Application of an Estimation of Distribution Algorithm to Financial Statement Fraud Classification

Brenton John Lessley

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science

University of Washington

2011

Program Authorized to Offer Degree:
Computing and Software Systems

University of Washington
Graduate School

This is to certify that I have examined this copy of a master's thesis by

Brenton John Lessley

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____
Donald Chinn

_____
Matthew Alden

_____
Daniel Bryan

_____
Ehsan Feroz

Date: _____

Signature_____

Date_____

University of Washington

Abstract

Application of an Estimation of Distribution Algorithm to Financial Statement Fraud Classification

Brenton John Lessley

Chair of the Supervisory Committee:
Associate Professor Donald Chinn
Institute of Technology

Published in 2002 by the Auditing Standards Board (ASB), the *Statement on Auditing Standards (SAS) No. 99* outlines a set of 42 risk factors, or "red flags," which can aid independent auditors in assessing the risk of financial statement fraud and errors arising from the material misstatement of financial statements. Due to the increase in corporate accounting fraud in the early 2000s, this statement significantly redefined fraud detection procedures and extensively revised the original 39 risk factors from *SAS Nos. 53* and *82*. The goal of this thesis is to assess the efficacy of these risk factors in predicting financial statement fraud and subsequent Securities and Exchange Commission (SEC) enforcement action. Using MARLEDA — a sophisticated estimation of distribution algorithm (EDA) — a fuzzy rule-based classifier (FRBC) system was generated and employed to classify a corporation as a fraudulent or non-fraudulent financial reporter. Each resulting classifier took the form of a comprehensible if-then rule set, with each if-condition containing the value of a corporation's risk factor variable and the then-condition representing the classification.

The classification performance of MARLEDA was compared to the performance of competing logistic regression, artificial neural network (ANN), and genetic algorithm (GA) models. Demonstrating moderate success in detecting patterns of fraudulent behavior, the

GA and MARLEDA models yielded average training accuracy rates of 62.10 and 69.53 percent, respectively. Reflecting the "tradeoff" condition of multi-objective optimization, the GA maintained a lower accuracy than MARLEDA, but possessed fewer active logic rules within its FRBCs. This reduction in classifier complexity possibly allowed the GA to achieve a higher average validation accuracy rate (59 percent) than that of MARLEDA (53.05 percent). Overall, both of the evolutionary algorithm models outperformed the benchmark logistic regression and ANN models. Finally, a subset of 6 financial variables, representing 6 *SAS No. 99* red flags, were considered to hold the most promise in detecting patterns of fraud.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This thesis is the culmination of both the direct and indirect contributions of several individuals.

First, each of my committee members contributed time and effort in his own unique way. Donald Chinn, my committee chairperson, diligently reviewed the multiple drafts of this thesis and suggested ways to improve the precision, clarity, and grammatical correctness of my writing. His insighful input during our many discussions helped shape the overall form of the thesis. Matthew Alden offered invaluable advice, help, and guidance throughout the experimental phase of this thesis, and I especially thank him for his willingness to meet with me on several occasions during the final weeks of the research. Daniel Bryan provided tireless support during the data collection and pre-processing phase, and his enthusiasm and business perspectives were greatly appreciated throughout the entire process. Ehsan Feroz furnished my initial curiosity in the field of artificial intelligence and greatly inspired this thesis work. During our many conversations, his expert insight and advice were utilized to enhance the quality and exposition of this thesis.

Second, I wish to express sincere appreciation to all of the teachers and academic advisors who have guided me through the completion of my M.S. work and/or fostered my curiosity in computer science, mathematics, and business.

Finally, I cannot thank my parents enough for their support and encouragement throughout my educational pursuits.

DEDICATION

To my parents.

Chapter 1

INTRODUCTION

Since the introduction of the Internet more than two decades ago, the amount of data and information available to decision makers has increased significantly. Accompanying this growth has been a corresponding desire for information that is credible, reliable, and free of material misstatements. In the face of several, high-profile corporate accounting scandals in the early 2000s (e.g. Enron, Worldcom, Adelphia, and RiteAid), these information qualities are now valued at a premium by investors, creditors, and other shareholders, who must make informed decisions regarding the optimal allocation of limited resources. In order to regulate the financial reporting activities of publicly-traded corporations, the Financial Accounting Standards Board (FASB) has established an auditing and assurance framework that requires the auditing profession to identify patterns of fraud within the financial accounts of a corporation [17]. Due to the potential ramifications of financial statement fraud, it is imperative that auditors and regulators develop and employ a system that can accurately detect the presence of such an occurrence. This thesis develops such a system using a fuzzy logic rule classification approach that is unique to the specific problem domain. The remainder of this chapter presents the motivation, related research, and approach of this financial statement fraud classification system.

1.1 Motivation

Published in 1988 by the Auditing Standards Board (ASB), *Statement on Auditing Standards (SAS) No. 53* outlined a set of 14 *risk factors*, or "red flags," which would aid independent auditors in assessing the risk of errors and irregularities arising from the material

misstatement[1] of financial statements [45]. Based on this risk assessment, each auditor was required to design and conduct an audit such that the detection of the errors in the financial statements could be reasonably assured [45]. While *SAS No. 53* provided guidance on detecting accounting errors, it did not explicitly require auditors to assess the risk of intentional financial statement fraud (e.g. falsification of accounting records and misappropriation of assets) [17]. Through its issuance of *SAS No. 82* in 1997, the ASB introduced much-needed fraud detection audit procedures and added 25 new red flags that could be used by auditors to assess the risk of financial statement fraud [17].

Due to the increase of corporate accounting fraud in the early 2000s, the ASB, in 2002, issued a revised SAS — *SAS No. 99: Consideration of Fraud in a Financial Statement Audit* — that significantly redefined fraud detection procedures and extensively revised the original 39 red flags [38]. Additionally, three new red flags were established, raising the total number of factors to 42 [38]. In order to facilitate the usefulness and effectiveness of *SAS No. 99*, the ASB categorized each red flag based on its relation to one of the following elements of the *Fraud Triangle*: *pressure* to commit fraud, *opportunity* to commit fraud, and *rationalization* to commit fraud. One of the primary goals of this thesis research is to assess the effectiveness of these red flags in detecting financial statement fraud and subsequent SEC enforcement.

1.1.1 Accounting and Auditing Enforcement Releases

According to the Committee of Sponsoring Organizations (COSO), the number of detected, public-company financial reporting fraud cases between the years 1998 and 2007 has increased by over 18 percent from the level in the prior 10-year period [5]. Furthermore, a 2002 study by the Association of Certified Fraud Examiners revealed that financial fraud cost U.S. businesses over $\$600$ billion in 2001 alone [26]. Based on these less-than-positive statistics, the SEC has increased its role in fraud detection and prevention through the is-

[1] A financial misstatement is considered to be *material* by an independent auditor when the misstatement affects the judgment of a knowledgeable individual who relies on the information [17].

suance of Accounting and Auditing Enforcement Releases (AAERs) to corporations and auditors who knowingly or excessively violate *generally accepted accounting principles* (GAAP).

An SEC enforcement process is instigated by various events, including the notification of misconduct within a corporation, a change in external auditor, the voluntary restatement of financial results, or the SEC's annual review of randomly-selected firms [15]. Initially, the SEC will privately request that a violating company restate its financial statements to address any errors or reporting deficiencies [15, 55]. If the company fails to comply, then the SEC may pursue legal action and, subsequent to its investigation, issue an AAER to the company. Faced with monetary and time constraints, the SEC typically pursues enforcement action against firms that provide more-than-sufficient evidence of securities or GAAP violations [15, 23]. Hence, AAERs represent a subset of all financial reporting violations that may have occurred. Since its first AAER issuance in 1982, the SEC has issued over 3,000 enforcement releases.

Since the SEC issues an AAER with the belief that a company engaged in fraudulent financial reporting (e.g. violations of GAAP), such a citation is highly suitable for financial fraud classification tasks [15, 23, 54, 55]. Hence, given the public availability of AAER citations and their potential fraud predictiveness, the primary goal of the financial statement fraud classification in this thesis is to detect patterns that are indicative of the issuance (or non-issuance) of an AAER to a corporation.

1.2 Related Research

Throughout the past thirty years, an extensive amount of research has been conducted to investigate the types of entities and red flag variables that are indicative of financial fraud. In this section, we briefly review the key findings of research that utilized machine learning classification techniques to predict or detect financial statement fraud and/or subsequent SEC enforcement.

The early work of Persons [51] (1995) investigated the ability of financial statement

data to identify factors associated with fraudulent financial reporting. With a set of 10 financial ratios/variables cited in previous accounting literature, stepwise-logistic models were used to estimate the expected cost of misclassifying a corporate data instance in both the first year of fraud and the preceding year. To train the models, a sample of 103 and 100 AAER firms were used for the fraud year and preceding year, respectively. After testing the models, the fraud-year model correctly classified 47 percent of fraud firms, while misclassifying 14 percent of non-fraud firms and assuming a relative error cost of $30:1$ of misclassifying fraud versus misclassifying non-fraud. The preceding-year model correctly classified 64 percent of fraud firms, while misclassifying 21 percent of non-fraud firms at the same relative error cost. Persons concluded that financial leverage, capital turnover, asset composition, and firm size are significant indicators of financial statement fraud. While conducting this study it was also assumed that the cost of misclassifying a fraud company (i.e. a false negative) is higher than the cost of misclassifying a non-fraud company (i.e. a false positive).

A study by Green and Choi [25], in 1997, deployed a set of three neural network models to predict financial statement fraud that is related to the revenue cycle (e.g. improper revenue recognition). Using a sample of 86 AAER/non-AAER cases between 1982 and 1990, and 8 financial variables, the best neural network model yielded an overall test accuracy rate of 63 percent. A similar neural network study by Fanning and Cogger [22] in 1998 removed the revenue cycle prediction constraint of [25], expanded the number of variables to 20, and increased the sample size to 204 AAER/non-AAER cases. However, the sample firms were matched based on the first year of fraud; thus, the goal was to *detect* fraud once it had already occurred. Overall, the results obtained by Fanning and Cogger [22] mirrored those of Green and Choi [25], primarily because the datasets were similar in composition.

Research investigating the prediction ability of the *SAS No. 53* red flags has also been conducted. Feroz et al. [23] (2000) used a feed-forward, multi-layered artificial neural network and a conventional logistic regression model to detect target corporations of the SEC's investigation of fraudulent financial reporting. Training their models with 90 non-AAER

records and 42 AAER records from the 1982-1990 time period, Feroz et al. [23] obtained classification accuracy rates of around 80 percent. The authors noted that an updated study using additional red flag variables and a larger sample size could be beneficial.

In 2004, Kaminski et al. [35] conducted an exploratory study to determine if financial ratio values of fraudulent companies differ from those of non-fraudulent companies. A data set of 79 corporations that were issued AAERs between 1982 and 1999 was constructed, based on the condition that each corporation possess ratio values for a seven-year time period (i.e. the fraud year +/- 3 years). Then, the AAER firms were matched with non-AAER firms on the basis of firm size, time period, and industry, to obtain another 79 non-AAER data instances. With this data set, a linear discriminant analysis classification technique was employed to correctly classify non-fraud firms between 84 and 94 percent of the time and fraud firms between 2 and 42 percent of the time. These results provided empirical evidence that financial ratios can possess limitations in predicting fraudulent financial reporting.

In 2006, Skousen and Wright [55] surveyed extant accounting research to identify financial ratios that could serve as proxies for the pressure, opportunity, and rationalization red flags of *SAS No. 99*. Using the most *significant* variables (based on statistical p-values), a logistic regression classification model was developed to correctly classify non-fraud firms 74.42 percent of the time and fraud firms 65.12 percent of the time; overall, the model correctly classified firms 69.77 percent of the time. To train the classification model, a data set of 86 AAER corporate data records between 1992 and 2001 was constructed. A control set of non-fraudulent firms was then developed by matching the AAER firms according to industry membership (4 digit SIC code), year of fraudulent activity, and size (Net Sales +/- 30%) in the year prior to fraud. One of the noted weaknesses of the study was the inability to identify significant financial variables to serve as proxies for the rationalization dimension of the *SAS No. 99* Fraud Triangle.

One of the first applications of evolutionary algorithms to financial statement fraud classification was initiated by Hoogs et al. [28] in 2007. This study used a genetic algorithm to detect temporal, or time-based, patterns that are indicative of financial statement fraud.

Using quarterly financial data from over 76 comparative metrics and of a target class of 51 AAER corporations, the genetic algorithm evolved patterns that consisted of multiple phrases, each of the form "n out of m quarters of metric i are *operator* than *threshold*", where m is the maximum number of quarters allowed in a pattern phrase and *operator* is a binary operator such as $\geq$ or $=$. A match between the pattern and the actual company data, at any time during the quarters, constitutes a match, resulting in the company being labeled as potentially-fraudulent by the algorithm. Overall, a set of three top-performing patterns correctly classified 63 percent of the target companies, while misclassifying 5 percent of the peers, or non-AAER companies. Due to the use of small training datasets, it was noted that the evolved patterns may have a difficult time generalizing to other datasets.

The work of Chai et al. [8] complemented the genetic algorithm approach of [28] by assigning to each learned pattern phrase a fuzzy score representing the degree to which a company's financial data matches the conditions (antecedents) of the phrase.

Providing a follow-up study to [55], Skousen et al. [54], in 2008, incorporated additional proxy variables and filtered the sample of 86 AAER fraud firms by both removing firms with significant outlier observations and mean-adjusting each variable. The improved classification models correctly classified non-fraud firms between 72 and 77 percent of the time and fraud firms between 68 and 70 percent of the time; overall, the models correctly classified firms between 70 and 73 percent of the time. The results of the study also revealed that rapid asset growth, increased cash needs, and external financing are positively-related to the likelihood of fraud (or the issuance of an AAER). Furthermore, the cumulative percentage of outstanding common stock owned by insiders and the control expressed by the board of directors are also linked to increased incidence of financial statement fraud.

In 2009, McKee [42] combined, or "stacked," the outputs of neural network, logistic regression, and decision tree models to predict financial statement fraud. This "meta-learning" approach first stacked a 71.4 percent-accurate neural network model into a logistic regression model, increasing the combined classification accuracy to 76.5 percent. Then the logistic regression model was subsequently stacked into a decision tree model, resulting

in an 83 percent combined accuracy rate. The study was based on the following parameters: 50 AAER corporate data instances between 1995 and 2002; 50 matched, non-AAER instances; and 15 financial red flag variables.

In 2010, Comunale et al. [10] demonstrated one of the first applications of fuzzy logic to the financial fraud domain. An expert system based on fuzzy logic was developed to assess the risk of financial statement fraud. First, an auditor is allowed to input data (e.g. a binary indicator) regarding the presence or absence of each *SAS No. 99* red flag variable. The system then uses the principles of fuzzy logic to evaluate the degree to which each red flag is present and to compute the fraud risk associated with various types (combinations) of financial statement fraud.

The recent work of Dechow et al. [15] (2011) examined the characteristics of 676 AAER firms along five dimensions: accrual quality, financial performance, non-financial measures, off-balance sheet activities, and market-based measures. Using a logistic regression model, an *F-Score* was derived to estimate the probability that a firm had engaged in an earnings misstatement. With a cut-off *F-Score* of 1.00, the implemented logistic models correctly identified over 60 percent of misstating firm-years, with a high frequency of false positives (i.e. firms that were not issued AAERs by the SEC are predicted to have misstated their earnings).

1.3 Approach

Using sophisticated evolutionary algorithms, a *fuzzy rule-based classifier* (FRBC) system was generated and employed to classify whether a corporation engaged in financial statement fraud. Each classifier was represented by a set of if-then logic rules, where each if-condition contains the associated fuzzy set of a red flag variable (e.g. Low, Medium, or High) and the then-condition represents the predicted classification. By applying evolutionary multi-objective optimization to the fraud classification task, the resulting FRBCs represented a "tradeoff" between classifier accuracy and complexity (comprehensibility). Thus, a financial regulatory body could utilize an FRBC that conforms to its specific pri-

orities, such as preferring a comprehensible FRBC over one that is more accurate. This flexibility is difficult to emulate in current regression classifiers and optimization models that are based on a single criterion. Furthermore, the assignment of red flag variables to fuzzy sets added an additional layer of comprehensibility and user-friendliness to the resulting FRBCs. A user can conveniently refer to the range of values for a fuzzy set to determine whether an actual red flag value falls within the limits of the set.

By applying a modern estimation of distribution algorithm (EDA) model to the *SAS No. 99* classification problem for the first time (to the best of our knowledge), we were able to compare the performance of this model to that of competing logistic regression, neural network, and genetic algorithm models. Since EDAs resolve several important issues that many machine learning models are known to possess, the results of this study corroborated the theoretical and statistical evidence of previous research. Furthermore, the work of this thesis built upon the *SAS No. 99* research of [54, 55] by using a larger and more-current set of data observations. These improvements enhanced the robustness of the classification results.

Finally, to assess the effectiveness of the *SAS No. 99* red flags in detecting fraudulent behavior, financial variables were devised to serve as proxies for the red flags. The composition of these variables was largely influenced by both the work of [23, 54, 55] and the public availability of corresponding financial data.

1.4 Overview of the Thesis

The remainder of this thesis consists of four main components: background (Chapter 2), financial data collection (Chapter 3), classification experiments (Chapter 4), and conclusion and future research (Chapter 5). Chapter 2 reviews the basic theory and operation of the classification models that are used in this thesis. Chapter 3 enumerates the procedures that are employed to construct and preprocess the financial data set. Chapter 4 discusses the design and implementation of the classification models, and presents the results of a suite of classification experiments. Finally, Chapter 5 contains a set of concluding remarks

that summarize the primary contributions of this thesis research, and introduces possible avenues of future research.

Chapter 2

BACKGROUND

During a typical day, each one of us, knowingly or not, engages in several forms of *pattern recognition*. From a trivial *classification* of the color of a stoplight to the intricate *clustering* of faraway galaxies, we each rely on our internal knowledge base and complex neural functionalities to reason about the world and infer our next plan of action. The ability to model and computationally automate these fundamental human activities has been one of the driving goals of the *machine learning* discipline.

Since the rise of machine learning over forty years ago, several pattern recognition models have been devised, each specializing in a specific problem domain or improving upon the contributions of predecessor models [18]. An important feature of these models is their ability to detect patterns within complex, high-dimensional, and potentially noisy data sets. Since corporate financial data typically consists of multiple variables and variable dependencies, it is imperative that robust and sophisticated algorithms be employed to run financial statement fraud classification tasks. In this chapter, the classification models that were utilized within this thesis research are presented and defined. Additionally, foundational concepts related to the understanding and operation of these models are provided.

2.1 Machine Learning

Within a learning task, an agent will first make observations about the present state of the world and then perform an action using a base of prior knowledge. Then, the type of learning will dictate whether the agent receives a form of performance feedback from the environment (e.g. a correct/incorrect response from a teacher). In machine learning, there are traditionally four main types of learning: *unsupervised*, *reinforcement*, *supervised*, and

semi-supervised [46]. Unsupervised learning does not support a feedback mechanism and includes such pattern recognition tasks as clustering and parameter estimation of probability models (parameter learning). Reinforcement learning enables an agent to learn an optimal plan of action (policy) based on a series of positive and negative rewards that are assigned to a set of actions by an unknown function (e.g. learning when to eat based on the negative rewards of hunger). In supervised learning, an agent receives direct feedback on the correctness of each action and attempts to learn the function defining the action-outcome pairs. Finally, in semi-supervised learning, an agent initially conducts supervised learning, but then engages in unsupervised learning to detect any systematic inaccuracies in the feedback.

