SCIENCE CHINA

December 2012 Vol. 55 No. 12: 1–18
doi:

Auxiliary Material of "Misleading Classification"

JIANG He[1,2*], XUAN JiFeng[1], REN ZhiLei[1], WU YouXi[3] & WU XinDong[2]

[1]*School of Software, Dalian University of Technology, Dalian* 116621*, China,*
[2]*Computer Science Department, University of Vermont, Burlington, Vermont* 05403*, USA,*
[3]*School of Computer Science and Software, Hebei University of Technology, Tianjin* 300130*, China*

Received xx, 20xx; accepted xx, 20xx

Abstract In this material, we present the auxiliary data for [Jiang H, Xuan J F, Ren Z L. et al. Misleading classification. Sci China Inf Sci]. We first show the performance of KRA and GRA on 12 instances, which are not presented in the paper. In addition, we also investigate the effect of the candidate set size. Finally, we present the proofs of two claims of running time complexity.

Keywords misleading classification, naive Bayes, K-nearest neighbor

Citation Jiang H, Xuan J F, Ren Z L. et al. Misleading classification. Sci China Inf Sci,

1 Additional Experimental Results of KRA and GRA

In this section, we present some additional results for 12 UCI instances. All the experiments are conducted on a Pentium(R) IV CPU 3.4G PC with 1G memory running Windows XP. We implemented those algorithms in Java compiled by NetBeans IDE 6.8 using Weka 3.6.2 library.

1.1 Evaluation of KRA

In this subsection, we shall evaluate KRA on 12 UCI data sets. Since KRA employs KNN to calculate the similarities between instances, we also investigate the effect of parameter K on KRA in this subsection. For every ranked list generated by those KRA algorithms taking different K values, the top $4\%, 8\%, \ldots, 100\%$ candidate instances are selected out as the training sets respectively to build classifiers. Then we use every classifier to predict its test set for calculating the leaking and misleading rates. In such a way, Figure 1,2,and 3 report the curves of the leaking and misleading rates against the fraction of ranked candidate instances used for the training set. Four instances are tested and reported in every figure, respectively.

As shown in Figure 1, 2, and 3, the leaking rates gradually increase along with the growth of the fraction of ranked candidate instances used. On the other hand, the misleading rates follow a decreasing trend against the fraction of ranked candidate instances used. It indicates that our ranking algorithm KRA could surely contribute to misleading classification that the front part of ranked candidate instances are more likely to mislead the rival than the back part.

We can also find that performance (i.e., the leaking and misleading rates) of KRA will slightly improve along with the increase of K value. For those representative data sets, the leaking rate will be usually

*Corresponding author (email: hejiang@ieee.org)

lower when a larger K is used in KRA. In contrast, the misleading rate will generally grows when a larger K is used.

It can be concluded for every data set that all curves of those KRA algorithms taking different K values will eventually converge to the same pixel when 100% candidate instances are employed as the training set. The reason for this phenomenon is that our classification algorithm NB is independent of the order of instances in the training set. Once the whole candidate set is taken as the training set, NB will always produce the same results for the test set regardless of the order of candidate instances.

In addition, we also report the running time of KRA taking different parameter K in Figure 4, 5, and 6.

