Fraud Detection in Financial Statements Using Evolutionary Computation Based Rule Miners

Ganghishetti Pradeep[1,2], Vadlamani Ravi[1(✉)], Kaushik Nandan[3], B.L. Deekshatulu[1], Indranil Bose[4], and A. Aditya[1]

[1] Center of Excellence in CRM and Analytics,
Institute for Development and Research in Banking Technology,
Castle Hills Road #1 Masab Tank, Hyderabad 500057
Andhra Pradesh, India
{pradeepghyd, padmarav, adityaachanta2}@gmail.com,
bldeekshatulu@idrbta.c.in
[2] SCIS, University of Hyderabad, Hyderabad 500046
Andhra Pradesh, India
[3] Indian Institute of Technology, Patna 800013, Bihar, India
kaushalta@gmail.com
[4] IIM Calcutta, Kolkata 700104, West Bengal, India
bose@iimcal.ac.in

Abstract. In this paper, we propose new rule based classifiers based on Firefly (FF) and Threshold Accepting (TA) Algorithms viz., Improved Firefly Miner, Threshold Accepting Miner, Hybridized Firefly-Threshold Accepting (FFTA) based Miner for classifying a company as fraudulent or non fraudulent with respect to their financial statements. We apply t-statistic based feature selection and investigate its impact on the results. FFTA and TA miners turned to be statistically similar. Both algorithms outperformed standard decision tree both in terms of sensitivity and the length of rules.

Keywords: Firefly algorithm · Threshold accepting algorithm · Evolutionary computing rule miner and financial statement fraud detection

1 Introduction

There has been a huge increase in the number of frauds over the past decade or so. The Lehman Brothers Scandal [1, 2] in 2008 in the US and Satyam Scandal [3] in 2009 in India are among the worst accounting scandals to have occurred in the recent times. In 2007, Lehman Brothers was ranked at the top in the "Most Admired Securities Firm" [1] by Fortune Magazine. Just a year after, the company had gone bankrupt and it was found out that it had hidden $50 billion in loans disguised as sales. It was a serious case of an accounting fraud. The Satyam Computer Services scandal was a corporate scandal that surfaced in India in 2009 where the Chairman confessed that the company's accounts had been falsified. The Global corporate community was shocked when the chairman confessed that he had manipulated the accounts by US$1.47-Billion [3]. These frauds could have been avoided had proper audits taken place. While we have auditors for the

B.K. Panigrahi et al. (Eds.): SEMCCO 2014, LNCS 8947, pp. 239–250, 2015.
DOI: 10.1007/978-3-319-20294-5_21

job, due to the number of cases they have to deal with as well as with the huge amount of data present, it is practically not possible for them to be always accurate. Here the Data Mining techniques come to our rescue. In this paper, we take recourse to employing Evolutionary Computational algorithms in order to generate 'if-then' rules that classify the companies into fraudulent or not. Rule based outputs are of interest to us because they are transparent and yield us knowledge in the human comprehensible form. With proper rule based outputs, we can analyze which variables are more important in determining frauds and take appropriate measures well in advance.

2 Literature Review

There has been a minor research in the field of financial statement fraud detection using data mining techniques. The techniques to have been used include case based reasoning, decision tree methods, text mining, logistic regression, neural networks etc.

According to Kirkos et al. [4], some estimates stated that fraud cost US business more than $400 billion annually. Spathis et al. [5] compared multi-criteria decision aids with statistical techniques such as logit and discriminant analysis in detecting fraudulent financial statements. A novel financial kernel for the detection of management fraud is developed using support vector machines on financial data by Cecchini et al. [6]. Huang et al. [7] developed an innovative fraud detection mechanism on the basis of Zipf's Law. Kirkos et al. [4] used the ID3 decision tree and Bayesian belief network to detect financial statement fraud. Sohl and Venkatachalam [8] used back-propagation NN for the purpose. Cerullo and Cerullo [9] explained the nature of fraud and financial statement fraud along with the characteristics of NN and their applications. Calderon and Cheh [10] examined the efficacy of NN as a potential enabler of business risk based auditing. Koskivaara [11, 12] investigated the impact of various pre processing models on the forecast capability of NN when auditing financial accounts. Busta and Weinberg [13] used six designs of NN by taking different subsets of 34 variables to distinguish between 'normal' and 'manipulated' financial data. They examined the digit distribution of the numbers in the underlying financial data based on Benford's law. This law demonstrated that the digits of naturally occurring numbers are distributed on a predictable and specific pattern. Feroz et al. [14] observed that the relative success of the NN models was due to their ability to 'learn' what were important. Brooks [15] also applied various NN models to detect financial statement fraud with great success. Fanning and Cogger [16] used NN (AutoNet) for detecting management fraud on the important publicly available predictors of fraudulent financial statements. Ramamoorti et al. [17] compared the performance of the multilayer perceptron with a Delphi study. Zhang et al. [18] conducted a review of the papers that reported the use of NN in forecasting during 1988–98.

