Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A cost-sensitive classification algorithm: BEE-Miner

Pınar Tapkan[a,*], Lale Özbakır[a], Sinem Kulluk[a], Adil Baykasoğlu[b]

[a] *Department of Industrial Engineering, Erciyes University, Kayseri 38039, Turkey*
[b] *Department of Industrial Engineering, Dokuz Eylül University, İzmir 35210, Turkey*

ARTICLE INFO

Article history:
Received 4 June 2015
Revised 1 December 2015
Accepted 20 December 2015
Available online 31 December 2015

Keywords:
Bees Algorithm
Cost-sensitive classification
Data mining

ABSTRACT

Classification is a data mining technique which is utilized to predict the future by using available data and aims to discover hidden relationships between variables and classes. Since the cost component is crucial in most real life classification problems and most traditional classification methods work for the purpose of correct classification, developing cost-sensitive classifiers which minimize the total misclassification cost remains a subject of much interest. The purpose of this study is to present an effective solution method that configurates and evaluates learning systems from previous experiences, thus aiming to obtain decisions and predictions. Since most real life problems are cost-sensitive and developing effective direct methods for cost-sensitive multi-class classification is still an attractive area, a cost-sensitive classification method, the BEE-Miner algorithm, is proposed by utilizing the recently developed Bees Algorithm (BA). The main advantages of BEE-Miner are its capability to handle both binary and multi-class problems and to incorporate misclassification cost into the algorithm via generating neighbor solutions and evaluating the quality of the solutions. An extensive computational study is also performed on cost-insensitive and cost-sensitive versions of the proposed BEE-Miner algorithm and effective results on different types of problems are obtained with high test accuracy and low misclassification cost.

1. Introduction

As a result of the rapid development of information technology and data collection resources, some difficulties are experienced in revealing meaningful and useful information from large sized datasets. Unfortunately, traditional statistical methods are insufficient to resolve these kinds of data. Data mining (DM), which is a convenient method for handling this type of data, can be defined as the acquisition of valid, enforceable, and previously unknown information from large sized databases and its effective use in the decision process. On the other hand, classification, which is the most widely used DM technique, is utilized to predict the future by using the available data and aims to discover hidden relationships between variables and classes when given a set of samples with known output classes.

Moreover most traditional classification methods, which are referred to as cost-insensitive classifiers, work for the purpose of correct classification. These methods aim to minimize the percentage of false predicted class labels, namely misclassification errors, and assume the presence of symmetric classification costs (all misclassification costs are equal to each other) by ignoring the difference between misclassification errors. However, the cost is not a discardable component and in many real life classification problems the cost difference between the misclassification errors can be quite high. For example, in a cancer diagnosis database with positive and negative classes representing having cancer or not, respectively, classifying a positive patient as negative will have a much greater cost than classifying a negative patient as positive. In other words, classifying a patient with cancer as healthy is much more serious than classifying a healthy patient as having cancer, since the wrong diagnosis may cause a delay in treatment or the death of the patient. In fact, it is possible to obtain low misclassification costs by utilizing effective traditional classifiers for solving such problems, but the cost component must be handled during the classification process. The area of machine learning dealing with such problems with non-uniform costs is known as cost-sensitive learning. Cost-sensitive learning, which aims at minimizing the cost of misclassified cases, has an effective structure that can be used for solving real life classification problems by removing the shortcomings of traditional classifiers.

Cost-sensitive learning is generally utilized on datasets that have imbalanced class distribution, such as encountered in many real life applications. To classify such a dataset with binary classes by cost-insensitive classifiers will give rise to major problems since almost all samples will be classified as negative. Under this circumstance, cost-sensitive classification is an effective method for the

* Corresponding author. Tel.: +90 352 2076666; fax: +90 352 4375784.
E-mail address: pinartan@erciyes.edu.tr (P. Tapkan).

http://dx.doi.org/10.1016/j.knosys.2015.12.010
0950-7051/© 2015 Elsevier B.V. All rights reserved.

classification of this kind of dataset. For this reason, the number of studies on cost-sensitive classification has gained momentum in recent years. These studies are generally divided into two groups: studies on developing direct cost-sensitive learning algorithms (direct methods) and studies on converting a cost-insensitive learning algorithm into a cost-sensitive one (meta-learning methods). Direct methods involve cost-sensitive classifiers that include the misclassification costs in the learning algorithm whereas meta-learning methods perform a pre-processing stage in order to accommodate training data for cost-sensitive learning or a post-processing stage on the outcomes of the cost-insensitive learning algorithm. On the other hand, the studies on cost-sensitive learning generally focus on developing meta-learning approaches and solving binary classification problems. Nowadays, the development of direct cost-sensitive classifiers, especially the handling of multi-class problems is an area that must be analyzed both theoretically and practically.

The purpose of this study is to present an effective solution method that configurates and evaluates learning systems from previous experiences and aims to obtain decisions and predictions. Since most real life problems are cost-sensitive and the scientific literature on direct cost-sensitive classifiers still requires new approaches, the BEE-Miner is proposed by utilizing the recently developed Bees Algorithm (BA). The performance of different discretization methods is also analyzed and complex neighborhood structures based on entropy and misclassification costs are developed. Since the BEE-Miner has a multi-rule structure, to find the best rule combination among the rule pool, which consist of the rules covering all samples in the training set, the genetic algorithm is used. The main advantages of the BEE-Miner are its capability to handle both binary and multi-class problems and to incorporate misclassification costs into the algorithm's operation principles rather than updating input or output. The stages that include the cost component are generating neighbor solutions and evaluating the quality of the solutions by using the fitness function. The remainder of the paper is organized as follows. Sections 2 and 3 clarify cost-sensitive classification and the Bees Algorithm, respectively. Section 4 presents the cost-insensitive and cost-sensitive BEE-Miner algorithm, Section 5 gives computational studies and Section 6 concludes the paper.

2. Cost-sensitive classification

The main distinction of cost-sensitive learning from cost-insensitive learning is that the former considers misclassification costs, in other words, it focuses on minimizing expected misclassification cost instead of minimizing expected misclassification errors. When we analyze the cost matrices of both cases, there is no difference between misclassification costs for the cost-insensitive case. On the other hand, to predict a sample of a certain class as a different class differs according to the predicted class for the cost-sensitive case. Consequently, cost-insensitive classification is a special form of cost-sensitive classification.

Elkan [1] and Zadrozny and Elkan [2] present the general structure of cost-sensitive learning for a binary class and describe the role of misclassification cost on different cost-sensitive learning algorithms in detail. An example of a cost matrix for binary classification including two classes, named as negative (0) and positive (1), is given as follows:
where

TN	number of samples that are actual negative and also predicted as negative;
FN	number of samples that are actual positive but predicted as negative;
FP	number of samples that are actual negative but predicted as positive;
TP	number of samples that are actual positive and also predicted as positive;
$C(i, j)$	cost of predicting a sample of class i as class j;
$C(i, i)$	samples that are predicted truly and characterized as benefit.

In cost-sensitive learning, such a matrix is assumed to be known and the rows and columns can be easily expanded for multi-class classification. Depending on the cost matrix, the samples are classified according to the minimum expected cost values.

2.1. Literature review of cost-sensitive classification

Cost-sensitive classification algorithms are analyzed in two groups as direct methods, which have been studied rarely in the literature, and meta-learning techniques that are divided into thresholding and sampling.

One of the direct methods, Inexpensive Classification with Expensive Tests (ICET), is a hybrid combination of the genetic algorithm and the decision tree induction algorithm and is based on the inclusion of test cost as well as misclassification cost in the fitness function of the genetic algorithm [3]. On the other hand, the Cost-sensitive Decision Tree (CSTree) includes the misclassification cost in the tree generation process, directly. In other words, the CSTree selects the best attribute according to the total misclassification cost [4,6]. Drummond and Holte [4] investigate the effect of misclassification cost on splitting and pruning methods in decision tree learning algorithms. Ling et al. [5] propose a simple splitting criterion that minimizes the sum of misclassification and test costs for attribute selection during the generation of decision trees. Ling et al. [6] deal with two-class problems and estimate the probability of the positive class using the relative cost of both classes and use this to calculate the expected cost. Ling et al. [7] propose a lazy decision tree learning algorithm that minimizes the sum of attribute and misclassification costs. Since it is hard to minimize misclassification cost or test cost simultaneously, Qin et al. [8] aim to minimize one kind of cost and control the other in a given budget. Sheng and Ling [9] propose a hybrid cost-sensitive decision tree composed of decision trees and Naive Bayes that benefits from the advantages of both techniques.

On the other hand, AdaCost presented by Fan et al. [10] is the cost-sensitive binary class version of AdaBoost which acts as a meta-learning technique. In AdaCost, the weight updating rule increases the weights of costly wrong classifications more aggressively, but decreases the weights of costly correct classifications more conservatively. Under this updating rule, the weights for expensive samples are higher and the weights for inexpensive samples are comparatively lower. Moreover, Margineantu [11] presents two multi-class versions of cost-sensitive bagging and shows that these versions can reduce the cost for some datasets. Tu [12] and Tu and Lin [13] utilize one-sided support vector regression for solving multi-class cost-sensitive learning problems.

Furthermore, from the metaheuristic's perspective, Li et al. [14], Ashfaq et al. [15], and Song et al. [16] utilized the genetic programming, the ant colony algorithm, and the genetic algorithm, respectively, in order to solve cost-sensitive classification problems in recent years.

When the studies on cost-sensitive classification in the literature are analyzed, it can be concluded that studies on binary class have attained a certain level but those on multi-class still going on. In conclusion, developing effective direct methods for cost sensitive multi-class classification is still an attractive area for DM researchers.

```
1. Initialize population with random solutions
2. Evaluate fitness of the population
3. While (stopping criterion not met) // Forming new population
4. Select sites for neighborhood search
5. Recruit bees for selected sites (more bees for best e sites) and evaluate fitnesses
6. Select the fittest bee from each patch
7. Assign remaining bees to search randomly and evaluate their fitnesses
8. End While
```

Fig. 1. The pseudo-code for BA.

3. Bees algorithm

The BA presented by Pham et al. [17] forms the basis of the BEE-Miner and is one of the metaheuristic methods based on swarm intelligence that models the foraging behavior of bees. Although the BA was developed especially for solving function optimization problems, in later years it was also utilized for both functional and combinatorial optimization problems [18–20]. The general structure of the BA is presented in Fig. 1.