2.1.1 Supervised Learning

Given the accessibility of corporate financial data with labels indicating the issuance or non-issuance of an AAER to a corporation, this thesis will employ supervised learning for all classification tasks. In terms of detecting patterns within a set of data, supervised learning can be formally defined as follows. Let

$$\mathbf{D} = \left(\begin{array}{c|cccc} & \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_d \\ \hline \mathbf{x}_1 & x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ \mathbf{x}_2 & x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_n & x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{array}\right)$$

be an $n \times d$ *training* data matrix consisting of n data instances and d *attributes* (or *dimensions*, *fields*, *variables*). Each instance $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{id})$ is a d-tuple with d input values and an associated output value y_i, which was derived from an unknown function

$y = f(\mathbf{x})$, and is the $(i, 1)^{th}$ entry of

$$\mathbf{Y} = \begin{pmatrix} y_{1,1} \\ y_{2,1} \\ \vdots \\ y_{n,1} \end{pmatrix}.$$

Given this training data, the goal of supervised learning is to learn a function, or *hypothesis*, h that both approximates the true function $y = f(\mathbf{x})$ and belongs to the space of possible hypotheses $\mathcal{H}$. This learning can be accomplished by conducting a *search* of $\mathcal{H}$ for a *consistent* hypothesis that maps each data instance to the correct output value (refer to §2.2 for a review of common search techniques); hence, a zero-error approximation. If a consistent hypothesis does not exist, then the learning task is not *realizable* and h will contain a non-zero approximation error. Finally, a *test* data matrix $\mathbf{T}$ ($\mathbf{T} \neq \mathbf{D}$) of $m \geq 0$ unseen data instances $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{id})$ is used to test the generalization ability of h.

If the number of possible output values y_i of $\mathbf{Y}$ is finite, then the supervised learning task is called *classification*. Otherwise, the learning is known as *regression*.

2.1.2 Loss Functions

During the search phase of supervised learning, potential hypotheses are evaluated based on their overall error in approximating the output value of each data instance. Three notable *loss* functions have commonly been used to quantify this approximation error: *absolute value* (L_1 norm; Manhattan distance), *Gaussian squared error* (L_2 norm; Euclidean distance), and *zero-one* ($L_{0/1}$; symmetrical loss) [18, 46]. Each loss function accepts as input both an actual output value y and the hypothesized output y', and returns the approximation error.

The absolute value loss function $L_1(y, y') = |y - y'|$ and the Gaussian squared error loss function $L_2(y, y') = (y - y')^2$ are both based on the assumption that smaller magnitude ap-

proximation errors are better than larger ones. Function L_2 is used in linear regression and classification tasks because it can be minimized to yield a set of optimal function *weights*, or coefficients (assuming normally distributed variation among the actual output values). If the supervised learning task outputs discrete values, then the *zero-one* loss function

$$L_{0/1}(y, y') = \begin{cases} 1 & \text{if } y \neq y'; \\ 0 & \text{if } y = y'. \end{cases} \tag{2.1}$$

presents a suitable option. $L_{0/1}$ assumes that all errors are equally costly and does not penalize predictions that are "more wrong" than others (assuming that there is not a natural ordering among the discrete values) [18]. Note that, for each of these functions, the loss is zero when $y' = y$.

2.2 Linear Classification

(Binary) *linear classification* seeks to learn a *hyperplane* (or *decision boundary*, *surface*) that partitions the attribute space into two distinct *classes* (or *regions*, *half-spaces*), such that the separation between the classes is maximized. In this binary classification task[1], the hypothesis to be learned is a function f that, when set equal to zero, defines a d-dimensional hyperplane

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} \tag{2.2}$$

$$= w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d \tag{2.3}$$

$$= 0, \tag{2.4}$$

where $\mathbf{w} = \langle w_0, \cdots, w_d \rangle$ is a (normal) vector of weights, $\mathbf{x} = \langle x_1, \cdots, x_d \rangle$ is a d-dimensional data instance, and x_0 is a dummy input attribute that is always equal to 1. If the hyperplane perfectly separates the data into two distinct classes, without any overlap, then we say that the data is *linearly separable*. The classification of each data instance $\mathbf{x}_i$

is determined by the decision rule

$$Threshold(z) = \begin{cases} 1 & \text{if } z > 0; \\ 0 & \text{if } z < 0, \end{cases} \tag{2.5}$$

where 0 and 1 represent the two distinct classes and z is the value of $f(\mathbf{x})$. If $z = 0$, then $Threshold(z)$ is undefined and $\mathbf{x}$ will not be classified.

Traditionally, the hypothesis f is learned by identifying the vector of weight values $\mathbf{w}$ that minimizes the Gaussian squared error loss function $L_2(y, Threshold(f))$, summed over all of the n training data instances, and where y is the actual binary class of a data instance [46]. This summation can be computed by means of the $Loss$ function:

$$Loss(g) = \sum_{i=1}^{n} L_2(y_i, g(\mathbf{x}_i)) = \sum_{i=1}^{n} (y_i - g(\mathbf{x}_i))^2, \tag{2.6}$$

where g is a function[2], $\mathbf{x}_i$ is an observation in training data matrix $\mathbf{D}$, and y_i is the actual class of $\mathbf{x}_i$. Depending on whether $Loss$ is both continuous and differentiable (smooth) at all instances $\mathbf{x}_i$, $\mathbf{w}$ can be determined either *analytically*, *exhaustively*, or *numerically* [18, 19, 27].

In general, if $Loss$ is continuous and differentiable[3], then it can be minimized analytically by computing each first-order partial derivative $\frac{\partial Loss}{\partial w_i}$ with respect to $w_i \in \mathbf{w}$, setting them to 0, and solving for $\mathbf{w}$. However, in binary classification, $Threshold$ is discontinuous at the point $x = 0$ and cannot be differentiated. Hence, $Loss(Threshold(f))$ must be

---

[1]Several approaches exist to perform linear classification with $m > 2$ classes. One notable approach partitions the attribute space into m distinct regions using m hypothesis functions

$$f_i(\mathbf{x}) = \mathbf{w}_i \cdot \mathbf{x} \qquad i = 1, \ldots, m,$$

where data instance $\mathbf{x}$ is assigned to the i^{th} class if $f_i(\mathbf{x}) > f_j(\mathbf{x})$ for all $j \neq i$. If $f_i(\mathbf{x}) = f_j(\mathbf{x})$, then the classification is undefined. The reader is referred to [18] for a detailed account of the various m-ary linear classification techniques.

[2]The function g may be composite, such as $g = Threshold(f)$, where f is the linear classification hypothesis.

minimized via an exhaustive or numerical search procedure.

Exhaustive search algorithms, in the worst case, enumerate every possible solution to an optimization task. If allowed to run to completion, these algorithms are guaranteed to identify an optimal solution(s), but possibly in an exponential amount of time [27]. Common exhaustive techniques that have been applied to function optimization include *brute force*, *branch-and-bound*, and *backtracking* [27, 40].

Since an exhaustive search can become computationally expensive or infeasible, *numerical search* procedures, or *heuristics*, may be utilized to approximate, if not identify, an optimal solution. These heuristics explore a solution space by iteratively generating solutions that follow the local curvature and gradient of the space (assuming the objective function of interest is differentiable). The search procedure will constantly generate *stronger* solutions by exploring in the direction of the greatest improvement, or objective function value (opposite direction for minimization tasks). Classic numeric search heuristics include the *bisection method*, *Newton's method*, and *gradient search* [18, 27, 40, 46].

2.2.1 Logistic Regression

Instead of passing $f = \mathbf{w} \cdot \mathbf{x}$ through the linear, hard threshold function $Threshold$, f can be mapped to a non-linear space by means of the historic *logistic*, or *sigmoid*, function

$$L(z) = \frac{1}{1 + e^{-z}}. \tag{2.7}$$

This function is defined over the range $[0, 1]$ and assigns to each data instance $\mathbf{x}_i$ a probability $y' = L(\mathbf{w} \cdot \mathbf{x}_i)$ of belonging to the class 1 (for binary classification tasks). Figure 2.1 presents a plot of the logistic function. y' can then be assigned to a class via a custom

[3]The $Loss$ function is *strictly convex* and guaranteed to possess a global minimum point $\mathbf{w}^*$ if and only if the function is defined over both a linear hypothesis f and a continuous and differentiable decision rule g. If these conditions do not hold, then a global minimum cannot be guaranteed. Several relative (local) minimum weight vectors may exist, or possibly none at all (e.g. when saddle or inflection points exist).

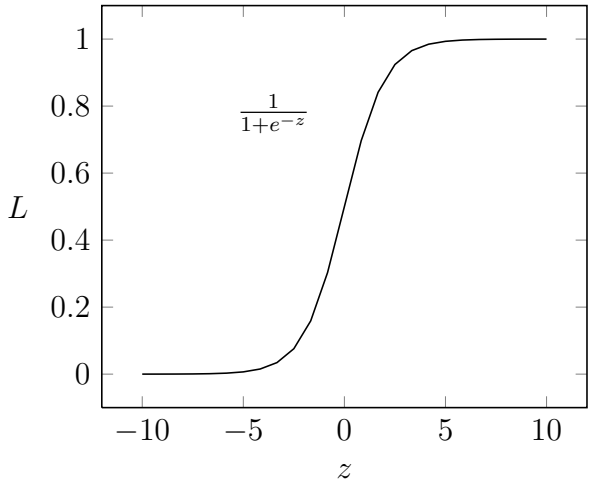


Figure 2.1: Graphical plot of the logistic function.

threshold function

$$Class_t(x) = \begin{cases} 1 & \text{if } x \geq t; \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

where $t \in [0, 1]$ is a threshold value used to separate the two classes.

The process of learning the weight set $\mathbf{w}$ that minimizes the loss function $Loss(L)$ is known as *logistic regression*, and, due to its simplicity, has been frequently employed to solve classification tasks such as medical diagnosis, corporate bankruptcy detection, and college basketball tournament prediction [4, 12, 36]. Since the logistic function is non-linear, $Loss(L)$ might not possess a unique optimal weight set $\mathbf{w}$, and, thus, an exhaustive or numeric search procedure must be used (e.g. gradient descent).

2.3 Non-Linear Classification

While linear classification models such as logistic regression provide a simple and convenient means of linearly partitioning training data into distinct classes, they fail to generalize to more complex real-world problem spaces, which demand non-linear decision boundaries

[18]. Essentially, the linear classifiers can be inadequate at minimizing the classification error within non-linearly separable domains, despite the use of heuristic techniques such as gradient search. By means of *non-linear classification*, we can learn the non-linear decision boundaries that lead to minimum classification error within a training data space. The following subsections present classification algorithms that have demonstrated proficiency in modeling complex, non-linear problem domains. These algorithms were employed within this thesis research to classify financial statement fraud.

2.3.1 Artificial Neural Networks

Since the first inception of an artificial neuron in 1943 by McCulloch and Pits, artificial neural networks (ANNs) have been used extensively to solve various classification and modeling tasks [49]. Based on the processes and properties of the human brain, ANNs have been trained and represented in a variety of fashions, most notably supervised learning with a feed-forward, multi-layered network structure. Using one or more *hidden* layers, ANNs can be adjusted to model non-linear functions and are ideal tools for solving complex classification problems.

As can be seen in Figure 2.2, an ANN emulates the biological neural system with three primary components: neurons (nodes), synaptic connections (edges), and connection strength values (weights). Given an input vector $\mathbf{x} = \langle x_1, x_2, \cdots, x_d \rangle$ with d attribute values, each value x_j is "fed" into the corresponding input node I_j; the set of input nodes define an *input layer*. Then, each input node I_j feeds x_j towards each of the hidden nodes H_i for which it shares a connection. Upon receipt of all input layer signals x_j, H_i computes its *net activation*, net_i, as the linear combination of the input layer signals and the weights w_{ji}, which exist at each edge:

$$net_i = \sum_{j=1}^{c} w_{ji} x_j, \tag{2.9}$$

where c is the number of input edges directed into hidden node H_i. The net activation net_i is subsequently mapped onto a bounded interval by means of a (typically non-linear)

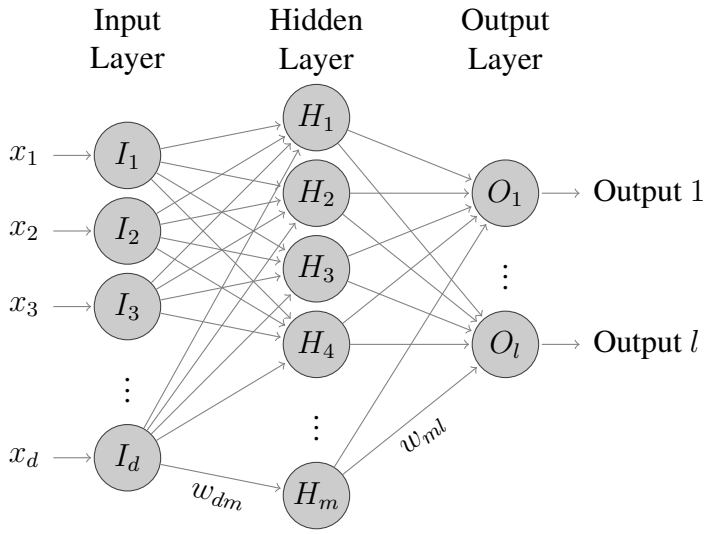


Figure 2.2: An example of a feed-forward, multi-layered ANN and its primary components.

activation function $t_i = f(net_i)$ at H_i, and fed to the next layer in the network. Once an output node O_k receives signals from each of its incident hidden nodes H_i, it computes the net activation net_k and the activation function value $z_k = f(net_k)$, in the same manner as the hidden nodes do. Finally, output signal z_k is compared to the actual output value of $\mathbf{x}$, y_k, and any difference (error) is used in training the edge weights throughout the network.

This supervised learning activity is known as *backpropogation* and attempts to minimize the error according to the $Loss$ function (refer to §2.2). Typically, the gradient descent numerical search technique is used to enforce this optimization by updating the network weight values in a direction that will reduce the training error [46]. Since gradient descent depends on the partial differentiation of an activation function, each hidden and output node must support an activation function that is both continuous and differentiable, across all net activation input values. Due to this constraint, the logistic function, L, is traditionally used as the activation function within hidden and/or output nodes [18]. Additionally, since L is monotonic (always increasing) and bounded to the range $[0, 1]$, undesirable local extrema and large changes in value can potentially be eliminated from the $Loss$ training error [18].

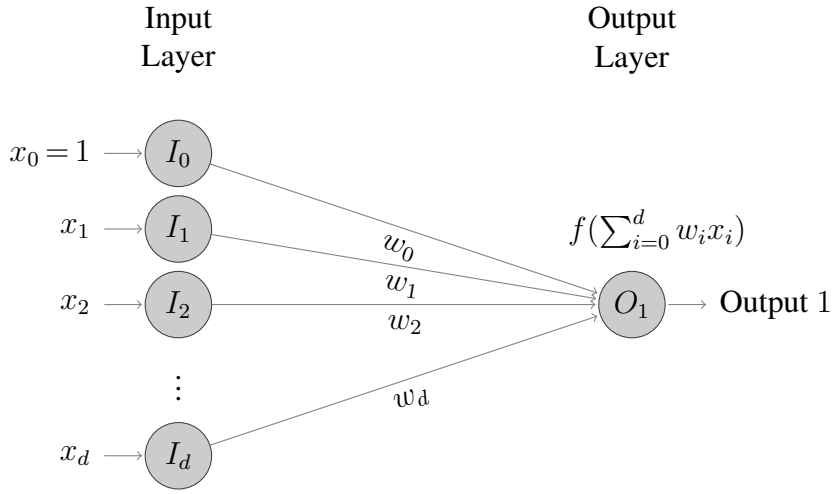


Figure 2.3: Illustration of a single-layered ANN structure that performs logistic regression.

Furthermore, ANNs are subject to *overfitting*[4] when there are too many free model parameters (e.g. input nodes, hidden nodes, and non-zero weights), and the relationship between the output and input values cannot be easily discerned [41]. Hence, a tradeoff between performance and comprehensibility arises while implementing such network models [34]. An optimal network topology will depend on the classification problem at hand and the proper selection of free model parameters. If too many free parameters are utilized, then model generalization will be poor; conversely, if too few parameters are used, then the training data will not be learned adequately [34]. *Regularization* techniques attempt to address such decisions by minimizing the total cost of a hypothesis, while incorporating both the classification loss and the complexity of the hypothesis [34, 46].

Finally, note that single-layered ANNs with logistic activation functions are essentially performing logistic regression. As can be seen in Figure 2.3, each input node I_i and its corresponding edge weight w_i represent the weighted term $w_i x_i$ in a regression equation.

[4]A classifier model is subject to *overfitting* when it begins to memorize the patterns in a training dataset. Typically, these models possess poor predictive performance and do not generalize well to inputs that have not been seen before. A small deviation from the memorized training patterns could generate a large fluctuation in classification error [18, 46].

Hence, via an ANN with only one input layer and output node, we can compute the hyperplane, or decision boundary, that separates the training data into distinct classes (or an approximation thereof if the data is not linearly separable).

2.3.2 Evolutionary Algorithms

When a pattern classification hypothesis space becomes too complex (e.g. slope discontinuity, non-linearity, or a large problem domain), traditional analytic and numeric search techniques can become ineffective and computationally slow [21]. To speed up the search process, we can employ *stochastic* pattern recognition techniques that rely largely on randomness to find an ideal hypothesis or model. *Evolutionary algorithms* (EAs) are a class of search algorithms that are suitable for performing optimization tasks within complex problem domains. In the following subsections, two of the most successful EAs — *genetic algorithms* and *estimation of distribution algorithms* — will be introduced.

Genetic Algorithms

Based on the fundamental principles of natural selection and survival of the fittest, genetic algorithms (GAs) have been successfully applied to a diverse range of optimization tasks, such as sports scheduling, traveling salesman tours, computer circuitry design, portfolio optimization, and protein secondary structure prediction, to name a few [3, 21, 24]. A canonical GA "evolves" a *population* of candidate solutions, or *chromosomes*, according to a *fitness function* that indicates the quality of a chromosome. Each chromosome consists of a set of *genes*, each representing a parameter of an optimization task. The value of each gene is known as an *allele* and represents one of the possible values for the associated task parameter.

During each algorithm iteration, or *generation*, *mutation* and *recombination* variation operators are applied to *parent* chromosomes with the intent of generating *offspring* chromosomes of higher fitness and diversity. This variation allows GAs to explore uncharted

areas of the solution space and avoid becoming trapped in local minima (maxima) regions [21]. Pseudocode of a simple GA and its basic operations is presented in Figure 2.4.