Aamodt and Plaza [19] and Kotsiantis et al. [20] used case based reasoning to identify the fraudulent companies. Further, Deshmukh and Talluru [21] employed a 15 rules-based fuzzy reasoning system to assess the risk of management fraud. Pacheco et al. [22] developed a hybrid system consisting of NN and a fuzzy expert system for

the purpose. Further, Magnusson et al. [23] used text mining and demonstrated that the language of quarterly reports provided an indication of the change in the company's financial status. Variable selection was used in a rule-based system by eliminating variables that were either redundant or possessed little predictive information [24].

Many researchers proposed rule extraction algorithms based on global optimization techniques such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony optimization (ACO) and Differential Evolution (DE). Firstly, Mahfoud and Mani [25] used GA and extracted rules to predict the performance of individual stocks. Shin and Lee [26] extracted rules from GA for predicting bankruptcy of firms. Then, Parpinelli et al. [27] proposed Ant-Miner, which uses ACO for extracting classification rules. Later, Kim and Han [28] used GA in discovering the rules for predicting bankruptcy of firms. Sousa et al. [29] proposed Constricted PSO (CPSO) for rule mining. Thereafter, Liu et al. [30] proposed PSO based rule extraction method, where they proposed fitness function different from [29]. Ji et al. [31] improved the Ant-miner proposed in [27]. Later, Zhao et al. [32] proposed Fuzzy-PSO, where the binary PSO generates fuzzy rules. Then, Holden and Freitas [33] hybridized PSO and ACO for discovering classification rules. Most recently, Su et al. [34] employed DE for rule extraction. Ravisankar et al. [35] used Classification and Regression Trees (C&RT) along with other techniques for determining fraud companies in the same Chinese Bank dataset. Naveen et al. [36] proposed Rule Extraction using firefly optimization and its applications to Banking. However, it failed when confronted with more than 10 dimensions of the feature space.

3 Proposed Methodology

3.1 Rule Encoding

We introduced a new rule encoding scheme as opposed to Naveen et al. [36], where each dimension of the firefly i.e. a financial attribute is represented by 3 bits. So, according to the number of attributes that we started with, i.e. 10, 18 or 35, we have 30 bits, 54 bits or 105 bits respectively allotted for rules. Here, the first bit gives us the benchmark value of the attribute. The second bit depicts us whether the attribute is less than or greater than the benchmark in the first bit. This is simply done using a random number between 0 and 1. If the number is less than 0.5, it denoted less than sign; else, it denotes greater than sign. And the last bit indicates whether to include this attribute in the rule or not. It is implemented using 0 and 1, where 0 represents that the attribute should not be included in the rule and 1 indicates otherwise (Table 1).

where D is the number of financial attributes which is 10, 18, 35 in our case. Suppose a rule is represented as in Table 1, the corresponding rule would be

If Attribute 1 < 0.87 and Attribute 3 > 0.57 then belong to a particular class.

3.2 Improved Firefly Miner (FF-miner)

Naveen et al. [36] proposed the FF miner for rule extraction using firefly optimization for application to banking. When we tried to execute the existing FF miner on our

Table 1. Rule encoding scheme

Dimension 1			Dimension 2			Dimension 3			.....			Dimension D		
0.87	0.3	1	0.45	0.4	0	0.54	0.7	1	..	..	..	0.67	0.78	0

dataset, it ran into an infinite loop. So we modified the existing FF miner. We modified the way the position of the firefly is updated. To ensure the global solution space coverage, we incorporate *gbest* and *pbest* concepts from PSO [37] and update the position of the firefly accordingly. Also to reduce the rule length, we introduce a new rule encoding scheme which has been explained in Sect. 3.1. The previous encoding scheme followed by Naveen et al. [36] consisted of only first 2 bits as opposed 3 bits in our case. This new encoding scheme serves the purpose of producing variable rules as opposed to fixed maximum length rules in our previous case. This encoding scheme has been very effective in smooth functioning of our rule miners.

