Increasing the Accuracy of Hidden Naive Bayes Model

Sotiris Kotsiantis[1], Vasilis Tampakas[2]
TEI of Patras, 1 M. Alexandrou str., Koukouli, 263 34 Patras, Greece
[1] sotos@math.upatras.gr
[2]tampakas@teipat.gr

***Abstract*- In this study, we attempt to increase the prediction accuracy of the Hidden Naive Bayes model. Because the concept of combining classifiers is proposed as a new direction for the improvement of the performance of individual classifiers, we make use of Adaboost, with the difference that in each iteration of Adaboost, we replace the missing values, we use a discretization method and we remove redundant features using a filter feature selection method. Finally, we perform a large-scale comparison with other attempts that have tried to improve the accuracy of the simple Bayes algorithm as well as other state-of-the-art algorithms and ensembles on 24 standard benchmark datasets and the present method gives better accuracy in most cases using less time for training, too.**

I. INTRODUCTION

Probabilistic classifiers are among the most popular classifiers used in the machine learning community and increasingly in many applications. Probabilistic classifiers operate on data where each example x consists of feature values $<a_1, a_2 ... a_i>$ and the target function y can take on any value from a pre-defined finite set $V=(v_1, v_2 ... v_j)$. Classifying unseen examples involves calculating the most probable target value $\mathbf{v}_{max}$ and is defined as: $v_{\max} = \max_{v_j \in V} P(v_j \mid a_1, a_2, ..., a_i)$.

Using Bayes theorem v_{max} can be rewritten as: $v_{\max} = \max_{v_j \in V} P(a_1, a_2, ..., a_i \mid v_j) P(v_j)$.

Bayes rule is used to estimate the conditional probability of a class label y, and then assumptions are made on the model, to decompose this probability into a product of conditional probabilities. Under the assumption that features values are conditionally independent given the target value, the formula used by the simple Bayes classifier is:

$v_{\max} = \max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$, where V is the target output of the classifier and $P(a_i|v_j)$ and $P(v_i)$ can be calculated based on their frequency in the training data.

The application of previous formula is straightforward for the categorical features. For numerical features, one can model the component marginal distributions in a wide variety of ways. The simplest would be to adopt some parametric form e.g. marginal Gaussian estimators [5]. Another problem with this formula is the zero counts. Zero counts are obtained when a given class and feature value never occur together in the training set, and is problematic because the resulting zero probabilities will wipe out the information in all the other probabilities when they are multiplied. A solution to this problem is to incorporate a small-sample correction into all probabilities, such as the Laplace correction [5]. If a feature value does not occur given some class, its probability is set to 1/N, where N is the number of examples in the training set.

The assumption of independence is clearly almost always wrong. However, a large-scale comparison of simple Bayesian classifier with state-of-the-art algorithms for decision tree induction and instance-based learning on standard benchmark datasets found that simple Bayesian classifier sometimes is superior to each of the other learning schemes even on datasets with substantial feature dependencies [5]. An explanation why simple Bayes method remains competitive, even though it provides very poor estimates of the true underlying probabilities can be found in [9].

Zhang et al [29] proposed hidden naive Bayes (HNB). In an HNB, a hidden parent is created for each attribute which combines the influences from all other attributes. Zhang et al [29] presented an approach to creating hidden parents using the average of weighted one-dependence estimators. HNB inherits the structural simplicity of naive Bayes and can be easily learned without structure learning.

In this study, we increase the training time of the HNB algorithm so as to take better results. We made use of boosting, with the difference that in each iteration of boosting, we replaced the missing values, we used a discretization method and we removed redundant features using a filter feature selection method. We performed a large-scale comparison with other attempts that have tried to improve the accuracy of the simple Bayes algorithm as well as other state-of-the-art algorithms and ensembles on 24 standard benchmark datasets and we actually took better accuracy in most cases using less time for training, too.

Description of some of the attempts that have been tried to improve the performance of simple Bayesian classifier is given in section 2. Section 3 discusses the presented algorithm for improving the performance of Bayesian classifier. Experiment results in a number of data sets are presented in section 4, while brief summary with further research topics are given in Section 5.