Genetic Algorithm

1. The initial population of chromosomes, $P(0)$, is uniformly sampled from X.
2. At iteration t, a subcollection, $P'(t) \subseteq P(t)$, of high-fitness chromosomes is selected.
3. The members of $P'(t)$ are recombined to form a collection of new chromosomes, $C(t)$.
4. The members of $C(t)$ have some of their genes mutated.
5. A subcollection, $R(t) \subseteq P(t)$, of low-fitness chromosomes is selected.
6. $P(t+1) \leftarrow P(t) - R(t) + C(t)$.
7. Unless termination criteria are met, return to step 2.

Figure 2.4: Pseudocode of a simple genetic algorithm.

The performance and search time of a GA largely depend on the algorithm parameters (e.g. population size, mutation and recombination rates, and chromosome representation) and their suitability for the problem domain [21]. Without properly tuning these parameters, the search process can become computationally expensive and fail to identify global optimum solutions (if such solutions exist) [21, 39]. Thus, care must be taken in adapting the original problem context to the solution space where evolution occurs. Finally, an additional drawback of GAs is that they are based on the *building block* assumption that high-fitness solutions are located "near" other high-fitness solutions in the search space [3, 21, 39]. Thus, GAs might not be as effective when this assumption is violated, such as when specific dependencies among genes dictate higher fitness values.

Estimation of Distribution Algorithms

In response to the notable limitations of GAs, such as parameter tuning and the building block assumption, *estimation of distribution algorithms* (EDAs) were introduced in the late 1990s [39]. Operating much like a GA, an EDA generates a new population of candidate solutions by statistically sampling from a gene-wise probability distribution, which is estimated from the selected solutions of the previous generation. Due to this statistical feature, an EDA is able to explicitly model the interrelations, or building blocks, among the genes of a chromosome solution. Furthermore, the sampling procedure has historically replaced the mutation[5] or recombination operations within the evolution process. The primary issue of an EDA is the procedure by which its probability distribution is estimated, or "learned" [39]. Since it may be infeasible to construct the best (according to some criterion) structure and parameter set of a distribution, heuristic search methods (e.g. greedy search and simulated annealing) are typically employed to estimate the distribution [3, 39].

Several EDAs have been proposed for both discrete and continuous (real-value) optimization tasks, and each algorithm induces a probability distribution that models one of three types of gene dependencies: *univariate*, *bivariate*, or *multivariate*. These different EDA models and their leading implementations will be discussed in the following paragraphs. (Note that only discrete probability distributions and EDAs will be reviewed. Refer to [39] for a detailed account of the continuous variants.)

Univariate EDAs do not learn dependencies among genes, but instead record the frequency of alleles for each gene. Offspring chromosomes are then composed by sampling a joint probability distribution that factorizes as a product of n univariate and independent

[5]To compensate for possible imprecision in the learned probability distribution models, mutation can be implemented as a form of *local search* that complements the sampling procedure and fine-tunes the search for high-fitness chromosomes. *Hybrid* EDAs utilize such a mutation operator and have demonstrated success in conducting more-efficient searches than mutation-less EDAs, within the space of feasible chromosomes [3, 39, 50].

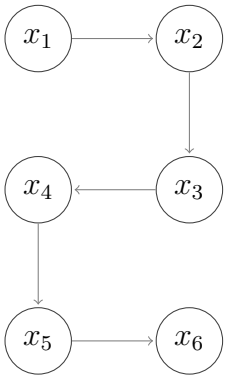


Figure 2.5: Illustration of a bivariate gene-dependency graph for a hypothetical optimization task with six parameters. Each numerically-labeled gene x_i represents the i^{th} task parameter and is connected to either a single parent gene, child gene, or both. This particular graph structure is enforced by the MIMIC EDA.

probability distributions:

$$P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i), \tag{2.10}$$

where x_i is the i^{th} gene of a chromosome $\mathbf{x}$. Notable univariate EDAs include the *Univariate Marginal Distribution Algorithm* (UMDA) and *Population-Based Incremental Learning* (PBIL) [39].

Bivariate EDAs model dependencies between pairs of genes, typically in the form of a dependency network or tree graph. These pairwise dependencies can be represented by a probability distribution model

$$P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|parent\text{-}of(x_i)), \tag{2.11}$$

where $parent\text{-}of(\cdot)$ indicates a "parent" gene that is connected to x_i in a dependency graph structure. Leading examples of bivariate EDAs include the *Mutual Information Maximization for Input Clustering* (MIMIC), *Combining Optimizers with Mutual Information Trees* (COMIT), and *Bivariate Marginal Distribution Algorithm* (BMDA) [1, 3, 39]. Figure 2.5

illustrates a bivariate gene-dependency structure for a hypothetical optimization task.

Providing a more-realistic representation of difficult optimization problems, multivariate EDAs model the dependencies among potentially larger sets of genes via probability graph models, such as Bayesian networks, trees, forests, and polytrees [1, 7, 39]. A general multivariate probability distribution model can be defined as

$$P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i | parents\text{-}of(x_i)), \tag{2.12}$$

where $parents\text{-}of(\cdot)$ is the set of parent genes of x_i. While multivariate models allow for greater accuracy in representing underlying relationships among genes, such models are computationally more difficult to infer, as compared to univariate and bivariate versions. It has been shown that both exact and approximate inference algorithms for multiply-connected[6] network models can possess an exponential time and space complexity in the worst case [11, 46]. Notable multivariate EDAs include *Bayesian Optimization Algorithm* (BOA), its hierarchical extension (hBOA), *Extended compact Genetic Algorithm* (EcGA), and *Estimation of Bayesian Networks Algorithm* (EBNA) [1, 3, 39]. Figure 2.6 illustrates the multivariate gene-dependency structure for a hypothetical optimization problem.

Each of the EDA implementations cited above maintain directed, acyclic graph (DAG) structures, which define a natural dependency ordering among genes and can be efficiently evaluated via graph traversal algorithms (e.g. depth-first search or breadth-first search) [40]. Recent research has experimentally verified that *undirected* graph models may perform better than their directed counterparts on many optimization tasks [3]. Since undirected graph models lack a natural gene-dependency ordering, it is costly to learn and sample from them. To overcome this bottleneck, existing undirected EDAs have reduced the complexity of the graph models and/or partially converted them into simpler, directed structures. However, these approaches can potentially reduce the model flexibility and the effectiveness of the

[6] A network graph is *multiply-connected* when there is more than one simple path (directed or undirected) between any two nodes in the network.

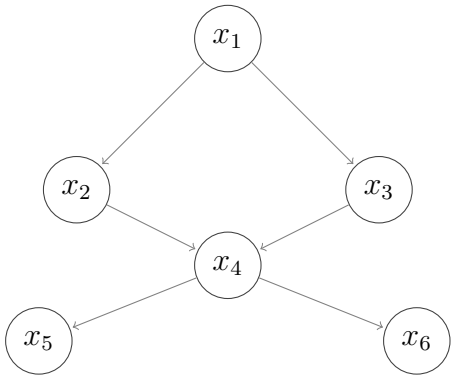


Figure 2.6: An example multivariate gene-dependency graph for a hypothetical optimization task with six parameters. Each numerically-labeled gene x_i represents the i^{th} task parameter and is connected to one or more parent and child genes. This particular graph structure is enforced by the BOA and EBNA EDAs, to name a few.

model learning procedure [3].

The *Markovian Learning Estimation of Distribution Algorithm* (MARLEDA) [3] is a recently-introduced EDA that learns and samples from a *Markov random field* (MRF) probability model to address the before-mentioned constraints posed by undirected graph structures. An MRF defines the joint probability distribution of a set of finite random variables, $\{X_1, \cdots, X_n\}$, in terms of (typically smaller) joint or conditional probability distributions of subsets of the random variables (or *local characteristics*). Each MRF consists of a *random field*

$$\begin{cases} P(X) > 0, & \forall X \in \mathbf{X} \\ \sum_{X \in \mathbf{X}} P(X) = 1, \end{cases} \tag{2.13}$$

and a *neighborhood system*, ∂, defined by an undirected graph:

$$\begin{cases} i \notin \partial(i), & \forall (1 \leq i \leq n) \\ i \in \partial(j), & \text{if and only if } j \in \partial(i) \end{cases} \tag{2.14}$$

where $\mathbf{X} = \prod_{i=1}^{n} X_i$ is the space of *configurations* of $\{X_1, \cdots, X_n\}$. Given the neighbor-

hood ∂, the *Markov property* induces an MRF on P:

$$P(x_i|x_j, i \neq j) = P(x_i|x_k, k \in \partial(i)). \tag{2.15}$$

Modeling the discrete (nominal) genes $\{x_1, \cdots, x_n\}$ with the random variables $\{X_1, \cdots, X_n\}$, MARLEDA learns the statistical dependencies between random variable pairs by means of Pearson's χ^2 nonparametric hypothesis test. This test compares two frequency distributions — observed and expected — to determine the degree of similarity, or confidence level of dependence, between two genes. If the similarity is significant, then the two genes become neighbors; conversely, if the two genes began as neighbors and the similarity is not significant, then they should become non-neighbors. Based on this concept of gene similarity, MARLEDA constructs the MRF neighborhood system. Starting from an empty neighborhood $\partial(i) = \emptyset$, one of the following two tests is randomly selected and iteratively performed:

1. Randomly select $ModelAdds$ pairs of non-neighbor genes and test each pair for neighbor status. Each pair with a confidence level of dependency of at least $ModelAddThresh$ is declared neighbors within the MRF neighborhood.

2. Randomly select $ModelSubs$ pairs of neighbor genes and test each pair for non-neighbor status. Each pair with a confidence level of dependency less than $ModelSubThresh$ is declared non-neighbors within the MRF neighborhood.

This learning procedure allows MARLEDA to effectively learn the multivariate depedencies among genes without placing artificial constraints on the complexity of the undirected MRF model.

Finally, to generate new chromosomes, MARLEDA samples the MRF model via a *Markov chain Monte Carlo* (MCMC) process, which iteratively proceeds for a specified number of iterations or until the allele distribution of a new chromosome reaches a steady-state. While complete convergence to a stationary allele distribution may not be attainable,

Markov Chain Monte Carlo Sampling

1. $\mathbf{x}^{\text{new}} \leftarrow$ a random chromosome from a (selected) subcollection $\mathcal{P}$ of the population.
2. Randomly select a gene x_i^{new}.
3. Compute $P(x_i|x_k, k \in \partial(i))$.
4. $x_i^{\text{new}} \leftarrow$ sample according to $P(x_i|x_k, k \in \partial(i))$.
5. Unless termination criteria are met, return to step 2.

Figure 2.7: Pseudocode of the Markov chain Monte Carlo sampling procedure.

the MCMC procedure at least enables MARLEDA to efficiently sample from an undirected probability graph model. The sampling algorithm is presented in Figure 2.7 and pseudocode of MARLEDA is displayed in Figure 2.8.

Essentially, MARLEDA leverages the potential benefits of undirected graph models (via the MRF neighborhood structure), while overcoming many of the learning and sampling constraints inherent in predecessor EDA models. Due to these contributions, MARLEDA was utilized within this thesis research to perform financial statement fraud classification.

2.4 Multi-Objective Optimization

Typically, in many real-world optimization and classification problems, more than one criterion, or objective function, will be needed to measure the quality of a candidate solution. Each of these objectives may be incommensurable and represent conflicting goals. Hence, a solution that is globally optimal across all objectives might not exist, and tradeoffs among objective values will be made. A decision maker will then implicitly select one or more ac-

---

MARLEDA

1. The initial population of chromosomes, $P(0)$, is uniformly sampled from X and the Markov Random Field model, $M(0)$, is initialized.

2. At iteration t, a subcollection, $P'(t) \subseteq P(t)$, of high-fitness chromosomes is selected.

3. $M(t)$ is learned to model the dependencies among members of $P'(t)$.

4. A collection of new chromosomes, $C(t)$, is produced by sampling $M(t)$, using the Markov chain Monte Carlo algorithm. The sampling procedure continues until the allele (gene value) distribution of each new chromosome stabilizes.

5. A subcollection, $R(t) \subseteq P(t)$, of low-fitness chromosomes is selected.

6. $P(t+1) \leftarrow P(t) - R(t) + C(t)$.

7. Unless termination criteria are met, return to step 2.

Figure 2.8: Pseudocode of the MARLEDA algorithm.

ceptable solutions based on specified goals. The following hypothetical multiple-objective scenario captures the essence of this tradeoff condition. Consider a company ABC that hires individuals based on their performance on an intelligence test IQ and the number of years of previous work experience n_{work}. Assume, without loss of generality, that ABC intends to maximize IQ and n_{work}. If no applicant maximizes both of the job criterion, then ABC will have to select an individual that did not perform optimally on one of the criteria. Thus, various criterion priorities or preferences by ABC will largely dictate the individual that is hired.

The simultaneous optimization of all objective functions is known as *multi-objective optimization* (or *multi-objective programming*). To formalize the notion of multi-objective

optimization, four relevant concepts will be defined. Assume, without loss of generality, that we seek to minimize m objective functions, $f_i(\mathbf{x}), 1 \leq i \leq m$, where $\mathbf{x}$ is a candidate solution in domain Ω. The four concepts are defined as follows:

1. **Pareto Dominance:** A solution $\mathbf{x}$ is said to (Pareto) dominate a solution $\mathbf{y}$ (denoted by $\mathbf{x} \succ \mathbf{y}$) if and only if

$$(\forall i, 1 \leq i \leq m : f_i(\mathbf{x}) \leq f_i(\mathbf{y})) \wedge (\exists i, 1 \leq i \leq m : f_i(\mathbf{x}) < f_i(\mathbf{y})). \tag{2.16}$$

   So, $\mathbf{x}$ must be as good as $\mathbf{y}$ on all objective values and better on at least one objective. Note that the notion of Pareto dominance defines a partial order on the set of all solutions Ω. Furthermore, two solutions are **Pareto independent** when neither of them dominates the other.

2. **Pareto Optimality:** A solution $\mathbf{x}$ is said to be Pareto optimal if and only if $\nexists \mathbf{y} : \mathbf{y} \succ \mathbf{x}$. That is, $\mathbf{x}$ is not dominated by any other solution in Ω. A **zenith** solution possesses the optimal value for each of the m objectives. Typically, this solution is infeasible and is used to measure the quality of tradeoff solutions.

3. **Pareto Optimal Set:** The set $\mathcal{P}_\mathcal{S}$ of all Pareto optimal solutions: $\mathcal{P}_\mathcal{S} = \{\mathbf{x} | \nexists \mathbf{y} : \mathbf{y} \succ \mathbf{x}\}$.

4. **Pareto Optimal Front:** The set $\mathcal{P}_\mathcal{F}$ of all objective function values corresponding to the solutions in $\mathcal{P}_\mathcal{S}$:

$$\mathcal{P}_\mathcal{F} = \{(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})) | \mathbf{x} \in \mathcal{P}_\mathcal{S}\}. \tag{2.17}$$

Given these formal definitions, the primary goal of a multi-objective optimization procedure is to identify as many solutions as possible that are close to the Pareto optimal front or that exist in the Pareto optimal set [9, 20]. Due to the various preferences that a decision

maker may possess in regards to an acceptable solution, it is also desirable to search for a *diverse*, or uniformly spread out, set of solutions along the Pareto front [9, 16]. It should be noted that the true Pareto optimal front and set are typically unknown during the operation of an optimization procedure because many real-world problem domains are too complex to enumerate all solutions in a polynomial (or efficient) amount of time [7]. Thus, the resulting Pareto set of an optimization algorithm typically serves as an approximation to the true Pareto optimal set.

Multi-objective optimization has traditionally been performed by means of two techniques: *mathematical programming* and *evolutionary computation* [9]. These techniques are presented in the following subsections.

2.4.1 Mathematical Programming Approach

Since the advent of the operations research discipline around the time of World War II, numerous optimization techniques have been devised to minimize (maximize) both linear and non-linear objective functions, usually subject to a set of constraints [27]. Termed *mathematical programming*, these techniques (e.g. linear, quadratic, and integer programming) have been applied to both single- and multiple-objective optimization problems. Several approaches have been utilized within this discipline to simultaneously optimize multiple objective functions.

The *weighted-sum* approach (also refer to §2.4.2) transforms the multi-objective problem into a single-objective version by linearly-combining the objectives into a single, weighted scalar function, with a real-value weight factor for each objective function value. This single-objective task can then be solved via traditional convex programming algorithms. While simple, this approach possesses several drawbacks, such as an ad hoc determination of weights, combination of incommensurable (conflicting) objectives, and inability to represent non-convex regions of the Pareto front [9, 16, 20, 21, 34].

A *lexicographic* procedure also maps a multi-objective problem into a single-objective version by treating each objective separately, sorting the objectives from the most important

to least important, and then ranking solutions according to the lexicographic order of their objective values [9, 16]. This rank defines a total order on the solutions. Hence, when two or more solutions are compared against each other, the best solution will be the one possessing the lexicographically-superior set of objective values.

Finally, another approach involves converting all but one of the objective functions into constraints and solving the single-objective problem with mathematical programming techniques such as integer or linear programming [20].

2.4.2 Evolutionary Computation Approach

A significant body of research has utilized evolutionary algorithms (EAs) to evolve candidate solutions (chromosomes) whose fitness is measured by a set of objective functions. EAs present an ideal approach because they can operate on problems with discontinuous or concave Pareto fronts and maintain an entire population of solutions, which fosters solution diversity [3, 9, 16, 34]. The main motivation, however, for using EAs to solve multi-objective optimization tasks is that EAs can identify several members of the Pareto optimal set in a single run of the algorithm, instead of conducting a series of connected linear or integer programs, as is the case of mathematical programming. Two approaches have most-commonly been employed to represent and simultaneously optimize multiple objectives: *weighted-sum* and *Pareto-based*.

The *weighted-sum* approach, which was introduced in §2.4.1, uses the weighted scalar function of the multiple objectives to represent the fitness measure of a solution. Thus, during the selection and population reduction phases of an EA, each chromosome will be evaluated according to this single, weighted sum. The drawbacks of this approach mirror those inherent in the mathematical programming weighted-sum approach.

In the *Pareto-based* approach, each objective function is treated separately (i.e. not linearly combined) and the concept of Pareto optimality is incorporated into the selection mechanism of an EA. A few of the most widely-used Pareto-based EAs are described as follows.

The Multi-Objective Genetic Algorithm (MOGA) [9] of Fonseca and Fleming assigns each chromosome in the population a rank based on the number of other chromosomes that the chromosome is dominated by; this rank serves as the fitness metric of a chromosome. All non-dominated chromosomes have the same probability of being selected for evolution and are assigned the same rank.

Combining the properties of different multi-objective EAs, the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [2, 9, 29, 31] assigns to each chromosome a "strength" fitness value that is based on the number of non-dominated chromosomes that dominate the chromosome. Non-dominated chromosomes are stored in a secondary population and randomly released into the current population as elite chromosomes. Additionally, a nearest-neighbor clustering technique is utilized to promote diversity among the set of non-dominated chromosomes.

Nondominated Sorting Genetic Algorithm-II (NSGA-II)

Finally, as one of the most successful and applied multi-objective EAs, the Nondominated Sorting Genetic Algorithm-II (NSGA-II) [14] uses elitism and a crowded comparison operator to efficiently evolve diverse non-dominated chromosomes and to help prevent the loss of quality chromosomes once they have been identified. During each iteration of NSGA-II, three primary activities are conducted to generate a Pareto set of chromosomes.