Existing update formula of x_i

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r^2}\left(x_i - x_j\right) + \alpha(\text{rand}() - 0.5) \tag{1}$$

Modified update formula of x_i

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r^2}\left(x_i - x_j\right) + c \times \left(gbest_i - x_i(t)\right) \tag{2}$$

Where t is the iteration number.

Firefly Algorithm [39, 40]

1. Initialize the population of fireflies i.e., rules in our case
2. For each of the iterations
3. Calculate fitness for each of the firefly using equation (5)
4. For each firefly i in the population
5. For each firefly j in the population
6. If fitness(i)<fitness(j)
7. Calculate Euclidean distance between firefly i and j
8. For each dimension in the firefly
9. Calculate beta using following equation.
10. $$\beta = \beta_0 e^{-\gamma r^2} \tag{3}$$
11. Update firefly position using following equation.
12. $$x_i(t+1) = x_i(t) + \beta\left(x_i(t) - x_j(t)\right) + \alpha(rand() - 0.5) \tag{4}$$
13. End For
14. End If
15. End For
16. End For

Pseudocode of FF miner

```
For each fold of the dataset
For run=1 to no_of_runs
        Load Training, Testing and Validation datasets
        Identify the total number of classes as n
        For each class of n classes(here n=two, fraud and non fraud)
                Reload Training Records
                Count the no of records satisfying the class as tr_class_count
                Repeat
                        Apply Firefly Algorithm given above to generate a rule
                        Mark and count Training Records which covered by the
                        rule as rule_cover_count
                Until rule_cover_count>=90% of (tr_class_count)
        Reload Training Records
        Sort The rules Based on Fitness

Compute Accuracy, Sensitivity & Specificity for training, test and validation data.
```

A particular firefly moves towards a brighter firefly keeping in mind that intensity also decreases with increase in distance. These form the essence of the overall firefly algorithm. Finally, we obtain the optimal value of the fitness function after some iterations, which gives us the optimal set of rules in our case. Equation 3 indicates attractiveness that varies with distance, while Eq. 4 indicates position update formula. Here, t, β_0 and γ are iteration number, attractiveness constant and light absorption coefficient respectively.

3.3 TA Miner

In the TA miner, we apply the Threshold Acceptance Algorithm [38] for the Fraud Detection Problem. TA being a local search method helped us find the local optimal solutions around different points. The parameters of TA such as thresh, threshtol, delta, acc are also identified after trial and error.

TA algorithm:

```
1. initialize candidate solution/rule
2. for each of the global iterations
3.                 for each of the inner iterations
4.                         Generate new candidate solution
5.                         calculate delta which is the difference of fitness between
                           current and previous candidate solution
6.                         if delta<thresh
7.                                 Make the candidate solution as the new solution
8.                 if thresh<threshtol
9.                         calculate delta2 which is the difference of fitness between
_                          old and new solution
10.                if ABS(delta2)<acc
11.                        break;
12.  thresh=thresh*(1-eps);
13.  Replace Solution in the population with the candidate solution
```

Pseudo code of TA miner

```
For each fold of the dataset
For run=1 to no_of_runs
        Load Training, Testing and Validation datasets
        Identify the total number of classes as n
        For each class of n classes(here n=two, fraud and non fraud)
                Reload Training Records
                Count the number of training records satisfying satisfying the class
                as tr_class_count
                Repeat
                        Apply TA algorithm given above to generate rule
                        Mark and count Training Records which covered by the
                        rule asrule_cover_count
                Until rule_cover_count>=90%of(tr_class_count)
        Reload Training Records
        Sort The rules Based on Fitness
Calculate Accuracy, Sensitivity &Specificity for training, test and validation data.
```

3.4 FFTA Miner

We propose a hybrid of the Firefly Algorithm and TA as general purpose rule miner and apply it for Fraud Detection. For each run, the algorithm runs at least as many times as the number of classes in the dataset. First of all, the dataset we have is divided into Training, Testing and Validation datasets. The total number of classes is identified from the dataset. Now for each class, the training records are loaded and we count the number of training records satisfying a particular class as tr_class_count. The Firefly – TA Algorithm (FFTA) is next applied to get the rules. This process is repeated until the number of classes covered by the rules is greater than 90 % of the tr_class_count. These rules are applied on Training, Testing and Validation Datasets. In the FFTA algorithm, we invoke the TA to replace the weakest firefly every 95 % of the iterations. The TA algorithm replaces the weakest firefly with the best solution in the neighborhood of the weakest firefly. The incorporation of TA in the algorithm thus ensured faster convergence. The pseudo code is as follows