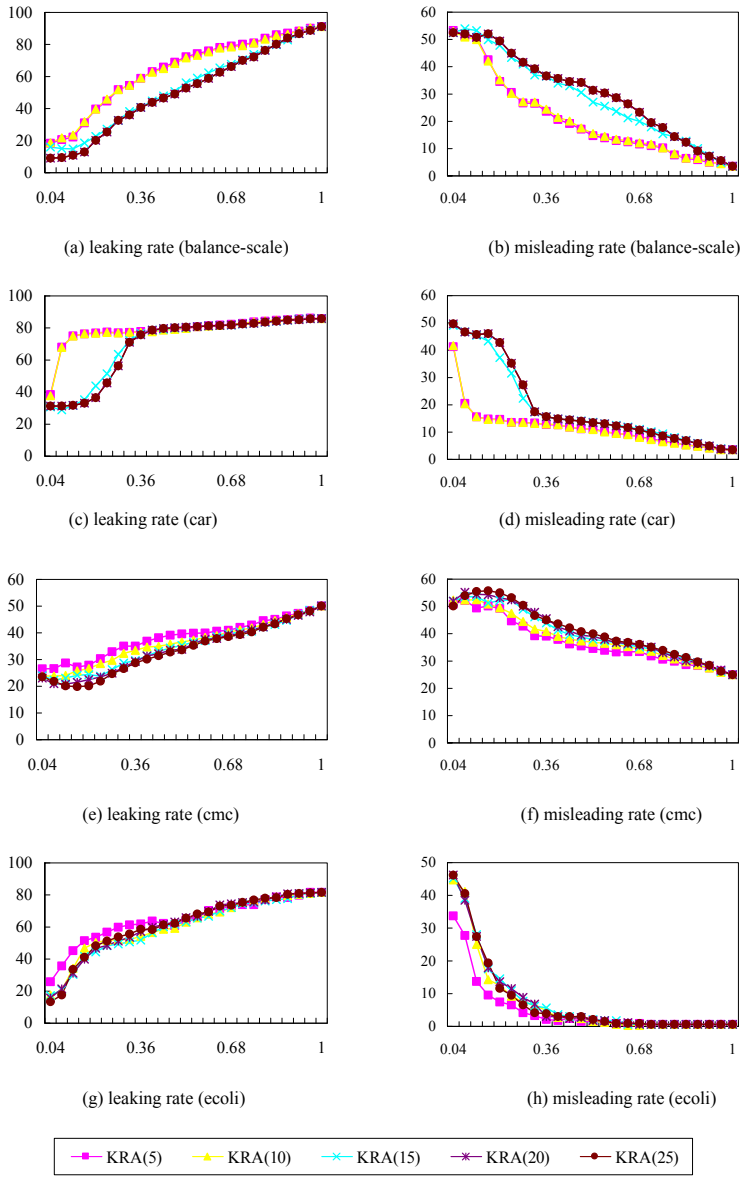

Figure 1: Performance of KRA taking different K values on balance-scale, car, cmc, and ecoli.

1.2 Evaluation of GRA

In this subsection, we evaluate the performance of GRA algorithms on 12 UCI instances. For comparison, the results of KRA taking $K = 25$ are also shown. In addition, we also implement a random algorithm (denoted by *rand*) which randomly shuffle those candidate instances to form the ranked list.

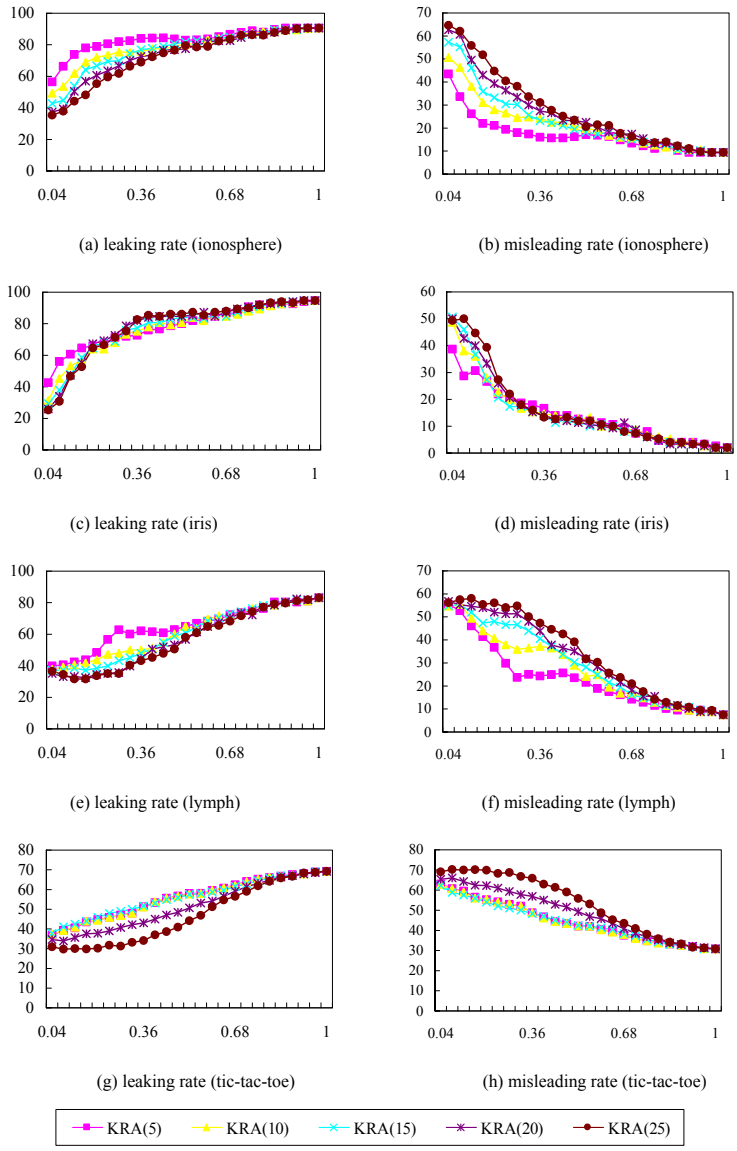

Figure 2: Performance of KRA taking different K values on ionosphere, iris, lymph, and tic-tac-toe.