First, a *non-dominated sorting* procedure assigns each chromosome s_i (parent or offspring) to a *front* F_j ($1 \leq j \leq i$), based on the number of chromosomes that it is dominated by. All non-dominated chromosomes belong to front F_0, all chromosomes that are dominated by the next fewest number of chromosomes are assigned to front F_1, and so on until all chromosomes have been assigned to a front.

Second, a *crowding distance*, $i_{distance}$, is computed for each chromosome s_i that is eligible to advance to the next generation. This diversity metric estimates the density of chromosomes surrounding a particular chromosome s_x in a front F_j. The following set of procedures are used to calculate the crowding distance for each $s_i \in F_j$:

1. Set $i_{distance}(s_i) = 0 \;\;$ for all $1 \leq i \leq |F_j|$.

2. For each objective $k \leq m$, sort the chromosomes in front F_j in ascending order with respect to the objective function f_k, and set $i_{distance}(s_1^k) = i_{distance}(s_{|F_j|}^k) = \infty$, so that the most extreme-valued chromosomes can be preserved.

3. Calculate $i_{distance}$ for all s_i in which $i_{distance}(s_i) \neq \infty$:

$$\sum_{k=1}^{m} \sum_{i=2}^{|F_j|-1} i_{distance}(s_i) + f_k(s_{i+1}^k) - f_k(s_{i-1}^k), \tag{2.18}$$

   where s_{i+1}^k and s_{i-1}^k are the two neighboring chromosomes of s_i in front F_j with respect to f_k.

Finally, a *crowding comparison* operator $\geq_n$ sorts each of the eligible chromosomes in descending order according to the following partial order:

$$i \geq_n j : \text{ if } (i_{rank} < j_{rank}) \vee ((i_{rank} = j_{rank}) \wedge (i_{distance} > j_{distance})), \tag{2.19}$$

where i_{rank} is the index of the front of which chromosome s_i is a member. The top n chromosomes are then advanced to the next generation.

Within this thesis research, both the weighted-sum and Pareto-based approaches to evolutionary multi-objective optimization will be utilized in the classification experiments. Due to the documented success and applications of NSGA-II [2, 9, 29, 31], it will be employed as the algorithm behind Pareto-based multi-objective GA and EDA rule classification models. The *multiobjective MARLEDA* (mMARLEDA) model developed by Alden [3] is based on the NSGA-II framework and will be used to perform the Pareto-based EDA experiments. mMARLEDA is a multi-objective variant of MARLEDA, which was introduced in §2.3.2.

2.5 Fuzzy Sets and Fuzzy Logic

2.5.1 Fuzzy Set Theory

First conjectured in 1965 by mathematician Lotfi Zadeh, *fuzzy set theory* is a logical system that measures the degree to which an event or proposition is true [53]. Contrary to traditional Boolean logic, which maps a proposition to a hard threshold value of *true* or *false*, fuzzy set theory assigns to the proposition a degree of truth in the range $[0, 1]$. Consider the proposition "The *Temperature* is *Cold*." The goal of fuzzy set theory is to measure the degree to which the *fuzzy variable* (or *linguistic variable*) "Temperature" is a member of the *fuzzy set* (or *linguistic value*) "Cold". A fuzzy set $\mathcal{F}$ defined on a domain of input values $\mathcal{U}$ is characterized by a *membership function* $\mu_{\mathcal{F}}$ that maps an input value $x \in \mathcal{U}$ to the interval $[0, 1]$. The process of assigning a membership value $\mu_{\mathcal{F}}(x)$ to a value x is known as *fuzzification*.

In regards to the example temperature proposition, the *Temperature* fuzzy variable may possess one or more fuzzy sets, such as *Cold*, *Mild*, and *Hot*. Figure 2.9 presents a graphical representation of these fuzzy sets and their corresponding membership functions. In this example, the *Mild* fuzzy set is characterized by a *trapezoidal* membership function, which is defined as

$$\mu_{Trap}(x) = \begin{cases} 0, & x < a, x > d \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b < x < c \\ \frac{d-x}{d-c}, & c \leq x \leq d \end{cases} \tag{2.20}$$

where a and b are the minimum (leftmost) and maximum (rightmost) base points, respectively, and c and d are the left and right "plateau" points, respectively.

The *Cold* and *Hot* sets are characterized by *ramp* functions, which behave like a trapezoidal function up to a plateau point b, at which the membership value remains constant for all points $b' > b$ (or $b' < b$ for a leftward ramp). Two other common membership functions

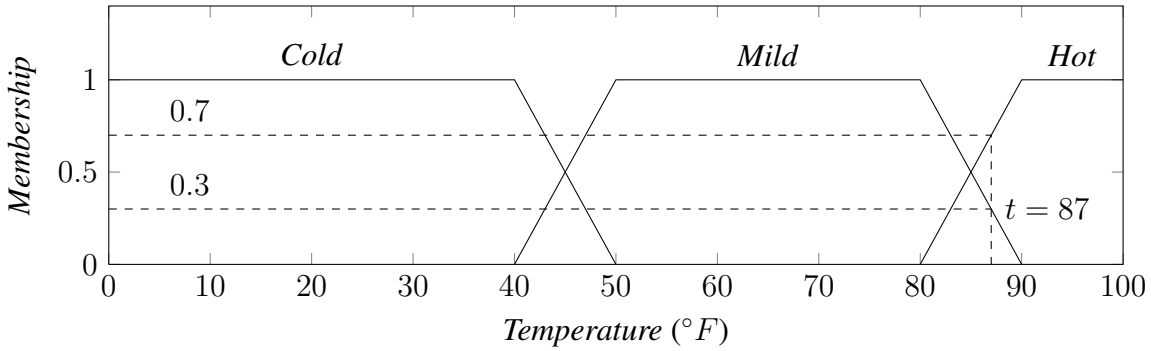


Figure 2.9: An illustration of fuzzy sets and membership functions, applied to a hypothetical *Temperature* fuzzy variable. Consider a daily temperature $t = 87\ ^\circ F$. A vertical line through $t = 87$ intersects both the *Mild* and *Hot* membership functions, implying that the temperature is at least partially *Mild* and *Hot*, and not *Cold* at all. The intersection points — $\mu_{Cold}(87) = 0$, $\mu_{Mild}(87) = 0.3$, and $\mu_{Hot}(87) = 0.7$ — indicate the degree of *Membership* in each set. (Note that these membership values were calculated with the trapezoidal and ramp functions.)

are the *triangular* and *normalized Gaussian*, and they are defined, respectively, as follows:

$$\mu_{Triangle}(x) = \begin{cases} 0, & x < min, x > max \\ \frac{x-min}{center-min}, & min \leq x \leq center \\ \frac{x-max}{center-max}, & center < x \leq max \end{cases} \tag{2.21}$$

where $center$ is the midpoint of the function;

$$\mu_{normal}(x) = e^{-\frac{(x-\zeta)^2}{2\sigma^2}}, \tag{2.22}$$

where ζ is a population mean and σ is a population standard deviation [24, 52].

Since many of the SAS No. 99 red flags are defined in terms of the degree of some event, fuzzy set theory presents an ideal approach to representing the red flag values. As outlined in Chapter 4, each red flag is characterized by a fuzzy variable in a fuzzy logic

rule, and possesses the fuzzy sets *Very Low*, *Low*, *Medium*, *High*, and *Very High*.

2.5.2 Fuzzy Logic

Fuzzy logic is a methodology for reasoning with *logical expressions* that describe membership in fuzzy sets. A logical expression consists of a conjunction or disjunction of fuzzy propositions, and is assigned a truth value that is a function of the truth value of each component proposition. For instance, the logical expression "*Sales* are *Low* and *Expenses* are *Very High*" is a conjunction of propositions. The most standard operators for evaluating the truth μ of a logical expression with proposition fuzzy sets A and B are defined as follows [29, 46, 52]:

$$Intersection: \quad \mu_{A\wedge B}(x,y) = \min\{\mu_A(x), \mu_B(y)\}, \tag{2.23}$$

$$\mu_{A\wedge B}(x,y) = \mu_A(x) \cdot \mu_B(y) \tag{2.24}$$

$$Union: \quad \mu_{A\vee B}(x,y) = \max\{\mu_A(x), \mu_B(y)\} \tag{2.25}$$

$$Complement: \quad \mu_{\neg A}(x) = 1 - \mu_A(x). \tag{2.26}$$

A *fuzzy rule-based system* (FRBS) is composed of a set of fuzzy logic *rules*, or logical implications. A hypothetical fuzzy logic rule is "**if** *Sales* are *Low* and *Expenses* are *Very High* **then** *Profit* is *Very Low*." Each rule is expressed in an **if** . . . **then** form, in which the **if** condition (*antecedent*) contains a logical expression and the **then** assignment (*consequent*) represents an output fuzzy proposition. The truth value of the consequent proposition is inferred by applying an *implication* operator to the antecedent and consequent fuzzy sets. Two of the most common rule implication operators for an implication $A \to C$ are defined as follows [46, 52]:

$$Mamdani\ Implication: \quad \mu_{A\to C}(x,y) = \min\{\mu_A(x), \mu_C(y)\} \tag{2.27}$$

$$Dienes\text{-}Rescher\ Implication: \quad \mu_{A\to C}(x,y) = \max\{1-\mu_A(x), \mu_C(y)\}, \tag{2.28}$$

where x is an input value for the antecedent set A and y is an input value for the consequent set C. An implication operator is applied at each point y in the domain of C to trace a new membership function $\mu_{A \to C}$, defined over the domain of C. The center of mass, or centroid, of this function is then (traditionally) used to represent the final truth value of the consequent proposition.

If the consequent of a fuzzy logic rule is represented by a distinct class, then the rule can serve as a classifier. A set of such rules is known as a *fuzzy rule-based classifier* (FRBC). More formally, in an n-dimensional pattern classification task with M classes, an FRBC rule takes the following form:

$$\text{Rule } R_q : \textbf{ if } x_1 \text{ is } A_{q1} \text{ and } \cdots \text{ and } x_n \text{ is } A_{qn} \textbf{ then } \text{Class } C_q \text{ with weight } CF_q, \tag{2.29}$$

where R_q is the q^{th} fuzzy logic rule, $\mathbf{x} = \langle x_1, \cdots, x_n \rangle$ is an n-dimensional input (pattern) vector, A_{qi} is an antecedent fuzzy set, $C_q \in M$ is the consequent class, and $CF_q \in [0, 1]$ is a *rule weight* representing the degree of certainty of the classification C_q. Given a training data matrix $\mathbf{D} = \{\mathbf{x}_p\}_{p=1}^m$ and a matrix $\mathbf{Y} = \{y_p\}_{p=1}^m$ of corresponding (actual) classifications, an FRBC engages in supervised learning to improve its classification accuracy. [2, 29, 30, 31, 32] detail the various methods for computing the consequent class and rule weight of an FRBC rule. [32] also outlines an approach for updating the rule weight of each rule after a classification is made.

During each iteration of the supervised learning procedure, one of several approaches may be employed to classify each input vector $\mathbf{x}_p \in \mathbf{D}$ with an FRBC S [29, 30, 31, 32]. The *single winner* rule method classifies an input vector $\mathbf{x}_p$ with the rule $R_{q*} \in S$ that has the maximum product of the *compatibility* grade $\mu_{\mathbf{A}_q}(\mathbf{x}_p) = \mu_{A_{q1}}(x_{p1}) \times \cdots \times \mu_{A_{qn}}(x_{pn})$ and the rule weight CF_q:

$$\max_{R_q \in S} \{\mu_{\mathbf{A}_q}(\mathbf{x}_p) \times CF_q\}, \tag{2.30}$$

where $\mu_{A_{qi}}(\cdot)$ is the membership function of the fuzzy set A_{qi}. A *weighted vote* scheme classifies $\mathbf{x}_p$ with the class $h^* \in M$ that receives the maximum total strength of vote from

the rules in S:

$$\max_{h \in M} \{ \sum_{R_q \in S} \mu_{\mathbf{A}_q}(\mathbf{x}_p) \times CF_q | C_q = h \}, \tag{2.31}$$

where $\mu_{\mathbf{A}_q}(\cdot) \times CF_q$ is the weighted vote cast by a rule R_q for its consequent class C_q. Finally, $\mathbf{x}_p$ can also be classified with the consequent class C_{q*} of the rule R_{q*} that possesses the maximum rule weight CF_{q*} among all rules in S:

$$\max_{R_q \in S} \{CF_q\}. \tag{2.32}$$

Note that if two or more rules each possess the maximum value of either of the above-mentioned objective functions, then a rule will be randomly selected to classify $\mathbf{x}_p$.

Essentially, FBRCs merge the concepts of fuzzy logic and classification, and present a viable alternative to learning accurate and comprehensible classifiers. Encoding each FRBC as a candidate solution (chromosome), EAs have demonstrated success in evolving these systems, using classification accuracy or error as a fitness dimension [2, 3, 29, 30, 31, 32, 34]. This EA approach to generating FRBC solutions was adapted to the financial statement fraud classification task, and its implementation is outlined in Chapter 4.

Chapter 3

FINANCIAL DATA COLLECTION, PREPARATION, AND ANALYSIS

Within the *data mining* discipline[1], a five-step iterative process is customarily employed to perform a *pattern recognition* experiment or task (see Figure 3.1) [16, 19]. Initially, a set of raw data must be collected (step 1). Then, in the second and often most important step, the dataset is *preprocessed* to remove extreme (outlier) observations, estimate missing data, and/or normalize[2] attribute values to a common scale of measurement. After this preprocessing phase, the data can then be *transformed* into a lower-dimensional space with potentially fewer attributes (step 3). This dimensionality reduction helps eliminate redundant and strongly-correlated attributes, and improves the comprehensibility of the data. Finally, the data is used to train and test a pattern recognition model (step 4), and the results of the experiment can be subsequently *analyzed* and *deployed* (step 5).

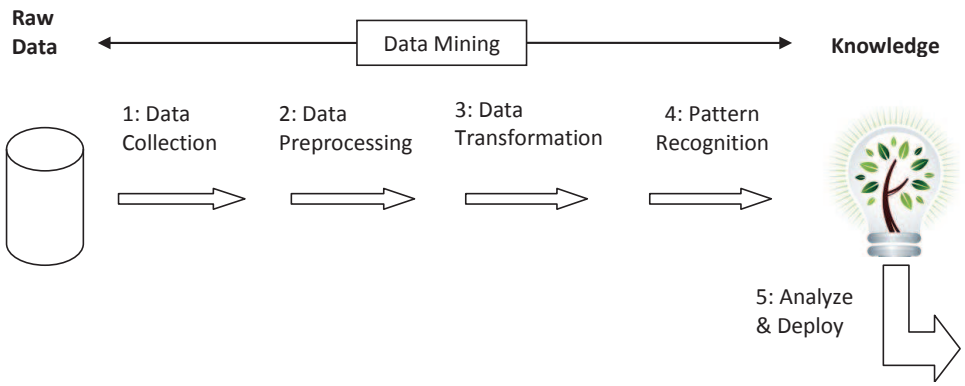


Figure 3.1: The five-step data mining process.

This chapter introduces the financial data that was used within this thesis research, and outlines the data preprocessing and transformation procedures that were applied to the data. Additionally, insight into the important data assumptions and factors that can influence the prediction of financial statement fraud is provided and discussed. Throughout the chapter, the terms *attribute* and *variable* will be used interchangeably to represent a fraud risk factor, or red flag.

3.1 Financial Dataset

The complete collection of data used to classify financial statement fraud is represented by an $n \times (d+1)$ data matrix

$$\mathbf{D} = \left(\begin{array}{c|cccc:c} & \mathbf{Flag}_1 & \mathbf{Flag}_2 & \cdots & \mathbf{Flag}_d & \mathbf{AAER} \\ \hline \mathbf{x}_1 & x_{1,1} & x_{1,2} & \cdots & x_{1,d} & y_1 \\ \mathbf{x}_2 & x_{2,1} & x_{2,2} & \cdots & x_{2,d} & y_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{x}_n & x_{n,1} & x_{n,2} & \cdots & x_{n,d} & y_n \end{array}\right),$$

where each of the n corporate data records $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{id}, y_i)$ is defined by both its set of red flag values for the first year of fraudulent activity and whether it was issued an *Accounting and Auditing Enforcement Release* (AAER) by the Securities and Exchange Commission (SEC) during that year. This binary AAER indicator will serve as a proxy for financial statement fraud (refer to §1.1.1 for an account of this relation).

After conducting the data collection and preprocessing procedures (refer to the following sections below), $n = 479$ unique corporate data records were acquired, consisting of 243 AAER-cited corporations and 236 non-cited corporations, which were "matched" by

[1] In its broadest sense, *data mining* is the act of extracting knowledge from raw, real-world data. When useful information is discovered by means of an automated process, the term *machine learning* is often used synonymously with data mining.

[2] Data can also be *standardized* during the preprocessing step by multiplying each attribute value by the factor (σ/μ), where σ and μ are the standard deviation and mean of the attribute, respectively.

the algorithm presented in §3.2.3. The collection of AAER data records represent a subset of AAER Nos. 623 through 3217, which were issued between November 1994 and December 2010. Additionally, $d = 16$ financial variables were composed to serve as proxies for a subset of the 42 *SAS No. 99* red flags. These variables are denoted in data matrix $\mathbf{D}$ by the column vectors $\langle \mathbf{Flag}_1, \mathbf{Flag}_2, \ldots, \mathbf{Flag}_d \rangle$.

3.2 Data Collection

Before entering the initial, preprocessing phase of the five-step data mining process, the financial data was collected and verified for accuracy. This data acquisition task consisted of the following three procedures. First, custom algorithms were used to extract and compile an initial list of AAER corporations. Second, through a review of related research and the red flag definitions in *SAS No. 99*, financial variables were composed to serve as proxies for a subset of the 42 red flags. Finally, via a financial database, a list of "matching" non-AAER corporations was compiled and the red flag values were extracted for each corporation $\mathbf{x}_i$ in the data file $\mathbf{D}$.

3.2.1 Web Crawler Tool and Text Parsing Algorithm

To conveniently obtain an up-to-date list of AAER-cited corporations, a custom web crawler program was designed and implemented. Using the SEC's main AAER website as a seed URL address, this tool *crawls* through the entire AAER website domain, searching for URL addresses that are related to official AAER documents published between October 1999 and December 2010[3]. Since the SEC maintains consistent web address formats for each electronic document, the crawler was programmed to identify these formats and store the corresponding URL in a data structure for further processing. In a queue-like fashion, the crawler hops from one URL to another, loading each web page (document) in turn.

[3]Since the web crawler extracted information from post-1998 AAER citations, a data file provided by one of the author's committee members was used to obtain a listing of the corporations that received AAERs between 1992 and 1998, inclusive.

By placing constraints on the crawling depth, duplicate or irrelevant URL addresses were minimized and isolated from the processing queue. Finally, for each document that the crawler tool opened, an AAER identification number (e.g. AAER-3000), date of citation, and corporation/employee/auditor name were extracted and written to an external file.