FF-TA algorithm:

```
[1-3] lines of FF algorithm
If(rand<prob_ta)
        Call TA
[4-16] lines of FF algorithm
```

FFTA Miner Pseudo Code

```
For each fold of the dataset
For run=1 to no_of_runs
        Load Training, Testing and Validation datasets
        Identify the total number of classes as n
        For each class of n classes (here two fraud and non fraud)
                Reload Training Records
                Count the no. of records satisfying the class as tr_class_count
                Repeat
                        Apply FF-TA algorithm given above to get a rule
                        Mark and count Training Records which covered by the
                        rule as rule_cover_count
                Until rule_cover_count>=90% of (tr_class_count)
        Reload Training Records
        Sort The rules Based on Fitness
        Compute Accuracy, Sensitivity, and Specificity.
```

The different parameters in the Firefly Algorithm such as β_0, γ, α etc. are determined by trial and error method. Similarly the parameters in the Threshold Acceptance such as thresh, threshtol, delta, acc are also identified in the overall TA algorithm.

$$\text{Fitness Function} = \text{Sensitivity} * \text{Specificity} \tag{5}$$

$$\text{Where Sensitivity} = \frac{\text{Number of True Positives}}{\text{Number of True Positives} + \text{Number of False Negatives}}$$

$$\text{Specificity} = \frac{\text{Number of True Negatives}}{\text{Number of True Negatives} + \text{Number of False Positives}}$$

True Positives = # Fraudulent Companies correctly identified
True Negatives = # Non-fraudulent Companies correctly identified
False Negatives = # Fraudulent Companies wrongly identified as non- fraudulent
False Positives = # Non-fraudulent Companies wrongly identified as fraudulent

4 Dataset Description and Experiment Methodology

The dataset taken from [35], consists of 35 financial variables of 202 Chinese companies of which 101 are fraudulent and 101 are not. First, the dataset is divided into 80 % and 20 % groups and the latter one is designated as validation set. Then, 10-fold cross validation is performed on the 80 % dataset and we tested the obtained model in every fold on the validation set. Owing to the evolutionary nature of the Firefly and TA Algorithms, the results varied slightly, whenever the random seed is changed. Therefore, each miner is run 20 times for every fold and then the average results are computed and tabulated. Thus, we have 20*10 i.e. 200 runs for each algorithm. Following Ravisankar et al. [35], we performed the analysis with the top 18 and the top 10

features obtained through the t-statistic based feature selection method. Thus, the 3 algorithms (FF, FFTA and TA) were run on the dataset with 35, 18 and 10 features.

5 Results and Discussion

The best, worst, average sensitivities of the 10 folds have been presented in Table 2 for the miners viz., FF, FFTA and TA. These are compared in different cases i.e., 35 features, Top-18, Top-10 obtained features. First, when compared with the results of decision trees [42] reported in [35], both TA and FFTA yielded superior sensitivity and rule length also turned to be smaller for them.

Table 2. Comparison of fitness values and sensitivity within braces

No of features	Improved FF miner			TA miner			FF-TA miner		
	Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
35	5294 (100)	1381.25 (25)	4468.94 (65.05)	5833 (100)	1666.5 (33.33)	5370.29 (75.13)	5526 (100)	2500 (50)	**5603.86 (79.05)**
18	6429 (100)	1765.16 (33.33)	4775.71 (70.20)	5385 (100)	1623.84 (33.33)	5367.40 (74.88)	5122 (100)	1444.19 (33.33)	**5703.17 (79.52)**
10	6333 (100)	1299.9 (30)	5191.18 (72.19)	6296 (100)	2432 (50)	5582.51 **(79.05)**	5833 (100)	3428.8 (66.67)	**5751.74 (79.40)**

