Learning k-Nearest Neighbor Naive Bayes For Ranking *

Liangxiao Jiang[1], Harry Zhang[2], and Jiang Su[2]

[1] Faculty of Computer Science, China University of Geosciences
Wuhan, Hubei, P.R.China, 430074
ljiang@cug.edu.cn
[2] Faculty of Computer Science, University of New Brunswick
P.O.Box 4400, Fredericton, NB, Canada E3B 5A3
hzhang@unb.ca

Abstract. Accurate probability-based ranking of instances is crucial in many real-world data mining applications. KNN (k-nearest neighbor) [1] has been intensively studied as an effective classification model in decades. However, its performance in ranking is unknown. In this paper, we conduct a systematic study on the ranking performance of KNN. At first, we compare KNN and KNNDW (KNN with distance weighted) to decision trees and naive Bayes in ranking, measured by AUC (the area under the Receiver Operating Characteristics curve). Then, we propose to improve the ranking performance of KNN by combining KNN with naive Bayes (simply NB). The idea is that a naive Bayes is learned using the k nearest neighbors of the test instance as the training data and used to classify the test instance. A critical problem in combining KNN with naive Bayes is the lack of training data when k is small. We propose to deal with it using cloning to expand the training data. That is, each of the k nearest neighbors is "cloned" and the clones are added to the training data. We call our new model instance cloning local naive Bayes (simply ICLNB). We conduct extensive empirical comparison for the related algorithms in two groups in terms of AUC, using the 36 UCI datasets recommended by Weka[2]. In the first group, we compare ICLNB with KNN, NB, NBTree[3], C4.4[4]. In the second group, we compare ICLNB with KNN, KNNDW and LWNB[5]. Our experimental results show that ICLNB outperforms all those algorithms significantly. From our study, we have two conclusions. First, KNN-relates algorithms performs well in ranking. Second, our new algorithm ICLNB performs best among the algorithms compared in this paper, and could be used in the applications in which an accurate ranking is desired.

1 Introduction

Classification is one of the most important tasks in data mining. In classification, a classifier is learned from a set of training instances with class labels, and an in-

* This work was supported by Excellent Youth Foundation of China University of Geosciences(No.CUGQNL0505) and Natural Science Foundation of Hubei of China(No.2001ABB006 and No.2003ABA043).

stance x is often represented by a tuple of attibutes $< a_1(x), a_2(x), \ldots, a_n(x) >$, where $a_i(x)$ denotes the value of the ith attribute A_i of x. The performance of a classifier is typically measured by its classification accuracy. Many classifiers can also produce the class probability estimates $\tilde{p}(c|x)$ that is the probability of an instance x in the class c, as a by-product. Thus, a ranking of instances based on the class probabilities would be generated. Indeed, in many data mining applications, such a ranking is useful. For example, in direct marketing, we often need to deploy different promotion strategies to customers with different likelihoods of buying some products, in which a ranking of customers in terms of their likelihoods of buying is desired.

KNN has been widely used for decades as an effective classification model. KNN is based on a distance function that measure the difference or similarity between instances. Given a test instance x, its k closest neighbors $y_1, \cdots, y_k$, are found and a vote are conducted to assign the most common class to x. That is, the class of x, denoted by $c(x)$, is determined by the following equation.

$$c(x) = \arg\max_{c \in C} \sum_{i=1}^{k} \delta(c, c(y_i)), \tag{1}$$

where $c(y_i)$ is the class of y_i, and δ is a function that $\delta(u, v) = 1$ if $u = v$.