II. PREVIOUS ATTEMPTS FOR IMPROVING THE PERFORMANCE OF SIMPLE BAYES CLASSIFIER

The most well known attempt for improving the performance of the simple Bayes algorithm is the discretization of the continuous features into intervals, instead of using the default

option to utilize the normal distribution to calculate probabilities [28]. Numerous discretization methods have been examined such as the partition of the range of the features into k equal sized intervals and the partition of the range into intervals containing the same number of instances. A brief survey of these attempts and an empirical study comparing the performance of the most well-known discretization methods is given in [1], concluding that all the tested discretization methods can outperform the version of the simple Bayesian classifier that assumes normal distributions for continuous features. The authors state that MDL (Minimum Description Length) discretization seems to be the best performer compared to that of EWD (Equal Width) and EFD (Equal Frequency) discretizations.

The performance of the simple Bayesian Classifier on domains with redundant features can be also improved by removing redundant features [15]. The 'selective Bayesian classifier', has been explored by numerous researchers such as [2], [15], [16]. A well-known filter that computes the empirical mutual information between features and the class, and discards low-valued features was used by [23]. Other researchers have explored the possibility of using a decision tree algorithm as a pre-processor to discover useful feature subsets for simple Bayes classifier. A simple method that uses C4.5 decision trees [18] to select features has been described by [20].

On the other hand, a wrapping forward search strategy to select features for use with simple Bayes was used in [15]. In [16], the author uses constructive induction and feature deletion (BSEJ algorithm) to alleviate the feature inter-dependence prob-lem of the simple Bayesian classifier. BSEJ uses the wrapper model to join and delete features. A greedy search, at each step, either deletes one feature or creates a new one through joining (generating the Cartesian product of) two features. BSEJ starts from the set of all the original features, and stops when neither joining nor deleting can improve upon the accuracy of the simple Bayesian classifier.

More elaborate attempts to overcome the assumption of independence between features are based on including extra terms describing the relationship between the features [10]. Of course, this means that the simple form of the simple Bayes model is sacrificed. In fact, these particular authors approached things from the opposite direction, identifying weaknesses of the approach based on constructing an overall model of the joint distribution, and restricting themselves to models which included all the two way marginals of which had the class label as one component.

An iterative approach of simple Bayes is presented in [11]. The iterative Bayes begins with the distribution tables built by the simple Bayes and then the algo-rithm iteratively cycles through all the training examples using a hill-climbing technique. Experimental evaluation of iterative Bayes showed minor but consistent gain in accuracy in relation to simple Bayes [11]. However, the contingency tables are incre-mentally updated each time a training example is seen, which implies that the order of the examples could influence the final decision.

Another attempt for improving the simple Bayes model was the Bayesian trees [31]. A Bayesian tree-learning algorithm builds a decision tree, and generates a local simple Bayesian classifier at each leaf. The tests, leading to a leaf, can alleviate fea-ture inter-dependencies for the local simple Bayesian classifier. In [31], the authors also proposed the application of lazy learning techniques to Bayesian tree induction and presented the resulting lazy Bayesian rule-learning algorithm, called LBR. For each test example, it builds a most appropriate rule with a local simple Bayesian classifier as its consequent.

Naive Bayes with attributes weighted differently results in a model called weighted naïve Bayes (WNB) [30]. Another study which comes under this group it the Attribute Weighted Bayesian classifier (AWNB), presented by [12], which used decision trees to estimate the weights of attributes based on the minimum depth at which each attribute is tested in the tree.

AODE achieves highly accurate classification by averaging over all of a small space of alternative naive-Bayes-like models that have weaker (and hence less detrimental) independence assumptions than naive Bayes [26]. LE-AODE (Lazy Elimination for AODE) [32] is the locally application of AODE.

Zhang et al [29] proposed hidden naive Bayes (HNB). In an HNB, a hidden parent is created for each attribute which combines the influences from all other attributes. Zhang et al [29] presented an approach to creating hidden parents using the average of weighted one-dependence estimators. HNB inherits the structural simplicity of naive Bayes and can be easily learned without structure learning.

Lately in the area of ML the concept of combining classifiers is proposed as a new direction for the improvement of the performance of individual classifiers [13]. In [3], the researchers built an ensemble of simple Bayes classifiers using bagging [4] and boosting procedures [8]. Another way that has been examined for generation of ensemble of simple Bayesian classifiers is by using different feature subsets randomly and taking a vote of the predictions of each classifier that uses different feature subset [25].