From the results, we can observe that hybridized FFTA miner produced highest sensitivity compared to individual FF and TA miners in all 3 cases i.e., with 35, 18 and 10 features. However, TA miner closely followed FFTA miner. But, in case of Top-10 features, TA miner with sensitivity 79.05 % is very much closer to that of FFTA miner which produced 79.40 %. In all the cases, FF miner was found to be inferior compared to other two algorithms. Finally, we conclude that FFTA with 10 features produced best results in terms of sensitivity and fitness value. Further, only FFTA miner could yield more than 79 % sensitivity in all cases. A t-test (see Table 3) was performed at 5 % level of significance to find if the difference between FFTA and TA is statistically significant. It turned out that the two methods are statistically similar for 10 features and different for 18 and 35 features. We didn't compare FF miner with FFTA miner as their sensitivities differ by greater amounts. Furthermore, FFTA miner yielded higher fitness values compared to FF miner and TA miner, thereby indicating that it also produced less false positives. Therefore, the hybrid FFTA miner has turned to be the holistic winner among the three miners. This is a significant outcome of the study. Also, since FFTA miner produced superior results, we performed statistical t-tests with respect to sensitivity across three cases i.e., with 35, 18, 10 features within FFTA miner and found that they are all statistically insignificant. Consequently, we recommend FFTA miner with 10 features as the best case for this dataset.

Table 3. t-test values of FFTA miner versus TA miner with 35, 18 and 10 features

35 features	7.09
18 features	8.35
10 features	0.43

The rules obtained by FFTA, FF and TA miners for the case of 10 features, which obtained a maximum of 100 % sensitivity, are presented in Tables 4, 5 and 6 respectively. We found that FFTA miner obtained just 2 rules, while the rest produced 3 rules each. Further, we found that the variable *Net_profit/Total_assets* appeared in almost all the rules produced by all the three miners. Therefore, we can conclude that this variable is the most significant variable in this study. This variable is followed by *Gross Profit*, which was picked by FFTA and FF miners but not TA miner. The optimal parameter combinations chosen for the FF, TA and FFTA algorithm are presented in Tables 7, 8 and 9.

Table 4. Rules of FFTA miner in its best run of 100 % sensitivity

Rule-1: *If(Inventory/Total_assets>0.44 & Gross_profit/Total_assets>0.96 & Net_profit/Total_assets>0.95& Inventory/Current_liabilities>0.40) Then Fraud* *Rule-2:* *If(Gross_profit<0.28 & Net_profit<0.36 & Net_profit/Total_assets<0.95 & Inventory/Current_liabilities>0.49) Then Non- Fraud*

Table 5. Rules of FF miner in its best run of 100 % sensitivity

Rule-1: *If(Gross_profit>0.99 & Net_profit>0.99 & Primary_business_income>1.0 & Net_profit/Total_assets<0.95) Then Fraud* *Rule-2:* *If(Gross_profit>0.99 & Net_profit>0.99 & Net_profit/Total_assets>0.95& Net_profit/Primary_business_income>1.0 & Primary_business_income/Fixed_assets >1.0) Then Fraud* *Rule-3:* *If(Gross_profit<0.52 & Net_profit<0.50 & Net_profit/Primary_business_income <0.44) Then Non – Fraud*

Table 6. Rules of TA miner in its best run of 100 % sensitivity

Rule-1: *If(Inventory/Total_assets>0.67 & Net_profit/Total_assets>0.95 & Inventory/Current_liabilities>0.14) Then Fraud* *Rule-2:* *If(Gross_profit/Total_assets>0.97 & Net_profit/Total_assets<0.95 & Primary_business_income/Fixed_assets>0.87) Then Fraud* *Rule-3:* *If(Net_profit/Total_assets<0.95) Then Non- Fraud*

Table 7. Parameters for FF

Features	n	β_0	Υ	Δ	α	MaxIterations	Prob_rule_length
35	35	1	2.5	1	0.5	100	0.05
18	35	1	2.5	1	0.5	100	0.175
10	35	1	2.5	1	0.5	100	0.197

Table 8. Parameters for TA miner

Features	Eps	Acc	ThreshTol	Thresh	MaxInerIteration (A)	MaxOuterIterations (B)	Prob_rule_length (C)
35	0.001	0.5	0.001	0.1	250	25	0.05
18	0.001	0.5	0.001	0.1	250	25	0.175
10	0.001	0.5	0.001	0.1	250	25	0.197