KNN also produces an estimate $\tilde{p}(c|x)$ using a simple vote. That is, $\tilde{p}(c|x)$ is the fraction of instances of class c in the k nearest neighbors, shown in the following equation.

$$\tilde{p}(c|x) = \frac{\sum_{i=1}^{k} \delta(c, c(y_i))}{k}. \tag{2}$$

Essentially, KNN can be also viewed as a probability-based classifier, shown in Equation 3. Thus, improving the probability estimates of KNN will also lead to an improvement in its classification performance.

$$c(x) = \arg\max_{c \in C} \tilde{p}(c|x). \tag{3}$$

From Equation 2, intuitively, the probability estimates yielded by KNN should be poor, since they are estimated from only the k nearest instances, instead of the whole training data. Thus, its ranking performance should be poor too. We will see later that, in fact, this intuition is sort of wrong.

When we study the ranking performance of a classifier, how to evaluate it is a question. In most scenarios in data mining, the underlying true ranking of training instances is unknown, and only a set of instances with class labels is given. Fortunately, The area under Receiver Operating Characteristics curve, or simple AUC, could be used for this purpose[6].

In recent years, AUC has attracted considerable attention in machine learning and data mining community. Hand and Till[7] show that, for binary classification, AUC is equivalent to the probability that a randomly chosen instance of class $-$ will have a smaller estimated probability of belonging to class $+$ than a randomly chosen instance of class $+$. They present a simple approach to calculating the AUC of a classifier G below.

$$\hat{A} = \frac{S_0 - n_0(n_0+1)/2}{n_0 n_1}, \tag{4}$$

where n_0 and n_1 are the numbers of negative and positive instances respectively, and $S_0 = \sum r_i$, where r_i is the rank of i_{th} positive instance in the ranking.

From Equation 4, it is clear that AUC is essentially a measure of the quality of a ranking. For example, the AUC of a ranking is 1 (the maximum value of AUC) if there is no positive instance preceding a negative instance.

In recent years, reseachers have started to study the ranking performance of traditional classification models, such as decision trees, naive Bayes and SVM (support vector machine) [4, 8, 9]. To our knowledge, there has been no work on studying the ranking performance of KNN-related algorithms (KNN and its variants). The motivation of this paper is to study systematically the performance of KNN-related algorithms in ranking, measured by AUC.

The rest of the paper is organized as follows. In Section 2, we introduce the related work on learning classifiers with accurate ranking. In Section 3, we empirically compare KNN-related algorithms with decision trees and naive Bayes in AUC, and then propose a novel algorithm for learning k-nearest neighbor naive Bayes for ranking. We also present the experimental results of an extensive empirical comparions on various algorithms in Section 3. In Section 4, we draw a conclusion.

2 Related Work

In recent years, researchers have paid considerable attention to exploring learning algorithms for yielding accurate ranking, since classification is not enough in many applications, such as data mining. A substantial amount of research work has been focussed on decision trees. It has been observed that decision trees produce poor probability estimates. Provost and Domingos[4] point out that the decision tree representation can approximate any probability distribution as accurately as possible, but modern decision tree algorithms are biased against building a tree with accurate probability estimates. They propose using Laplace correction and turning off the reduced-error pruning in C4.5[10] to improve the probability estimates. The resulting algorithm is called C4.4. They compared C4.4 to C4.5 by empirical experiments, and showed that C4.4 is a significant improvement over C4.5 with regard to AUC.

Ling and Yan also propose a method to improve the AUC of a decision tree[8]. They present a novel probability estimation algorithm, in which the class probability of an instance is an average of the probability estimates from all leaves of the tree, instead of only using the leaf into which it falls. In other word, each leaf contributes to the class probability estimate of an instance.

Kohavi[3] presents a model NBTree to combine a decision tree with naive Bayes. In an NBTree, a local naive Bayes is deployed on each leaf of a traditional decision tree, and an instance is classified using the local naive Bayes on the leaf

into which it falls. The experiments showed that NBTree outperforms naive Bayes significantly in terms of classification.