Figure 7,8,and 9 report the curves of the leaking and misleading rates against the fraction of ranked candidate instances used for the training set. Since AGRA-NB will produce the same ranked candidate instances as GRA-NB, only the results of AGRA-NB are presented in these figures. We run KRA(25) and *rand* as well on those UCI data sets for comparison. To show how well the classification algorithm will fit for those test sets, we also build the classifiers on those test instances with misleading classes and apply such classifiers to predict the test sets themselves (the leaking and misleading curves of such classifiers are marked as self).

Obviously, the leaking rate curves of *rand*, KRA(25), and AGRA-NB follow similar trends that those leaking rates increase along with the growth of the fraction of ranked list used for the training set. On the other hand, the most parts of those misleading rate curves of *rand*, KRA(25), and AGRA-NB gradually drop against the growth of the fraction of ranked list used. For both the leaking rate and the misleading rate, all curves of *rand*, KRA(25), and AGRA-NB for every data set converge to the same pixel when the total candidate set is used as the training set.

In contrast to *rand*, KRA(25) achieves far lower leaking rates and higher misleading rates on those data sets. Among those three algorithms, our AGRA-NB is the best one which usually returns the

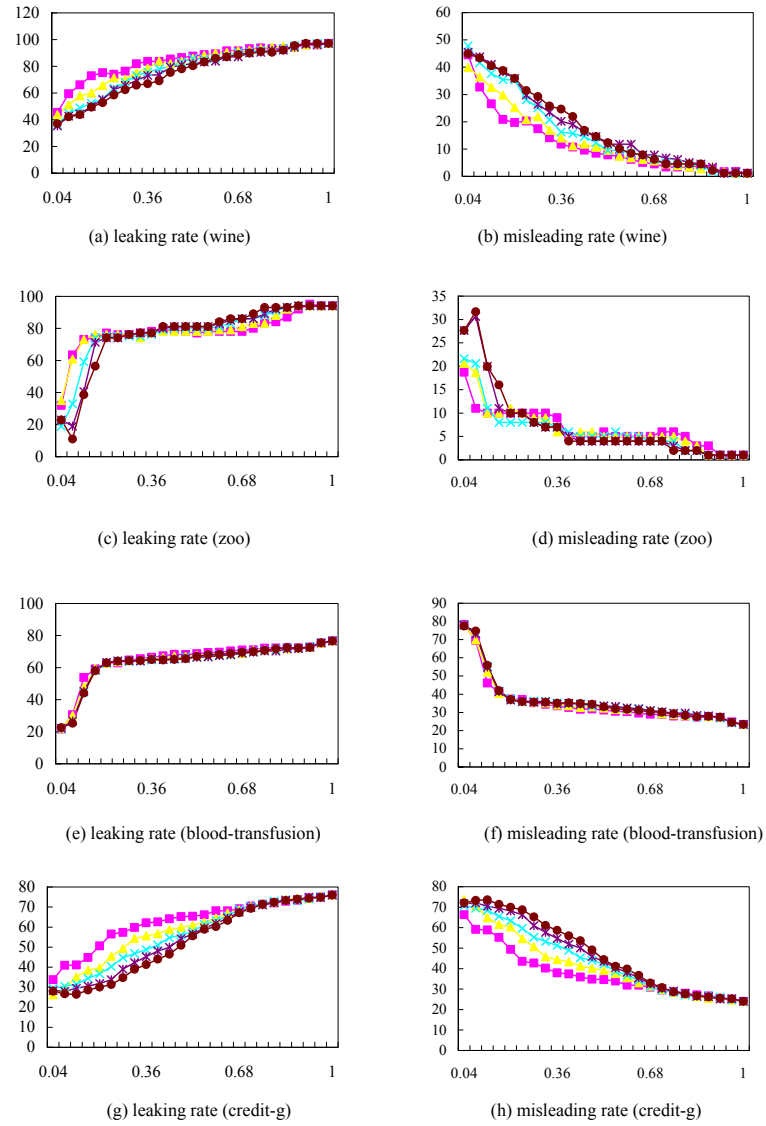

Figure 3: Performance of KRA taking different K values on wine, zoo, blood-transfusion, and credit-g.