The food foraging process in nature starts with scout bees selecting random food sources. Also during the food foraging process, a certain percentage of the population remains as scout bees. Similarly, the BA starts with the scout bees (n) being placed randomly in the search space. In step 2, the fitnesses of the sites visited by the scout bees are evaluated. In principle, food sources that have a high nectar amount and can be consumed with less effort will be visited by more bees. Based on this fact, in step 4, bees that have the highest fitnesses are chosen as "selected bees" (m) and the sites visited by them are chosen for neighborhood search. Then, in step 5, the algorithm conducts searches in the neighborhood of the selected sites, assigning more bees to search the area near the best e sites. Searches in the neighborhood of the best e sites which represent more promising solutions are made in a more detailed way by recruiting more bees to follow them than the other selected bees. Together with scouting, this differential recruitment is a key operation of the BA. However, in step 6, for each patch only the bee with the highest fitness will be selected to form the next bee population. In step 7, the remaining bees (n-m) in the population are assigned randomly around the search space to scout for new potential solutions. These steps are repeated until a stopping criterion is met. At the end of each iteration, the colony will have two parts to construct the new population: those that were the fittest representatives from a patch and those that were sent out randomly.

4. Bee-Miner

Since the BEE-Miner is developed as a direct method, it is based on incorporating misclassification cost into the algorithm's operation principles rather than updating input or output. The stages that include the cost component are generating neighbor solutions and evaluating the quality of the solutions using the fitness function. However, firstly the cost-insensitive BEE-Miner is developed in order to exhibit the effectiveness of the algorithm and the obtained results are compared with the results of algorithms presented in the literature.

4.1. Cost-insensitive BEE-Miner

The BEE-Miner algorithm starts with the pre-processing stage that completes the missing values and discretizes the continuous attributes. Afterwards, the initial solutions are generated and the fitness function values of these solutions are evaluated. Then the Bees Algorithm is utilized to form the new population and, finally, the rule set is generated via the genetic algorithm. The flowchart of the BEE-Miner algorithm is presented in Fig. 2 and the details of the proposed algorithm are described in the following sections.

4.1.1. Generating initial solutions

Due to the high number of attributes in real life data, it is not effective to use the binary form or original attribute values at the solution string. In this context, the binary and discrete attribute values are split as "=", "≠", "null" and continuous attribute values are split as "≤", "≥", "null", where the operator of an attribute considered as "null" indicates that such an attribute does not have any effect on the rule generation stage. An example of the solution string for the BEE-Miner is illustrated in Fig. 3.

In Fig. 3, attributes 1–3 are continuous, attributes 4–6 are discrete, and attributes 7–9 are binary. The length of the solution string is "# of attributes+1" depending on all attributes and the class label. In this structure, each attribute is connected by the AND logical operator to generate a rule. The classification rule demonstrated in Fig. 3 is as follows:

If A1≤5.25 *AND* A2≥12.50 *AND* A4=6 *AND* A5≠7 *AND* A7=1 *AND* A8≠1 *then* Class 1.

The initial solutions are generated by selecting an operator-value combination for each attribute among possible alternatives, randomly. Table 1.

On the other hand, at the discretization stage the equal-width discretization (EWD) [21], equal-frequency discretization (EFD) [21], fixed-frequency discretization (FFD) [22], multi-interval-entropy-minimization discretization (MIEMD) [23], ChiMerge [24] and InfoMerge [25] discretization methods are evaluated. While discretizing a quantitative attribute, EWD predefines the number of intervals (k) and then divides the number line between v_{min} and v_{max} into k intervals of equal width, where v_{min} is the minimum observed value and v_{max} is the maximum observed value. On the other hand, EFD predefines the number of intervals (k) and then divides the sorted values into k intervals so that each interval contains approximately the same number of training instances. Note that training instances with identical values must be placed in the same interval. In consequence it is not always possible to generate k equal-frequency intervals. While discretizing a quantitative attribute, FFD predefines a sufficient interval frequency k and discretizes the sorted values into intervals so that each interval has approximately the same number k of training instances with adjacent (possibly identical) values. To discretize an attribute, MIEMD evaluates as a candidate cut point the midpoint between each successive pair of sorted values. For evaluating each candidate cut point, the data are discretized into two intervals and the resulting class information entropy is calculated. The binary discretization is applied recursively, always selecting the best cut point. ChiMerge discretization uses the χ^2 statistic to determine if the relative class frequencies of adjacent intervals are distinctly different or if they are similar enough to justify merging them into a single interval. Merging continues until all pairs of intervals have χ^2 values exceeding a predefined χ^2 threshold. This threshold is determined as 0.95 significance level, as recommended by Yang et al. [26]. InfoMerge uses information loss, which is calculated as the amount

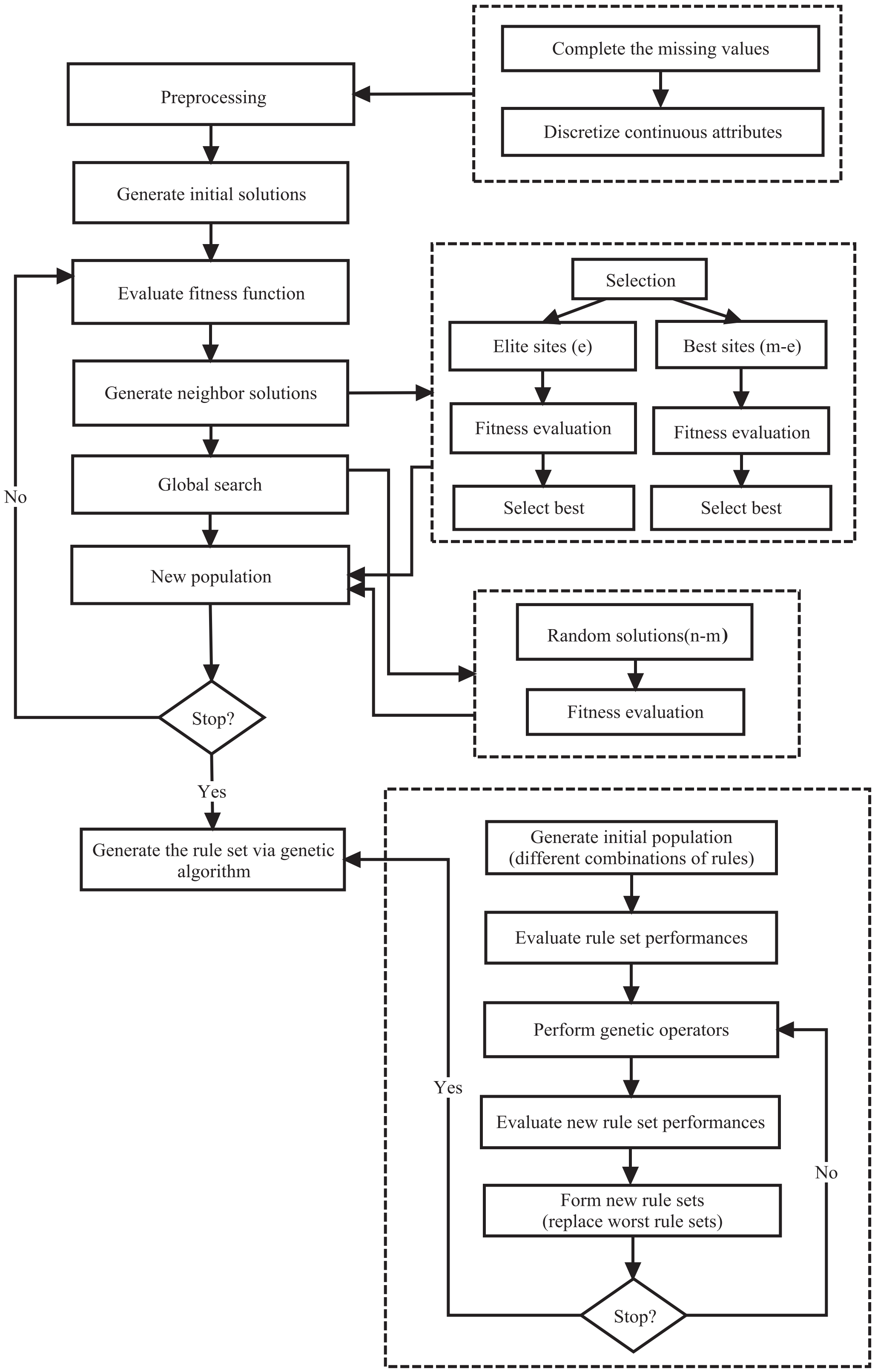
Complete the missing values
Discretize continuous attributes
Preprocessing
Generate initial solutions
Evaluate fitness function
Selection
Elite sites (e)
Best sites (m-e)
Generate neighbor solutions
Fitness evaluation
Fitness evaluation
Global search
Select best
Select best
No
New population
Random solutions(n-m)
Fitness evaluation
Stop?
Yes
Generate the rule set via genetic algorithm
Generate initial population (different combinations of rules)
Evaluate rule set performances
Perform genetic operators
Yes
Evaluate new rule set performances
No
Form new rule sets (replace worst rule sets)
Stop?

Fig. 2. Flowchart of BEE-Miner.

of information necessary to identify the class of an instance after merging and the amount of information before merging, to direct the merge procedure [26].

In order to determine the discretization method that will be used in later computational studies, 3 different datasets with continuous attributes are selected. These datasets consist of both binary and multi-class cases and the characteristics are summarized in Table 2.

In order to obtain an accurate comparison, the parameters of the discretizaton methods are adjusted as having 4 cut points for

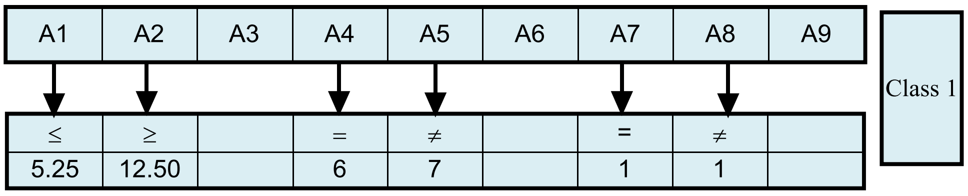

Fig. 3. Example for the structure of solution string.

Table 1
An example of a cost matrix for binary classification.

		Predicted	
		Negative	Positive
Actual	Negative	*C(0, 0)* TN	*C(1, 0)* FP
	Positive	*C(0, 1)* FN	*C(1, 1)* TP

Table 2
The selected datasets for comparing discretization methods.