Some other traditional learning algorithms, such as decision trees[4, 8], naive Bayes[11] and SVM[9], have been studied in terms of ranking. To our knowledge, there has been no systematic study on the performance of KNN-related algorithms in producing accurate probability estimates or probability-based rankings. Instead, a substantial amount of work has been done on improving the classificaiton accuracy of KNN. Recently, researchers have observed that an significant improvement can be achieved by combining KNN with naive Bayes[5, 12]. That is, a naive Bayes is deployed in the neighborhood of the test instance, consisting of its k nearest neighbors. Indeed, naive Bayes is a simple but effective classifier, which is based on the assumption that all the attributes are independent given the class (the conditional independence assumption). In addition, it performs well when the size of training data is small[3]. Thus, naive Bayes is a suitable local model within KNN. The idea for combining KNN with naive Bayes is quite straightforward. Whenever a test instance is classified, a local naive Bayes is trained using the k nearest neighbors of the test instance, with which the test instance is classified. The classification of the local naive Bayes is based on the following equation.

$$c(x) = \arg\max_{c \in C} p(c) \prod_{i=1}^{k} p(a_i(x)|c), \tag{5}$$

where x is the test instance. The parameters of the local naive Bayes are the probabilities $p(c)$ and $p(a_i(x)|c)$ in Equation 5 that are estimated from the local training data (the k nearest neighbors of x) based on frequency.

Frank et al.[5] present an model to combine KNN with naive Bayes, called locally weighted naive Bayes(LWNB). In LWNB, each of nearest neighbors is weighted in terms of its distance to the test instance. Then a local naive Bayes is built from the weighted training instances. Their experiments show that LWNB outperforms naive Bayes significantly.

Most of the existing research works on combining KNN with naive Bayes are motivated by improving naive Bayes through relaxing the conditional independence assumption using lazy learning. It is expected that there are no strong dependences within the k nearest neighbors of the test instance, although the attribute dependences might be strong in the whole data.

3 Probability-based Ranking of KNN

3.1 Experiment Methodology

The study in this paper is mostly based on experiments. Thus, we first introduce the setup of our experiments. We run our experiments on the 36 UCI data sets recommended by Weka[2]. All the preprocessing stages of data sets are carried out by the Weka[13]. They mainly include the following three processes:

1. We use the filter of ReplaceMissingValues in Weka to replace the missing values of attributes.
2. We use the filter of Discretize in Weka to discretize numeric attributes.
3. It is well-known that, if the number of values of an attribute is almost equal to the number of instances in the data set, this attribute does not contribute any information to classification. So we use the filter of Remove in Weka to delete these attributes. In these 36 data sets, there only exists three this type of attributes, namely Hospital Number in colic.ORIG, Instance Name in Splice and Animal in zoo.

In our experiments, we use the Laplace estimation to avoid the zero-frequency problem. Assume that there are p instances of the class c, N total instances, and C total classes in the training data. The frequency-based estimation calculates the estimated probability $p(c) = \frac{p}{N}$. The Laplace estimation calculates the estimated probability $p(c) = \frac{p+1}{N+C}$. In the Laplace estimation, $p(a_i(x)|c) = \frac{1+N_{ic}}{N_i+N_c}$, where N_{ic} is the number of instances in class c and with $A_i = a_i(x)$, N_c is the number of instances in class c, and N_i is the number of values for attribute A_i.

All algorithms are implemented within the Weka[13]. Multi-class AUC is calculated by the M-measure[7]. The AUC of a classifier on a data set is obtained by averaging the result from a ten-fold cross validation. Runs with the various algorithms are carried out on the same training sets and evaluated on the same test sets. Finally, we conduct two-tailed t-test with significantly different probability of 0.95 to compare each pair of algorithms. That is, we speak of two results for a data set as being "significantly different" only if the difference is statistically significant at the 0.05 level according to the corrected two-tailed t-test.

