Machine Learning and Sampling Scheme: An Empirical Study of Money Laundering Detection

Yan Zhang¹ · Peter Trubey²

Accepted: 12 October 2018 / Published online: 25 October 2018

© This is a U.S. Government work and not under copyright protection in the US; foreign copyright protection may apply 2018

Abstract

This paper studies the interplay of machine learning and sampling scheme in an empirical analysis of money laundering detection algorithms. Using actual transaction data provided by a U.S. financial institution, we study five major machine learning algorithms including Bayes logistic regression, decision tree, random forest, support vector machine, and artificial neural network. As the incidence of money laundering events is rare, we apply and compare two sampling techniques that increase the relative presence of the events. Our analysis reveals potential advantages of machine learning algorithms in modeling money laundering events. This paper provides insights into the use of machine learning and sampling schemes in money laundering detection specifically, and classification of rare events in general.

Keywords Bootstrap · Machine learning · Money laundering · Rare event · Sampling scheme

JEL Classification G21 · G28

The authors thank Mike Carhill, Chau Do, Shermineh Haghani, and Irina Paley for helpful comments, and Sebastian Bautista for editorial assistance. The views expressed in this paper are those of the authors alone and do not necessarily reflect those of the Office of the Comptroller of the Currency or the U.S. Department of the Treasury.

✉ Yan Zhang
yan.zhang@occ.treas.gov

¹ Enterprise Risk Analysis Division, Office of the Comptroller of the Currency, 400 7th Street SW, Mail Stop 6E-3, Washington, DC 20219, USA

² Department of Applied Mathematics and Statistics, University of California Santa Cruz, Santa Cruz, USA

1 Introduction

Money laundering is the criminal practice of processing illegal money through a series of transactions to gain access to the legal financial system. It is a type of “financial crime with potentially devastating social and financial effects” (FFIEC 2014, p. 7). The amount of money laundered globally in a year is estimated to be 2–5% of global GDP, or 0.8–2 trillion in US dollars.¹ The United States Congress passed the Bank Secrecy Act (BSA) in 1970, which mandates that financial institutions identify and prevent dirty money from entering the U.S. financial system. The Money Laundering Control Act of 1986 further establishes money laundering as a federal crime. The USA PATRIOT Act of 2001 provides a legal framework to prevent international money laundering, focusing on terrorist financing.

Suspicious activity can be detected manually, whereupon observing abnormal behavior by a customer, an employee of the financial institution files a report. However, as financial activities become prolific, the tremendous amount and speed of financial transactions call for an effective monitoring system that can process transactions quickly, preferably in real time. Moreover, as technologies advance, more and more transactions are conducted in the absence of personal interaction, eliminating the opportunity for manual scrutiny. This trend of change has accelerated recently, showcased by the rapid growth of financial technology companies and financial disintermediation. Through this need, automated transaction monitoring systems have been established as an indispensable part of the anti-money laundering regime.

Typical transaction monitoring systems utilize rules to detect money laundering activities. Rules are basically if and then conditions with thresholds pre-defined based on historical data or business intuition. As money launderers continuously attack monitoring systems and modify their strategies to find loopholes, the rules need to be continuously validated and calibrated, which could be labor intensive and thus subject to delay. Moreover, rule-based systems focus on individual transactions or simple transaction patterns, therefore they might not be sufficient to detect complicated transaction patterns.

Conversely, machine learning develops models that are able to independently adapt when exposed to new data. They learn from previous computations to produce reliable and repeatable decisions and results. With the development of high-speed computing, machine learning can analyze large amounts of data quickly. Many of the machine learning algorithms are designed to capture complex patterns like nonlinear relationships between a dependent variable and explanatory variables, where the linear rules fall short. The abilities of machine learning to handle large data, analyze unstructured information, identify complicated patterns, and self-update with new information make it a good candidate model for detecting money laundering activity.

Despite an expanding strand of literature documenting applications of machine learning to credit risk and fraud,² literature on using machine learning in detecting

¹ United Nations Office on Drugs and Crime, “Money-laundering and globalization,” <https://www.unodc.org/unodc/en/money-laundering/globalization.html>.