Dataset	# of samples	# of categoric attributes	# of continuous attributes	# of classes
Heart disease (Statlog)	270	7	6	2
Wisconsin breast cancer	699	–	9	2
Wine	178	–	13	3

Table 4
p values of *t*-test for the comparison of discretization methods.

Dataset	Heart disease (Statlog)	Wisconsin breast cancer	Wine
EWD-EFD	0.421	0.255	0.210
EWD-FFD	**0.002**	**0.042**	0.349
EWD-MIEMD	–	0.344	0.85
EWD-ChiMerge	0.205	0.191	0.392
EWD-InfoMerge	0.333	0.396	0.187
EFD-FFD	0.180	0.354	**0.030**
EFD-MIEMD	–	**0.047**	0.220
EFD-ChiMerge	0.951	0.314	0.876
EFD-InfoMerge	0.195	0.594	0.874
FFD-MIEMD	–	**0.003**	0.285
FFD-ChiMerge	**0.034**	0.407	0.162
FFD-InfoMerge	**0.000**	0.183	**0.015**
MIEMD-ChiMerge	–	0.122	0.467
MIEMD-InfoMerge	–	0.109	0.212
ChiMerge-InfoMerge	**0.037**	0.260	0.768

each method and each dataset. Each dataset is executed for each discretization method for 30 times by using 10-fold cross validation and any case having a missing value is removed from the dataset. The average test accuracies, the number of rules, and CPU time in seconds are summarized in Table 3. In addition, since obtaining a cut point for two attributes of the heart disease dataset is impossible by the MIEMD method, the heart disease dataset could not be discretized by the MIEMD method.

To compare the discretization methods, the *t*-test is applied to the average test accuracy results for each dataset by assuming that the significance level is 0.05. The obtained *p*-values are presented in Table 4 and indicate that there are statistically significant differences between 9 comparisons.

The obtained *p*-values indicate that EWD outperformed FFD on the Wisconsin breast cancer dataset. FFD outperformed EWD, ChiMerge, and InfoMerge on the heart disease dataset; it also outperformed EFD and InfoMerge on the wine dataset. MIEMD outperformed EFD and FFD on the Wisconsin breast cancer dataset. ChiMerge outperformed InfoMerge on the heart disease dataset. Hence, FFD is determined as the discretization method for later computational study.

4.1.2. Neighborhood structures

The solutions that have the best fitness value are selected for neighborhood search. The algorithm searches for more promising best *e* solutions which are more detailed than the other selected solutions. Afterwards, each solution is updated and only the solution with the best fitness is selected to form the next bee population. At the stage of generating neighbor solutions that correspond to the position update of the bees, a structure based on changing the operator-value combination of a certain attribute on the solution string is utilized. This structure is based on entropy which is the underlying essence of the ID3 and C4.5 algorithms. Assume that the samples on the *S* database are distinguished as negative and positive. According to this binary classification, the entropy of the database is calculated by Eq. (1).

$$Entropy(S) = -p_{\oplus}\log_2 p_{\oplus} - p_{\ominus}\log_2 p_{\ominus} \tag{1}$$

where $p_{\oplus}$ represents the positive samples and $p_{\ominus}$ represents the negative samples. Under the assumption of having multi-class, the entropy of the database is calculated by Eq. (2).

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{2}$$

where p_i represents the portion of the database belonging to class *i* and *c* represents the number of classes. On the other hand, the information gain of an attribute is the expected reduction in entropy caused by partitioning the samples according to this attribute, i.e.,

Table 3
The comparison of discretization methods.

		EWD	EFD	FFD	MIEMD	ChiMerge	InfoMerge
Heart disease (Statlog)	Test accuracy	71.60 ± 8.82	73.08 ± 8.07	75.67 ± 7.06	–	73.20 ± 8.17	70.00 ± 8.21
	# of rules	3.13 ± 0.86	3.33 ± 0.66	3.56 ± 0.56	–	3.26 ± 0.58	2.80 ± 0.71
	CPU	56.53 ± 3.43	42.26 ± 2.33	42.3 ± 2.49	–	39.16 ± 2.47	44.30 ± 2.21
Wisconsin breast cancer	Test accuracy	94.32 ± 2.84	93.75 ± 3.12	93.42 ± 2.96	94.80 ± 3.32	91.84 ± 10.14	93.99 ± 2.66
	# of rules	4.33 ± 0.88	3.83 ± 1.01	3.66 ± 0.88	4.76 ± 1.10	4.33 ± 1.26	3.53 ± 0.97
	CPU	53.93 ± 2.54	58.70 ± 3.01	55.50 ± 2.88	49.13 ± 2.84	63.06 ± 3.56	56.63 ± 2.78
Wine	Test accuracy	95.33 ± 4.40	94.01 ± 4.87	96.08 ± 4.42	95.16 ± 4.78	94.23 ± 6.27	93.85 ± 5.12
	# of rules	3.16 ± 0.37	3.80 ± 0.48	3.13 ± 0.34	3.3 ± 0.53	3.13 ± 0.34	3.33 ± 0.47
	CPU	35.23 ± 1.10	34.4 ± 1.16	32.86 ± 1.43	37.16 ± 2.13	35.8 ± 1.27	33.43 ± 1.69

Table 5
Lenses dataset.

	A1	A2	A3	A4	C		A1	A2	A3	A4	C
1	1	1	1	1	3	13	2	2	1	1	3
2	1	1	1	2	2	14	2	2	1	2	2
3	1	1	2	1	3	15	2	2	2	1	3
4	1	1	2	2	1	16	2	2	2	2	3
5	1	2	1	1	3	17	3	1	1	1	3
6	1	2	1	2	2	18	3	1	1	2	3
7	1	2	2	1	3	19	3	1	2	1	3
8	1	2	2	2	1	20	3	1	2	2	1
9	2	1	1	1	3	21	3	2	1	1	3
10	2	1	1	2	2	22	3	2	1	2	2
11	2	1	2	1	3	23	3	2	2	1	3
12	2	1	2	2	1	24	3	2	2	2	3

information gain determines the effectiveness of an attribute during the classification of training data and is calculated by Eq. (3).

$$InformationGain(S, A) = Entropy(S) - \sum_{v \epsilon V(A)} \frac{|S_v|}{|S|} Entropy(S_v) \quad (3)$$

where *V(A)* is the set of all possible values for attribute *A; S_v* is the dataset where attribute *A* takes the value of *v*. The first part of the information gain is the entropy of the database and the second part is the expected value of entropy when the database is partitioned according to attribute *A* [27].

The neighborhood structures utilized in BEE-Miner and the usage of information gain in these neighborhood structures are as follows. The proposed algorithm uses 3 types of neighborhood structures as insertion, extraction, and alteration. Having derived a neighbor solution, one of these neighborhood structures is selected randomly. The insertion neighborhood structure is based on the inclusion of an attribute that does not take place in the current solution string; similarly, the extraction neighborhood structure is based on removing an attribute that takes place in the current solution string. The selection of the attribute that will be inserted or extracted is carried on the information gain value. By using a probabilistic structure, the insertion probability of the attributes with high information gain value will be high for the insertion neighborhood and the extraction probability of the attributes with low information gain value will be high for the extraction neighborhood. On the other hand, the alteration neighborhood structure is based on selecting an attribute on the current solution string and changing the operator or value information. These 3 neighborhood structures are visually explained on the Lenses dataset with 3 classes, as presented in Table 5.

The entropy of the Lenses dataset is calculated by Eq. (2) as 1.32. The information gains of all attributes are calculated by Eq. (3) as 0.83, 0.39, 0.03, and 0.63, respectively. Afterwards, the probability vector is constructed according to information gain values. The implementation of neighborhood structures on a solution string is presented in Table 6.

The current solution string is composed of attributes 1 and 3. According to the possible attributes that can be inserted into the solution string, the probability vector is constructed for attributes 2 and 4 by using information gain values such as 0.39/(0.39 + 0.63) = 0.38 and 0.38 + 0.63/(0.39 + 0.63) = 1, respectively. The probability of inserting attribute 4 is higher than that of attribute 2, since the information gain of attribute 4 is higher than that of attribute 2. After the implementation of the insertion neighborhood, attribute 4 is added to the solution string according to the reflection of the generated random number on the probability vector. Similarly, the attributes that can be extracted from the current solution string are attributes 1 and 3. The implementation of the extraction neighborhood is achieved by only utilizing attribute 1. Eventually, for the

Table 6
The implementation of neighborhood structures for cost-insensitive BEE-miner.

						Insertion				
						A_1	A_2	A_3	A_4	C
						= 3		≠ 1	= 2	1
Current solution string						Probability vector [0.38, 1] Random number = 0.52 Inserted attribute is A_4				
						Extraction				
A_1	A_2	A_3	A_4	C		A_1	A_2	A_3	A_4	C
= 3		≠ 1		1		= 3				1
						Probability vector [0.03, 1] Random number = 0.62 Removed attribute is A_3				
						Alteration				
						A_1	A_2	A_3	A_4	C
						= 1		≠ 1		1
						Randomnumber1 = 1 (the attribute that will be changed is A_1) Randomnumber2 = 2 (value change)				

alteration neighborhood, two different random numbers are generated to decide which attribute will be changed and whether the operator or the value of this attribute will be altered, respectively. For the example in Table 6, the first random number indicates that the operator or the value of attribute 1 will be changed and the second random number indicates that the value of this attribute will be randomly altered through possible values.

4.1.3. Fitness function

The fitness function of the algorithm is constructed by maximizing the product of sensitivity and specificity as in Eq. (4),

$$fit = S_e * S_p = \frac{TP}{TP + FN} * \frac{TN}{TN + FP} \quad (4)$$

where S_e and S_p denote the sensitivity and specificity measurements representing the proportion of positive and negative samples that are classified truly, respectively.

4.1.4. Generating the rule set

BEE-Miner is an algorithm that uses a multi-rule structure, operated separately for each class, so that at least one rule is obtained for each class. The rules are generated starting with the classes that have fewer samples in the training set and are collected in a rule pool. There, before a pre-processing stage is applied to the original rules in order to avoid conflicts between rules, namely rules that are exactly the same as other rules in the rule pool and those that are covered by another rule are not taken into the pool. In the formation of the rule pool, in other words to find the best rule combination covering all samples in the training set, the BEE-Miner utilizes the genetic algorithm. A gene is composed of 0 and 1 values that indicate whether the relevant rule is included in the rule pool or not, respectively. In addition, accuracy is used in calculating the fitness value of each chromosome and the chromosome length of the genetic algorithm states the number of rules in the rule pool.