Table 9. Parameters for FFTA miner

Features	N	β_0	Υ	Δ	α	MaxIterations	Prob_TA	
35	35	0.5–2.5	0.5–2.5	1	0.5	25–200	0.95	Parameters of TA from Table 8 are selected here too
18	35	0.5–2.5	0.5–2.5	1	0.5	25–200	0.95	
10	35	0.5–2.5	0.5–2.5	1	0.5	25–200	0.95	

Observing that the sensitivities yielded by all the three algorithms are no very high, we tried to find the reason behind it and performed Principal Component Analysis (PCA) visualization using NeuCom [41]. It was found out that most of the 202 companies are concentrated along a line, thereby making them difficult to discriminate. Most probably, the same behavior could be observed in the original input space also. This might be the reason why the rule mining algorithms failed to yield sensitivity of 90 % and above.

6 Conclusion

We proposes hybrid FFTA miner for financial fraud detection. It yielded far superior results compared to the traditional decision trees in terms of sensitivity with less number of rules. Further, TA and FFTA miners are not statistically significantly different, as evidenced by t-test. But since TA miner is comparatively less complex than the hybrid FFTA miner, it's preferable to apply the former for this dataset, when 10 variables are considered. But, with more variables, FFTA miner is stable and superior.

References

1. http://www.accounting-degree.org/scandals/. Accessed 10 June 2014
2. http://en.wikipedia.org/wiki/Accounting_scandals. Accessed 10 June 2014
3. http://en.wikipedia.org/wiki/Satyam_scandal. Accessed 10 June 2014
4. Kirkos, E., Spathis, C., Manolopoulos, Y.: Data mining techniques for the detection of fraudulent financial statement. Expert Syst. Appl. **32**, 995–1003 (2007)
5. Spathis, C., Doumpos, M., Zopounidis, C.: Detecting falsified financial statements: a comparative study using multi criteria analysis and multivariate statistical techniques. Eur. Acc. Rev. **11**(3), 509–535 (2002)
6. Cecchini, M., Aytug, H., Koehler, G.J., Pathak, P.: Detecting management fraud in public companies. http://warrington.ufl.edu/isom/docs/papers/DetectingManagementFraudInPublicCompanies.pdf
7. Huang, S.-M., Yen, D.C., Yang, L.-W., Hua, J.-S.: An investigation of Zipf's Law for fraud detection. Decis. Support Syst. **46**(1), 70–83 (2008)
8. Sohl, J.E., Venkatachalam, A.R.: A neural network approach to forecasting model selection. Inf. Manage. **29**(6), 297–303 (1995)
9. Cerullo, M.J., Cerullo, V.: Using neural networks to predict financial reporting fraud: part 1. Comput. Fraud Secur. **5**, 14–17 (1999)
10. Calderon, T.G., Cheh, J.J.: A roadmap for future neural networks research in auditing and risk assessment. Int. J. Acc. Inf. Syst. **3**(4), 203–236 (2002)
11. Koskivaara, E.: Different pre-processing models for financial accounts when using neural networks for auditing. In: Proceedings of the 8th European Conference on Information Systems, vol. 1, pp. 326–3328. Vienna, Austria (2000)
12. Koskivaara, E.: Artificial neural networks in auditing: state of the art. ICFAI J. Audit Pract. **1** (4), 12–33 (2004)
13. Busta, B., Weinberg, R.: Using Benford's law and neural networks as a review procedure. Manage. Auditing J. **13**(6), 356–366 (1998)
14. Feroz, E.H., Kwon, T.M., Pastena, V., Park, K.J.: The efficacy of red flags in predicting the SEC's targets: an artificial neural networks approach. Int. J. Intell. Syst. Acc. Finan. Manage. **9**(3), 145–157 (2000)
15. Brooks, R.C.: Neural networks: a new technology. CPA J. http://www.nysscpa.org/cpajournal/old/15328449.htm1994
16. Fanning, K.M., Cogger, K.O.: Neural network detection of management fraud using published financial data. Int. J. Intell. Syst. Acc. Finan. Manage. **7**(1), 21–41 (1998)
17. Ramamoorti, S., Bailey Jr., A.D., Traver, R.O.: Risk assessment in internal auditing: a neural network approach. Int. J. Intell. Syst. Acc. Finan. Manage. **8**(3), 159–180 (1999)
18. Zhang, G., Patuwo, B.E., Hu, M.Y.: Forecasting with artificial neural networks: the state of the art. Int. J. Forecast. **14**(1), 35–62 (1998)
19. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. Artif. Intell. Commun. **7**(1), 39–59 (1994)
20. Kotsiantis, S., Koumanakos, E., Tzelepis, D., Tampakas, V.: Forecasting fraudulent financial statements using data mining. Int. J. Comput. Intell. **3**(2), 104–110 (2006)
21. Deshmukh, L.Talluru: A rule-based fuzzy reasoning system for assessing the risk of management fraud. Int. J. Intell. Syst. Acc. Finan. Manage. **7**(4), 223–241 (1998)
22. Pacheco, R., Martins, A., Barcia, R.M., Khator, S.: A hybrid intelligent system applied to financial statement analysis. In: Proceedings of the 5th IEEE Conference on Fuzzy Systems, vol. 2, pp. 1007–10128. New Orleans, USA (1996)

