the proposed hybrid algorithms for each of the 10 training datasets.

The results in Table 7 indicate that the proposed DT algorithm (Algorithm 1) outperforms the C4.5 DT classifier for the classification of the diabetes dataset by 7.55%, the tic-tac-toe dataset by 4.7%, the contact lenses dataset by 4.17% and the NSL-KDD dataset by 3.89%. Algorithm 1 is capable of identifying the noisy instances from each dataset before the DT induction. This DT classifier generated from the updated noise free high quality representative training dataset is less likely to become overfitting and thus able to carry more generalization capabilities comparing to the DT generated directly from the original training instances using C4.5 algorithm.

Moreover, in Table 7, the proposed NB algorithm (Algorithm 2) also outperforms the traditional NB classifier for the classification of all 10 training datasets, since it is able to identify the most important and discriminative subset of attributes for the production of naïve assumption of class conditional independence comparing to the basic NB classifier. Among all the 10 datasets, the proposed NB algorithm respectively improves the classification rates of the glass dataset by 18.23%, the tic-tac-toe dataset by 8.98%, the image seg. dataset by 4.87% and the contact lenses dataset by 4.17%.

Secondly, we have used classification accuracy, precision, sensitivity, specificity, and 10-fold cross validation to measure the performances of the proposed algorithms (Algorithms 1 and 2) using all 10 datasets. We consider the weighted average for precision, sensitivity and specificity analysis for each dataset. The weighted average is similar to an arithmetic mean, where instead of each of the data points contributing equally to the final average, some data points contribute more than others. For example, the weighting for the evaluation of this research is calculated using the number of instances belonging to one class divided by the total number of instances in one dataset. The detailed results for the weighted average of precision, sensitivity and specificity analysis for the experiments conducted are presented in Tables 8–11.

Evaluated using all the instances in the 10 datasets, the traditional C4.5 DT achieved an average accuracy rate of 83.5% using 10-fold cross validation. The proposed DT algorithm (Algorithm 1) obtained an average accuracy rate of 88.3% for the classification of the 10 datasets. Tables 8 and 9 respectively tabulate the performances of the classic C4.5 classifier and the proposed DT algorithm on 10 datasets using 10-fold cross validation.

Table 7
The classification accuracies of classifiers using training datasets.

Training datasets	C4.5 DT classifier (%)	NB classifier (%)	Proposed DT classifier (%) (Algorithm 1)	Proposed NB classifier(%) (Algorithm 2)
Breast cancer	96.33	93.7	98.82	95.9
Contact lenses	91.66	95.83	95.83	100
Diabetes	84.11	76.30	91.66	79.55
Glass	96.26	55.60	97.66	73.83
Iris plants	98	96	100	98.66
Soybean	96.33	93.7	99.85	96.63
Vote	97.24	90.34	98.85	93.33
Image seg.	99.0	81.66	99.6	86.53
Tic-Tac-Toe	93.73	69.83	98.43	78.81
NSL-KDD	73.37	79.94	77.26	83.44

Table 8
The classification accuracies and precision, sensitivity and specificity values for a C4.5 classifier with 10-fold cross validation.

Datasets	Classification accuracy (%)	Precision (weighted avg.)	Sensitivity (weighted avg.)	Specificity (weighted avg.)
Breast cancer	75.52	0.752	0.755	0.245
Contact lenses	83.33	0.851	0.833	0.166
Diabetes	73.82	0.735	0.738	0.261
Glass	66.82	0.67	0.668	0.331
Iris plants	96	0.96	0.96	0.04
Soybean	91.50	0.842	0.849	0.065
Vote	96.32	0.963	0.963	0.036
Image seg.	95.73	0.819	0.957	0.042
Tic-Tac-Toe	85.07	0.849	0.851	0.149
NSL-KDD	71.11	0.711	0.711	0.288

Table 9
The classification accuracies and precision, sensitivity and specificity values for the proposed hybrid DT classifier with 10-fold cross validation.

Datasets	Classification accuracy (%)	Precision (weighted avg.)	Sensitivity (weighted avg.)	Specificity (weighted avg.)
Breast cancer	81.46	0.834	0.814	0.185
Contact lenses	91.66	0.931	0.916	0.083
Diabetes	79.03	0.788	0.79	0.209
Glass	76.27	0.767	0.762	0.237
Iris plants	98.66	0.986	0.986	0.013
Soybean	92.97	0.866	0.871	0.057
Vote	97.70	0.977	0.977	0.022
Image seg.	96.53	0.826	0.965	0.034
Tic-Tac-Toe	88.1	0.88	0.881	0.118
NSL-KDD	81.92	0.826	0.819	0.18

Table 10
The classification accuracies and precision, sensitivity and specificity values for a NB classifier with 10-fold cross validation.