III. PRESENTED ALGORITHM

As we have already mentioned, comparing the performance of the most well known discretization methods, in [1], the authors concluded that the entropy-based method [6] performed slightly better. For this reason, we use the entropy-based method as discretization method in our algorithm. Full description of the discretization method is beyond the scope of this paper since it can be found in [6].

Feature wrappers selection techniques often achieve better results than filters due to the fact that they are tuned to the specific interaction between the induction algorithm and its training data. However, they tend to be much slower than feature filters because they must repeatedly call the induction algorithm. In the authors' opinion, the advantage of filter approach outweighs its disadvantage. In general, filters execute many times faster than wrappers, and therefore stand a

much better chance of scaling to databases with a large number of features than wrappers do. Thus, we have selected to use the filter approach embedded to our algorithm. The well-known filter that computes the empirical mutual information between features and the class [23], and discards low-valued features was modified for our algorithm. The mutual information gain criterion has a strong bias in favor of features with many different values, thus we rectify this bias by a kind of normalization – gain ratio that sometimes is used from decision tree algorithms [18] was used by the proposed algorithm. Features are selected by keeping those for which gain ratio exceeds a fixed threshold e. In order to have a robust selection, we set e to 0.02 of the gain ratio filter, in an attempt to discard only features with negligible impact on predictions. However, such a low threshold can discard many features.

The ensembles of Bayesian classifiers have traditionally not been a focus of research. The reason is that Bayes models are extremely stable learning algorithms, and bagging is mainly variance reduction technique, thus not being able to benefit from its integration. However, Bayes models can be effectively used in ensemble techniques, which perform also bias reduction, such as boosting [3]. The authors of [3] also report that there is a problem with boosting which is the robustness to noise. This is expected because noisy examples tend to be misclassified, and the weight will be increased for these examples. For this reason, we have chosen to use Adaboost with only 10 classifiers. Thus, the proposed algorithm (BoostFSHNB) is summarized in (Figure 1). It makes use of Adaboost, with the difference that in each iteration of Adaboost, we replaced the missing values for nominal and numeric attributes with the modes and means from the training data, we used a discretization method and we removed redundant features using a filter feature selection method.

Because the proposed algorithm is a combination of models, the produced classifier is less easily interpretable. However, BoostFSHNB as a weighted sum of Bayes models, remains much more easily interpretable than Neural Networks and Support Vector Classifiers [21]. In [24], the authors showed that boosting improves the accuracy of the simple Bayesian classifier in 9 out of the tested 14 data sets. However, they concluded that the mean relative error reduction of boosting over the simple Bayesian classifier in the 14 data sets was only 1%, indicating very marginal improvement due to boosting. On the contrary, the proposed algorithm has mean relative error reduction about 30% over the simple Bayesian classifier in the tested data sets. The main reason is that the embedding feature selection technique makes Bayes models slightly unstable and as a result more suitable for Adaboost.

In the following section, we present the experiments. It must be mentioned that the comparisons of the proposed algorithm are separated in three phases: a) with the other attempts that have tried to improve the accuracy of the simple Bayes algorithm, b) with other state-of-the-art algorithms and c) with other well-known ensembles.

```
Assign weight=1  to each training example
for i=1 to 10
   {  for i=1 to number of features
         {
         Replaces all missing values for nominal and numeric attributes in a
dataset with the modes and means from the training data.
         S= Sorted values of i feature;  Splitting(S)
         if Entropy-MDLP-Criterion=SATISFIED then break
         else {
           T = GetBestSpitPoint(S); S1 = GetLeftPart(S,T);  S2 =
GetRightPart(S,T); Splitting(S1)
           Splitting(S2)   }
        }
       for i=1 to number of features
       { G= Gain Ratio of i feature
        if (G<0.02) then remove i feature from data set
       }
Apply HNB  to weighted data set and store the resulting model
Compute error e of model and store error;  if (e=0) or (e>0.5) then break
      for k=1 to number of training examples
           {
           if (Classification(k)=correct) then multiply weight of example by
e/(1-e)
          Normalize weight of all examples
          }
Output the final classifier:
For each model add –log(e/(1-e)) to weight of class predicted and return
class with highest sum
```