Once a preliminary list of 1,608 AAER records (between the citation years 1992 and 2010) was compiled, the list was filtered to remove every employee- or auditor-related record. Then, a unique corporate identifier had to be obtained and matched to each of the remaining corporate AAER records. This proprietary identifier, known as a GVKEY, is used by the Compustat® financial database to extract financial data for an identified corporation. Using a data file containing information on every SEC-registered corporation, a simple parsing algorithm was constructed to match each AAER firm with its entry in the data file and extract the corresponding GVKEY number. Those firms without a matching entry in the data file were temporarily removed from the AAER listing. After briefly reviewing these removed companies, some were reinstated into the AAER list because they were listed under an incomplete or parent-company name. Overall, 997 records were removed from the AAER list during this parsing procedure.

Finally, the citation documents of the remaining 611 corporate AAERs were manually reviewed to extract two pieces of information: time period of fraudulent activity and reason for the issuance of the citation. By collecting the corporate financial data from the first fiscal year of the fraud period, instead of the AAER citation year, we can increase the chance of detecting fraudulent patterns. As is evident from Figure 3.2, the time lag between the first year of fraudulent activity and AAER issuance is quite large. During this time period, a corporation could have restated its financial results or removed any evidence of fraudulent behavior. Given the fraud period and reason for citation, the AAER list was filtered according to the following criteria:

1. If the reason for citation of a corporation was neither directly related to any financial accounts (e.g. the failure to disclose notes to a financial statement or the use of illegal

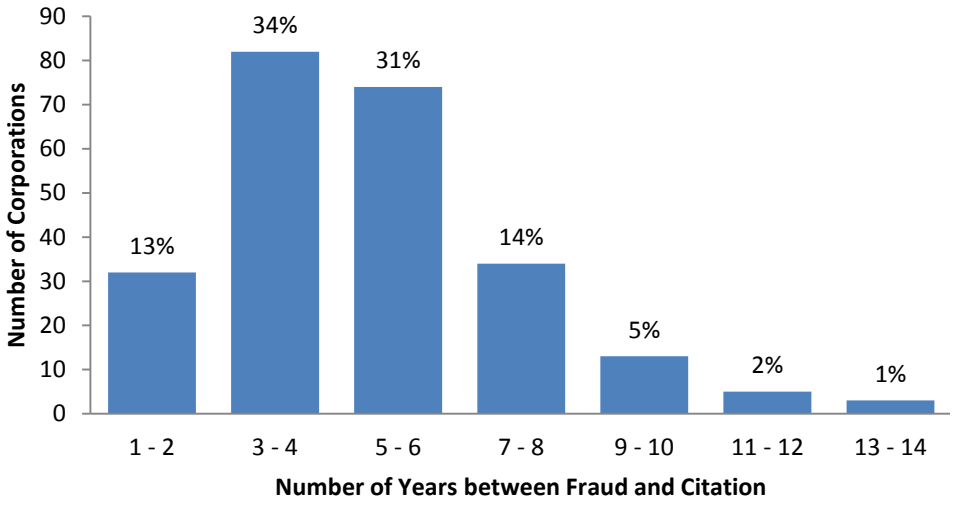


Figure 3.2: Illustration of the time lag between the first commission of financial statement fraud and eventual SEC enforcement for a sample of 243 AAER corporate data records that were utilized in this thesis research.

bribes and kickbacks) nor of a material nature, then the corresponding record was removed from the AAER list.

2. Corporations that had been issued multiple AAER citations were temporarily segregated. If each of these corporate records was related to the same financial reporting violation, then all but one record was removed from the AAER list.

3. All firms with a fiscal first year of fraudulent activity before 1992 were removed from the AAER list. This rule was implemented because the Compustat® database provides broader access to financial data and variables post-1991.

During these procedures, 273 corporations were removed, leaving a net total of 338 records in the AAER list.

3.2.2 Variable Selection and Composition

The second phase of the three-step data collection process consisted of identifying a set of financial variables that would represent a subset of the *SAS No. 99* red flags in the fraud classification task. For a variable to remain in the financial dataset, the following two criteria had to be satisfied. First, the financial variable needed to possess a sufficient ability to discriminate between fraudulent and non-fraudulent financial reporters (i.e. the variable values of a fraudulent firm should likely be distinguishable from a non-fraudulent firm). This assessment was largely guided by the statistical regression results of [15, 23, 54, 55]. Finally, and most importantly, the variable needed to be accessible via a publicly-available database or other electronic source. Red flags that involved internal corporate data and or unquantifiable events (e.g. ineffective accounting and information systems, and frequent auditor disputes) were not represented because of their inaccessibility to the public.

Through the consultation of related financial statement fraud research [23, 54, 55] and discussions with the author's committee members, an initial total of 20 variables was proposed. Based on the *SAS No. 99* red flag descriptions, these variables serve as proxies for 16 red flags. Note that the number of variables was eventually reduced in size during the data preprocessing phase of the data mining process. The reader is referred to §3.3.1 for a review of the variable preprocessing and to §A.4 for tabular summaries of the final variable set and corresponding red flags.

3.2.3 Utilization of Financial Database

After compiling an initial set of financial variables for the fraud classification task, the Compustat® financial database was utilized to extract the corporate data associated with these variables. Statistical Analysis Software (SAS®) database scripts were designed and employed to query Compustat® for the necessary data.

To generate and obtain the non-AAER corporate data records, a SAS® "matching" algorithm was implemented. This algorithm performed a sequence of steps to match each

AAER data record in the input file with a single, non-AAER firm, based on the criteria of first year of fraud, industry, and beginning of the year total assets. This procedure was applied to each of the corporate data records in the AAER input file. Although a one-to-one matching may not reflect the actual distribution of AAER and non-AAER firms, empirical research has demonstrated that this combination yields decent classification accuracy compared to other combinations (particularly for artificial neural network models) [23, 48, 54]. Pseudocode for the matching algorithm appears in Figure 3.3.

3.3 Exploratory Data Analysis

Encompassing the second and third steps of the data mining process, *exploratory data analysis* seeks to learn the basic characteristics of a data set and to hypothesize possible explanations for these properties. Within this phase, the data is preprocessed and possibly transformed to a lower-dimensional attribute space, with the intent of reducing data noise and redundancy [18]. Additionally, through an initial exploration process, relevant assumptions can be defined and posited regarding the data set.

3.3.1 Data Preprocessing and Transformation

As an initial preprocessing step, all incomplete data records were removed from the dataset. An incomplete data record is one in which the Compustat® database was unable to extract non-empty values for every financial variable in the record. This procedure eliminated 95 AAER records and 102 Non-AAER records, resulting in a final dataset size of 479 records (243 AAER records and 236 non-AAER records). §A.1 and §A.2 summarize the preprocessing procedures that were applied to the AAER and non-AAER data records, respectively, throughout the data collection and exploratory analysis phases of the data mining process.

Given the official dataset of 479 data records, a set of summary statistics was produced for each of the initial 20 financial variables. For each variable, a two-sample *t*-test of sig-

```
Algorithm: AAER Matching
Input:
  Set S of AAER data patterns
  Set T of non-AAER data patterns
Output:
  Set M of non-AAER data patterns that are matched to set S

for each AAER data pattern x of S do
  industry := SIC (x) // The industry that x belongs to.
  fraudYear := YEAR (x)
  assets_x := ASSETS (x)
  Group T by industry AND fraudYear
  minDist := INFINITY
  for each non-AAER data pattern y of T do
    assets_y := ASSETS (y)
    dist := |assets_y - assets_x|
    if dist <= minDist then
      minDist := dist
      M (x) := y // Pattern y is now matched to x.
      T := T - {y}
    end if
  end for
end for
return M
```

Figure 3.3: Pseudocode of the AAER Matching Algorithm.

nificance was also conducted to assess whether a significant difference existed between the mean values of the AAER and non-AAER groups, using a *p*-value of 0.05. The summary statistics and complete *t*-test results are presented in §A.5. Based on a review of the frequency distributions of each variable, a decision was made to eliminate the following four financial variables: *COMPLEXTRANS*, Δ*AUD*, *AUDOPINION*, and *BIG4* (refer to §A.4.4 for a review of these variables). Each of these four variables — the first continous and the

latter three binary — were heavily skewed toward one particular value (e.g. 87.68 percent of the *AUD* values were 0). Preliminary experiments with evolutionary algorithms (refer to Chapter 4) revealed that the skewed data was negatively impacting the learning and classification procedures of the algorithms, and reducing the effectiveness of the constructed fuzzy sets. Due to the removal of the four variables, a total of six *SAS No. 99* red flags were left unrepresented.

Finally, the *Principal Component Analysis* (PCA) and *Information Gain* (IG) dimensionality reduction techniques were applied to the data. After assessing the results of these algorithms, it was determined that no variable possessed enough statistical evidence to support its removal from the data collection. However, the *CATA*, *FREEC*, *FINANCE*, *TACC*, and *CACC* variables tended to provide the most information, or variance, as measured by the PCA and IG procedures. Except for *CAAC*, each of these variables possessed a significant ($p < 0.05$) difference between the sample means of the AAER and non-AAER observations.

After completing these standard preprocessing and transformation steps, the financial data set with 16 variables was considered complete and ready for use in the classification algorithms.

3.3.2 Data Assumptions and Hypotheses

Despite the removal of undesirable data via the preprocessing procedures, there are still a couple of notable factors that may at least partially influence the financial statement fraud classification results.

First, there is the possibility that one or more of the non-AAER corporate data instances actually committed financial statement fraud, but were not investigated and cited by the SEC. Hence, the financial variable values of these non-cited firms could be reflective of AAER firms and potentially mislead the classification models during the pattern detection (learning) phase.

Finally, various economic and industry factors could affect the ability of the classifi-

cation models to accurately discriminate patterns of fraud. Between the fiscal years 1994 and 2010 — the fraud time range of the financial data — three global recessionary periods[4] have occurred: 1998, 2001–2002, and 2008–2009 [44]. During each of these periods, the financial variable values may have been unintentionally low for both AAER and non-AAER firms. This condition could potentially add noise to the classification task if the "depressed" values of non-AAER firms happened to mirror the values of AAER firms in non-recessionary periods.

[4]The International Monetary Fund defines a global recession as a year-over-year global change in *real* gross domestic product (GDP) of 3 percent or less [37]. In this context, real GDP refers to the inflation (deflation)-adjusted market value of the final goods and services that are produced within a country.

Chapter 4

CLASSIFICATION EXPERIMENTS

In this chapter the fraud classification performance of MARLEDA is compared to that of the following three competing models: logistic regression classifier, multi-layered ANN, and standard GA. §4.1 provides an account of the design and implementation of each model, §4.2 defines the performance indicators that were used to evaluate each model, and §4.3 presents the results of a suite of classification experiments.

4.1 Experimental Design

The following subsections introduce the different models that were used to perform financial statement fraud classification. These models can be categorized into two groups: evolutionary algorithm and function-based.

4.1.1 Evolutionary Algorithm Models

MARLEDA and the standard GA both evolved candidate solutions (chromosomes) in the form of fuzzy rule-based classifiers (FRBCs). An FRBC chromosome R consists of 20 fuzzy logic rules, each of the form

$$\text{Rule } R_q: \textbf{ if } x_1 \text{ is } A_{q1} \text{ and } \cdots \text{ and } x_n \text{ is } A_{qn} \textbf{ then } \text{Class } C_q,$$

where R_q is the q^{th} logic rule, $\mathbf{x} = \langle x_1, \cdots, x_{16} \rangle$ is a corporate data pattern of 16 financial variables, $A_{qi} \in \{$*Very Low*, *Low*, *Medium*, *High*, *Very High*$\}$ is an antecedent fuzzy set, and $C_q \in \{Fraudulent, Non\text{-}Fraudulent\}$ is the consequent class. The *Very Low* and *Very High* fuzzy sets are each characterized by a ramp membership function, while the

	Active	V_1		V_2		$\cdots$	V_n	
R_1	1	4	0	1	1	$\cdots$	0	1
R_2	0	0	1	2	1	$\cdots$	4	1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
R_m	1	1	0	3	0	$\cdots$	3	0

Figure 4.1: Example of the internal representation of an FRBC. Each rule R_q, $1 \leq q \leq m$, maintains an *Active* bit, indicating whether the rule can be used to classify any of the data patterns. Each financial variable V_i, $1 \leq i \leq n$, is associated with a pair of values that represent, respectively, $1.$) the fuzzy set of which the variable is a member and $2.$) the *active* status of the variable. If V_i is active, then its rightmost value is set to 1 (0, otherwise) and the variable can be used in the classification task. Additionally, each fuzzy set $s \in \{$*Very Low*, *Low*, *Medium*, *High*, *Very High*$\}$ is encoded with an integer $i \in \{0, 1, 2, 3, 4\}$. Note that the consequent (class) of each rule is not included in the chromosome representation. These values are stored in an external data structure that cannot be altered by the evolutionary algorithm.

three inner-most sets are each defined by a triangular membership function[1]. Furthermore, every rule R_q and (x_i, A_{qi}) pair is either *active* or *inactive*. Figure 4.1 illustrates an FRBC in the form of a multi-dimensional matrix; this internal representation was utilized within the GA models. Since MARLEDA represents chromosomes in a "string" (or one-dimensional matrix) format, each FRBC was instead modeled by a concatenation of the individual logic rules (matrix rows) from Figure 4.1. Finally, to interpret the logic rules of an FRBC, we can decode each rule into a human-readable format, such as in Figure 4.2.

Each of the five polygonal fuzzy sets for a fuzzy variable x_i were constructed from experimental data by means of a custom algorithm that attempts to evenly partition the attribute space of x_i. Given a data matrix $\mathbf{D}$, a desired number of fuzzy sets per variable, $numSets$, and a percentage overlap between adjacent fuzzy sets (25 percent in this thesis research), ϕ, the algorithm proceeds as follows. First, all the n attribute values of variable

[1]The five fuzzy sets were selected because of their ability to linguistically model financial data and sufficiently represent the distribution of attribute values. Furthermore, ramp membership functions were used to model the *Very Low* and *Very High* sets because of their ability to encapsulate extreme data observations, without necessitating additional triangular functions (e.g. *Extremely Low* or *Extremely High*), which would increase the complexity of the fuzzy logic rules and rule sets.

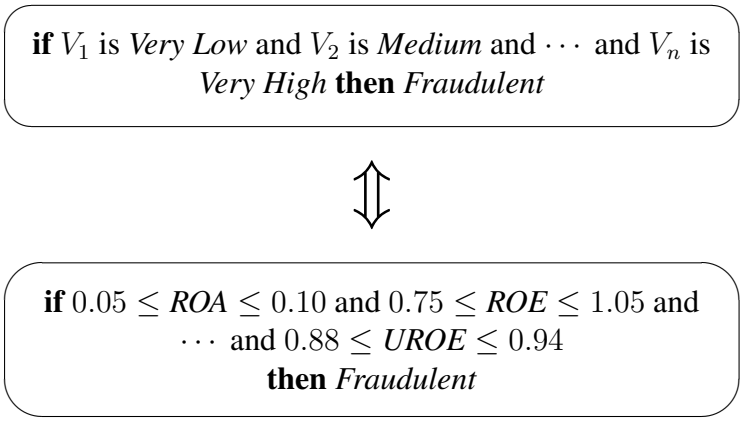


Figure 4.2: Example of the human-comprehensible form of Rule R_2 in the FRBC from Figure 4.1. The first format is a direct representation of the encoded structure presented in Figure 4.1, while the second format is obtained by mapping each fuzzy set to its hypothetical real-value interval (defined by minimum and maximum values), and replacing each variable V_i with its actual name. The two rule interpretations are equivalent.

x_i in $\mathbf{D}$ are sorted in ascending order and filtered to remove duplicates. Then, each unique value is assigned a rank that reflects the position of the value in the resulting totally-ordered set $\mathbf{S}$ of $m \leq n$ elements; hence, the minimum and maximum attribute values possess ranks of 1 and m, respectively. Starting at the median of the first set (rank 1), the median value of each successive fuzzy set is located by iteratively advancing an inter-set distance of $\delta = \frac{m-1}{numSets-1}$ in rank space. The minimum and maximum boundary points of a set are determined by adding half of the total fuzzy set length, $l = \frac{\delta}{1-\frac{\phi}{2}}$, to each side of the median set value in rank space (for convenience, the median is used as the minimum and maximum value for the *Very Low* and *Very High* sets, respectively). Finally each rank space value j is mapped back to its real-value (attribute) space equivalent in one of two ways. If j is an integer, then map j to the attribute value indexed at j within $\mathbf{S}$; otherwise, compute the difference $\Delta = S(j+1) - S(\lfloor j \rfloor)$ and estimate j as $j = S(\lfloor j \rfloor) + \Delta \cdot (j - \lfloor j \rfloor)$, where $(j - \lfloor j \rfloor)$ is the fractional component of rank j. This fuzzy set construction algorithm is iteratively applied to each fuzzy variable $x_i \in \mathbf{x}$.

During each algorithm iteration and for each FRBC chromosome, the single-winner ap-

proach was employed to select the rule that was most compatible with a given training data pattern. The consequent (class) of this rule was then used to classify the data pattern on behalf of the FRBC; refer to §2.5.2 for a review of this approach. During the classification experiments, the rule weight, CF, of each logic rule was held constant at 1.0. This decision was made because CF did not significantly influence the selection of the winner rule throughout multiple trial runs of the evolutionary algorithm models.

Additionally, the fitness of each FRBC was measured along two objectives: classification accuracy rate, $accuracy$, and ratio of active logic rules, $activeRules$, where

$$accuracy = \frac{\text{\# of correctly classified patterns}}{\text{\# of total patterns}} \tag{4.1}$$

and

$$activeRules = \frac{\text{\# of active rules}}{\text{\# of total rules}}. \tag{4.2}$$

The goal of the evolutionary process is to evolve chromosomes that maximize $accuracy$ and minimize $activeRules$. These two objectives were linearly-combined into a weighted fitness function

$$fitness = w_1 \cdot accuracy - w_2 \cdot activeRules, \tag{4.3}$$

where w_1 and w_2 are arbitrarily-assigned weights from the interval $[0, 1]$. To maintain accurate FRBCs in the population and prevent negative fitness scores, w_1 and w_2 were set to 0.99 and 0.01, respectively, for the classification experiments. Note that the second term of $fitness$ is subtracted from the first term to simulate the minimization of $activeRules$. Hence, the evolutionary algorithms attempted to maximize $fitness$.

Finally, each of the evolutionary algorithms was implemented and tested in the C++ programming language. The base source code for MARLEDA was written by Alden [3]. All of the source code related to the financial statement fraud classification task and the GA was custom written for this thesis research.

4.1.2 Function-based Models

Providing a performance benchmark from the early work of [22, 23, 25], a multi-layered, feed-forward ANN was applied to the classification task. This model was implemented in C++ code and maintains the following fixed, structural parameters: 16 input nodes, 1 output node, 1 hidden layer, and 2 bias nodes (1 for each of the hidden and output layers). The learning rate, momentum rate, and number of hidden nodes were each optimized according to the experiments of §4.3. Finally, prior to inserting data into the ANN model, the financial dataset was normalized onto the range $[0, 1]$ in an attribute-wise fashion [23].

Due to the historical and popular usage of logistic regression in previous accounting/finance literature, a standard logistic regression model was run using the Waikato Environment for Knowledge Analysis (Weka) data mining software. Similar to the ANN implementation, 16 independent variables were defined to represent the financial variables.