After giving the details of the cost-insensitive BEE-Miner, the related pseudo-code with the notations used in the rest of the paper are presented in Table 7 and Fig. 4, respectively Table 8.

Table 7
The notations.

Ps	Population size of genetic algorithm
Cp	Crossover probability
Mp	Mutation probability
Pc	Percentage of chromosome that will be protected
maxin	Maximum iteration number of genetic algorithm
S	Number of scout bees ($s = 1,\ldots,S$)
P	Number of employed bees ($p = 1,\ldots,P$)
e	Number of best employed bees
nep	Number of onlooker bees assigned to best *e* employed bees
nsp	Number of onlooker bees assigned to remaining *P-e* employed bees ($nsp<nep$)
MaxIter	Maximum iteration number of Bees Algorithm
C	Number of classes
σ^p	Solution of *p*th employed bee
$\sigma^{neighbor}$	A neighbor solution obtained by local search
σ^{best}	Best solution
$fit(\sigma^p)$	Fitness function value of *p*th employed bee solution

4.2. Cost-sensitive BEE-Miner

The main stages of the cost-sensitive BEE-Miner are the same as those of the cost-insensitive BEE-Miner except for the following details.

4.2.1. Neighborhood structures

The neighborhood structure used in the cost-sensitive BEE-Miner is similar to the cost-insensitive case but cost information is also included in the neighborhood mechanism. According to this structure, the entropy of a database is calculated by Eqs. (5) and (6).

Table 8
The cost matrix of the Lenses dataset.

		Predicted		
		Class 1	Class 2	Class 3
Actual	Class 1	0	5	10
	Class 2	15	0	5
	Class 3	5	20	0

$$Entropy(CS) = \sum_{i=1}^{C} -A_i p_i \log_2 p_i \tag{5}$$

$$A_i = \sum_{j=1}^{C} c_{ij} \tag{6}$$

where c_{ij} is the misclassification cost of a sample of class *i* predicted as class *j*. On the other hand, the information gain of an attribute is calculated based on the rule, unlike the case of the cost-insensitive BEE-Miner. Since corresponding misclassification costs depend on actual and predicted classes, the information gain must be calculated within this context by (Eq. (7)),

$$InformationGain(CS, A, K) = Entropy(CS) - \sum_{v\in V(A)} \frac{|S_v|}{|S|} \left(\sum_{i=1}^{C} -c_{iK} p_i \log_2 p_i \right) \tag{7}$$

where *K* represents the class of the current solution string.

```
t=0
Do
    Generate S number of scout bee solutions, randomly
    Evaluate the fitness of scout bee solutions
        $fit(\sigma^s) = S_e * S_p = \frac{TP}{TP+FN} * \frac{TN}{TN+FP}$
    l=0
    Do
        Sort $_{s=1.....S}$ $fit(\sigma^s)$ in descending order and select best P solutions as employed bees
        Select best e employed bees
        Assign nep number of onlooker bees to each best e employed bees
        Assign nsp number of onlooker bees to each remaining P-e employed bees
        k=0
        Do
            For each neighbor solution obtained via entropy
            If $fit(\sigma^{neighbor}) > fit(\sigma^p)$ then $\sigma^p = \sigma^{neighbor}$
            If $fit(\sigma^{neighbor}) = fit(\sigma^p)$ then $\sigma^p = \min_{\#\ of\ rules}(\sigma^p, \sigma^{neighbor})$
            Update the best solution
            Ifmax $_{p=1....p}$ $fit(\sigma^p) > fit(\sigma^{best})$ then $\sigma^{best} = \sigma^p$
            k=k+1
        While (k<P )
        Generate S-P number of new scout bee solutions, randomly
        l=l+1
    While (l<MaxIter )
    Sort $_{s=1.....S}$ $fit(\sigma^s)$ in descending order
    t=t+1
While (t< C)
// Genetic algorithm
Generate the initial population
r=0
Do
        Generate a new population from the previous population (crossover and mutation)
        Evaluate the fitness of new population and select the best chromosome
        r=r+1
While (r<maxin)
Select the best chromosome
Simplify the obtained rules and make linguistic
```

Fig. 4. The pseudo-code of cost-insensitive BEE-Miner.

The cost-sensitive BEE-Miner uses the same neighborhood structures as the cost-insensitive case, however, since the objective is mainly to minimize the misclassification cost, the attribute is selected from among the attributes with low cost information value during insertion and from among the attributes with high cost information value during extraction according to a probabilistic choice. Moreover, after determining the attribute that will be inserted or altered, the relevant operator-value combination is again selected probabilistically depending on the misclassification cost ratio rather than selecting randomly. The mentioned misclassification cost ratio is calculated by Eq. (8) [28].

$$MCSR_{a_k,s,i} = \frac{\sum_{\forall(x,y)\in S} C_{y,i}}{\sum_{\forall(x,y)\in S} C_{y,i} + \left(\sum_{j=1}^{C} \sum_{\forall(x,y)\in S \in y=i} C_{y,i}\right) * \frac{1}{C-1}} \tag{8}$$

Let $MCSR_{a_k,s,i}$ be the misclassification cost ratio caused by classifying all the records in the sth bin of attribute a_k as class i whose real class is not actually class i, where (x, y) represents the attribute and class information of a sample in the training set, respectively. The denominator of Eq. (8) represents the potential misclassification cost caused by classifying all the records in the given interval as class i whose class is not actually i. The numerator of Eq. (8) represents the potential misclassification cost caused by not classifying all of the records in the given interval to their true class.

The insertion, extraction, and alteration neighborhood structures are again visualized on the Lenses dataset within the following cost matrix.

The entropy of the Lenses dataset is calculated by Eq. (5) as 26.48. The information gain values of attributes depending on Class 1 are calculated by Eq. (7) as 20.20, 18.81, 20.33, and 20.76, respectively. After implementation of the neighborhood structures to the current solution string, the obtained solutions are as given below.

The current solution string is composed of attributes 2 and 3. Almost similar to the cost-insensitive case, attribute 1 is selected to be added to the solution string. However, unlike to the cost-insensitive case, the operator-value combination is determined by utilizing MCSR values instead of random assignation. The probability vector is constructed in accordance with the MCSR values of the possible operator-value combinations. By reflecting the generated random number on the probability vector, ‘$A_1 = 2$’ is added to the solution string. The implementation of extraction neighborhood is the same as in the cost-insensitive case except for the calculation of information gain values. For the example in Table 9, the probability of extracting attribute 2 is higher than that of attribute 3, since the cost information (based on the class of the rule) of attribute 2 is lower than that of attribute 3. For the alteration neighborhood, the difference from the cost-insensitive case lies in again using MCSR values for determining the operator-value combination of the altered attribute.

4.2.2. Fitness function

The performance of a classifier using k number of class labels is measured by the k*k confusion matrix (CoM). The rows and columns of CoM represent actual and predicted class labels, respectively. CoM_{ij} is the number of samples that are classified as j, but are actually i. Cost-sensitive classifiers can be tested by a cost-weighted sum of misclassifications divided by the number of classified instances based on one-to-one multiplication of the confusion matrix and cost matrix [29]. For the fitness function of the cost-sensitive BEE-Miner, Eq. (9) is used according to the mentioned structure.

$$fit = \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} (CoM_{ij} c_{ij})}{\sum_{i=1}^{k} \sum_{j=1}^{k} CoM_{ij}} \tag{9}$$

However, there are some difficulties in determining the confusion matrix of the multi-class case. Suppose that the dataset contains 3 classes and the rule that will be evaluated belongs to class 1. The number of samples that satisfy both the rule part and class part of the evaluated rule will be multiplied with the C_{11} entry of the cost matrix, while the number of samples that satisfy the rule part but don’t satisfy the class part will be multiplied one-to-one with the C_{i1} entries of the cost matrix according to the actual class (i) to which the sample belongs. However, even though the actual classes of the samples that satisfy the class part but don’t satisfy the rule part of the evaluated rule are known in advance, the predicted classes of these samples are not known. A different situation arises here, since it is uncertain whether the number of this type of samples will be multiplied by C_{12} or C_{13}. In order to overcome this problem the expected cost is utilized for these samples.

$$ExpectedCost_i = \sum_{\substack{j=1 \\ i\neq j}}^{k} P(j) c_{ij} \tag{10}$$

where $P(j)$ denotes the occurrence probability of class j. By adding the value obtained by Eq. (10) to the one-to-one multiplications with the cost matrix and dividing them by the total number of samples, the adjustment to Eq. (9) is achieved. On the other hand, the samples that don’t satisfy both the rule part and class part of the evaluated rule are not considered since these samples are not classified by the evaluated rule. These samples represent TN and with the general rationality, the misclassification cost of these samples is equal to 0.

Table 9
The implementation of neighborhood structures for cost-sensitive BEE-miner.

	A_1	A_2	A_3	A_4	C
Current solution string		= 1	≠ 2		1
Insertion	= 2	= 1	≠ 2		1
	Probability vector [0.49, 1]				
	Random number = 0.12				
	Inserted attribute is A_1				
	The possible values for A_1				
	A_1=1, A_1=2, A_1=3, A_1≠1, A_1≠2, A_1≠3				
	MCSR 0.44, 0.42, 0.47, 0.22, 0.24, 0.20				
	Probability vector [0.22, 0.43, 0.67, 0.78, 0.90, 1]				
	Random number = 0.25				
	A_1 = 2				
Extraction			≠ 2		1
	Probability vector [0.52, 1]				
	Random number = 0.41				
	Removed attribute is A_2				
Alteration		≠ 1	≠ 2		1
	Randomnumber1 = 1 (the attribute that will be changed is A_2)				
	The possible values for A_2				
	A_2=2, A_2≠1, A_2≠2				
	MCSR 0.72, 0.72, 0.55				
	Probability vector [0.30, 0.60, 1]				
	Random number = 0.51				
	A_2≠1				

5. Computational study

5.1. The datasets

In order to evaluate the performance of BEE-Miner, the datasets obtained from the UCI machine learning repository are used. The characteristics of the datasets are summarized in Table 10 [30].

Table 10
The characteristics of datasets used in the evaluation of BEE-miner.

Dataset	# of samples	# of categoric attributes	# of continuous attributes	# of classes
Credit approval	690	9	6	2
Pima Indians diabetes	768	–	8	2
Echocardiogram	132	1	6	2
Heart disease (Statlog)	270	7	6	2
Hepatitis	155	13	6	2
Horse colic	368	15	7	2
Congressional voting records	435	16	–	2
Wisconsin breast cancer	699	–	9	2
Iris	150	–	4	3
Lymphography	148	18	–	4
Wine	178	–	13	3
Zoo	101	16	–	7

Table 11
The performance of different parameter sets for BEE-miner.