Fig. 1. The proposed algorithm

IV. COMPARISONS AND RESULTS

For the purpose of our study, we used 24 well-known datasets from many domains from the UCI repository [7]. These data sets were hand selected so as to come from real-world problems and to vary in characteristics. In order to calculate the classifiers' accuracy, the whole training set was divided into ten mutually exclusive and equal-sized subsets and for each subset the classifier was trained on the union of all of the other subsets. Then, cross validation was run 10 times for each algorithm and the median value of the 10-cross validations was calculated. In Table I, one can see the comparisons of the proposed algorithm with the other attempts that have tried to improve the classification accuracy of the simple Bayes algorithm. Four well-known algorithms were used for the comparison: selective simple Bayes with forward selection [15], Tree Augmented Naïve Bayes [10], HNB [29] and lazy Bayesian rule-learning algorithm [31]. It must be mentioned that we also present the accuracy of the simple Bayes algorithm as borderline. In addition, a representative algorithm for each of the other sophisticated machine learning techniques was tested. The most well known learning algorithm to estimate the values of the weights of a neural network - the Back Propagation (BP) algorithm [27] - was the representative of the ANNs. SMO algorithm was the representative of the Support Vector Machines [17]. We also used the 3-NN algorithm as a representative of kNN[27].

In Table I, we represent with "*v*" that the proposed BoostFSHNB algorithm *looses* from the specific algorithm. That is, the specific algorithm performed statistically better than BoostFSHNB according to t-test with $p<0.05$. Furthermore, in Table I, "*" indicates that BoostFSHNB performed statistically better than the specific classifier

according to t-test with $p<0.05$. In all the other cases, there is no significant statistical difference between the results (*Draws*)[22]. In the last rows of the Table I one can see the aggregated results in the form (*a/b/c*). In this notation "*a*" means that the proposed algorithm is significantly more accurate than the compared algorithm in *a* out of 24 data sets, "*c*" means that the proposed algorithm is significantly less accurate than the compared algorithm in *c* out of 24 data sets, while in the remaining cases (*b*), there is no significant statistical difference between the results.

Thus, the proposed algorithm is significantly more accurate than simple Bayes (NB) in 11 out of the 24 data sets, while it has significantly higher error rates than simple Bayes in three data sets. The average error rate of our algorithm is also about 32% less than that of simple Bayes algorithm. In addition, the proposed algorithm has significantly lower error rates in 6 out of the 24 data sets than the simple HNB model, while it is significantly less accurate in none data set. Furthermore, the proposed algorithm is significantly more accurate than the wrapping feature selection version of simple Bayes (WrapperNB) in 11 out of the 24 data sets using less time for training, too. In only 2 data sets, the proposed algorithm has significantly higher error rate.

TABLE I. COMPARING THE PROPOSED ALGORITHM WITH OTHER ATTEMPTS TO IMPROVE THE SIMPLE BAYES

Dataset	Boost FSHNB	NB	HNB	Wrapper NB	TAN	LBR	BP	SMO	3NN
anneal	99.33	86.59*	98.20	89.57*	98.11	98.01	94.97*	96.89*	97.29*
autos	86.29	57.41*	82.00*	63.61*	74.78*	73.80*	48.84*	56.55*	67.23*
balance	84.97	90.53v	69.67*	90.53v	70.80*	72.17*	85.67	87.62v	86.74
breast-cancer	68.58	72.7v	70.23	71.64v	69.28	72.35v	72.95v	69.92	73.13v
breast-w	96.57	96.07	96.37	95.5	97	97.21	96.35	96.81	96.61
colic	83.96	78.70*	81.69	83.91	82.8	82.33	83.07	82.69	80.95*
credit-a	84.49	77.86*	84.84	85.3	86.07	86.1	85.94	84.91	84.96
diabetes	73.96	75.75	74.57	76.06	75.36	75.38	77.04v	77.07v	73.86
heart-c	81.49	83.34	81.95	78.75*	82.97	83.54	82.98	84.03	81.82
heart-h	84.39	83.95	84.87	82.63	84.84	84.54	84.16	83.26	82.33
heart-statlog	82.22	83.59	82.74	82.78	82.63	82.59	83.3	83.81	79.11*
hepatitis	85.75	83.81	84.64	82.55*	84.83	84.91	84.29	85.03	80.85*
Hypo-thyroid	97.93	95.3	99.07	95.19	99.16	99.12	93.44*	93.49*	93.21*
ionosphere	91.74	82.17*	91.48	90.14	91.03	90	87.07*	87.93*	86.02*
iris	95.33	95.53	92.07*	94.27	92.93*	93.2	84.80*	84.87*	95.2
kr-vs-kp	95.37	87.79*	92.35*	94.34	92.28*	96.79	98.92v	95.78	96.56
Lympho-theraapy	84.43	83.13	85.57	81.22*	86.3	85.45	82.26	85.94	81.74
mushroom	100	95.76*	99.96*	99.62*	99.96*	99.96*	99.97*	100	100
prim. -tumor	46.60	49.71v	47.85	43.45*	47.76	48.85	24.90*	29.20*	44.98
segment	96.03	80.16*	96.47	89.17*	95.16	93.94	91.35*	89.72*	96.12
sonar	81.21	67.71*	76.13*	71.25*	76.41*	76.04*	78.67	77.88*	83.76
vote	95.17	90.02*	94.32	95.63	94.62	94.11	96.32	96.23	93.08
waveform	84.51	80.01*	84.87	81.94*	80.1*	83.42	86.56	86.94	77.67*
zoo	96.09	94.97	97.11	92.67*	95.45	93.21*	60.43*	93.16*	92.61*
Average error	*13.48*	*17.81*	*14.62*	*16.18*	*14.97*	*14.71*	*18.16*	*16.26*	*15.59*
W-D-L		*11/10/3*	*6/18/0*	*11/11/2*	*6/18/0*	*5/18/1*	*10/13/3*	*9/13/2*	*9/14/1*