Finally, the logistic regression and ANN models both utilized a *threshold* value of 0.5. In logistic regression, a data observation $\mathbf{x}$ was classified as *Fraudulent* (or "1") if the value of the hypothesis $f(\mathbf{x})$ was greater than or equal to 0.5; otherwise, $\mathbf{x}$ was classified as *Non-Fraudulent* (or "0"). Similarly, an ANN classified $\mathbf{x}$ as *Fraudulent* if the activation function value $z_1 = f(net_1)$ at the single output node O_1 was greater than or equal to 0.5.

4.2 Performance Indicators

This section introduces a suite of indicator functions that were used to assess the quality of a classifier and enable performance comparisons among multiple classification models.

4.2.1 Confusion Matrix-Based Indicators

During the training and testing phases of a classifier, the classification results were documented in a 2×2 *confusion matrix* that provides a concise summary of the classifier's performance. Figure 4.3 depicts a typical confusion matrix in which the first dimension represents the distribution of the actual binary classes among the classified data instances, and

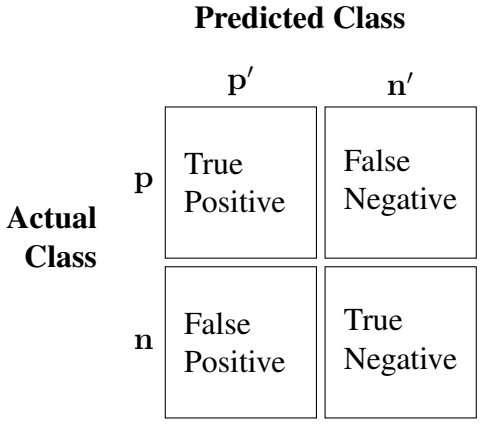


Figure 4.3: Illustration of a confusion matrix.

the second dimension indicates the distribution of the hypothesized, or predicted, classes. The values of the four matrix entries sum to the size of either the training or test dataset, and each entry can be defined as follows:

1. **True Positive:** A data pattern $\mathbf{x}$ of positive class p was correctly classified with class p'.

2. **True Negative:** A data pattern $\mathbf{x}$ of negative class n was correctly classified with class n'.

3. **False Positive (Type I Error):** A data pattern $\mathbf{x}$ of negative class n was incorrectly classified with positive class p'.

4. **False Negative (Type II Error):** A data pattern $\mathbf{x}$ of positive class p was incorrectly classified with negative class n'.

In this thesis research, the *Fraudulent* (or *AAER*) class was considered positive and labeled as "1", while the *Non-Fraudulent* class was considered negative and represented by a "0".

Given the four confusion matrix components, a set of classification performance indicators can conveniently be defined. These indicators are presented in Table 4.1, and were computed for each run of a classification model.

Indicator	**Definition**	**Description**
Accuracy Rate	$\textit{Accuracy} = \frac{\lvert TP\rvert+\lvert TN\rvert}{n}$	The number of correct classifications divided by the size of a sample.
False Positive Rate	$\textit{FPR} = \frac{\lvert FP\rvert}{\lvert TN\rvert+\lvert FP\rvert}$	The ability of a classifier to correctly identify negative data instances; $(1 - \textit{TNR})$.
True Positive Rate	$\textit{TPR} = \frac{\lvert TP\rvert}{\lvert TP\rvert+\lvert FN\rvert}$	The ability of a classifier to correctly identify positive data instances.
Precision	$\textit{Precision} = \frac{\lvert TP\rvert}{\lvert TP\rvert+\lvert FP\rvert}$	The ability of a classifier to correctly classify positive data instances.
True Negative Rate	$\textit{TNR} = \frac{\lvert TN\rvert}{\lvert TN\rvert+\lvert FP\rvert}$	The ability of a classifier to correctly classify negative data instances; $(1 - \textit{FPR})$.

Table 4.1: Classification performance indicators that are derived from a confusion matrix.

Furthermore, the average training and test classification errors of the ANN model were also measured by the *mean squared error* (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i)^2 \tag{4.4}$$

$$= \frac{1}{n}Loss(f), \tag{4.5}$$

where f is a hypothesis function and $Loss(\cdot)$ is the error function defined in §2.2.

All of the above-defined statistics were computed for each classification model by means of a *cross-validation* procedure that partitions the financial dataset into k equally-sized *folds* ($\lfloor\frac{n}{k}\rfloor$ observations per fold) and performs k classification experiments (model runs), where in each experiment one *validation* fold is held-out for testing and the other $k - 1$ folds are used for training. This procedure helps minimize the bias that the training dataset may insert in the classifier learning phase, and allows each data instance to eventually participate in the testing process [19].

Additionally, while optimizing the parameters of a classification model, cross-validation may be used to identify the parameter combination that yields the highest average validation accuracy, over the k validation folds (validation sets). This optimal and fully-trained classifier should then be further validated against a separate test dataset of unseen data observations. Since the validation dataset was used to select the classifier, the average validation accuracy rate of the classifier could be biased (i.e. greater than the true accuracy rate) [18]. Thus, it is important to assess the performance of such a model on a separate test dataset.

4.3 Empirical Results

The following subsections review the results of the financial statement fraud classification experiments that were conducted for each of the four machine learning models presented in §4.1. The summary statistics for the models are presented in Table 4.4.

4.3.1 Logistic Regression Results

Using the complete financial dataset of 479 observations, a 5-fold cross validation procedure was employed to train and validate the logistic regression model (hypothesis)

$$\begin{aligned} f_{Logistic}(\mathbf{x}) \;=\; & \beta_0 + \beta_1 \textit{ROA} + \beta_2 \textit{ROE} + \beta_3 \textit{ACHANGE} + \beta_4 \textit{TACC} + \beta_5 \textit{CACC} \\ & + \beta_6 \textit{LEV} + \beta_7 \textit{DIFFAUDIT} + \beta_8 \textit{FREEC} + \beta_9 \textit{FINANCE} \\ & + \beta_{10} \textit{CATA} + \beta_{11} \textit{RECEIVABLE} + \beta_{12} \textit{INVENTORY} \\ & + \beta_{13} \textit{SCHANGE} + \beta_{14} \textit{MDOM} + \beta_{15} \textit{UMARGIN} + \beta_{16} \textit{UROE}, \end{aligned}$$

where β_i, $0 \leq i \leq 16$, is a function weight to be learned, and β_0 is an intercept term. Overall, the model correctly classified 256 of the 479 observations for an average validation accuracy rate of 53.44 percent. Additionally, $f_{Logistic}$ correctly classified 47.33 percent of the 243 AAER firms and 59.75 percent of the 236 non-AAER firms. The confusion matrix of the cumulative 479 classifications made throughout the validation runs is presented in Table 4.2.

A test of significance, with a p-value of 0.05, was performed to identify the financial variables that significantly influenced the hypothesis $f_{Logistic}$. Only the *CATA* variable was significant at $p = 0.007 < 0.05$. However, *SCHANGE* ($p = 0.053$) and *CACC* ($p = 0.095$)

		Predicted			
		AAER		*Non-AAER*	
		#	% of Total	#	% of Total
Actual	*AAER*	115	24.0%	128	26.7%
	Non-AAER	95	19.8%	141	29.5%

Table 4.2: The confusion matrix of 479 cumulative classifications that the logistic regression classifier made on the five different validation datasets.

were both moderately significant at $p < 0.10$. A detailed account of the significance value for each financial variable can be found in §B.1.

4.3.2 Artificial Neural Network Results

The artificial neural network (ANN) experiment consisted of two phases. First, a *wrapper* algorithm was designed to enumerate ANN models according to distinct parameter combinations of hidden nodes, learning rates, and momentum rates. For each set of parameter values, 5-fold cross-validation was employed to evaluate the average validation accuracy rate of an ANN model that was built upon the parameter values. The parameter set that yielded the maximum average validation accuracy was selected and used to construct an optimized ANN. This model, $NeuralNet_{OPT}$, was then re-trained on a training dataset of 319 observations (approximately 2/3 of the complete dataset) and tested against a dataset of 160 unseen observations.

During the parameter-tuning phase, 128 distinct combinations of hidden nodes, learning rates, and momentum rates were evaluated. With an average validation accuracy rate of 55.87 percent, the ANN model with 14 hidden nodes, a learning rate of 0.2, and momentum rate of 0.4, was identified as optimal[2]. The top-5 performing parameter combinations are presented in §B.2. After training the optimal ANN for 5,000 epochs, it was able to correctly classify 55.63 percent of the observations in the test dataset. Furthermore, the model correctly classified 43.9 percent of AAER firms and 67.95 of non-AAER firms. A confusion matrix of the 160 test set classifications is displayed in Table 4.3.

[2]An ANN model with a parameter set of 16 hidden nodes, 0.2 learning rate, and 0.3 momentum rate also possessed an average validation accuracy rate of 55.87 percent. Since this model possessed a higher average mean-square error (*MSE*) over the validation sets, it was not selected as the "best-tuned" model.

		Predicted			
		AAER		*Non-AAER*	
		#	% of Total	#	% of Total
Actual	*AAER*	36	22.5%	46	28.8%
	Non-AAER	25	15.6%	53	33.1%

Table 4.3: The confusion matrix of 160 test set classifications performed by the artificial neural network model.

4.3.3 Evolutionary Algorithm Results

Both the GA and MARLEDA attempted to evolve accurate and comprehensible FRBCs by combining the $accuracy$ and $activeRules$ objectives into a single, weighted fitness value. Each model was run for a specific number of generations within a 5-fold cross validation procedure and maintained parameter settings that were tuned during preliminary trial runs. The technical specifications of the GA and MARLEDA are as follows:

- **GA** : generations = 3000, population size = 100, population selection ratio = 0.75, uniform crossover, mutation rate = 0.05, crossover rate = 0.15, and tournament size = 3.

- **MARLEDA** : generations = 5000, population size = 100, population selection ratio = 0.75, tournament size = 2, mutation rate = 0.1, Markov chain Monte Carlo sampling iterations = 4100, $ModelAdds = 4100$, $ModelSubs = 4100$, $ModelAddThresh = 0.6$, and $ModelSubThresh = 0.5$.

Furthermore, the performance of each model was stated in terms of the average classification accuracy and active rules ratio of the most-fit rule-based classifier from each of the five cross-validation runs. These five classifiers possessed the highest fitness function value within their population after the final training generation.

Over the five cross-validation iterations, both of the models demonstrated a more-superior ability to detect patterns of fraud within the training dataset, as compared to the $f_{Logistic}$ and $NeuralNet_{OPT}$ classifiers. The GA yielded an average training accuracy rate of 62.1 percent and an average validation accuracy rate of 59 percent, with 280 of 475 data observations correctly classified. Throughout the validation test phases, the most-fit GA rule-based classifiers correctly identified 58.9 percent of the AAER observations and 59

		Predicted			
		AAER		*Non-AAER*	
		#	% of Total	#	% of Total
Actual	*AAER*	139	29.3%	97	20.4%
	Non-AAER	98	20.6%	141	29.7%

(a) GA Confusion Matrix

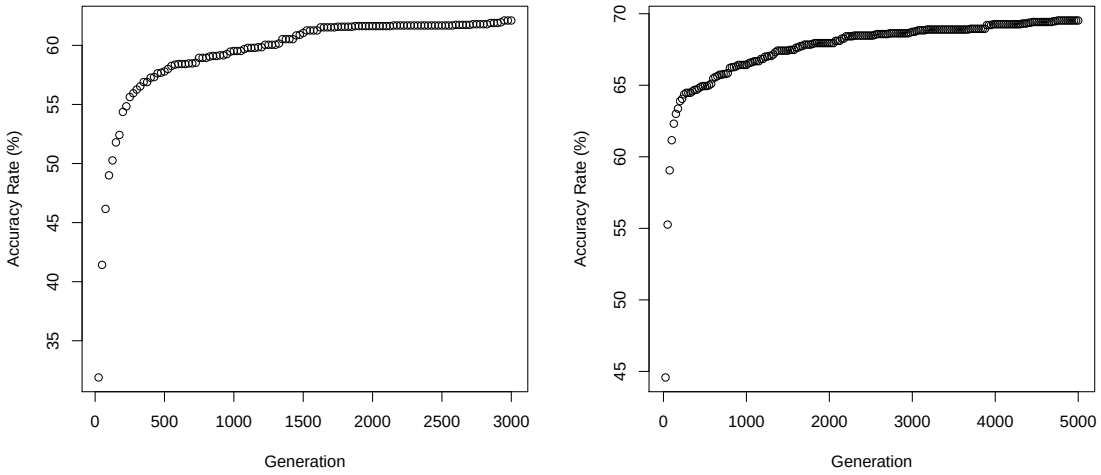
		Predicted			
		AAER		*Non-AAER*	
		#	% of Total	#	% of Total
Actual	*AAER*	128	26.9%	110	23.2%
	Non-AAER	113	23.8%	124	26.1%

(b) MARLEDA Confusion Matrix

Figure 4.4: The confusion matrices of 475 cumulative classifications that the most-fit rule-based classifiers of each model made on the five validation datasets.

percent of the non-AAER observations. Demonstrating an even greater potential of detecting fraud, MARLEDA reported an average training accuracy rate of 69.53 percent, with a maximum accuracy of 70.53 percent in the fifth cross-validation run. Throughout the validation test runs, MARLEDA recorded an unexpectedly low average validation accuracy of 53.05 percent, with 223 of 475 data observations incorrectly classified and a minimum validation accuracy of 48.42 percent for the classification of the first validation dataset. These results reveal that MARLEDA had difficulty generalizing to unseen corporate data observations and may be overlearning (or memorizing) the patterns of fraud within the training data. The confusion matrices of the GA and MARLEDA are presented in Figure 4.4.

From Figures 4.5 and 4.6 the effect of the weighted-sum fitness function can be observed, as the GA and MARLEDA both attempted to extract as much accuracy as possible from a small set of active rules. Starting with 100 percent of the logic rules active (20 rules) in each rule-based classifier, the average $activeRules$ ratio for the GA, over all 5 training runs, was eventually reduced to 39 percent by the $3{,}000^{th}$ generation. This reduction in classifier complexity was largely influenced by the 0.01 weight factor that was assigned to the $activeRules$ term in the fitness function. Since the weight of the $accuracy$ term was set at a disproportionately high value of 0.95, the evolutionary processes naturally favored more-accurate classifiers and, thus, a point was reached at which no more rules could be deactivated without sacrificing significant classification accuracy. Hence, new classifiers



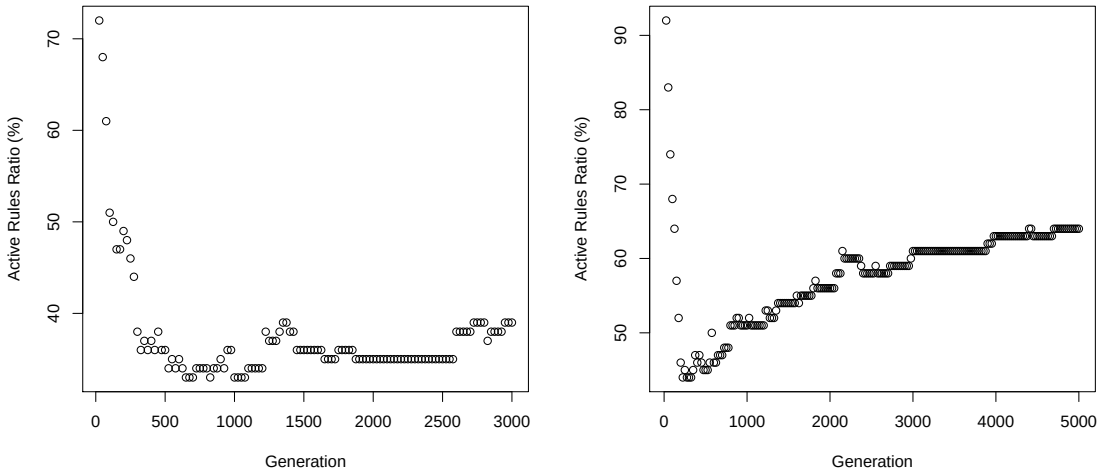
(a) Genetic Algorithm

(b) MARLEDA

Figure 4.5: Scatter plots of the average generational accuracy rate for the GA and MARLEDA. Each data point represents the average accuracy rate value of the most-fit rule-based classifier from each of the five cross-validation runs.

steadily increased their fitness and accuracy by activating more logic rules. This effect was clearly evident in the final $4{,}500$ generations of the MARLEDA model. During this period, MARLEDA had to increase the average ratio of active rules by approximately 20 percentage points just to raise the average classification accuracy rate from 64.95 percent to 69.53 percent.

Among the GA validation runs, the fourth run yielded a 7-rule classifier that possessed the maximum validation accuracy rate of 63.16 percent. Likewise, during the fifth validation run of the MARLEDA experiment, a 13-rule classifier was evolved that yielded a maximum validation accuracy rate of 57.89 percent. These fuzzy rule-based classifiers (FRBCs) are displayed in a human-readable format in Figures 4.7 and 4.8, respectively. An analysis of the classifiers reveals that MARLEDA tended to evolve more-complex logic rules, with multiple conjunctions of fuzzy propositions, while the GA generated a rule set with fewer active rules and active variables. These outcomes could be attributed to the differences in evolutionary learning between the GA and MARLEDA models. Given enough training generations and gene-similarity tests ($ModelAdds$ and $ModelSubs$), MARLEDA



(a) Genetic Algorithm (b) MARLEDA

Figure 4.6: Scatter plots of the average generational active rules ratio for the GA and MARLEDA. Each data point represents the average active rules ratio value of the most-fit rule-based classifier from each of the five cross-validation runs.

was likely utilizing its MRF probability model to infer the inter-dependencies among genes (and, thus, financial variables) more-effectively than the GA. This learning ability was validated by the training accuracy of 70.53 percent for the top MARLEDA classifier. Furthermore, the number of active rules within each of the two classifiers clearly portrayed the tradeoff between classifier accuracy and complexity. By maintaining a smaller set of active rules (7) from which to select a single-winner, the GA classifier had to accept a lower rate of training accuracy (60 percent). Similarly, the MARLEDA classifier achieved a higher accuracy rate (70.53 percent) by activating a larger set of logic rules (13).

Finally, a review of the most-fit rule-based classifier from each of the five cross-validation runs revealed that a specific subset of financial variables was frequently utilized by both the GA and MARLEDA. Across all of the active rules within the five classifiers, the variables *FINANCE*, *FREEC*, *INVENTORY*, *TACC*, *CATA*, *ACHANGE*, and *SCHANGE* were ranked in the list of top-10 most active variables for both machine learning models. Each of the variables, except *INVENTORY* and *ACHANGE*, possessed significant p-values in the two-sample t-test of means (§A.5), indicating that the single-objective GA and

1. **if** $0.085 \leq FREEC \leq 0.437$ (*Very High*) **then** *Fraudulent*
2. **if** $-2.378 \leq FINANCE \leq -0.274$ (*Very Low*) **then** *Non-Fraudulent*
3. **if** $0.00 \leq DIFFAUD \leq 0.073$ (*Very Low*) **then** *Fraudulent*
4. **if** $0.016 \leq FREEC \leq 0.085$ (*High*) **then** *Fraudulent*
5. **if** $-0.274 \leq FINANCE \leq -0.046$ (*Low*) **then** *Non-Fraudulent*
6. **if** $-0.046 \leq FINANCE \leq 0.029$ (*Medium*) **then** *Fraudulent*
7. **if** $-0.016 \leq ROA \leq 0.040$ (*Medium*) **then** *Non-Fraudulent*

Figure 4.7: The most-fit fuzzy rule-based classifier that was evolved by the GA during the fourth cross-validation run.