Dataset	Parameter set	Test accuracy	# of rules	CPU time
Credit approval	I	84.73 ± 6.01	2.80 ± 0.84	108.53 ± 13.14
	II	84.83 ± 6.11	3.43 ± 0.93	276.70 ± 14.94
	III	84.30 ± 6.43	3.66 ± 1.24	1369.93 ± 8.09
Zoo	I	92.42 ± 8.07	7.00 ± 0.00	35.46 ± 0.62
	II	91.09 ± 8.37	7.00 ± 0.00	153.73 ± 2.75
	III	90.42 ± 8.43	7.03 ± 0.18	745.46 ± 9.43
Iris	I	94.66 ± 6.16	3.86 ± 0.43	17.06 ± 0.90
	II	96.00 ± 5.42	3.90 ± 0.30	57.50 ± 1.25
	III	89.55 ± 13.17	4.13 ± 0.68	265.73 ± 24.92

Table 12
The *p* values of *t*-test for the comparison of different parameter values.

Dataset	Parameter sets (I–II)	Parameter sets (I–III)	Parameter sets (II–III)
Credit approval	0.601	**0.009**	0.069
Zoo	0.203	0.078	0.536
Iris	0.056	**0.035**	**0.006**

The selected datasets are composed of 8 binary and 4 multi-class datasets; 4 of them contain only continuous attributes, 3 of them contain only categoric attributes and 5 of them contain both types of attributes with different sizes.

5.2. BEE-Miner parameter setting

In order to determine the parameters of the algorithm, different parameter sets are generated and the set that has the best performance is selected. $\{S, P, e, nep, nsp, MaksIter\}$ = I {30,20,10,8,4,250}, II {50,40,20,10,5,500}, and III {70,50,25,20,10,1000} sets are formed by considering that the number of iterations is not greater than the number of bees in the algorithm. The representatives of the binary and multi-class datasets with different sizes are determined as credit approval, zoo, and iris. The average test accuracy, the number of rules, and CPU time in seconds are given in Table 11 with standard deviations for each dataset.

In order to compare the parameter sets statistically, each pair of parameter sets is evaluated in terms of average test accuracy. The obtained *p*-values of the *t*-test based on average test accuracy are presented in Table 12.

The *t*-test results indicate that there is a statistically significant difference between the 3 comparisons. The parameter set I outperformed set III on the credit approval and iris datasets; the parameter set II outperformed set III on the iris dataset. Consequently, parameter set I {30,20,10,8,4,250} is selected for later computational studies. Additionally the parameters of the genetic algorithm are determined as $\{ps, cp, mp, pc, maxin\} = \{40, 0.8, 0.2, 0.2, 100\}$.

On the other hand, cost matrices are generated within the algorithm for each dataset specifically. Because of the absence of cost matrices for all of the datasets in the literature, the cost matrix generation technique of MetaCost [31], one of the studies used in the comparisons, is utilized. According to this technique, the c_{ij} values are calculated by $1000P(j)/P(i)$ and the c_{ii} values are determined as 0 for the multi-class case, where $P(i)$ and $P(j)$ denote the occurrence probability of class i and j in the training set. Hence, high cost values will emerge in the misclassification of the minority class. This situation also reflects the real life problems and reveals the importance of correct classification of the minority classes. On the other hand, for the binary class case, the misclassification cost values are determined as $c_{11} = c_{22} = 0$, $c_{12} = 5000$, $c_{21} = 1000$ where 1 and 2 denote the minority and majority classes, respectively.

5.3. Comparison results for cost-insensitive BEE-Miner

The selected datasets are executed 30 times by using 10-fold cross validation and the cost-insensitive BEE-Miner is compared with the results in the literature in terms of average test accuracy and average number of rules as presented in Table 13. However, since the horse colic dataset has an original distinction in the training and test sets, this dataset does not use 10-fold cross validation. Also, in order to establish a correlation between test accuracy and the number of rules, the Minimum Deviation Method (MDM) is used as in Eq. (11) and the comparisons are based on this calculation [32]. Moreover, the missing values in the datasets are replaced with the average value of the class to which they belong for continuous attributes and with the frequent value of the class to which they belong for categoric attributes.

$$MDM_i = \frac{test\ accuracy_{max} - test\ accuracy_i}{test\ accuracy_{max} - test\ accuracy_{min}} + \frac{\#\ of\ rules_i - \#\ of\ rules_{min}}{\#\ of\ rules_{max} - \#\ of\ rules_{min}} \tag{11}$$

According to the results summarized in Table 13, the cost-insensitive BEE-Miner has an above-average performance in terms of test accuracy and the number of rules. In addition to less test accuracy being yielded for some of the datasets, comparable results are obtained with fewer rules. Even though the ranking on MDM is low, the test accuracy is close to the other results and the number of rules is much better than the compared results, such as for the credit approval dataset. Furthermore, the test accuracy values obtained by BEE-Miner are better than the results that have a similar number of rules, such as for the echocardiogram dataset. It is a fact that test accuracy is more crucial than the number of rules. However, obtaining a solution that has similar test accuracy with a lower number of rules is also important. Among the compared studies, BEE-Miner is the best for the wine, hepatitis, horse colic, and voting datasets. On the other hand, for the lymphography and zoo datasets, BEE-Miner ranked as 2 and 4 among 13 studies.

Moreover, BEE-Miner achieves the first rank for the horse colic, hepatitis, voting, and wine datasets which have the highest number of attributes. For the lymphography and zoo datasets, which also have a high number of attributes, BEE-Miner achieves the second and fourth rank, respectively. Therefore, it can be concluded that the performance of the cost-insensitive BEE-Miner is more effective on datasets with a high number of attributes.

5.4. Comparison results for cost-sensitive BEE-Miner

The same benchmark problems are executed with the cost-sensitive BEE-Miner for 30 times by using 10-fold cross

Table 13
The comparison results of cost-insensitive BEE-miner.

	Compared algorithms	Test accuracy	# of rules	MDM
Credit approval	cAnt-Miner [33]	85.30 ± 0.93	7.07 ± 0.19	0.74 (3)
	μcAnt-Miner [34]	86.70 ± 0.84	17.42 ± 0.31	0.50 (1)
	Jrip [35]	85.51 ± 1.46	4.10 ± 0.64	0.55 (2)
	PART [36]	84.35 ± 1.08	31.90 ± 2.93	2.00 (7)
	ATM [37]	85.90 ± 0.10	29.60 ± 0.20	1.26 (6)
	cAnt-MinerPB [38]	85.70 ± 0.10	12.30 ± 0.10	0.75 (4)
	BEE-Miner	84.87 ± 6.47	2.66 ± 0.92	0.77 (5)
Pima Indians diabetes	Ant Miner [39]	70.99	12.20	0.97 (5)
	Ant Miner with Imp. Quick Red. [39]	76.58	11.30	0.74 (3)
	RCA [40]	81.00	153.00	1.06 (7)
	EBA [41]	76.00	285.00	1.73 (10)
	GA [42]	73.00	228.00	1.65 (9)
	Johnson [42]	80.00	81.00	0.84 (4)
	Holte 1R [42]	71.00	42.00	1.07 (8)
	EDGAR [43]	94.00 ± 0.17	128.00 ± 9.34	0.44 (2)
	REGAL [44]	94.00 ± 0.17	127.00 ± 9.49	0.43 (1)
	BEE-Miner	69.53 ± 5.53	2.96 ± 1.06	1.00 (6)
Echocardiogram	RCA [40]	92.00	143.00	0.60 (5)
	EBA [41]	91.00	376.00	1.26 (11)
	GA [42]	92.00	241.00	0.87 (8)
	Johnson [42]	91.00	19.00	0.30 (3)
	Holte1R [42]	100.00	44.00	0.11 (1)
	NBTree [45]	67.45	2.30	0.94 (9)
	Decision Table [46]	65.48	3.30	1.00 (10)
	PART [36]	71.39	4.00	0.83 (6)
	C4.5 [47]	70.17	3.20	0.86 (7)
	Rule extraction from adaptive ANN using AIS [48]	94.59	24.00	0.21 (2)
	BEE-Miner	84.67 ± 9.68	2.46 ± 0.73	0.44 (4)
Heart disease (Statlog)	ATM [37]	77.30 ± 0.30	18.10 ± 0.20	1.26 (7)
	cAnt-MinerPB [38]	77.00 ± 0.40	8.60 ± 0.10	0.99 (3)
	C4.5 [47]	81.29	2.00	0.00 (1)
	RIPPER [35]	78.23	2.00	0.54 (2)
	CBA [49]	79.20	22.00	1.06 (6)
	MCAR [50]	81.14	31.00	1.02 (4)
	BEE-Miner	75.67 ± 7.06	3.56 ± 0.56	1.05 (5)
Hepatitis	cAnt-Miner [33]	76.84 ± 3.13	4.91 ± 0.11	0.83 (10)
	μcAnt-Miner [34]	80.27 ± 2.04	7.80 ± 0.36	0.72 (6)
	Jrip [35]	78.13 ± 2.66	2.70 ± 0.21	0.74 (7)
	PART [36]	83.25 ± 3.47	8.40 ± 0.34	0.59 (2)
	ATM [37]	82.50 ± 0.30	7.50 ± 0.10	0.61 (3)
	cAnt-MinerPB [38]	80.10 ± 0.50	7.60 ± 0.10	0.72 (5)
	C4.5Rules [47]	79.33 ± 9.47	6.30 ± 1.42	0.74 (8)
	RIPPER [35]	72.21 ± 14.18	7.90 ± 1.73	1.08 (13)
	BNGE [51]	82.54 ± 4.55	37.90 ± 2.81	1.07 (12)
	RISE [52]	83.88 ± 6.95	50.80 ± 3.49	1.20 (14)
	INNER [53]	79.38 ± 2.38	14.70 ± 1.95	0.86 (11)
	SIA [54]	76.88 ± 9.47	69.40 ± 6.90	1.79 (15)
	EHS-CHC [55]	79.38 ± 2.38	7.50 ± 1.35	0.75 (9)
	Filtered EHS-CHC [55]	79.38 ± 2.38	4.40 ± 1.26	0.71 (4)
	BEE-Miner	94.62 ± 5.35	2.23 ± 0.43	0.00 (1)
Horse colic	cAnt-Miner [33]	80.45 ± 2.58	7.27 ± 0.21	0.91 (3)
	μcAnt-Miner [34]	70.23 ± 2.16	14.57 ± 0.82	2.00 (7)
	Jrip [35]	83.54 ± 1.87	3.70 ± 0.34	0.46 (2)
	PART [36]	82.39 ± 2.10	9.60 ± 0.45	1.02 (5)
	ATM [37]	83.80 ± 0.10	10.20 ± 0.10	1.01 (4)
	cAnt-MinerPB [38]	83.50 ± 0.30	10.80 ± 0.20	1.08 (6)
	BEE-Miner	92.66 ± 3.97	3.06 ± 0.52	0.00 (1)
Voting	ATM [37]	94.90 ± 0.10	6.10 ± 0.00	0.74 (3)
	cAnt-MinerPB [38]	93.30 ± 0.20	6.40 ± 0.10	1.19 (5)
	EDGAR [43]	97.00 ± 0.21	24.00 ± 7.13	1.00 (4)
	REGAL [44]	96.00 ± 4.42	8.00 ± 9.36	0.53 (2)
	BEE-Miner	96.45 ± 3.16	2.16 ± 0.37	0.14 (1)
Wisconsin breast cancer	EDGAR [43]	98.00 ± 0.56	25.00 ± 4.10	1.30 (14)
	REGAL [44]	100.00 ± 0.26	25.00 ± 3.63	1.00 (12)
	MPPSONf1f2 [56]	97.66 ± 1.66	5.80 ± 2.50	0.52 (6)
	MPPSONf1-f3 [56]	97.81 ± 1.24	5.10 ± 1.50	0.46 (4)
	MEPGANf1f2 [56]	96.78 ± 1.49	6.60 ± 3.60	0.68 (10)
	MEPGANf1-f3 [56]	97.80 ± 1.23	5.40 ± 2.80	0.48 (5)
	MEPDENf1f2 [56]	96.93 ± 1.86	3.40 ± 1.30	0.52 (7)
	MEPDENf1-f3 [56]	97.66 ± 1.01	3.80 ± 0.90	0.43 (3)