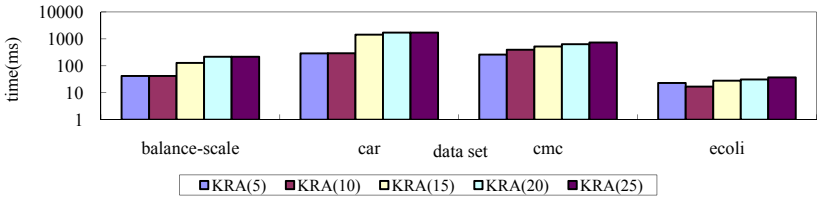

Figure 4: Running time (ms) of KRA with different K values on balance-scale, car, cmc, and ecoli.

lowest leaking rates and the highest misleading rates. In addition, the leaking and misleading curves of AGRA-NB are obviously the closest to those of self.

In addition, we report the running time (ms) of AGRA-NB and GRA-NB on UCI data sets in Figure 10, 11, and 12. It should be noted that a logarithm coordinate is adopted for presenting the running time. As shown in Figure 10,11,and 12, *rand* takes the least running time among all 4 algorithms. In general, KRA(25) costs far less running time than either AGRA-NB or GRA-NB. At the mean time, AGRA-NB is dramatically faster than GRA-NB on all 12 data sets.

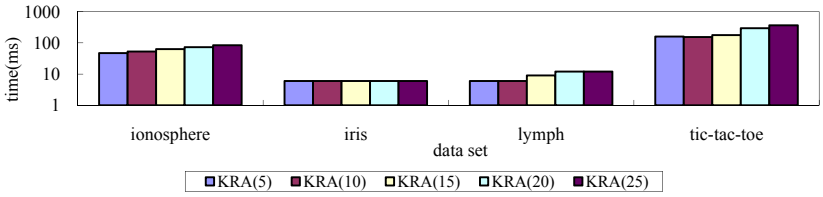

Figure 5: Running time (ms) of KRA with different K values on ionosphere, iris, lymph, and tic-tac-toe.

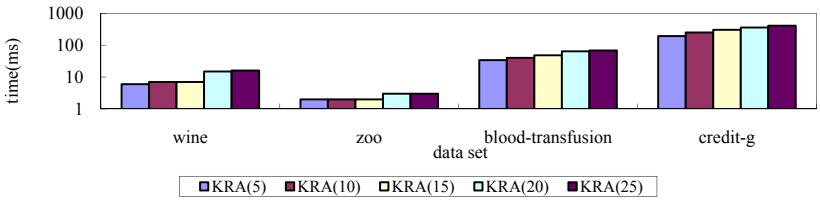

Figure 6: Running time (ms) of KRA with different K values on wine, zoo, blood-transfusion, and credit-g.

2 Effect of Candidate Set Size

In this subsection, we shall investigate the effect of the candidate set size. Every UCI data set is partitioned into 10 folds, where each fold will be used for the test set and we select n ($n = 1, 3, 5, 7, 9$) folds out of the other 9 folds respectively as the candidate set. So, for every candidate set size, each data set will be used 10 times for an algorithm and the average results over those 10 runs are given.

Figure 13 reports the experimental results of KRA taking $K = 25$. In fact, no clear relationship between the candidate size and the leaking rate (or the misleading rate) can be found. It may implies that KRA is not very sensitive to the candidate set size.

Similarly, the experimental results of AGRA-NB are presented in Figure 14. Out of 4 UCI data sets, we can find that AGRA-NB can usually achieve lower leaking rates and higher misleading rates over 3 data sets, including *yeast*, *vehicle*, and *glass*. It indicates that AGRA-NB may work better when more candidate instances are given.

A Proof of Claim1

Every instance y_i in the test set can be classified by NB in $O(N_L M + N_L) = O(N_L M)$ running time, since it takes $O(M)$ running time to calculate every joint probability $p(c, F_1, F_2, \ldots, F_M)$, and $O(N_L)$ comparisons should be conducted for finding the class with the largest joint probability. Given a test set T of N_T instances, NB takes totally $O(N_T N_L M)$ running time to calculate the leaking rate and the misleading rates. As shown in the previous subsection, GRA-NB will build $N_C(N_C+1)/2$ classifiers and classify the test set for $N_C(N_C+1)/2$ times. Obviously, the total running time for classifying the test set is $O(N_T N_L M N_C^2)$.