² For examples, Khandani et al. (2010) used decision tree to forecast consumer credit risk; Butaru et al. (2016) introduced decision tree, random forest, and logistic regression to combined consumer tradeline,

money laundering activity is still limited. Most of the relevant literature are of survey nature (Bolton and Hand 2002; King et al. 2018; Phua et al. 2010; Sudjianto et al. 2010). While a few authors study the use of machine learning in anti-money laundering by utilizing actual bank or firm data in foreign countries (Tang and Yin 2005; Wang and Yang 2007; Liu et al. 2008; Lv et al. 2008; Le-Khac et al. 2010; Villalobos and Silva 2017), documentation of machine learning applications in anti-money laundering in the U.S. is often descriptive or suggestive, with little detail on data, model, and performance (Senator et al. 1995; Kirkland et al. 1999). This is largely due to the lack of public datasets that can be used to analyze money laundering detection. While general transaction data can be utilized to perform unsupervised machine learning so as to identify outliers, it is hard to obtain money laundering detection outcomes that are needed by supervised machine learning to conduct predictive analysis and more importantly to evaluate model performance.³ Investigation outcome measures such as Suspicious Activity Report (SAR) or case indicator are considered sensitive information to be disclosed to the public; not to mention that law enforcement agencies typically do not provide feedback on whether reported suspicious activities are actually found to be of money laundering nature or not.

Another challenge in modeling money laundering activities is that they are rare events. Rarity of an event refers to the situation that the events of interest are few in number relative to the non-events. Examples of rare events include natural disaster, terminal disease, terrorist attack, war, financial fraud, or money laundering. These events are not frequent, yet their effects can be so detrimental that it is important to understand their causes in order to prevent them. In such cases, the false negative error (inaccurately classifying events as non-events) carries a much higher cost than the false positive error (inaccurately classifying nonevents as events). However, when models fail to take the cost differences associated with classification errors into account in cases of rare events, sub-optimal performance in discrimination is likely to occur. In addition, due to their rare occurrence, the information associated with the events tends to be buried by the dominating non-events, thus imposing challenges for the model to effectively sort through the various signals in the data and to accurately separate the two types of events. Because of these problems, models can perform sub-optimally in discriminating events from non-events when the events are rare.

Approaches proposed to address the rare event issue can be divided into two main categories. One is to tailor the algorithm, for example, adjusting the misclassification costs for the various classes. The other is to manipulate the data prior to modeling in order to magnify the information contained in the event observations. This category is not model specific as it preprocesses the data so that models are estimated on a then well-balanced dataset. There are two approaches loosely falling into this category. The first is to modify the target variable. An alternative target variable can be used so that it is closely related to the original target variable and yet occurs more often. For

Footnote 2 continued

credit bureau, and macroeconomic variables to predict credit card delinquency; Tam and Kiang (1992) and Altman et al. (1994) applied neural networks to bank/firm failure predictions; Cecchnin et al. (2010) used support vector machines for detecting corporate management fraud using basic financial data.

³ Depending on whether the dependent variable is available, machine learning algorithms can be divided into supervised and unsupervised, with the former has the dependent variable and the latter does not.

example, given the low frequency of corporate default, the agency rating can be used instead to model corporate default risk. But such an alternative is not always readily available and may not necessarily reflect the true relationship to be modeled, thus arises the second and more common technique, which involves sampling to magnify the event rate. Two popular sampling approaches to deal with rare event data are to either under-sample the majority class or over-sample the minority class.

This paper conducts an empirical study that provides insights into the use of machine learning and sampling schemes to better detect a particular type of rare event, money laundering activities. The analysis is based on actual transaction data from a U.S. financial institution, with investigation outcomes appended. This data enables us to evaluate predictive models. We study four representative supervised machine learning algorithms, including Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), and Artificial Neural Network (ANN). The traditional regression model used in handling a binary target, Maximum Likelihood Logistic Regression (MLLR), is included in our study as the benchmark. To address the impact of rare event data on MLLR fitting, a machine learning algorithm, Bayes Logistic Regression (BLR), is also studied. We introduce under- and over-sampling to alleviate the data imbalance and evaluate the sensitivity of the models to subsequent changes in the event rate. To the best of our knowledge, this is so far the most comprehensive empirical study of supervised machine learning models in the area of money laundering detection involving actual transaction data of a U.S financial institution. It is also believed to be the first attempt in modeling money laundering activities that applies under- and over-sampling to improve model performance.