(continued on next page)

Table 13 (*continued*)

	Compared algorithms	Test accuracy	# of rules	MDM
	MPANN [57]	98.10	4.10	0.38 (2)
	HMOEN_L2 [58]	96.26	4.70	0.68 (11)
	HMOEN_HN [58]	96.82	4.80	0.60 (9)
	MSCC [59]	97.60	2.00	0.36 (1)
	RBFN-TVMOPSO [60]	96.53	2.00	0.52 (8)
	BEE-Miner	93.42 ± 2.96	3.66 ± 0.88	1.07 (13)
Iris	cAnt-Miner [33]	94.21 ± 0.99	4.00 ± 0.00	0.55 (8)
	μcAnt-Miner [34]	95.65 ± 3.27	8.40 ± 0.61	0.52 (7)
	Jrip [35]	93.50 ± 3.84	3.58 ± 0.31	0.62 (10)
	PART [36]	93.02 ± 3.55	3.79 ± 1.20	0.68 (13)
	ATM [37]	96.20 ± 0.10	4.20 ± 0.00	0.33 (4)
	cAnt-MinerPB [38]	93.20 ± 0.20	4.80 ± 0.00	0.69 (14)
	C4.5Rules [47]	96.67 ± 4.71	5.00 ± 0.00	0.30 (3)
	RIPPER [35]	96.00 ± 3.44	6.20 ± 0.79	0.41 (5)
	BNGE [51]	96.00 ± 4.66	12.20 ± 1.32	0.59 (9)
	RISE [52]	94.00 ± 4.92	37.10 ± 11.21	1.56 (17)
	INNER [53]	96.00 ± 3.44	8.20 ± 1.23	0.47 (6)
	SIA [54]	94.67 ± 4.22	13.90 ± 1.79	0.80 (15)
	EHS-CHC [55]	94.00 ± 3.78	6.20 ± 0.79	0.64 (11)
	Filtered EHS-CHC [55]	96.67 ± 4.71	5.00 ± 0.00	0.30 (2)
	EDGAR [43]	96.00 ± 0.38	14.00 ± 5.55	0.65 (12)
	REGAL [44]	99.00 ± 0.17	11.00 ± 2.55	0.22 (1)
	BEE-Miner	90.22 ± 7.57	5.76 ± 1.07	1.06 (16)
Lymphography	DEREx [61]	80.79 ± 1.66	5.00	0.17 (3)
	OneR [62]	75.41 ± 0.57	4.00	0.36 (4)
	PART [36]	85.14 ± 0.00	12.00	0.10 (1)
	Ridor [63]	77.84 ± 3.77	14.00	0.40 (6)
	C4.5Rules [47]	74.27 ± 11.75	11.50 ± 1.18	0.50 (9)
	RIPPER [35]	75.80 ± 8.97	13.60 ± 1.65	0.47 (8)
	BNGE [51]	80.06 ± 9.66	29.20 ± 2.35	0.51 (10)
	RISE [52]	76.12 ± 9.93	81.00 ± 10.35	1.33 (13)
	INNER [53]	58.26 ± 14.65	8.30 ± 1.83	1.05 (12)
	SIA [54]	81.30 ± 7.52	33.30 ± 1.70	0.52 (11)
	EHS-CHC [55]	77.33 ± 9.11	13.40 ± 1.35	0.41 (7)
	Filtered EHS-CHC [55]	75.77 ± 9.49	7.70 ± 1.57	0.39 (5)
	BEE-Miner	80.82 ± 9.28	5.23 ± 0.67	0.17 (2)
Wine	cAnt-Miner [33]	91.38 ± 1.72	4.01 ± 0.01	0.51 (15)
	μcAnt-Miner [34]	93.82 ± 1.69	4.07 ± 0.25	0.29 (7)
	Jrip [35]	92.19 ± 2.22	4.00 ± 0.15	0.43 (13)
	PART [36]	92.75 ± 1.44	4.60 ± 0.16	0.39 (11)
	ATM [37]	96.20 ± 0.10	5.60 ± 0.00	0.08 (2)
	cAnt-MinerPB [38]	93.60 ± 0.30	5.40 ± 0.10	0.32 (8)
	C4.5Rules [47]	94.90 ± 6.19	5.00 ± 0.00	0.20 (5)
	RIPPER [35]	92.16 ± 5.34	5.60 ± 1.07	0.45 (14)
	BNGE [51]	96.60 ± 2.93	10.10 ± 0.99	0.08 (3)
	RISE [52]	94.38 ± 5.24	29.70 ± 10.78	0.40 (12)
	INNER [53]	85.92 ± 6.18	6.20 ± 1.03	1.01 (16)
	SIA [54]	94.38 ± 5.24	160.20 ± 0.42	1.23 (17)
	EHS-CHC [55]	94.31 ± 6.61	9.80 ± 1.81	0.28 (6)
	Filtered EHS-CHC [55]	95.42 ± 5.38	7.60 ± 0.97	0.17 (4)
	EDGAR [43]	97.00 ± 0.31	57.00 ± 13.51	0.34 (9)
	REGAL [44]	97.00 ± 0.31	60.00 ± 14.52	0.36 (10)
	BEE-Miner	96.08 ± 4.42	3.13 ± 0.34	0.08 (1)
Zoo	ATM [37]	88.20 ± 0.20	10.70 ± 0.00	0.51 (9)
	cAnt-MinerPB [38]	78.60 ± 0.90	6.00 ± 0.00	0.57 (10)
	C4.5Rules [47]	92.81 ± 6.92	8.70 ± 0.48	0.26 (7)
	RIPPER [35]	94.08 ± 7.03	8.70 ± 0.48	0.22 (5)
	BNGE [51]	96.83 ± 5.24	9.00 ± 0.47	0.15 (2)
	RISE [52]	96.83 ± 5.24	25.60 ± 3.75	1.00 (11)
	INNER [53]	87.61 ± 6.64	8.90 ± 0.88	0.43 (8)
	SIA [54]	96.17 ± 6.85	10.10 ± 0.74	0.22 (6)
	EHS-CHC [55]	95.00 ± 6.71	7.00 ± 0.00	0.10 (1)
	Filtered EHS-CHC [55]	91.31 ± 8.02	6.30 ± 0.67	0.18 (3)
	EDGAR [43]	69.00 ± 3.14	9.00 ± 5.035	1.02 (13)
	REGAL [44]	65.00 ± 3.11	6.00 ± 4.097	1.00 (12)
	BEE-Miner	91.42 ± 8.38	7.00 ± 0.00	0.22 (4)

validation and the results are summarized in Table 14, which represents the average and standard deviations of test accuracy, the number of rules, and misclassification cost. The computational results of the cost-sensitive BEE-Miner are compared with 2 popular cost-sensitive meta-learning algorithms, namely MetaCost and CostSensitiveClassifier, in WEKA 3.6.11 data mining software with the same training set, testing set and cost matrices. While executing the compared meta-learning algorithms, 5 cost-insensitive classifiers are taken into consideration. These algorithms are PART, C4.5, SimpleCART, RIPPER and DecisionTable. However, since the

Table 14
The comparison results of cost-sensitive BEE-Miner.