3.2 The Ranking Performance of KNN

We have studied the ranking performance of KNN, measured by AUC, by experimentally comparing KNN with naive Bayes and C4.4. Table 1 shows the detailed experimental results at $k = 10$, and a summary of t-test results at $k = 5, 10, 30$ is shown in Table 2. This paper, we only present the detailed experimental results at $k = 10$ for KNN-related algorithms, due to the space limit. But in the summary of t-test, we present comparison results at $k = 5, 10, 30$. From Table 1 and 2, we have a few observations on KNN as follows:

1. KNN performs worse than naive Bayes in ranking, when k is small. The AUC scores of KNN are lower than naive Bayes' in 10 data sets, and higher than naive Bayes' in 5 data sets at $k = 5$; and the cresponding numbers are 6 and 4, respectively, at $k = 10$.
2. KNN outperforms C4.4 in ranking. The AUC scores of KNN are higher than C4.4's at all the k values in Table 4. In addition, KNN outperforms C4.4 in larger margin at larger k values.
3. The ranking performance of KNN improves as k increases.

Generally, the ranking performance of the traditional KNN is poor when k is small. In real applications of KNN, a small k value is preferred, since the

classification performance of KNN typically degrades as the increase of k. In addition, small k conforms closer to the data.

It is a natural extension to KNN that weights the instances in the neighborhood in terms of their distance to the test instance. The resuling model is called k-nearest neighbor with distance weighted (KNNDW). The probability estimate $\tilde{p}(c|x)$ yielded by KNNDW is shown in Equation 6.

$$\tilde{p}(c|x) = \frac{\sum_{i=1}^{k} w_i \delta(c, c(y_i))}{\sum_{j=1}^{k} w_i}, \tag{6}$$

where w_i is the weight of y_i, which is a function of the distance $d(x, y_i)$. In our experiments, $w_i = \frac{1}{d^2(x,y_i)}$.

Our experiments show that KNNDW outperforms KNN in ranking. From Table 4, you can see that the number of data sets on which KNNDW has higher AUC scores is significantly greater than the converse.

3.3 *K*-nearest Neighbor Naive Bayes for Ranking

As we showed in Section 1, the probability estimate of KNN is based on a voting within the neighborhood of the test instance. It is believed that a more sophisticated local model within the neighborhood, instead of voting, would improve probability estimates. It is natural to learn a local naive Bayes for a test instance using only the k nearest neighbors. Although the conditional independence assumption of naive is always violated on the whole training data, it is expected that the dependencies within the neighborhood of the test instance are not strong and thus naive Bayes performs better. However, when the local naive Bayes is learned from only the k nearest neighbors, the training data tends to be insufficient, espeically when k is small. Thus, the parameters of naive Bayes cannot be accurately estimated. Then, the performance of a local naive Bayes would be poor. In NBTree[3], a threshold on the size of the training data on a decision node is set to avoid this problem. Each node should have at least 30 training instances. In LWNB[5], Laplace estimation has been used to smooth probability estimates, and a relatively large k, such as $k = 50$, is chosen.

We propose to an approach to handling the issue of lack of training data by expanding the neighborhood. We "clone" each neighbor in terms of its distance to the test instance and add the clones to the training data. Thus, the parameters in naive Bayes can be estimated more accurately and reliably, and the resulting local naive Bayes performs better.

Our cloning is based on an explicit function, defined in Equation 7, which measures the similarity between two instances with nominal attributes. Let x and y are two instances, their similarity, denoted by $s(x, y)$, is defined as:

$$s(x, y) = \sum_{i=1}^{n} \delta(a_i(x), a_i(y)). \tag{7}$$

Given a test instance x, for each instance y in its neighborhood, $s(x, y)$ clones of y are added to the training data. Then, a local naive Bayes is learned from the

expanded training data with which x is classified. We call our method *instance cloning local naive Bayes*, or simply ICLNB. Its algorithm is depicted below.