Moreover, the proposed algorithm is significantly more accurate than LBR algorithm in 5 out of the 24 data sets while the proposed algorithm has significantly higher error rates in 1 data set. Finally, the proposed algorithm is significantly more accurate than Tree Augmented Naïve Bayes (TAN) algorithm in 6 out of the 26 data sets, while it is significantly less accurate in none data set. The proposed algorithm is also significantly more precise than BP algorithm with one hidden layer in 10 out of the 24 data sets, whilst it has significantly higher error rates in 3 data sets. In addition, the proposed algorithm is significantly more accurate than SMO algorithm in 9 out of the 24 data sets, whereas it has significantly higher error rates in 2 data sets. The proposed algorithm is significantly more precise than 3NN algorithm in 9 out of the 24 data sets, while it has significantly higher error rates in one data set. In brief, we managed to improve the performance of the simple Bayes Classifier obtaining better accuracy than other well known methods that have tried to improve the performance of the simple Bayes algorithm. The proposed algorithm also gave better accuracy than other sophisticated state-of-the-art algorithms on most of the 24 standard benchmark datasets.

Subsequently, we compare the performance of BoostFSHNB with bagging decision trees and boosting decision trees that have been proved to be very successful for many machine-learning problems [19]. Similarly with the proposed algorithm, Quinlan (1996) used 10 classifiers for bagging and boosting C4.5 algorithm. In Table II, we also present the accuracy of the simple Bayes and HNB algorithms after applying the bagging and boosting procedures. In the last rows of the Table II one can see the aggregated results.

The proposed algorithm is significantly more accurate than boosting C4.5 algorithm with 10 classifiers in 8 out of the 24 data sets. In only one data set, the proposed algorithm has significantly higher error rates. In addition, the proposed algorithm is significantly more accurate than bagging C4.5 algorithm with 10 classifiers in 7 out of the 24 data sets, while on other 2 data sets, the proposed algorithm has significantly higher error rates. In brief, we took better accuracy than boosting and bagging decision trees on the most of the 24 datasets, using less time for training, too. In addition, the proposed algorithm is significantly more accurate than single boosting simple Bayes (AdaBoostNB) algorithm (using 25 classifiers) in 7 out of the 24 data sets, while in 2 data sets, the proposed algorithm has significantly higher error rates. Moreover, the proposed algorithm is significantly more accurate than single bagging simple Bayes algorithm (using 25 ensembles) in 11 out of the 24 data sets, whereas it has significantly higher error rates in only 3 data sets. Moreover, the proposed algorithm is significantly more accurate than boosting HNB (AdaBoostHNB) algorithm (using 10 classifiers) in 5 out of the 24 data sets, while in none data set, the proposed algorithm has significantly higher error rates. Moreover, the proposed algorithm is significantly more accurate than single bagging HNB algorithm (using 10 ensembles) in 4 out of the 24 data sets, whereas it has significantly higher error rates in none data set.