23. Magnusson, C., Arppe, A., Eklund, T., Back, B., Vanharanta, H., Visa, A.: The language of quarterly reports as an indicator of change in the company's financial status. Inf. Manage. **42** (4), 561–574 (2005)
24. Kim, Y.: Toward a successful CRM: variable selection, sampling, and ensemble. Decis. Support Syst. **41**(2), 542–553 (2006)
25. Mahfoud, S., Mani, G.: Financial forecasting using genetic algorithms. Appl. Artif. Intell. **10**, 543–565 (1996)
26. Shin, K.-S., Lee, Y.-J.: A genetic algorithm application in bankruptcy prediction modeling. Expert Syst. Appl. **23**(3), 321–328 (2002)
27. Parpinelli, R.S., Lopes, H.S., Frietas, A.A.: Data mining with an ant colony optimization algorithm. IEEE Trans. Evol. Comput. **6**(4), 321–332 (2002)
28. Kim, M.-J., Han, I.: The discovery of experts' decision rules from qualitative bankruptcy data using genetic algorithms. Expert Syst. Appl. **25**, 637–646 (2003)
29. Sousa, T., Neves, A., Silva, A.: A particle swarm data miner. In: 11th Portuguese Conference on Artificial Intelligence, Workshop on Artificial Life and Evolutionary Algorithms, pp. 43–53 (2003)
30. Liu, Y., Qin, Z., Shi, Z., Chen, J.: Rule discovery with particle swarm optimization. In: Chi, C.-H., Lam, K.-Y. (eds.) AWCC 2004. LNCS, vol. 3309, pp. 291–296. Springer, Heidelberg (2004)
31. Ji, J., Zhang, N., Liu, C., Zhong, N.: An ant colony optimization algorithm for learning classification rules. In: Proceedings of IEEE/WIC, pp. 1034–1037 (2006)
32. Zhao, X., Zeng, J., Gao, Y., Yang, Y.: Particle swarm algorithm for classification rules generation. In: Proceedings of the Intelligent Systems Design and Applications, IEEE, pp. 957–962 (2006)
33. Holden, N., Frietas, A.A.: A hybrid PSO/ACO algorithm for classification. In: Proceedings of Genetic and Evolutionary Computation conference, pp. 2745–2750 (2007)
34. Su, H., Yang, Y., Zha, L.: Classification rule discovery with DE/QDE algorithm. Expert Syst. Appl. **37**(2), 1216–1222 (2010)
35. Ravisankar, P., Ravi, V., Raghava Rao, G., Bose, I.: Detection of financial statement fraud and feature selection using data mining techniques. Dec. Support Syst. **50**, 491–500 (2010)
36. Naveen, N., Ravi, V., Raghavendra Rao, C., Sarath, K.N.V.D.: Rule extraction using firefly optimization: application to banking. In: IEEM (2012)
37. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ (1995)
38. Dueck, G., Scheuer, T.: Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. J. Comput. Phys. **90**, 161–175 (1990)
39. Yang, X.-S.: Firefly algorithms for multimodal optimization. In: Watanabe, O., Zeugmann, T. (eds.) SAGA 2009. LNCS, vol. 5792, pp. 169–178. Springer, Heidelberg (2009)
40. Yang, X.-S.: Firefly algorithm, stochastic test functions and design optimization. Int. J. Bio-Inspired Comput. **2**(2), 78–84 (2010)
41. Neucom. http://www.aut.ac.nz/research/research-institutes/kedri/research-centres/centre-for-data-mining-and-decision-support-systems/neucom-project-homepage#download
42. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**, 81–106 (1986)