Algorithm ICLNB(**T**, k, x)
Input : a set **T** of training instances, an integer k, and a test instance x.
Output : the probability estimate $\tilde{p}(c|x)$

1. Find x's k nearest neighbors $y_1, \cdots, y_k$, from **T**.
2. Local training set $\mathbf{L} = \{y_1, \cdots, y_k\}$
3. For each neighbor y_i of x
   - Compute $s(x, y_i)$ using the similarity function in Equation 7.
   - Add $s(x, y_i)$ clones of y_i to **L**.
4. Create a local naive Bayes **NB** using **L** as the training data.
5. Use **NB** to produce the probability estimate $\tilde{p}(c|x)$.
6. Return the probability estimate $\tilde{p}(c|x)$.

ICLNB is based on instance cloning, different from the instance weighting in LWNB[5]. ICLNB replicates instances in order to improve the parameter estimates of naive Bayes, and thus leads to more accurate probability estimates from the local naive Bayes. On the other hand, the instance weighting of LWNB aims to differentiate the contributions of instances to classification, and is not necessarily helpful to the probability estimates of the local naive Bayes, which will be shown by the experimental results in Section 3.4.

3.4 Experimental Results for ICLNB

We conducted two group of comparisons. In the first group, we compared ICLNB with KNN, NB, NBTree, and C4.4. Table 1 and 2 show the experimental results at $k = 10$ and a summary of t-test results at $k = 5, 10, 30$ respectively. In the second group, we compared ICLNB with the KNN-related algorithms, including KNN, KNNDW, LWNB, and the experimental results at $k = 10$ and a summary of t-test results at $k = 5, 10, 30$ are shown in Table 3 and 4 respectively.

From Table 1 and 2, we can see that ICLNB generally outperforms all the other types of algorithms compared in this paper in AUC. We summarize the highlights as follows:

1. ICLNB outperforms naive Bayes significantly. The $w/t/l$ values between ICLNB and NB respectively is 9/25/5, 9/24/3, and 9/22/5 at $k = 5, 10, 30$.
2. ICLNB outperforms C4.4 significantly. The $w/t/l$ values between ICLNB and C4.4 are 11/23/2, 13/22/1, and 14/21/1 at $k = 5, 10, 30$, respectively. Notice that C4.4 is the state-of-the-arts decision tree learning algorithm designed for yielding accurate rankings.
3. ICLNB performs better than NBTree in AUC. The $w/t/l$ values between ICLNB and NBTree are 4/28/4, 6/27/3, and 5/31/0 at $k = 5, 10, 30$, respectively. As k gets larger, ICLNB performs better than NBTree with larger margin. This fact is quite interesting, since NBTree is similar to ICLNB in the sense that both have naive Bayes as a local model. It indicates that KNN would have a better potential than decision trees in ranking.

Table 1. Experimental results on AUC and standard deviation. ICLNB: instance cloning local naive Bayes; KNN: k-nearest neighbor; NB: naive Bayes; NBTree: naive Bayes tree; C4.4: C4.5 with laplace correction and without tree pruning. The value of **K** in each related algorithm is **10**.