TABLE II. COMPARING THE PROPOSED ALGORITHM WITH WELL KNOWN ENSEMBLES

Dataset	Boost FSHNB	Adaboost C4.5	Bagging C4.5	Bagging NB	Adaboost NB	Bagging HNB	Adaboost HNB
anneal	99.33	99.59	98.82	87.27*	95.20*	98.59	99.48
autos	86.29	85.46	81.48*	57.71*	57.12*	82.05*	83.46 *
balance	84.97	78.35*	82.68	90.29v	92.11v	82.08	82.13
breast-cancer	68.58	66.89	73.10v	72.73v	68.57	70.34	67.49
breast-w	96.57	96.08	96.18	96.07	95.55	96.57	96.07
colic	83.96	81.63	84.99	78.94*	77.46*	81.98	78.78 *
credit-a	84.49	84.01	86.38	77.81*	81.16*	85.33	82.42
diabetes	73.96	71.69	75.23	75.57	75.88	75.30	75.04
heart-c	81.49	78.79*	79.86	83.24	83.14	82.58	79.46
heart-h	84.39	78.68*	80.24*	84.16	84.67	84.43	83.00
heart-statlog	82.22	78.59*	80.93	83.41	82.3	83.04	81.96
hepatitis	85.75	82.38*	82.05*	84.39	84.23	85.75	82.85 *
Hypo-thyroid	97.93	99.65	99.57	95.53	95.27	99.05	98.08
ionosphere	91.74	93.05	91.63	81.94*	91.12	91.68	92.74
iris	95.33	94.33	94.73	95.53	95.07	94.73	94.13
kr-vs-kp	95.37	99.59v	99.43v	87.80*	95.1	92.45*	97.88
Lympho-theraapy	84.43	80.87*	78.44*	83.5	80.67*	85.37	84.68
mushroom	100	100	100	95.73*	100	99.82*	99.92 *
prim. -tumor	46.60	41.65*	42.81*	49.47v	49.71v	47.87	46.40
segment	96.03	98.12	97.25	80.41*	80.16*	96.01	96.01
sonar	81.21	79.13	77.25*	67.96*	81.21	75.11*	78.10 *
vote	95.17	95.51	96.25	90.09*	95.19	94.34	94.59
waveform	84.51	81.40*	81.63*	80.00*	80.01*	83.11	83.21
zoo	96.09	95.18	93.7	95.07	97.23	96.31	97.42
Average error	13.48	14.97	14.39	17.72	15.91	14.00	14.36
W-D-L		*8/15/1*	*7/15/2*	*11/10/3*	*7/15/2*	*4/20/0*	*5/19/0*

V. CONCLUSIONS

Ideally, we would like to be able to identify or design the single best learning algorithm to be used in all situations. However, both experimental results and theoretical work indicate that this is not possible. The Bayes classifiers have much broader applicability than previously thought.

To sum up, we increased the prediction accuracy of the Hidden Naive Bayes model. We made use of Adaboost, with the difference that in each iteration of Adaboost, we replaced the missing values, we used a discretization method and we removed redundant features using a filter feature selection method. We performed a large-scale comparison with other attempts that have tried to improve the accuracy of the simple Bayes algorithm as well as other state-of-the-art algorithms and ensembles on 24 standard benchmark datasets and we took better accuracy in most cases using less time for training, too. Using less time for training the proposed algorithm stands a much better chance of scaling to data mining applications.

In future research it would be interesting to examine the performance of the presented algorithm in extremely imbalanced data sets. We believe that the embedded boosting process of the presented algorithm enables it to operate comparatively well in such difficult data sets, too.

REFERENCES

[1] R. Abraham, J.B. Simha and S.S. Iyengar, "A Comparative Analysis of Discretization Methods for Medical Data Mining with Naïve Bayesian Classifier," in 9th International Conference on Information Technology (ICIT'06), 2006: pp. 235-236.

[2] R. Abraham, J.B. Simha, and S.S. Iyengar, "Medical Data mining with a New Algorithm for Feature Selection and Naïve Bayesian Classifier," in ICIT. 2007: IEEE Computer Society, pp.44-49.

[3] E. Bauer, and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, Machine Learning, 36 (1999): 105–139.

[4] L. Breiman, Bagging Predictors. Machine Learning, 24 (1996): 123-140.

[5] P. Domingos and M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss, Machine Learning, 29(1997): 103-130.