On the other hand, the total running time for building $N_C(N_C+1)/2$ classifiers should be calculated in an iterative way. The first iteration of the greedy phase in GRA will build N_C NB classifiers, where the training set size in each classifier is 1. The second iteration will build $N_C - 1$ NB classifiers whose training set sizes are 2. In such a way, the total running time for building all NB classifiers is $O(\sum_{i=1}^{N_C}(N_C - i + 1)(iM + N_L N_F)) = O(N_C^3 M + N_C^2 N_L N_F)$.

Therefore, the overall running time of GRA-NB is $O(N_T N_L M N_C^2 + N_C^3 M + N_C^2 N_L N_F)$.

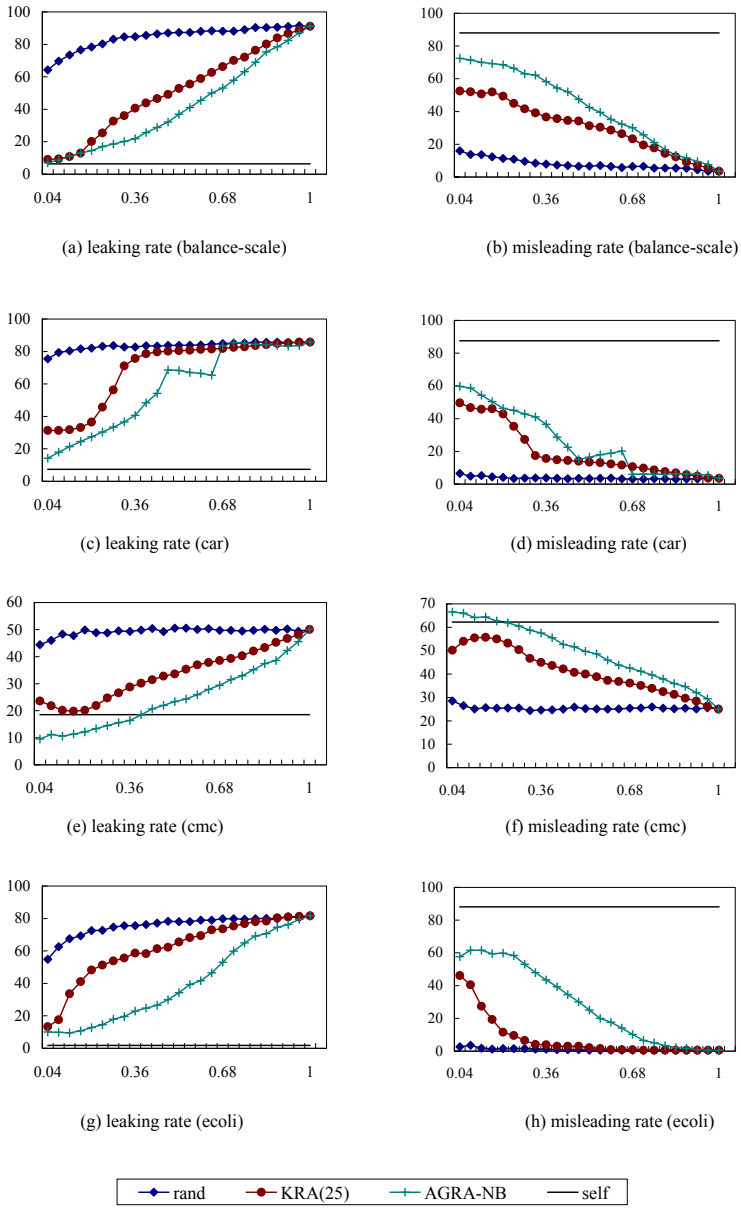

Figure 7: Performance of rand, KRA(25), AGRA-NB, and self on balance-scale, car, cmc, and ecoli.

B Proof of Claim2

The running time of AGRA-NB can be evaluated as follows. Step 1 and 5 can be conducted in $O(1)$ running time. Step 2 can be run in $O(N_C)$ time. It takes $O(N_L), O(N_L N_F)$, and $O(N_T N_L)$ running time to initialize V, W, and P respectively in Step 3.

For the greedy phase of AGRA-NB, the outer loop will be run for totally N_C iterations. Since P is available, it just needs $O(N_L N_T)$ running time to calculate the misleading rate and the leaking rate in either Step 4.2 or Step 4.4.2. Since Step 4.3, 4.4.3, 4.4.4, 4.6, 4.7, and 4.8 can be done in $O(1)$ running time respectively, the ith iteration of the outer loop in the greedy phase of AGRA-NB can be conducted in $(N_T - i + 2)O(MN_T) + (N_T - i + 1)O(N_T N_L)$ running time.