Datasets	ICLNB	KNN	NB	NBTree	C4.4
anneal	95.46±3.77	94.52±3.94	95.9±1.3	96.45±0.28	93.78±2.9
anneal.ORIG	94.77±3.74	92.19±7.17	94.49±3.67	94.71±3.74	92.69±3.15
audiology	71.11±0.7	70.93±0.74	70.96±0.73	71.14±0.71	70.58±0.63
autos	94.13±2.69	89.55±2.95	89.18±4.93	93.93±2.68	90.73±4.52
balance-scale	72.2±2.91	65.86±2.94	84.46±4.1	84.46±4.1	63.06±6.18
breast-cancer	61.03±12.37	62.92±12.49	69.71±15.21	68.95±11.27	59.3±12.03
breast-w	99.41±0.74	98.37±1.59	99.19±0.87	99.21±0.73	97.85±1.86
colic	82.19±4.82	86.74±5.7	83.71±5.5	85.92±6.3	85.02±7.03
colic.ORIG	76.39±6.37	76.95±6.5	80.67±6.98	80.06±8.69	80.56±8.94
credit-a	90.09±3.33	91.59±3.55	92.09±3.43	91.48±3.52	89.42±3.1
credit-g	75.11±5.64	75.97±5.23	79.27±4.74	77.75±5.97	69.62±5
diabetes	79.28±6.08	78.35±5.68	82.31±5.17	82.31±5.17	75.5±5.76
glass	84.43±6.05	82.53±4	80.5±6.65	82.53±8.46	82.36±4.38
heart-c	83.57±1.05	83.8±0.77	84.1±0.54	83.96±0.51	83.1±1.19
heart-h	83.51±0.86	83.6±0.79	83.8±0.7	83.78±0.62	83.04±0.85
heart-statlog	88.27±3.51	90.6±4.82	91.3±4.19	89.66±3.42	81.36±9.15
hepatitis	85.2±14.49	86.97±9.3	88.99±8.99	88.03±8.29	82.03±14.04
hypothyroid	84.99±10.95	82.25±10.83	87.37±8.52	87.01±9.1	81.58±8.8
ionosphere	98.14±1.27	95.11±4.24	93.61±3.36	96.84±2.16	93.1±3.76
iris	98.75±2.12	97.83±3.12	98.58±2.67	98.08±2.67	97.33±2.63
kr-vs-kp	99.54±0.36	99.07±0.46	95.17±1.29	99.17±0.68	99.95±0.06
labor	95±11.25	95.83±10.58	98.33±5.27	100±0	74.17±31.04
letter	99.75±0.05	99.25±0.15	96.86±0.24	98.47±0.15	95.39±0.39
lymph	89.61±2.02	89.33±3.06	89.69±1.49	89.08±2.08	87.26±3.75
mushroom	100±0	100±0.01	99.79±0.04	100±0	100±0
primary-tumor	77.77±1.58	77.72±1.66	78.85±1.35	78.26±1.75	75.48±2.33
segment	99.66±0.17	98.82±0.26	98.51±0.46	99.09±0.43	98.85±0.32
sick	98.97±0.37	97.28±1.33	95.91±2.35	94.46±2.65	99.07±0.35
sonar	89.98±8.08	86.65±8.53	85.48±10.82	79.72±12.51	77.01±8.59
soybean	99.38±0.75	96.77±1.79	99.53±0.6	99.33±0.64	91.43±2.6
splice	98.47±0.63	97.9±0.71	99.41±0.22	99.41±0.22	98.14±0.72
vehicle	88.88±2.12	88.49±2.87	80.81±3.51	85.83±2.9	86.5±2.28
vote	98.72±1.12	97.88±1.27	96.56±2.09	98.82±1.61	96.77±2.96
vowel	99.7±0.25	96.12±0.87	95.81±0.84	98.66±0.68	91.28±2.46
waveform-5000	91.42±0.89	91.62±0.65	95.27±0.58	93.35±1.32	80.83±1.24
zoo	89.88±4.05	89.42±4.44	89.88±4.05	89.88±4.05	88.88±4.5
Mean	89.30±3.53	88.58±3.75	89.61±3.54	89.99±3.34	85.92±4.71

Table 2. Summary on t-test of experimental results: AUC comparisons on KNN, NB, NBTree, and C4.4.

		KNN	NB	NBTree	C4.4
K=5	NB	10/21/5			
	NBTree	12/23/1	7/28/1		
	C4.4	5/24/7	4/20/12	2/20/14	
	ICLNB	12/13/1	9/22/5	4/28/4	11/23/2
K=10	NB	6/26/4			
	NBTree	8/26/2	7/28/1		
	C4.4	2/25/9	4/20/12	2/20/14	
	ICLNB	10/25/1	9/24/3	6/27/3	13/22/1
K=30	NB	5/25/6			
	NBTree	8/25/3	7/28/1		
	C4.4	3/23/10	4/20/12	2/20/14	
	ICLNB	11/20/5	9/22/5	5/31/0	14/21/1