MARLEDA models are successfully identifying the factors that discriminate between the AAER (fraud) and non-AAER (non-fraud) training observations. A pair of histograms of the 5 most frequently activated variables for the GA and MARLEDA appears in Figure 4.9. Additionally, the activation frequencies for every financial variable are presented in §B.5.

4.3.4 Summary and Discussion

From Table 4.4 it is apparent that the training accuracy monotonically increased along each of the machine learning models. In particular, MARLEDA demonstrated its sophistication by yielding the maximum average training accuracy rate, with the tradeoff of a more-complex model. This complexity may have contributed to MARLEDA's inability to generalize to unseen test observations, as was evident from the minimum average validation accuracy rate of 53.05 percent. Although the GA possessed a lower training accuracy than MARLEDA, it maintained a smaller percentage of active logic rules that likely contributed to its decent generalization to the validation data (59 percent average validation rate). Additionally, MARLEDA and the GA both outperformed the $f_{Logistic}$ and ANN_{OPT} classifiers, which were unsuccessful at correctly identifying AAER (fraud) and non-AAER (non-fraudulent) observations.

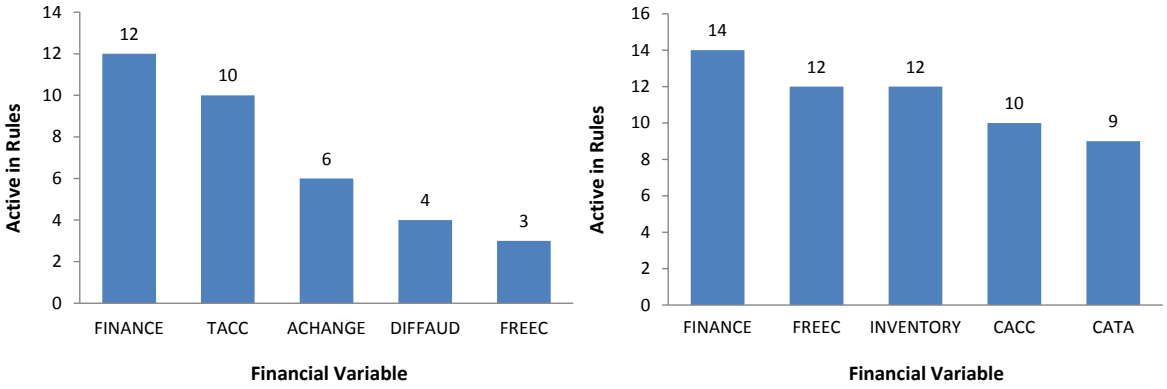
Furthermore, the multi-objective weighted-sum approach of the EAs was able to generate diverse rule-based classifiers that reflected the tradeoff between accuracy and comprehensibility (complexity). Multi-objective experiments with Pareto-based EAs, including mMARLEDA (refer to §2.4.2), were also unofficially conducted to assess the quality and diversity of a set non-dominated classifiers. After several trial runs, these models were un-

1. **if** $0.090 \leq ROE \leq 0.396$ (*High*) **then** *Fraudulent*
2. **if** $-2.378 \leq FINANCE \leq -0.181$ (*Very Low*) **then** *Fraudulent*
3. **if** $-0.325 \leq FINANCE \leq -0.031$ (*Low*) **then** *Fraudulent*
4. **if** $-0.060 \leq FREEC \leq 0.030$ (*Medium*) and $0.001 \leq CATA \leq 0.063$ (*Medium*) **then** *Non-Fraudulent*
5. **if** $-0.262 \leq TACC \leq -0.050$ (*Low*) and $0.012 \leq INVENTORY \leq 0.070$ (*High*) **then** *Non-Fraudulent*
6. **if** $-0.074 \leq INVENTORY \leq -0.005$ (*Low*) **then** *Non-Fraudulent*
7. **if** $-0.144 \leq ACHANGE \leq 0.088$ (*Low*) and $0.005 \leq FREEC \leq 0.111$ (*High*) **then** *Non-Fraudulent*
8. **if** $-1.256 \leq ROE \leq 0.006$ (*Low*) and $0.070 \leq FREEC \leq 0.377$ (*Very High*) and $0.081 \leq FINANCE \leq 1.049$ (*Very High*) and $-0.010 \leq CATA \leq 0.026$ (*Low*) and $311.830 \leq SCHANGE \leq 17{,}953.300$ (*Very High*) **then** *Fraudulent*
9. **if** $-673.750 \leq CACC \leq -0.056$ (*Very Low*) **then** *Non-Fraudulent*
10. **if** $-0.010 \leq INVENTORY \leq 0.005$ (*Medium*) and $0.010 \leq UROE \leq 1.622$ (*High*) **then** *Non-Fraudulent*
11. **if** $-0.010 \leq ROE \leq 0.141$ (*Medium*) and $-0.010 \leq CATA \leq 0.026$ (*Low*) and $0.000 \leq MDOM \leq 0.001$ (*Low*) **then** *Fraudulent*
12. **if** $0.042 \leq ACHANGE \leq 0.304$ (*Medium*) **then** *Non-Fraudulent*
13. **if** $0.0568 \leq CATA \leq 0.167$ (*High*) **then** *Fraudulent*

Figure 4.8: The most-fit fuzzy rule-based classifier that was evolved by MARLEDA during the fifth cross-validation run.

able to consistently yield classifiers with decent accuracy rates. Thus, detailed statistical results of these experiments were not included in this thesis.

According to the values of the *TPR* indicator, all four models were less-successful at correctly identifying AAER observations. Since a lower *TPR* (true positive) rate implies a higher false negative rate, the models were frequently misclassifying fraudulent observations as non-fraudulent. If the misclassification of an actual AAER firm is considered to be more costly than the misclassification of an actual non-AAER firm, then the *TPR* results should not be regarded positively. Furthermore, the *TNR* values indicate that the models were correctly classifying a relatively large percentage of non-AAER observations. Given a sufficiently accurate classifier, a larger number of true negatives could benefit a decision maker by reducing the number of firms that are unnecessarily investigated because of an



(a) Genetic Algorithm

(b) MARLEDA

Figure 4.9: Histograms of the top-5 most frequently used financial variables across the most-fit rule-based classifier from each of the five cross-validation runs.

incorrect fraud classification (false positive).

Finally, among the financial variable analyses of the $f_{Logistic}$, GA, and MARLEDA models, the *CATA* and *SCHANGE* variables were significant or frequently-utilized within each model. Both of these variables serve as proxies for Pressure red flags within the *SAS No. 99* Fraud Triangle and measure events such as recurring negative cash flows from operations, rapid growth or unusual profitability, and operating losses (refer to §A.4.5). Additionally, the *CAAC* financial variable was moderately significant in the logistic regression

Model	*Accuracy* — Training	*Accuracy* — Validation	*FPR*	*TPR*	*TNR*	*Precision*	*activeRules* — Training
$f_{Logistic}$	58.20	53.44	40.25	47.33	59.75	54.76	—
ANN_{OPT}	59.40	55.63	32.05	43.90	67.95	59.02	—
GA	62.10	59.00	41.00	58.90	59.00	58.65	39.00
MARLEDA	69.53	53.05	47.68	53.78	52.32	53.11	64.00

Table 4.4: Summary statistics for each of the classification models. Each statistic is stated as a percentage and, unless otherwise denoted, is derived from a validation confusion matrix.

experiment and was one of the top-5 most frequently used variables within the most-fit MARLEDA classifiers. This variable serves as a proxy for the two Rationalization red flags. Interestingly, the *FINANCE*, *ACHANGE*, and *FREEC* variables, which were each in the top-5 most active variables list of the GA, are the only significant ($p < 0.05$) "financial" variables within the logistic regression classifier of Skousen et al. [54]. Based on this finding and the GA's decent ability to generalize to unseen data, the three variables appear to hold promise in modeling patterns of fraud.

Chapter 5

CONCLUSION AND FUTURE RESEARCH

Throughout our modern economic history, the desire to detect fraudulent behavior within a corporation has captured the attention of regulators, auditors, and investors alike. Fostering this widespread interest has been the development of machine learning models that computationally discover patterns of fraud that humans would be unable to manually identify. Since humans must interpret the results of these models in order to classify the fraudulent state of a corporation, it is also imperative that the complexity of these models be minimized. The three primary contributions of this thesis research were to 1.) apply a sophisticated estimation of distribution algorithm (EDA) to the financial statement fraud classification task, 2.) assess the ability of evolutionary multi-objective optimization to generate both accurate and comprehensible fuzzy rule-based classifiers (FRBCs), and 3.) identify a subset of the *SAS No. 99* red flags that significantly contributed to the detection of patterns of fraud, or the issuance of an Accounting and Auditing Enforcement Release (AAER). This chapter reviews the contributions and findings of this thesis, and introduces potential avenues of future research.

5.1 Summary of Thesis Research

Within Chapter 1, the financial statement fraud classification task was defined and placed into context with the SEC enforcement process (e.g. the issuance of AAERs) and the work of previous research. The review of related literature revealed that little research has investigated the application of sophisticated evolutionary algorithms (EAs) to the classification of financial statement fraud. This opportunity inspired the work of this thesis and prompted the three primary contributions that are defined in the introductory paragraph of this chapter.

In Chapter 2, the foundational concepts of machine learning, pattern recognition, and supervised learning were introduced and associated with a suite of classification models that were utilized within this thesis. After reviewing logistic regression, artificial neural networks (ANNs), and genetic algorithms (GAs), estimation of distribution algorithms

(EDAs) were presented in terms of three underlying probability models: univariate, bivariate, and multivariate. Based on a multivariate and undirected Markov Random Field (MRF) neighborhood system, the MARLEDA EDA improves upon the learning and sampling procedures of predecessor EDAs and was employed as the centerpiece model within this research. A synopsis of multi-objective optimization, fuzzy logic, and rule classification (e.g. the single-winner method) was subsequently provided to facilitate an understanding of the method by which the EA models performed financial statement fraud classification.

Based on the five-step data mining process, Chapter 3 surveyed the broad set of procedures that were performed in regards to financial data collection, preprocessing, and transformation. Through the use of a financial database, data was extracted for 479 corporate data observations, each consisting of 16 financial variables that served as proxies for a subset of the *SAS No. 99* red flags. All of the 243 AAER observations included data from the first year of fraudulent activity, and each of the 236 non-AAER observations was acquired as a result of a one-to-one matching algorithm that paired a non-AAER firm with an AAER firm based on the criteria of fraud year, industry, and beginning of the year total assets. Finally, summary statistics revealed that the variables *SCHANGE*, *CATA*, *FREEC*, *FINANCE*, and *TACC* each possessed a significant difference between the mean values of the AAER and non-AAER observations.

In Chapter 4, a suite of classification experiments were enumerated and the results of these experiments were presented and analyzed. Demonstrating success in detecting patterns of fraud, the GA and MARLEDA models yielded average training accuracy rates of 62.10 and 69.53 percent, respectively. Reflecting the tradeoff condition of multi-objective optimization, the GA maintained a lower accuracy than MARLEDA, but possessed fewer active logic rules within its FRBCs. This reduction in classifier complexity possibly allowed the GA to achieve a higher average validation accuracy rate (59 percent) than that of MARLEDA (53.05 percent). Overall, both of the EA models outperformed the benchmark logistic regression and ANN models. Finally, the *CATA*, *SCHANGE*, *CACC*, *FINANCE*, *ACHANGE*, and *FREEC* financial variables were considered to hold the most promise in detecting patterns of fraud or non-fraud.

5.2 Future Research

Two primary avenues of future research include enhancing the financial dataset and incorporating sophisticated heuristic techniques into the learning and classification phases of

the EA models. Since the size and composition of the data are two of the most important factors in the ability of a classifier to accurately detect fraud, it may be advantageous to extract data for the corporations, if any, that are linked to the employee- and auditor-related AAER citations (only citations involving financial misstatements). If complete data is available, these new observations could increase the dataset size enough that additional financial variables can be introduced, such as compensation- and audit committee-related variables, which have demonstrated success in identifying fraudulent behavior [54]. Furthermore, the increase in observations would enable the dataset to be partitioned according to disjoint time periods and facilitate temporal-based classification experiments.

Due to the slow generational rate of change in classification accuracy during the latter stages of evolution for the GA and MARLEDA models, heuristics such as adaptable crossover and mutation could be utilized to spur evolutionary change. These dynamic variation operators could be increased in value when the fitness of solutions becomes stagnant, and decreased in value when the fitness is still monotonically increasing. Moreover, the EA models could be seeded with an initial set of hand-crafted FRBCs that are designed from human knowledge of financial statement fraud patterns. This knowledge could partially be acquired by manually studying the attribute values for a large subset of the financial dataset, and documenting "rules of thumb" that are prevalent for cited and non-cited corporations. Furthermore, this manual analysis of the data may provide insight into the percentage of corporate observations that possess fraud-like data, but that were not assessed an AAER by the SEC (i.e. those firms that were either not detected or legally pursued by the SEC).

Additionally, a sensitivity analysis could be performed with the evolutionary algorithms to assess the individual predictive ability of each financial variable. In an iterative fashion, this analysis would remove a variable from the classification task and assess the classifier performance without the variable. If the accuracy with the variable included is significantly different from the accuracy without the variable, then the variable is effective at influencing the detection of fraudulent and/or non-fraudulent patterns. The results of this analysis would hopefully corroborate the frequency distribution statistics of each variable being active in the most-fit rule-based classifiers (refer to §B.5), and validate that the EAs are utilizing the most-predictive variables within the final set of classifiers.

Based on MARLEDA's less-than-ideal performance in classifying test observations, it may be beneficial to run the model for an increasing number of generations until the average validation accuracy begins to substantially decrease. This generational cutoff could

reveal the point t at which the model may be overlearning the training data. By running MARLEDA for t or fewer generations, the complexity of the learned MRF neighborhood system may decrease enough to allow an improved generalization to the test data.

Finally, it would be interesting to experiment with different crowding distance metrics for the Pareto-based multi-objective EAs and observe the effect that each metric has on the spread of the non-dominated classifiers. The primary goal of this experiment would be to reduce the number of classifiers with a 0 percent accuracy rate and no active rules. Since these solutions are not of any use to a decision maker, they could feasibly be removed from the population during the selection phase of each generation and hopefully promote new non-dominated solutions with a non-zero number of active rules.

BIBLIOGRAPHY

[1] C. W. Ahn, R. S. Ramakrishna, and D. E. Goldberg. Real-coded Bayesian Optimization Algorithm. In J. Lozano, P. Larraaga, I. Inza, and E. Bengoetxea, editors, *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 51–73. Springer Berlin / Heidelberg, Berlin, Germany, 2006.

[2] R. Alcalá, J. Alcalá-Fdez, M. J. Gacto, and F. Herrera. On the usefulness of MOEAs for getting compact FRBSs under parameter tuning and rule selection. In A. Ghosh, S. Dehuri, and S. Ghosh, editors, *Multi-Objective Evolutionary Algorithms for Knowledge Discovery from Databases*, pages 91–107. Springer-Verlag, Berlin, Germany, 2008.

[3] M. Alden. *MARLEDA: Effective distribution estimation through markov random fields*. Ph.D. thesis, The University of Texas at Austin, Austin, Texas, 2007.

[4] T. Ayer, J. Chhatwal, O. Alagoz, C. E. Kahn, Jr., R. W. Woods, and E. S. Burnside. Comparison of logistic regression and artificial neural network models in breast cancer risk estimation. *RadioGraphics*, 30:13–22, 2010.

[5] M. S. Beasley, J. V. Carcello, D. R. Hermanson, and T. L. Neal. Fraudulent Financial Reporting 1998 - 2007: An Analysis of U.S. Public Companies. `http://www.coso.org/documents/COSOFRAUDSTUDY2010_001.pdf`, 2010.

[6] J. D. Becker, M. I. Hwang, and J. W. Lin. A fuzzy neural network for assessing the risk of fraudulent financial reporting. *Managerial Auditing Journal*, (18):657–665, 2003.

[7] P. Bosman and D. Thierens. Multi-objective optimization with the naive MIDEA. In J. Lozano, P. Larraaga, I. Inza, and E. Bengoetxea, editors, *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 123–157. Springer Berlin / Heidelberg, Berlin, Germany, 2006.

[8] W. Chai, B. K. Hoogs, and B. T. Verschueren. Fuzzy ranking of financial statements for fraud detection. In *2006 IEEE International Conference on Fuzzy Systems*, pages 152–158, 2006.

[9] C. A. C. Coello and G. B. Lamont. An introduction to multi-objective evolutionary algorithms and their applications. In C. A. C. Coello and G. B. Lamont, editors, *Applications of Multi-Objective Evolutionary Algorithms*, pages 1–28. World Scientific Publishing Co. Pte. Ltd., Toh Tuck Link, Singapore, 2004.

[10] C. L. Comunale, R. L. Rosner, and T. R. Sexton. The auditor's assessment of fraud risk: A fuzzy logic approach. *Journal of Forensic & Investigative Accounting*, 3(1):149–194, 2010.

[11] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60:141–153, 1993.

[12] S. Davalos, R. D. Gritta, and B. Adrangi. Deriving rules for forecasting air carrier financial stress and insolvency: A genetic algorithm approach. *Journal of the Transportation Research Forum*, 46(2):40–54, 2007.

[13] M.C.M. de Carvalho, M.S. Dougherty, A.S. Fowkes, and M.R. Wardman. Forecasting travel demand: a comparison of logit and artificial neural network methods. *Journal of the Operational Research Society*, 49(7):717–722, 1998.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2000.

[15] P. M. Dechow, W. Ge, C. R. Larson, and R. G. Sloan. Predicting material accounting misstatements. *Contemporary Accounting Research*, 28(1):17–82.

[16] S. Dehuri, A. Ghosh, and S. Ghosh. Genetic algorithm for optimization of multiple objectives in knowledge discovery from large databases. In S. Dehuri, A. Ghosh, and S. Ghosh, editors, *Multi-Objective Evolutionary Algorithms for Knowledge Discovery from Databases*, pages 1–22. Springer-Verlag, Berlin, Germany, 2008.

[17] R. J. Dery and A. Reinstein. AICPA standard aids in detecting risk factors for fraud – American Institute of Certified Public Accountants Statement on Auditing Standards No. 82. Consideration of Fraud in a Financial Statement Audit. *Healthcare Financial Management*, 53(10):48–50, 1999.

[18] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., New York, NY, first edition, 2001.

[19] M.H. Dunham. *Data Mining*. Pearson Education Inc., Upper Saddle River, NJ, 2003.

[20] P. Dutot, K. Rzadca, E. Saule, and D. Trystram. Multi-objective scheduling. In Y. Robert and F. Vivien, editors, *Introduction to Scheduling*, pages 219–251. CRC Press, Boca Raton, FL, 2010.

[21] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin, Germany, 2003.