Datasets			Credit approval	Pima Indians diabetes	Echocardiogram	Heart disease (Statlog)	Hepatitis	Horse colic	Congressional voting records	Wisconsin breast cancer	Iris	Lymphography	Wine	Zoo
MetaCost	DecisionTable	Test acc.	83.91	73.56	68.91	76.66	80.64	81.60	94.94	94.13	93.33	75.67	84.83	60.39
		# of rules	18	32	1	44	1	49	3	34	8	49	34	2
		Cost	208.69	294.27	337.83	288.88	200.00	224.00	80.45	71.53	66.66	193.43	141.38	90.31
	PART	Test acc.	83.91	76.04	70.27	81.48	81.29	80.80	96.32	95.13	93.33	78.37	91.57	83.16
		# of rules	23	9	3	17	7	3	9	11	4	8	4	6
		Cost	221.73	298.17	418.91	259.25	258.06	248.00	55.17	77.25	66.66	157.54	81.50	144.41
	C4.5	Test acc.	85.65	74.47	71.62	79.25	81.93	80.00	95.86	94.84	92.66	80.40	91.01	82.17
		# of rules	15	22	7	23	4	5	7	11	5	20	5	6
		Cost	194.20	324.21	351.35	270.37	200.00	288.00	62.06	75.82	73.33	157.64	86.08	150.60
	SimpleCART	Test acc.	85.07	75.39	68.91	80.00	81.29	81.60	95.63	94.56	94.66	81.75	88.20	83.16
		# of rules	4	7	7	8	3	20	6	9	5	3	8	9
		Cost	215.94	304.68	405.40	259.25	212.90	240.00	62.06	77.25	53.33	131.99	123.52	88.17
	RIPPER	Test acc.	85.50	74.86	62.16	80.00	82.58	82.40	95.63	95.56	94.66	74.32	91.01	87.12
		# of rules	6	5	3	4	3	3	2	4	4	6	4	5
		Cost	195.65	289.06	486.48	266.66	200.00	240.00	73.56	67.23	53.33	199.83	86.80	33.17
CostSensitiveClassifier	DecisionTable	Test acc.	85.50	73.17	67.56	78.88	74.19	79.20	94.02	94.13	93.33	74.32	91.57	91.08
		# of rules	38	8	7	12	28	62	68	23	3	14	18	10
		Cost	186.95	303.38	364.86	274.07	335.48	256.00	80.45	77.25	66.66	203.33	81.36	71.32
	PART	Test acc.	84.63	74.73	62.16	80.37	80.64	82.40	94.48	94.56	94.66	81.08	89.88	82.17
		# of rules	18	6	4	9	7	4	9	13	3	10	5	5
		Cost	192.75	**289.06**	527.02	251.85	219.35	232.00	73.56	75.82	53.33	168.25	91.33	117.02
	C4.5	Test acc.	84.92	74.08	62.16	76.66	83.22	82.40	96.78	94.84	95.33	78.37	89.88	82.17
		# of rules	8	6	6	19	3	4	5	8	5	11	9	5
		Cost	191.30	309.89	513.51	296.29	187.09	232.00	45.97	74.39	46.66	173.07	98.04	117.02
	SimpleCART	Test acc.	82.75	71.87	67.56	78.51	81.29	76.00	92.41	94.99	94.00	70.27	85.39	85.14
		# of rules	23	43	11	9	1	13	10	15	5	13	9	5
		Cost	223.18	367.18	405.40	262.96	225.80	320.00	105.74	70.10	60.00	269.31	138.12	127.81
	RIPPER	Test acc.	85.07	73.43	64.86	78.14	80.64	82.40	95.86	95.85	95.33	78.37	90.44	83.16
		# of rules	4	3	1	4	2	2	3	6	3	4	5	4
		Cost	189.85	325.52	445.94	259.25	219.35	232.00	68.96	67.23	46.66	153.54	91.61	51.91
Cost-sensitive BEE-Miner		Test acc.	86.67 ±	79.03 ± 4.63	87.50 ±	84.07 ±	98.04 ±	94.64 ±	98.14 ±	98.27 ±	96.66 ±	89.19 ±	99.44 ±	99.00 ±
		Test acc.	6.72	4.63	12.47	5.25	3.15	1.15	2.13	1.89	4.71	5.64	1.75	3.16
		# of rules	4.10 ± 1.10	6.20 ± 1.68	6.10 ± 0.73	3.90 ± 1.19	4.30 ± 1.15	3.33 ± 0.66	2.10 ± 0.31	5.60 ± 1.17	5.30 ± 0.82	5.00 ± 0.00	3.00 ± 0.00	8.60 ± 0.51
		Cost	**181.15** ± 91.97	290.90 ± 82.41	**84.61** ± 76.49	**218.51** ± 74.99	**25.00** ± 43.70	**88.26** ± 23.53	**29.54** ± 41.56	**25.71** ± 29.19	**33.33** ± 47.14	**84.39** ± 66.07	**6.68** ± 21.14	**1.21** ± 3.85

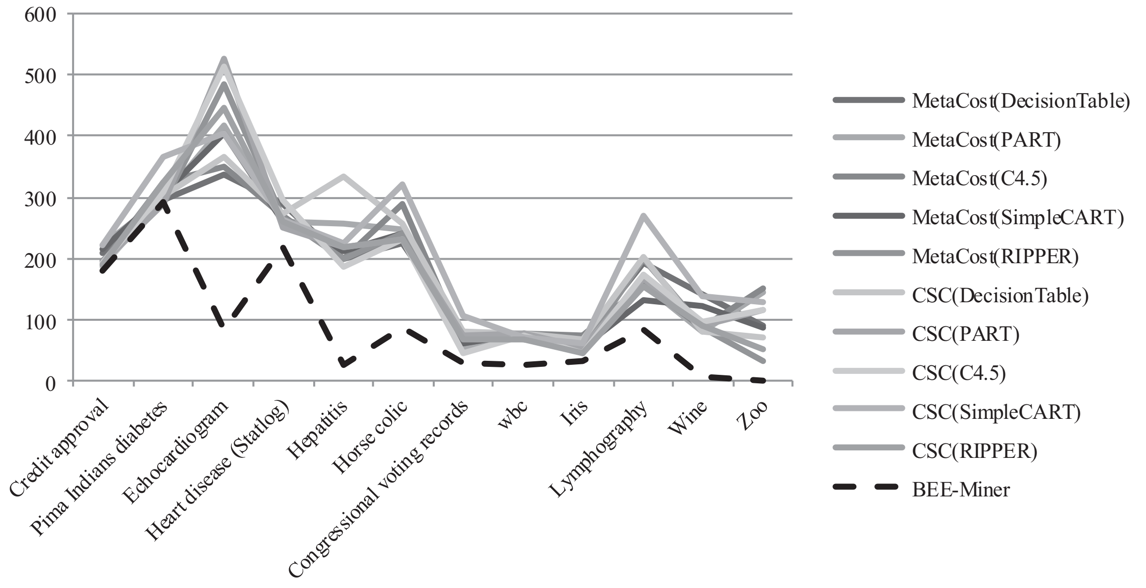

Fig. 5. The demonstration of misclassification cost values of each algorithm.

Table 15
The effect of cost-based mechanisms on cost-sensitive BEE-miner.

	Cost-insensitive BEE-Miner with cost-based neighborhood	Cost-insensitive BEE-Miner with cost-based fitness function	Cost-sensitive BEE-Miner
Credit approval	255.07 ± 127.80	244.44 ± 125.87	242.99 ± 121.53
Pima Indians diabetes	467.96 ± 117.72	357.57 ± 85.24	355.84 ± 90.40
Echocardiogram	289.74 ± 154.74	266.66 ± 156.12	200.00 ± 125.50
Heart disease (Statlog)	450.61 ± 155.34	343.20 ± 109.14	316.04 ± 112.89
Hepatitis	243.75 ± 171.63	120.83 ± 97.00	83.33 ± 88.89
Horse colic	262.40 ± 52.11	93.86 ± 20.89	88.26 ± 23.53
Congressional voting records	111.36 ± 60.67	81.06 ± 69.26	81.06 ± 66.64
Wisconsin breast cancer	89.04 ± 44.81	87.61 ± 49.73	81.42 ± 60.27
Iris	77.77 ± 70.21	77.77 ± 70.21	75.55 ± 69.44
Lymphography	338.80 ± 231.11	335.53 ± 511.41	178.48 ± 120.08
Wine	121.64 ± 102.65	42.26 ± 52.42	35.58 ± 44.44
Zoo	117.58 ± 199.85	91.12 ± 189.45	45.82 ± 42.69

horse colic dataset has an original distinction in the training and test sets, this dataset does not use 10-fold cross validation. Similar to the cost-insensitive case, the missing values are replaced with the average value of the class to which they belong for continuous attributes and the frequent value of the class to which they belong for categoric attributes.

The cost-sensitive BEE-Miner yields lower misclassification cost values for 11 out of 12 of the datasets which are marked in bold in Table 14. Also for the remaining dataset (Pima Indians diabetes) the BEE-Miner attained a comparable result with only 0.6% increase in misclassification cost. The reason behind the high standard deviations of test accuracy and especially misclassification cost values is due to the variation in the test and training sets of each fold. Therefore, it can be concluded that the cost-sensitive BEE-Miner algorithm attains high test accuracy with low misclassification cost. Furthermore, the effect of the neighborhood structure and fitness function on the proposed BEE-Miner algorithm is also obvious on the obtained results. The misclassification cost values of each algorithm are also illustrated in Fig. 5.

The performance of the BEE-Miner is also analyzed considering the attribute numbers of datasets. In this respect, the average of the difference between the misclassification cost value of the BEE-Miner and compared algorithms is calculated. It is observed that 82% of the decrease in misclassification cost value belongs to the datasets with more than 15 attributes. The average of the difference between the test accuracy value of the BEE-Miner and the compared algorithms is also calculated and it is also observed that 58% of the increase in test accuracy value belongs to the datasets with more than 15 attributes. Therefore, it can be concluded that the performance of the cost-sensitive BEE-Miner is more effective on datasets with a high number of attributes.

5.5. The effect of cost-based components on cost-sensitive BEE-Miner

The effect of the cost-based neighborhood mechanism and cost-based fitness function on the cost-sensitive BEE-Miner is also analyzed. The average and standard deviation values of misclassification costs under 30 executions for the selected datasets, which represent the binary and multi-class datasets with different sizes similar to the determination of the parameter values stage, are summarized in Table 15.

It can be generally concluded that the effectiveness of the cost-sensitive BEE-Miner relies mainly on the cost-based fitness function, because the single-handed cost-based neighborhood mechanism obtained higher misclassification costs than the single-handed cost-based fitness function mechanism. However, because the cost-sensitive BEE-Miner includes both the cost-based neighborhood and cost-based fitness function it has superior performance, as expected.

6. Conclusion

In this study, a cost-sensitive classification algorithm is introduced. The key feature of the BEE-Miner is that it is a direct method rather than a meta-learning method, namely it incorporates the cost component into the operation principles of the algorithm. Moreover, the developed algorithm is also capable of classifying both binary and multi-class datasets. The effect of the cost-sensitive BEE-Miner including misclassification cost to both the neighborhood structures and fitness function is also provided. The extensive computational study results indicate that the BEE-Miner is able to classify datasets with high accuracy and low cost. For future studies, a multi-objective approach that aims to maximize accuracy, minimize misclassification cost and the number of rules at the same time will be beneficial to obtain more significant results.

Acknowledgment

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under Project 111M528.