Datasets	Classification accuracy (%)	Precision (weighted avg.)	Sensitivity (weighted avg.)	Specificity (weighted avg.)
Breast cancer	71.67	0.703	0.716	0.283
Contact lenses	70.83	0.691	0.708	0.291
Diabetes	76.30	0.758	0.763	0.236
Glass	48.59	0.496	0.485	0.514
Iris plants	96	0.96	0.96	0.04
Soybean	92.83	0.87	0.872	0.055
Vote	90.11	0.905	0.901	0.098
Image seg.	81.06	0.708	0.81	0.189
Tic-Tac-Toe	69.62	0.682	0.696	0.3
NSL-KDD	76.27	0.764	0.762	0.237

Moreover, the results shown in Tables 8 and 9 indicate that the proposed DT algorithm outperforms the traditional C4.5 DT classifier in all the test cases. In comparison with the traditional DT, the proposed DT classifier has respectively improved the classification accuracy rates of the NSL-KDD dataset by 10.81%, the glass dataset by 9.45%, the contact lenses dataset by 8.33%, and the breast cancer dataset by 5.94%. Overall, it has improved the classification accuracy rates for the classification of the above 10 datasets by 4.8% on average comparing to the traditional DT classifier.

Furthermore, we have also evaluated the traditional NB classifier using all the 10 datasets and achieved an average accuracy rate of 77.3% using 10-fold cross validation, while the proposed NB classifier (Algorithm 2) obtained an average accuracy rate of 86.7%.

Table 11
The classification accuracies and precision, sensitivity and specificity values for the proposed hybrid NB classifier with 10-fold cross validation.

Datasets	Classification accuracy (%)	Precision (weighted avg.)	Sensitivity (weighted avg.)	Specificity (weighted avg.)
Breast cancer	75.87	0.75	0.758	0.241
Contact lenses	87.50	0.909	0.875	0.125
Diabetes	85.41	0.853	0.854	0.145
Glass	52.33	0.533	0.523	0.476
Iris plants	98	0.98	0.98	0.02
Soybean	94.15	0.891	0.893	0.047
Vote	94.48	0.945	0.944	0.055
Image seg.	85.19	0.742	0.852	0.148
Tic-Tac-Toe	78.91	0.786	0.789	0.21
NSL-KDD	82.39	0.836	0.823	0.176

Tables 10 and 11 respectively tabulate the performances of the traditional NB classifier and the proposed NB algorithm on 10 datasets using 10-fold cross validation.

The results shown in Tables 10 and 11 indicate that the proposed NB algorithm (Algorithm 2) also outperforms the basic NB algorithm for all the test cases. Comparing to the traditional NB classifier, the proposed NB algorithm has respectively improved the classification accuracy rates of the contact lenses dataset by 16.67%, the diabetes dataset by 9.11%, the tic-tac-toe dataset by 9.29%, and the NSL-KDD dataset by 6.12%. Algorithm 2 has improved the classification accuracy rates for all the 10 datasets by 9.4% on average in comparison to the classic NB classifier.

Figs. 2 and 3 respectively show the comparison of classification accuracy rates between the C4.5 DT and the proposed DT classifiers, and between a basic NB and the proposed NB classifiers for each dataset with 10-fold cross validation. Fig. 4 also shows the comparison of classification accuracy rates of all classifiers on 10 datasets with 10-fold cross validation.

Overall, Algorithm 1 is able to automatically remove noisy instances from training datasets for DT generation to avoid overfitting. It thus possesses more robustness and generalization capabilities. Algorithm 2 is capable of identifying the most discriminative subset of attributes for classification. The evaluation results prove the efficiency of the proposed DT and NB algorithms (Algorithm 1 and 2) for the classification of challenging real benchmark datasets. They respectively outperform the traditional C4.5 DT and NB classifiers in all the test cases (see Figs. 2 and 3).

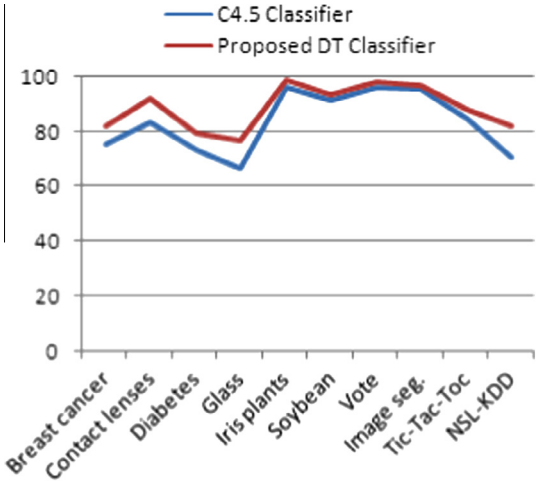

Fig. 2. The comparison of classification accuracy rates between the C4.5 DT and the proposed DT classifiers on 10 datasets with 10-fold cross validation.