The overall running time of AGRA-NB is $O(1)+O(N_C)+O(N_L)+O(N_L N_F)+O(N_T N_L)+\sum_{i=1}^{N_C}((N_C - i + 2)O(MN_T) + (N_C - i + 1)O(N_T N_L)) + O(1) = O(N_C^2 M N_T + N_C^2 N_T N_L + N_L N_F)$.

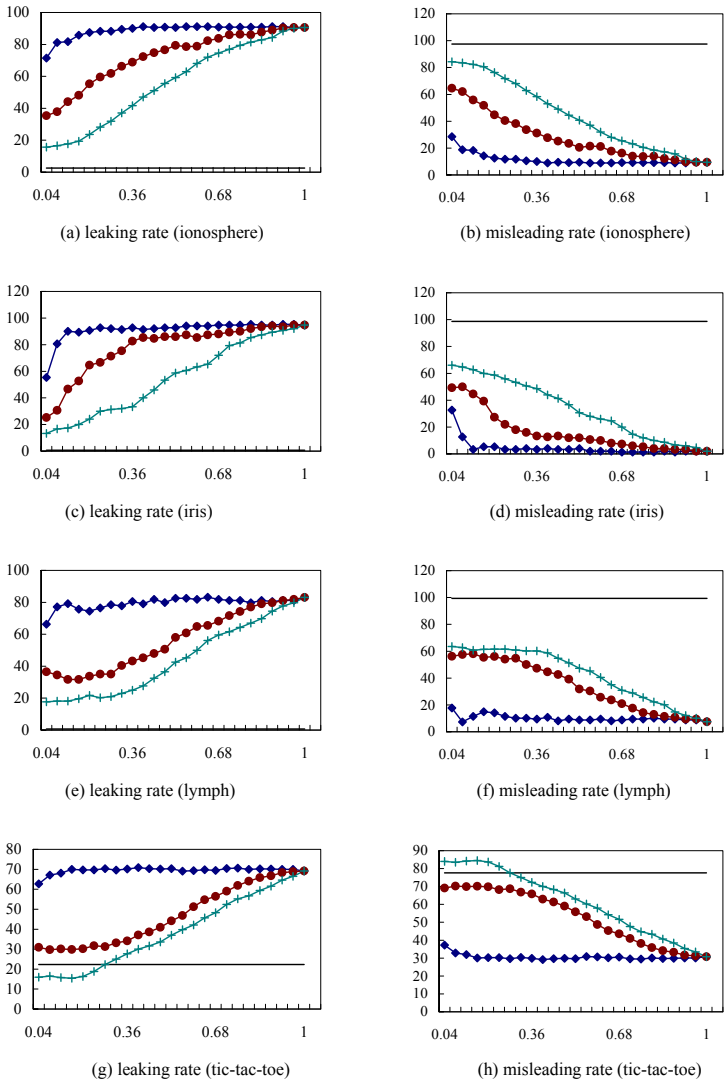

Figure 8: Performance of rand, KRA(25), AGRA-NB, and self on ionosphere, iris, lymph, and tic-tac-toe.