From Table 3 and 4, we can see that ICLNB achieves a significant improvement to all other KNN-related algorithms compared in AUC scores. Now, we summarize the highlights as follows:

1. ICLNB outperforms all other three algorithms significantly in AUC. For example, compared to LWNB, ICLNB wins in 6 data sets, ties in 29 data sets and loses in 1 data set, at both $k = 5$ and $k = 10$.
2. KNNDW outperforms LWNB $k = 5$ (wins in 7 data sets, loses in 1 data sets); and there is no significant difference when $k = 10$ and $k = 30$. This fact shows that weighting instances does not help to improve the probability estimates, although it results in a significant improvement in classification.
3. Both KNNDW and LWNB are significantly better than KNN. It shows that there is considerable space for improving the probability estimates of KNN.

4 Conclusions

In this paper, we have conducted a systematic empirical study on the ranking performance of KNN-related algorithms. We found that KNN-related algorithms performs well compared to naive Bayes and decision trees. We proposed an approach of combining KNN with naive Baye to improving the ranking performance of KNN, and presented an instance cloning based method to deal with the problem of lack of training data for the local naive Bayes. Our experimental results showed that our new algorithm ICLNB significantly outperforms the KNN-related algorithms KNNN, KNNDW and LWNB. It also performs better than the state-of-the-arts learning algorithms naive Bayes, C4.4 and NBTree. Our study suggests that KNN and its variants could be a good model for the data mining applications that requires accurate rankings.

Table 3. Experimental results on AUC. ICLNB: instance cloning local naive Bayes; KNN: k-nearest neighbor; KNNDW: k-nearest neighbor with distance weighted; LWNB: locally weighted naive Bayes. The value of **K** in each algorithm is **10**.

Datasets	ICLNB	KNN	KNNDW	LWNB
anneal	95.46±3.77	94.52±3.94	96.04±1.61	94.95±4.93
anneal.ORIG	94.77±3.74	92.19±7.17	94.41±3.61	94.38±2.35
audiology	71.11±0.7	70.93±0.74	71.14±0.6	71.06±0.65
autos	94.13±2.69	89.55±2.95	94.05±2.85	94.18±2.95
balance-scale	72.2±2.91	65.86±2.94	65.86±2.94	73.25±3.57
breast-cancer	61.03±12.37	62.92±12.49	65.04±12.54	62.44±12.56
breast-w	99.41±0.74	98.37±1.59	98.99±1.15	99.07±1.55
colic	82.19±4.82	86.74±5.7	87.32±4.55	84.37±4.2
colic.ORIG	76.39±6.37	76.95±6.5	75.22±7.88	73.62±10.18
credit-a	90.09±3.33	91.59±3.55	90.91±3.51	88.75±3.85
credit-g	75.11±5.64	75.97±5.23	76.06±3.89	74.53±5.02
diabetes	79.28±6.08	78.35±5.68	78.79±5.08	72.74±4.03
glass	84.43±6.05	82.53±4	85.65 ±6.55	82.88±7.01
heart-c	83.57±1.05	83.8±0.77	83.7±0.93	83.5±1.14
heart-h	83.51±0.86	83.6±0.79	83.5±0.74	83.38±0.97
heart-statlog	88.27±3.51	90.6 ±4.82	90.33±4.89	88.13±6.81
hepatitis	85.2±14.49	86.97±9.3	84.75±10.92	79.75±10.1
hypothyroid	84.99±10.95	82.25±10.83	79.22±10.68	80.07±12.51
ionosphere	98.14±1.27	95.11 ±4.24	94.16 ±2.52	96.89 ±1.4
iris	98.75±2.12	97.83±3.12	98.08±3.33	97.33±3.87
kr-vs-kp	99.54±0.36	99.07±0.46	99.55±0.21	99.38±0.29
labor	95±11.25	95.83±10.58	96.67±7.03	98.33±5.27
letter	99.75±0.05	99.25±0.15	99.44±0.07	99.43±0.06
lymph	89.61±2.02	89.33±3.06	89.64±2.36	89.19±2.76
mushroom	100±0	100±0.01	100±0	100±0
primary-tumor	77.77±1.58	77.72±1.66	76.9±1.98	76.99±2.68
segment	99.66±0.17	98.82±0.26	99.17±0.21	99.32±0.22
sick	98.97±0.37	97.28±1.33	98.08±0.85	98.03±1.64
sonar	89.98±8.08	86.65±8.53	88.57±8.27	90.27±7.45
soybean	99.38±0.75	96.77±1.79	99.28±0.76	99.31±0.8
splice	98.47±0.63	97.9±0.71	98.35±0.6	97.87±0.64
vehicle	88.88±2.12	88.49±2.87	88.2±2.91	87.84±2.91
vote	98.72±1.12	97.88±1.27	97.76±1.44	98.05±1.89
vowel	99.7±0.25	96.12±0.87	99.28±0.33	99.74±0.19
waveform-5000	91.42±0.89	91.62±0.65	91.44±0.67	87.36±1.17
zoo	89.88±4.05	89.42±4.44	89.88±4.05	89.88±4.05
Mean	89.30±3.53	88.58±3.75	89.04±3.40	88.51±3.66