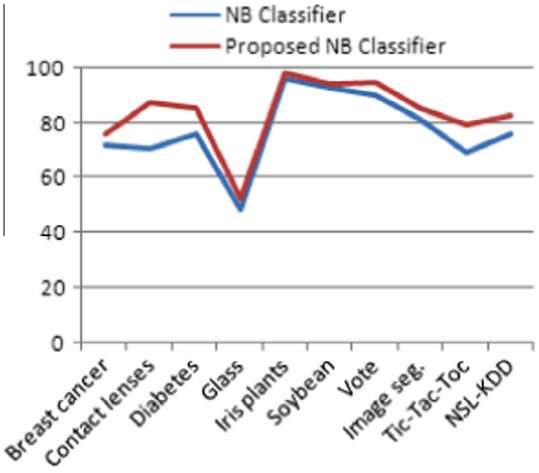

Fig. 3. The comparison of classification accuracy rates between the basic NB and the proposed hybrid NB classifiers on 10 datasets with 10-fold cross validation.

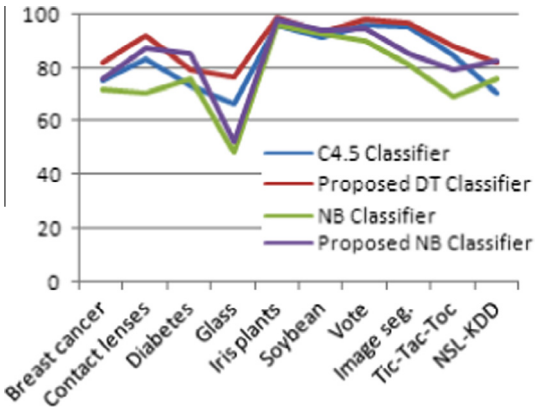

Fig. 4. The comparison of classification accuracy rates of all classifiers on each dataset with 10-fold cross validation.

6. Conclusions

In this paper, we have proposed two independent hybrid algorithms for DT and NB classifiers. The proposed methods improved the classification accuracy rates of both DT and NB classifiers in multi-class classification tasks. The first proposed hybrid DT algorithm used a NB classifier to remove the noisy troublesome instances from the training set before the DT induction, while the second proposed hybrid NB classifier used a DT induction to select a subset of attributes for the production of naïve assumption of class conditional independence. The performances of the proposed algorithms were tested against those of the traditional DT and NB classifiers using the classification accuracy, precision, sensitivity-specificity analysis, and 10-fold cross validation on 10 real benchmark datasets from UCI machine learning repository. The experimental results showed that the proposed methods have produced impressive results for the classification of real life challenging multi-class problems. In future work, other classification algorithms, such as naïve Bayes tree (NBTree), genetic algorithms, rough set approaches and fuzzy logic, will be used to deal with real-time multi-class classification tasks under dynamic feature sets.

Acknowledgment

We appreciate the support for this research received from the European Union (EU) sponsored (Erasmus Mundus) cLINK (Centre of Excellence for Learning, Innovation, Networking and Knowledge) project (Grant No. 2645).