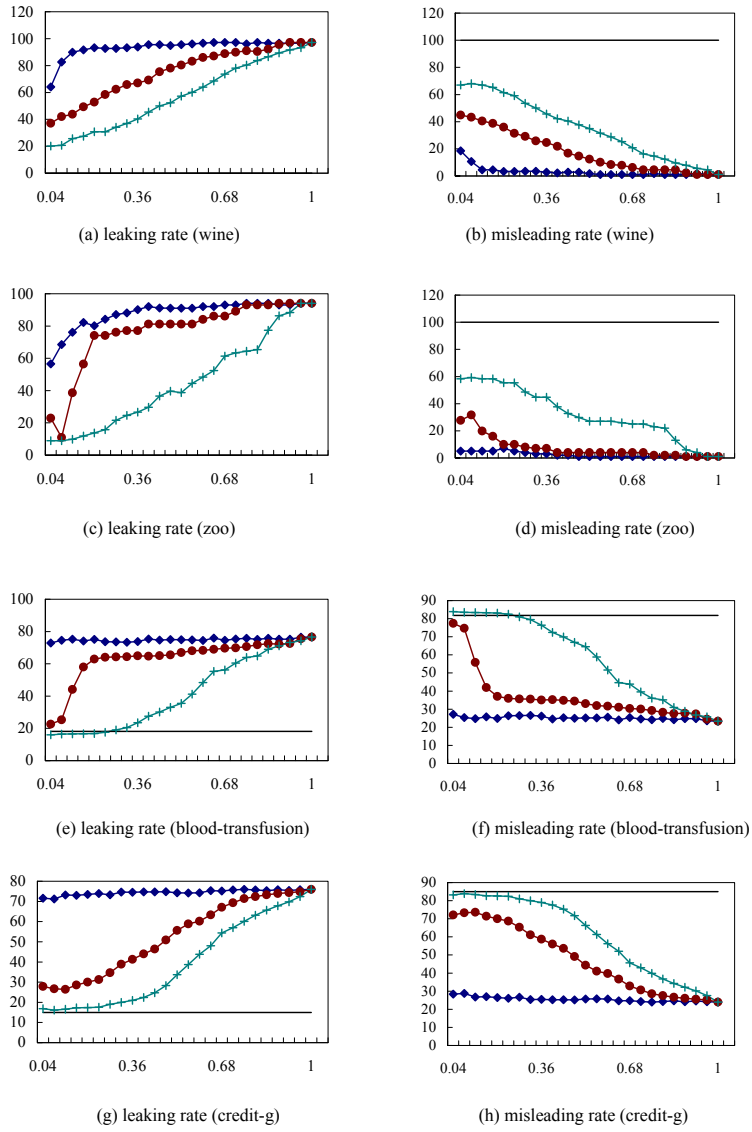

Figure 9: Performance of rand, KRA(25), AGRA-NB, and self on wine, zoo, blood-transfusion, and credit-g.

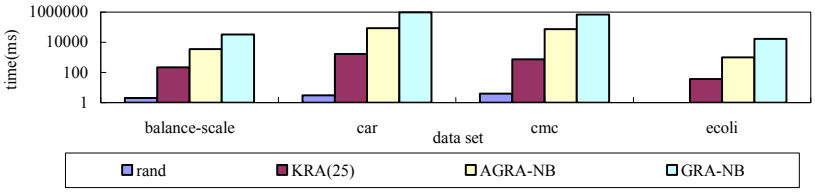

Figure 10: Running time (ms) of KRA with different K values on balance-scale, car, cmc, and ecoli.

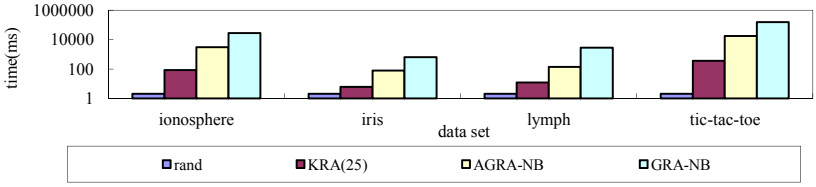

Figure 11: Running time (ms) of KRA with different K values on ionosphere, iris, lymph, and tic-tac-toe.

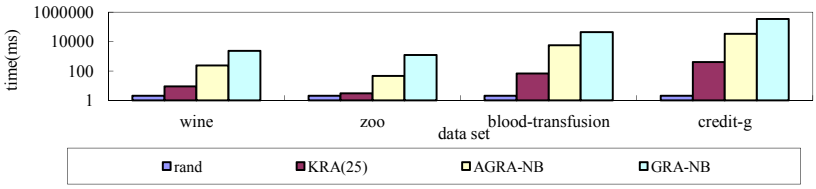

Figure 12: Running time (ms) of KRA with different K values on wine, zoo, blood-transfusion, and credit-g.