[6] U. Fayyad and K. Irani, Multi-interval discretization of continuous-valued attributes for classification learning. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann (1993): 1022-1027.

[7] A. Frank, & A. Asuncion (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[8] Y. Freund and R. E. Schapire, Experiments with a New Boosting Algorithm, In Proceedings of ICML'96, 148-156.

[9] J. H. Friedman, On bias, variance, 0/1-loss and curse-of-dimensionality. Data Mining and Knowledge Discovery, 1(1997): 55-77.

[10] N. Friedman, D. Geiger and M. Goldszmidt, Bayesian network classifiers. Machine Learning, 29(1997): 131-163.

[11] J. Gama, Iterative Bayes. Intelligent Data Analysis, 6(2000): 463 – 473.

[12] M. Hall, "A Decision Tree-Based Attribute Weighting Filtering for Naïve Bayes," Knowledge-Based Systems, 2007. 20(2): pp. 120-126.

[13] S. Kotsiantis, P. Pintelas: Logitboost of Simple Bayesian Classifier. Informatica, 29(1): 53-60 (2005).

[14] S.Kotsiantis, P. Pintelas, Increasing the Classification Accuracy of Simple Bayesian Classifier, Lecture Notes in Artificial Intelligence, AIMSA 2004, Springer-Verlag Vol 3192, pp. 198-207, 2004.

[15] P. Langley and S. Sage, Induction of selective Bayesian classifiers. In Proc. of the 10th Conference on Uncertainty in Artificial Intelligence, Seattle, (1994): 399-406.

[16] M. Pazzani, Searching for dependencies in Bayesian classifiers. Artificial Intelligence and Statistics IV, Lecture Notes in Statistics, Springer-Verlag: New York, 1997.

[17] J. Platt, Using sparseness and analytic QP to speed training of support vector machines. In M. S. Kearns, S. A. Solla, and D. A. Cohn (Eds.), Advances in neural information processing systems 11, MA: MIT Press, 1999.

[18] J.Quinlan, C4.5: Programs for machine learning. Morgan Kaufmann, San Francisco, 1993.

[19] J. R. Quinlan, Bagging, boosting, and C4.5. In Proceedings of the Thirteenth National Conference on Artificial Intelligence (1996), 725–730.

[20] C. Ratanamahatana and D. Gunopulos, Feature Selection for the Naive Bayesian Classifier using Decision Trees, Applied Artificial Intelligence, 17 (2003): 475–487.

[21] G. Ridgeway, D. Madigan and T. Richardson,. Interpretable boosted Naive Bayes classification. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, Menlo Park (1998): 101-104.

[22] S. Salzberg, On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach, Data Mining and Knowledge Discovery, 1(1997): 317–328.

[23] M. Singh, and G. Provan, Efficient learning of selective Bayesian network classifiers. In Proc of the 13th International Conference on Machine Learning (1996): 453-461, Bari.

[24] K. Ting and Z. Zheng,. Improving the Performance of Boosting for Naive Bayesian Classification, N. Zhong and L. Zhou (Eds.): PAKDD'99, LNAI 1574, pp. 296-305, 1999.
[25] Tsymbal, S. Puuronen and D. Patterson, Feature Selection for Ensembles of Simple Bayesian Classifiers, In Proceedings of ISMIS (2002): 592-600, Lyon, June 27-29.
[26] G. Webb, J. Boughton, Z. Wang (2005). Not So Naive Bayes: Aggregating One-Dependence Estimators. Machine Learning. 58(1):5-24.
[27] I. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). Morgan Kaufmann, San Mateo, 2005.
[28] Y. Yang and G. I. Webb, "On why discretization works for naive-Bayes classifiers," Lecture Notes in Computer Science, 2003: pp. 440–452.
[29] H. Zhang, L. Jiang, J. Su: Hidden Naive Bayes. In: Twentieth National Conference on Artificial Intelligence, 919-924, 2005.
[30] H. Zhang and S. Sheng, "Learning Weighted Naïve Bayes with Accurate Ranking," in Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04), 2004.
[31] Z. Zheng, and G.I. Webb, Lazy learning of Bayesian rules. Machine Learning, 41 (2000): 53-84
[32] F. Zheng and G. I. Webb, "Efficient Lazy Elimination for Averaged One-Dependence Estimators," in Proceedings of the 23rd International Conference on Machine Learning, 2006: pp. 1113-1120.