References

Aitkenhead, M. J. (2008). A co-evolving decision tree classification method. *Expert Systems with Applications, 34*, 18–25.
Aviad, B., & Roy, G. (2011). Classsification by clustering decision tree-like classifier based on adjusted clusters. *Expert Systems with Applications, 38*, 8220–8228.
Balamurugan, S. A. A., & Rajaram, R. (2009). Effective solution for unhandled exception in decision tree induction algorithms. *Expert Systems with Applications, 36*, 12113–12119.
Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. Chapman and Hall/CRC.
Bujlow, T., Riaz, M.T., & Pedersen, J.M. (2012). A method for classification of network traffic based on C5.0 machine learning algorithm. In *Proc. of the International Conference on Computing, Networking and Communications (ICNC)* (pp. 237–241).
Chandra, B., & Gupta, M. (2011). Robust approach for estimating probabilities in naïve Bayesian classifier for gene expression data. *Expert Systems with Applications, 38*, 1293–1298.
Chandra, B., & Paul Varghese, P. (2009). Moving towards efficient decision tree construction. *Information Sciences, 179*, 1059–1069.
Chandra, B., & Varghese, P. P. (2009). Fuzzifying gini index based decision trees. *Expert Systems with Applications, 36*, 8549–8559.
Chen, J., Huang, H., Tian, S., & Qu, Y. (2009). Feature selection for text classification with naïve Bayes. *Expert Systems with Applications, 36*, 5432–5435.
Chen, Y.-L., & Hung, L. T.-H. (2009). Using decision trees to summarize associative classification rules. *Expert Systems with Applications, 36*, 2338–2351.
Fan, L., Poh, K.-L., & Zhou, P. (2010). Partition-conditional ICA for Bayesian classification of microarray data. *Expert Systems with Applications, 37*, 8188–8192.
Farid, D. M., Harbi, N., & Rahman, M. Z. (2010). Combining naïve Bayes and decision tree for adaptive intrusion detection. *International Journal of Network Security & Its Applications, 2*, 12–15.
Farid, D. M., & Rahman, M. Z. (2010). Anomaly network intrusion detection based on improved self adaptive Bayesian algorithm. *Journal of Computers, 5*, 23–31.
Farid, D. M., Rahman, M. Z., & Rahman, C. M. (2011). Adaptive intrusion detection based on boosting and naïve Bayesian classifier. *International Journal of Computer Applications, 24*, 12–19.
Farid, D. M., Zhang, L., Hossain, A., Rahman, C. M., Strachan, R., Sexton, G., et al. (2013). An adaptive ensemble classifier for mining concept drifting data streams. *Expert Systems with Applications, 40*, 5895–5906.
Franco-Arcega, A., Carrasco-Ochoa, J. A., Sanchez-Diaz, G., & Martinez-Trinidad, J. F. (2011). Decision tree induction using a fast splitting attribute selection for large datasets. *Expert Systems with Applications, 38*, 14290–14300.
Frank, A., & Asuncion, A. (2010). UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 29.07.2013.
Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I.H. (2009). The weka data mining software: An update. SIGKDD Explorations, 11. http://www.cs.waikato.ac.nz/ml/weka/. Accessed 29.07.2013.
Hsu, C.-C., Huang, Y.-P., & Chang, K.-W. (2008). Extended naïve Bayes classifier for mixed data. *Expert Systems with Applications, 35*, 1080–1083.
Jamain, A., & Hand, D. J. (2008). Mining supervised classification performance studies: A meta-analytic investigation. *Journal of Classification, 25*, 87–112.
Koc, L., Mazzuchi, T. A., & Sarkani, S. (2012). A network intrusion detection system based on a hidden naïve Bayes multiclass classifier. *Expert Systems with Applications, 39*, 13492–13500.
Lee, L. H., & Isa, D. (2010). Automatically computed document dependent weighting factor facility for na"ıve Bayes classification. *Expert Systems with Applications, 37*, 8471–8478.
Liao, S.-H., Chu, P.-H., & Hsiao, P.-Y. (2012). Data mining techniques and applications – a decade review from 2000 to 2011. *Expert Systems with Applications, 39*, 11303–11311.
Loh, W. Y., & Shih, X. (1997). Split selection methods for classification tree. *Statistica Sininca, 7*, 815–840.
McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transaction on Information and System Security, 3*, 262–294.
Ngai, E. W. T., Xiu, L., & Chau, D. C. K. (2009). Application of data mining techniques in customer relationship management: A literature review and classification review article. *Expert Systems with Applications, 36*, 2592–2602.
Polat, K., & Gunes, S. (2009). A novel hybrid intelligent method based on C4.5 decision tree classifier and one-against-all approach for multi-class classification problems. *Expert Systems with Applications, 36*, 1587–1592.
Quinlan, J. R. (1986). Induction of decision tree. *Machine Learning, 1*, 81–106.
Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufman Publishers Inc..
Safavian, S. R., & Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man and Cybernetics, 21*, 660–674.
Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A.A. (2009). A detailed analysis of the KDD CUP 99 data set. In *Proc. of the 2nd IEEE Int. Conf. on Computational Intelligence in Security and Defense Applications* (pp. 53–58).
Turney, D. (1995). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 369–409.

Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine Learning, 4*, 161–186.

Utgoff, P.E. (1988). ID5: An incremental ID3. In *Proc. of the fifth National Conference on Machine Learning, Ann Arbor, Michigan, USA* (pp. 107–120).

Valle, M. A., Varas, S., & Ruz, G. A. (2012). Job performance prediction in a call center using a naïve Bayes classifier. *Expert Systems with Applications, 39*, 9939–9945.