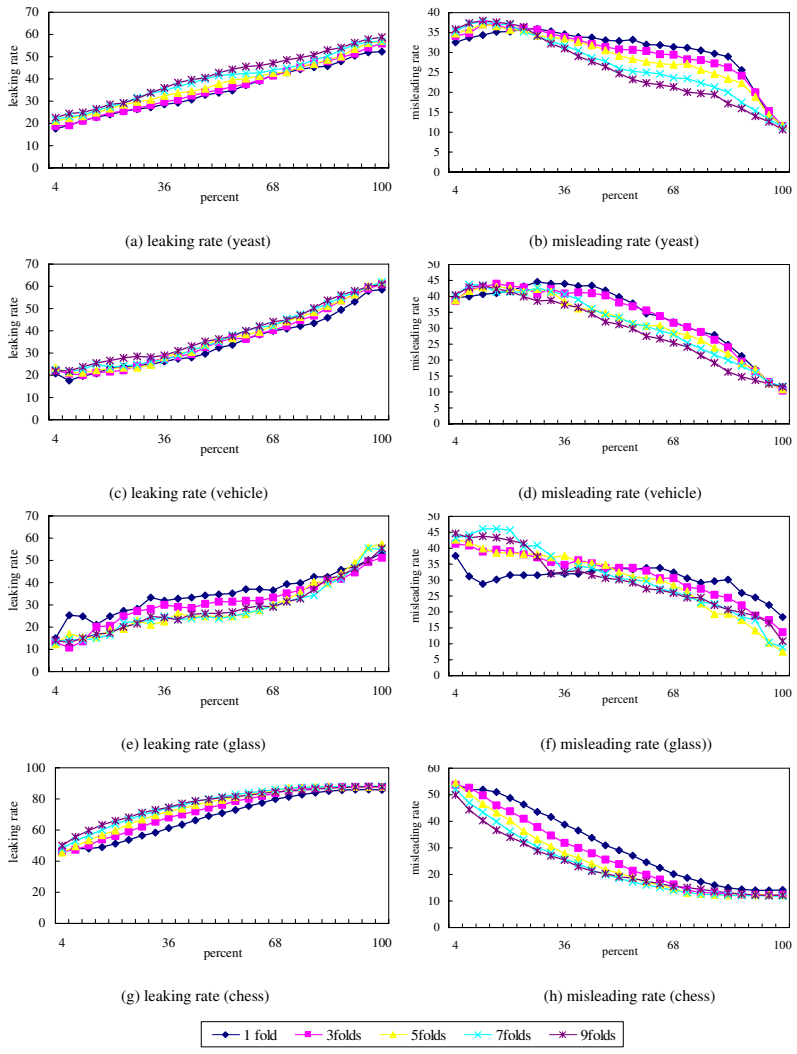

Figure 13: Performance of KRA(25) taking different candidate set size.

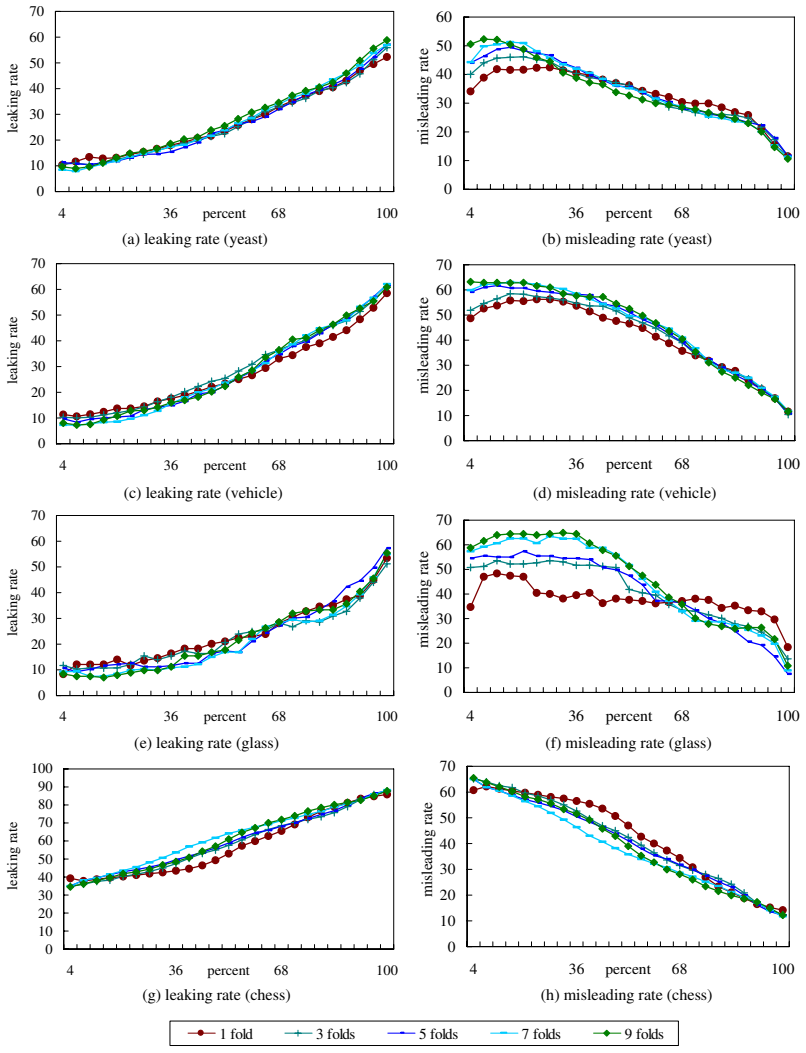

Figure 14: Performance of AGRA-NB taking different candidate set size.