References

[1] C. Elkan, The foundations of cost-sensitive learning, in: Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, WA, 2001, pp. 973–978.
[2] B. Zadrozny, C. Elkan, Learning and making decisions when costs and probabilities are both unknown, in: Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining, 2001, pp. 204–213.
[3] P.D. Turney, Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm, J. Artif. Intell. Res. 2 (1995) 369–409.
[4] C. Drummond, R. Holte, Exploiting the cost (in)sensitivity of decision tree splitting criteria, in: Proceedings of the 17th International Conference on Machine Learning, 2000, pp. 239–246.
[5] C.X. Ling, Q. Yang, J. Wang, S. Zhang, Decision trees with minimal cost, in: Proceedings of the 21th International Conference on Machine Learning, 2004.
[6] C.X. Ling, V.S. Sheng, T. Bruckhaus, N.H. Madhavji, Maximum profit mining and its application in software development, in: Proceedings of the 12th International Conference on Knowledge Discovery and Data Mining, 2006, p. 929.
[7] C.X. Ling, V.S. Sheng, Q. Yang, Test strategies for cost-sensitive decision trees, IEEE Trans. Knowl. Data Eng. 18 (8) (2006) 1055–1067.
[8] Z. Qin, S. Zhang, C. Zhang, Cost-sensitive decision trees with multiple cost scales, in: Proceedings of the 17th Australian Joint Conference on Artificial Intelligence, 2004, pp. 380–390.
[9] S. Sheng, A. Ling, Hybrid cost-sensitive decision tree, in: Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, in: LNCS, 3721, 2005, pp. 274–284.
[10] W. Fan, S.J. Stolfo, J. Zhang, P.K. Chan, AdaCost: misclassification cost-sensitive boosting, in: Proceedings of the 16th International Conference on Machine Learning, 1999, pp. 97–105.
[11] D.D. Margineantu, Building ensembles of classifiers for loss minimization, in: Proceedings of the 31st Symposium on the Interface, 1999, pp. 190–194.
[12] H. Tu, Regression Approaches for Multi-Class Cost-Sensitive Classification, National Taiwan University, 2009 M.S. thesis.
[13] H. Tu, H.T. Lin, One-sided support vector regression for multiclass cost-sensitive classification, in: Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 1095–1102.
[14] J. Li, X. Li, X. Yao, Cost-sensitive classification with genetic programming, IEEE Congr. Evol. Comput. 3 (2005) 2114–2121.
[15] S. Ashfaq, M.U. Farooq, A. Karim, Efficient rule generation for cost-sensitive misuse detection using genetic algorithms, Int. Conf. Comput. Intell. Secur. (2006) 282–285.
[16] D.L. Song, B. Yang, Z. Peng, W. Fang, Study of cost-sensitive ant colony data mining algorithm, Int. Conf. Ind. Mechatron. Automat. (2009) 488–491.
[17] D.T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, M. Zaidi, The Bees Algorithm – a novel tool for complex optimisation problems, in: Proceedings of the Innovative Production Machines and Systems Virtual Conference, 2006, pp. 454–461.
[18] L. Özbakır, A. Baykasoğlu, P. Tapkan, Bees algorithm for generalized assignment problem, Appl. Math. Comput. 215 (11) (2010) 3782–3795.
[19] L. Özbakır, P. Tapkan, Bee colony intelligence in zone constrained two-sided assembly line balancing problem, Expert Syst. Appl. 38 (2011) 11947–11957.
[20] P. Tapkan, L. Özbakır, A. Baykasoğlu, Modeling and solving constrained two-sided assembly line balancing problem via bee algorithms, Appl. Soft Comput. 12 (2012) 3343–3355.
[21] J. Catlett, On changing continuous attributes into ordered discrete attributes, in: Proceedings of the European Working Session on Learning, 1991, pp. 164–178.
[22] Y. Yang, G.I. Webb, Discretization for Naïve–Bayes learning: managing discretization bias and variance, Mach. Learn. 74 (2009) 39–74.
[23] U.M. Fayyad, K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: Proceedings of the 13th International Joint Conference on Artificial Intelligence, 1993, pp. 1022–1027.
[24] R. Kerber, Chimerge: discretization for numeric attributes, Nat. Conf. Artif. Intell. (1992) 123–128.
[25] A.A. Freitas, S.H. Lavington, Speeding up knowledge discovery in large relational databases by means of a new discretization algorithm, in: Proceedings of the 14th British National Conference on Databases, 1996, pp. 124–133.
[26] Y. Yang, G.I. Webb, X. Wu, Discretization methods, Data Mining and Knowledge Discovery Handbook, 2nd ed., Springer-Verlag, New York, 2010 pp. 101-116.
[27] T. Mitchell, Machine Learning, McGraw Hill, New York, USA, 1997.
[28] Y. Weiss, Y. Elovici, L. Rokach, The CASH algorithm-cost-sensitive attribute selection using histograms, Inf. Sci. 222 (2013) 247–268.
[29] T. Pietraszek, Alert Classification to Reduce False Positives in Intrusion Detection, Comput. Sci, Univ. of Freiburg, Germany, 2006 Ph.D. dissertation.
[30] http://archive.ics.uci.edu/ml/datasets.html, last accessed date June 04. 2015.
[31] P. Domingos, MetaCost: a general method for making classifiers cost-sensitive, in: Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining, 1999, pp. 155–164.
[32] M.T. Tabucanon, Multiple Criteria Based Decision Making in Industry, Elsevier, New York, USA, 1988.
[33] F. Otero, A. Freitas, C.G. Johnson, cAnt-Miner: an ant colony classification algorithm to cope with continuous attributes, LNCS 5217 (2008) 48–59.
[34] K.M. Salama, A.M. Abdelbar, F.E.B. Otero, A.A. Freitas, Utilizing multiple pheromones in an ant-based algorithm for continuous attribute classification rule discovery, Appl. Soft Comput. 13 (1) (2013) 667–675.
[35] W. Cohen, Fast effective rule induction, in: Proceedings of the 12th International Conference On Machine Learning, 1995, pp. 115–123.
[36] E. Frank, I.H. Witten, Generating accurate rule sets without global optimization, in: Proceedings of the 15th International Conference On Machine Learning, 1998, pp. 144–151.
[37] F.E.B. Otero, A.A. Freitas, C.G. Johnson, Inducing decision trees with an ant colony optimization algorithm, Appl. Soft Comput. 12 (11) (2012) 3615–3626.
[38] F.E.B. Otero, A.A. Freitas, C.G. Johnson, A new sequential covering strategy for inducing classification rules with ant colony algorithms, IEEE Trans. Evol. Comput. 17 (1) (2013) 64–76.
[39] P. Jaganathan, K. Thangavel, A. Pethalakshmi, M. Karnan, Classification rule discovery with ant colony optimization and improved quick reduct algorithm, Int. J. Comput. Sci. 33 (1) (2007) 286–291.
[40] A.A. Bakar, Z.A. Othman, A.R. Hamdan, R. Yusof, R. Ismail, An agent model for rough classifiers, Appl. Soft Comput. 11 (2) (2011) 2239–2245.
[41] Y. Hassan, E. Tazaki, Emergent rough set data analysis, LNCS 3135 (2006) 343–361.
[42] A. Ohrn, J. Komorowski, ROSETTA-a rough set toolkit for analysis of data, in: Proceedings of the 5th International Workshop on Rough Sets and Soft Computing, 1997, pp. 403–407.
[43] M. Rodriguez, D.M. Escalante, A. Peregrin, Efficient distributed genetic algorithm for rule extraction, Appl. Soft. Comput. 11 (1) (2011) 733–743.
[44] A. Giordana, F. Neri, Search-intensive concept induction, Evol. Comput. 3 (4) (1995) 375–416.
[45] G.H. John, P. Langley, Estimating continuous distributions in Bayesian classifiers, in: Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, 1995, pp. 338–345.
[46] R. Kohavi, The power of decision tables, in: Proceedings of the 8th European Conference on Machine Learning, 1995, pp. 174–189.
[47] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Francisco, USA, 1993.
[48] H. Kahramanlı, N. Allahverdi, Rule extraction from trained adaptive neural networks using artificial immune systems, Expert Syst. Appl. 36 (2009) 1513–1522.
[49] B. Liu, W. Hsu, Y. Ma, Integrating classification and association rule mining, in: Proceedings of the Knowledge Discovery and Data Mining, 1998, pp. 80–86.
[50] F. Thabtah, P. Cowling, S. Hammoud, Improving rule sorting, predictive accuracy and training time in associative classification, Expert Syst. Appl. 31 (2) (2006) 414–426.
[51] D. Wettschereck, T.G. Dietterich, An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms, Mach. Learn. 19 (1995) 5–27.
[52] P. Domingos, Unifying instance-based and rule-based induction, Mach. Learn. 24 (1996) 141–168.
[53] O. Luaces, A. Bahamonde, Inflating examples to obtain rules, international examples to obtain rules, Int. J. Intell. Syst. 18 (11) (2003) 1113–1143.
[54] G.S.I.A. Venturing, A supervised inductive algorithm with genetic search for learning attributes based concepts, LNCS 667 (1993) 280–296.
[55] S. Garcia, J. Derrac, J. Luengo, C.J. Carmona, F. Herrera, Evolutionary selection of hyperrectangles in nested generalized exemplar learning, Appl. Soft Comput. 11 (3) (2011) 3032–3045.
[56] S.N. Qasem, S.M. Shamsuddin, A.M. Zain, Multi-objective hybrid evolutionary algorithms for radial basis function neural network design, Knowl.-Based Syst. 27 (2012) 475–497.
[57] H.A. Abbass, Speed up backpropagation using multi-objective evolutionary algorithms, Neural Comput. 15 (11) (2003) 2705–2726.

[58] C. Goh, E. Teoh, K.C. Tan, Hybrid multi-objective evolutionary design for artificial neural networks, IEEE Trans. Neural Netw 19 (9) (2008) 1531–1548.

[59] W. Cai, S. Chen, D. Zhang, A multi-objective simultaneous learning framework for clustering and classification, IEEE Trans. Neural Netw 21 (2) (2010) 185–200.

[60] S.N. Qasem, S.M. Shamsuddin, Radial basis function network based on time variant multi-objective particle swarm optimization for medical diseases diagnosis, Appl. Soft Comput. 11 (1) (2011) 1427–1438.

[61] I. De Falco, Differential evolution for automatic rule extraction from medical databases, Appl. Soft Comput. 13 (2) (2013) 1265–1283.

[62] R.C. Holte, Very simple classification rules perform well on most commonly used datasets, Mach. Learn. 11 (1993) 63–91.

[63] P. Compton, R. Jansen, Knowledge in context: a strategy for expert system maintenance, LNCS 406 (1990) 292–306.