[22] K. M. Fanning and K. O. Cogger. Neural network detection of management fraud using published financial data. *International Journal of Intelligent Systems in Accounting Finance & Management*, 7:21–41, 1998.

[23] E. H. Feroz, T. M. Kwon, K. J. Park, and V. Pastena. The efficacy of red flags in predicting the SEC's targets: An artificial neural networks approach. *International Journal of Intelligent Systems in Accounting, Finance, and Management*, 9(3):145–157, 2000.

[24] A. Ghandar, Z. Michalewicz, M. Schmidt, T. Tô, and R. Zurbrugg. Evolving trading rules. In A. Yang, Y. Shan, and L. Bui, editors, *Success in Evolutionary Computation*, volume 92 of *Studies in Computational Intelligence*, pages 95–119. Springer Berlin / Heidelberg, Berlin, Germany, 2008.

[25] B. P. Green and J. H. Choi. Assessing the risk of management fraud through neural network technology. *Auditing: A Journal of Practice & Theory*, 16(1):14–28, 1997.

[26] T. Hersh. Fraud happens! Sarbanes-Oxley Its not just for public companies. *New Jersey TechNews*, 6(10):1–2, 2002.

[27] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research*. McGraw Hill, New York, NY, eighth edition, 2005.

[28] B. Hoogs, T. Kiehl, C. Lacomb, and D. Senturk. A genetic algorithm approach to detecting temporal patterns indicative of financial statement fraud: Research Articles. *International Journal of Intelligent Systems in Accounting and Finance Management*, 15:41–56, 2007.

[29] H. Ishibuchi, I. Kuwajima, and Y. Nojima. Evolutionary multi-objective rule selection for classification rule mining. In A. Ghosh, S. Dehuri, and S. Ghosh, editors, *Multi-Objective Evolutionary Algorithms for Knowledge Discovery from Databases*, volume 98 of *Studies in Computational Intelligence*, pages 47–70. Springer Berlin / Heidelberg, Berlin, Germany, 2008.

[30] H. Ishibuchi, T. Murata, and H. Tanaka. Construction of fuzzy classification systems with linguistic if-then rules using genetic algorithms. In S. K. Pal and P. P. Wang, editors, *Genetic Algorithms for Pattern Recognition*, pages 227–251. CRC Press, Boca Raton, FL, 1996.

[31] H. Ishibuchi and Y. Nojima. Multiobjective formulations of fuzzy rule-based classification system design. In *Joint 4th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2005) and the 11th Rencontres Francophones sur la Logique Floue et ses Applications (LFA 2005)*, pages 285–290. CRC Press, Boca Raton, FL, September 2005.

[32] H. Ishibuchi and T. Yamamoto. Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy Sets and Systems*, 141(1):59 – 88, 2004.

[33] H. Ishibuchi and T. Yamamoto. Rule weight specification in fuzzy rule-based classification systems. *IEEE Trans. on Fuzzy Systems*, 13:428–435, 2005.

[34] Y. Jin, B. Sendhoff, and E. Korner. Rule extraction from compact Pareto-optimal neural networks. In A. Ghosh, S. Dehuri, and S. Ghosh, editors, *Multi-Objective Evolutionary Algorithms for Knowledge Discovery from Databases*, pages 71–90. Springer-Verlag, Berlin, Germany, 2008.

[35] K. Kaminski, T. Wetzel, and L. Guan. Can financial ratios detect fraudulent financial reporting? *Managerial Auditing Journal*, 19(1):15–28, 2004.

[36] P. Kvam and J. S. Sokol. A logistic regression/Markov chain model for NCAA basketball. *Naval Research Logistics*, 53(8):788803, 2006.

[37] S. Lall. IMF predicts slower world growth amid serious market crisis. `http://www.imf.org/external/pubs/ft/survey/so/2008/RES040908A.htm`, April 2008.

[38] R. M. Landry, Jr., P. Lin, and G. D. Moyes. Raise the red flag: A recent study examines which SAS No. 99 indicators are more effective in detecting fraudulent financial reporting. *Internal Auditor*, 62(5):47–51, 2005.

[39] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms*. Kluwer Academic Publishers, Boston, MA, 2002.

[40] A. Levitin. *Introduction to the Design and Analysis of Algorithms*. Pearson Education, Inc., Upper Saddle River, NJ, second edition, 2007.

[41] K. Ma and F. Tzeng. Opening the black box - Data driven visualization of neural network. In *VIS 05: Proceedings of the IEEE Visualization 2005 (VIS05)*, pages 383–390. IEEE Computer Society, Los Alamitos, CA, 2005.

[42] T. E. McKee. A meta-learning approach to predicting financial statement fraud. *Journal of Emerging Technologies in Accounting*, 6:5–26, 2009.

[43] W. F. Messier, Jr., S. M. Glover, and D. F. Prawitt. *Auditing & Assurance Services*. McGraw-Hill Irwin, New York, NY, first edition, 2008.

[44] R. Miller. Global recession risk grows as U.S. 'Damage' spreads (Update2). `http://www.bloomberg.com/apps/news?pid=newsarchive&sid=arlKrFbn3pfY&refer=home`, January 2008.

[45] P. Munter and T. A. Ratcliffe. Auditor's responsibilities for detection of fraud. *National Public Accountant*, 43(6):37–43, 1998.

[46] P. Norvig and S. Russell. *Artificial Intelligence: A Modern Approach*. Pearson Education, Inc., Upper Saddle River, NJ, third edition, 2010.

[47] T. Okabe, Y. Jin, and B. Sendhoff. A critical survey of performance indices for multi-objective optimisation. In *Proc. of 2003 Congress on Evolutionary Computation*, pages 878–885. IEEE Press, 2003.

[48] D. E. O'Leary. Using neural networks to predict corporate failure. *Intelligent Systems in Accounting, Finance and Management*, 7(3):187–197, 1998.

[49] Oracle ThinkQuest Education Foundation. Brief history of neural networks. `http://library.thinkquest.org/C007395/tqweb/history.html`.

[50] M. Pelikan, A. Hartmann, and K. Sastry. Hierarchical BOA, cluster exact approximation, and Ising spin glasses. Technical Report 2006002, Missouri Estimation of Distribution Algorithms Laboratory (MEDAL) - University of Missouri in St. Louis, St. Louis, MO, 2006.

[51] O. Persons. Using financial statement data to identify factors associated with fraudulent financial reporting. *Journal of Applied Business Research*, 11(3):38–46, 1995.

[52] C. A. P. Reyes. *Coevolutionary Fuzzy Modeling*. Springer-Verlag, Berlin, Germany, first edition, 2004.

[53] A. Rooney. *The Story of Mathematics*. Arcturus Publishing Limited, London, England, 2008.

[54] C. J. Skousen, K. R. Smith, and C. J. Wright. Detecting and predicting financial statement fraud: The effectiveness of the fraud triangle and sas no. 99. In M. Hirschey, K. John, and A. K. Makhija, editors, *Corporate Governance and Firm Performance (Advances in Financial Economics)*, pages 53–81. Emerald Group Publishing Limited, New York, NY, 2009.

[55] C. J. Skousen and C. J. Wright. Contemporaneous risk factors and the prediction of financial statement fraud. *Journal of Forensic Accounting*, 9(1):37–62, 2006.

[56] G. Winkler. *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods*. Springer-Verlag, Berlin, Germany, second edition, 2003.

Appendix A

FINANCIAL DATA — VARIABLES AND SUMMARY STATISTICS

A.1 AAER Sample Composition and Selection Procedures

A statistical synopsis of the procedures that were conducted on the original list of AAER data records during the data mining process.

	AAER Records
Preliminary set	1,608
Less: Employee and auditor citations, and corporations without a GVKEY identifier	(997)
Less: Citations that were either duplicates, unrelated to a financial misstatement, or issued for pre-1992 fraud	(273)
Less: Incomplete data records	(95)
Final set	243

A.2 Non-AAER Sample Composition and Selection Procedures

A statistical synopsis of the procedures that were conducted on the original set of matched, non-AAER corporate data records during the data mining process.

	Non-AAER Records
Initial set of firms that were matched to AAER firms with a post-1991 fraud period	469
Less: Incomplete data records	(233)
Final set	236

A.3 Overall, Combined Sample Composition

Summary statistics regarding the composition of the final, combined financial dataset, which includes a mixture of both AAER and matched, non-AAER corporate data records.

	Dataset
AAER firms	243
Non-AAER, matched firms	236
Total firms	479

A.4 Financial Variable Definitions

Each of the 16 financial variables served as a proxy for a *SAS No. 99* red flag indicator that is associated with one of the three Fraud Triangle components: *pressure*, *opportunity*, and *rationalization*. In tables A.4.1 through A.4.3, the financial variables are defined and grouped according to the Fraud Triangle category for which they served as a proxies. Table A.4.4 defines the 4 financial variables that were removed during the data preprocessing phase of the data mining process. Finally, table A.4.5 introduces the *SAS No. 99* red flags that were represented by the financial variables.

A.4.1 Pressure Financial Variables

Variable	Definition	Source
SCHANGE	$\frac{\Delta Sales}{Sales_{t-1}} - \frac{\Delta Avg.\ Sales^{Industry}}{Avg.\ Sales_{t-1}^{Industry}}$	[54]; Compustat code: REVT
ROA	Return on Assets: $\frac{NetIncome}{Assets_{t-1}}$	Compustat codes: AT, NI
CATA	Cash Flows to Earnings Growth[1]	Compustat codes: AT, OIBDP, OANCF
ROE	Return on Equity: $\frac{NetIncome}{Equity_{t-1}}$	Compustat codes: NI, SEQ
LEV	Leverage: $\frac{Debt_{LongTerm}}{Assets}$	Compustat codes: AT, DLTT
ACHANGE	$\frac{Assets_{t-1} - Assets_{t-2}}{Assets_{t-2}}$	[54]; Compustat code: AT
FREEC	Free Cash Flow[2]	[54]; Compustat codes: AT, CAPX, DV, OANCF
FINANCE	Demand for Financing[3]	[54]; Compustat codes: AT, CAPX, OANCF

[1] $CATA = \frac{OIBDP - OperatingNetCash}{Assets}$, where *OIBDP* is operating income before depreciation.

[2] $FREEC = \frac{OperatingCash - CashDividends - CapitalExpenditures}{Assets}$

[3] $FINANCE = \frac{OperatingCash_t - Avg.\ CapitalExpenditures_{t-2\,\text{to}\,t}}{Assets_{t-1}}$

A.4.2 Opportunity Financial Variables

Variable	Definition	Source
RECEIVABLE	$\frac{Receivables_t}{Revenue_t} - \frac{Receivables_{t-1}}{Revenue_{t-1}}$	[54]; Compustat codes: RECT, REVT
INVENTORY	$\frac{Inventory_t}{Revenue_t} - \frac{Inventory_{t-1}}{Revenue_{t-1}}$	[54]; Compustat codes: INVT, REVT
MARKETDOM	Market Domination: $\frac{Revenue}{Avg.\ Revenue^{Industry}}$	Compustat code: REVT
DIFFAUD	Difficult to Audit Transactions: $\frac{Receivables}{Revenue}$	[23]; Compustat codes: RECT, REVT
UMARGIN	Unusual Margin[4]	Compustat code: GPM
UROE	Unusual Return on Equity[5]	Compustat code: ROE

A.4.3 Rationalization Financial Variables

Variable	Definition	Source
TACC	Total Accruals[6]	Compustat codes: RECT, INVT, ACO, AP, TXP, LCO, DP, AT
CACC	Current Accurals: $CACC = \frac{OIBDP - OperatingNetCash}{Revenue}$, where *OIBDP* is operating income before depreciation.	Compustat codes: OIBDP, OANCF, REVT

[4] $UMARGIN = \frac{Avg.\ GPM^{Industry}}{GPM}$, where $GPM = 1 - \frac{Cost\ of\ Goods\ Sold}{Revenue}$ is the gross profit margin.

[5] $UROE = \frac{Avg.\ ROE^{Industry}}{ROE}$, where *ROE* is the return on equity (refer to §A.4.1).

[6]

$$TACC = (\frac{1}{Assets_{t-1}}) \cdot (\Delta Receivables + \Delta Inventory + \Delta CurrentAssets - \Delta AccountsPayable - \Delta TaxesPayable - \Delta CurrentLiabilities - Depreciation)$$

A.4.4 Financial Variables Removed from Dataset

Variable	Definition	Source
ΔAUD	Equals 1 if a change in auditor occurred during the year of fraud; otherwise, 0.	[54]; Compustat code: AU
AUDOPINION	Equals 1 if an unqualified audit opinion was issued; otherwise, 0.	[54]; Compustat code: AUOP
BIG4	Equals 1 if the external auditing firm is a member of the Big 4 (Deloitte & Touche, Ernst & Young, KPMG, PricewaterhouseCoopers); otherwise, 0.	Compustat code: AU
COMPLEXTRANS	Complex (Unusual) Transactions: (Extraordinary Items $\div$ Revenue)	Compustat codes: XI, REVT

A.4.5 SAS No. 99 Red Flag Definitions

Fraud Category	Red Flag	Proxy Variable(s)
Pressure	Recurring negative cash flows from operations	*CATA*, *FREEC*
	Rapid growth or unusual profitability	*SCHANGE*, *ACHANGE*
	Need to obtain additional debt or equity financing	*LEV*, *FINANCE*
	Operating losses	*CATA*
	High degree of competition or declining profit margins	*ROA*, *ROE*
Opportunity	A strong financial presence or ability to dominate a certain industry sector that allows the entity to dictate terms or conditions to suppliers or customers	*MARKETDOM*, *UMARGIN*, *UROE*
	Accounts based on significant estimates	*RECEIVABLE*, *INVENTORY*
	Significant, unusual, or highly complex transactions	*DIFFAUD*, *RECEIVABLE*, *INVENTORY*
Rationalization	Aggressive or unrealistic forecasts	*TACC*, *CACC*
	Interest by management in employing inappropriate means to minimize reported earnings for tax	*TACC*, *CACC*

A.5 Summary Statistics

The following table specifies the standard deviation and mean values of the AAER and non-AAER observations for each financial variable, along with the results of a Welch Two-sample *t*-test that determined whether a significant difference existed between the means of the two groups (at the 95 percent confidence interval and significance level of 0.05). If the *p*-value is less than 0.05, then a significant difference exists between the two sample means (i.e. the alternate hypothesis of a non-zero difference is accepted).

	Standard Deviation		**Mean**		**Significance**	
Variable	AAER	Non-AAER	AAER	Non-AAER	*t*-Stat	*p*-value
SCHANGE	2603.320	1649.720	605.275	162.641	2.229	0.026
ROA	0.448	0.280	-0.115	-0.048	-1.96	0.050
CATA	0.151	0.145	0.049	0.014	2.555	0.011
ROE	1.108	1.748	-0.197	-0.088	-0.810	0.418
LEV	0.189	0.219	0.179	0.187	-0.443	0.658
ACHANGE	5.010	110.837	1.302	8.150	-0.948	0.344
FREEC	0.201	0.286	-0.076	-0.052	-0.069	0.020
FINANCE	0.346	0.240	-0.103	-0.021	-2.999	0.003
RECEIVABLE	0.878	0.579	-0.041	-0.008	-0.166	0.100
INVENTORY	5.482	10.794	-0.466	0.568	-1.316	0.189
MARKETDOM	0.038	0.029	0.014	0.011	0.947	0.344
DIFFAUD	1.017	1.095	0.384	0.315	0.723	0.470
UMARGIN	15.575	310.942	-1.593	18.310	-0.982	0.327
UROE	3.600	4.063	0.443	0.335	0.307	0.759
TACC	0.661	0.271	0.054	-0.040	2.035	0.043
CACC	14.766	44.245	-1.300	-2.783	0.490	0.625

Appendix B

CLASSIFICATION STATISTICS

This appendix chapter presents the detailed statistics of the classification experiments and analyses that were conducted within this thesis research.

B.1 Logistic Regression Classifier

The Wald statistical hypothesis test was employed to measure the significance of the financial variable coefficients within the regression hypothesis $f_{Logistic}$. Within this test a chi-square distribution was used to produce a p-value for the Wald statistic $z = \left(\frac{\hat{\theta}}{se(\hat{\theta})}\right)^2$, where $\hat{\theta}$ is the maximum likelihood estimate of a coefficient θ and $se(\hat{\theta})$ is the standard error of the maximum likelihood estimate. Each coefficient was considered significant if it possessed a p-value less than 0.05 (95 percent confidence interval). The hypothesis test results are presented in the following table.

Variable	***z*-Stat**	***p*-value**
ROA	2.434	0.119
ROE	0.078	0.781
ACHANGE	0.324	0.569
TACC	0.890	0.346
CACC	2.787	0.095
LEV	0.542	0.462
DIFFAUD	0.459	0.498
FREEC	1.556	0.212
FINANCE	1.398	0.237
CATA	7.412	0.007
RECEIVABLE	1.431	0.232
INVENTORY	0.627	0.428
SCHANGE	3.748	0.053
MDOM	0.062	0.804
UMARGIN	1.279	0.258
UROE	0.181	0.671

B.2 Artificial Neural Network

The top-5 performing neural network parameter combinations in terms of the average validation accuracy rate. Each model was trained for 5,000 epochs.

Hidden Nodes	Learning Rate	Momentum Rate	Training Accuracy	Validation Accuracy	Validation MSE
14	0.20	0.40	59.40	55.87	24.99
16	0.20	0.30	59.57	55.87	25.15
18	0.20	0.20	59.59	55.24	25.13
14	0.20	0.25	59.57	55.24	25.18
13	0.20	0.25	59.58	54.92	25.03

B.3 Genetic Algorithm

Classification statistics for the most-fit fuzzy rule-based classifier (chromosome) from each 5-fold cross-validation run. Each classifier possessed the highest fitness value at the end of the $3{,}000^{th}$ generation.

Cross-Validation Run	Training Accuracy Rate	Validation Accuracy Rate	Training Active Rules Ratio
1	63.42	58.95	35.00
2	60.26	61.05	40.00
3	63.16	60.00	45.00
4	60.00	63.16	35.00
5	63.68	56.84	40.00

B.4 MARLEDA

Classification statistics for the most-fit fuzzy rule-based classifier (chromosome) from each 5-fold cross-validation run. Each classifier possessed the highest fitness value at the end of the $5{,}000^{th}$ generation of MARLEDA.

Cross-Validation Run	Training Accuracy Rate	Validation Accuracy Rate	Training Active Rules Ratio
1	67.89	48.42	65.00
2	70.00	54.74	60.00
3	70.26	50.53	70.00
4	68.95	53.68	60.00
5	70.53	57.89	65.00

B.5 Financial Variable Frequency Distribution

Frequency distribution of the cumulative number of times each financial variable was active within the most-fit rule-based classifier from each of the five cross-validation runs of an evolutionary algorithm model.

Variable	***GA***	***MARLEDA***
ROA	1	6
ROE	1	7
ACHANGE	6	9
TACC	10	7
CACC	0	10
LEV	2	4
DIFFAUD	4	5
FREEC	3	12
FINANCE	12	14
CATA	2	9
RECEIVABLE	2	2
INVENTORY	2	12
SCHANGE	2	6
MDOM	3	4
UMARGIN	0	3
UROE	1	5