Table 4. Summary on t-test of experimental results: AUC comparisons on KNN-related algorithms, including KNN, KNNDW, LWNB.

		KNN	KNNDW	LWNB
K=5	KNNDW	8/27/1		
	LWNB	6/27/3	1/28/7	
	ICLNB	12/23/1	6/30/0	6/29/1
K=10	KNNDW	7/28/1		
	LWNB	6/27/3	2/31/3	
	ICLNB	10/25/1	6/30/0	6/29/1
K=30	KNNDW	8/26/2		
	LWNB	9/25/2	4/30/2	
	ICLNB	11/20/5	9/24/3	6/27/3

References

1. Aha, David W., Dennis Kibler, Marc K. Albert. 1991. Instance-Based Learning Algorithms. Machine Learning, vol. 6, pp. 37-66.
2. http://prdownloads.sourceforge.net/weka/datasets-UCI.jar
3. Kohavi, R.: Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press (1996) 202-207
4. Provost, F. J., Domingos, P.: Tree Induction for Probability-Based Ranking. Machine Learning **52(3)** (2003) 199-215
5. Frank, E., Hall, M., Pfahringer, B.: Locally Weighted Naive Bayes. Proceedings of the Conference on Uncertainty in Artificial Intelligence (2003). Morgan Kaufmann(2003), 249-256.
6. Provost, F., Fawcett, T.: Analysis and visualization of classifier performance: comparison under imprecise class and cost distribution. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining. AAAI Press (1997) 43-48
7. Hand, D. J., Till, R. J.: A simple generalisation of the area under the ROC curve for multiple class classification problems. Machine Learning **45** (2001) 171-186
8. Ling, C. X., Yan, R. J.: Decision Tree with Better Ranking. Proceedings of the 20th International Conference on Machine Learning. Morgan Kaufmann (2003) 480-487
9. Huang, J., Lu, J., Ling, C., X.: Comparing Naive Bayes, Decision Trees, and SVM with AUC and Accuracy. Proceedings of the Third IEEE International Conference on Data Mining. IEEE Computer Society Press(2003), 553-556.
10. Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann: San Mateo, CA (1993)
11. Zhang, H., Su, J.: Naive Bayesian Classifiers for Ranking. Proceeding of ECML 2004. Springer (2004) 501-512
12. Xie, Z., Hsu, W., Liu, Z., Lee, M.: SNNB: A Selective Neighborhood Based Naive Bayes for Lazy Learning. Proceedings of the Sixth Pacific-Asia Conference on KDD. Springer (2002) 104-114
13. Witten, I. H., Frank, E.: Data Mining –Practical Machine Learning Tools and Techniques with Java Implementation. Morgan Kaufmann (2000)