Our analysis reveals advantages of machine learning algorithms in modeling money laundering events given the data and model specifications used. In particular, ANN consistently outperforms parametric logistic regressions with and without sampling; in high event rate regimes with sampling, SVM surpasses logistic regressions and RF delivers comparable performance.

The rest of the paper is organized as follows. Section 2 describes the data used. Section 3 discusses the models, sampling schemes, model performance evaluation criteria, and model parameter optimization. Section 4 presents the results. Section 5 concludes the paper.

2 Data

Oftentimes, transaction surveillance systems detect money laundering activities in a hierarchical process.⁴ The first layer is alert detection in which transactions are flagged as alerts for potential risk. Alert detection aims to cast a wide net to capture suspicious activities, typically involving the use of outlier detection techniques such as rules and clustering analysis. Alerts deemed to potentially reflect higher risk are escalated to the next level, where closer scrutiny from experienced analysts is applied. In the banking industry these escalated alerts are typically called cases or productive alerts. For the remainder of the paper, we will use the term cases. Finally through investigation, cases

⁴ Sudjianto et al. (2010) provided a description of a general anti-money laundering process.

determined to present strong money laundering risk by the financial institution are filed as SARs to the Financial Crimes Enforcement Network (FinCEN).⁵

Our paper focuses on the application of machine learning in the later stages of the decision-making process from alerts to SARs. The analysis is conducted using actual data from a U.S. financial institution. The data comprise a sample of alerts from the institution's business for 13 months. The first 8 months' data are used for analysis and the remaining 5 months' data are used for test purposes. The test data come from a time period different from the analysis data, thus providing an out-of-time validation of model performance. The datasets report a SAR indicator that equals 1 if the alert is subsequently filed as a SAR, and 0 otherwise. As Table 1 shows, the analysis data contain 3922 alerts with 28 SARs, yielding a very low SAR rate of 0.71%. The test data have 6 SARs out of 2157 total alerts, leading to an even lower SAR rate of 0.28%. These two datasets highlight the typical challenges in capturing rare events that money laundering transaction monitoring faces, therefore providing a venue for various models to showcase their ability to tackle this issue. The case event rate is relatively higher at 5.48% and 4.13% for the analysis and test datasets, respectively. We use the case event as an alternative target variable to measure model performance.⁶

The datasets contain roughly a hundred explanatory variables, based on rich information on customer/account characteristics, recent transactions, and alert management outcomes if applicable. Through Know Your Customer (KYC) and Customer Due Diligence (CDD) processes, financial institutions obtain information about the customer/account, such as risk rating, account age and location. For each alert, the institution provides transaction activities summarized over a period of time. The transaction summary variables reflect elements such as count, amount, percent of transactions by type (for example, cash, wire, and deposit/withdrawal) and direction (inbound or outbound). Some variables are binary outcomes after imposing pre-defined detection rules to raw transactions.

Based on a detailed data exploration, we have focused on ten explanatory variables to build models. Our variable selection process includes variable transformations and missing/invalid imputation, variable clustering, and correlation analysis to identify related variables (to exclude them from simultaneous appearance in a model), variable importance measurements from randomForest, and stepwise regression from several starting sets (using Akaike's Information Criterion). To ensure a fair comparison, we use the same ten variables for all models. Since the data contain sensitive real transaction information, we are not allowed to disclose the definition of the ten explanatory variables here, and denote them generally as $x_1, x_2, \dots, x_{10}$.

As illustrated by Table 1, variables $x_1 - x_5$ are binary variables that take values of 0 or 1; $x_6 - x_7$ are ratios bounded within the $[0, 1]$ interval; and variables $x_8 - x_{10}$

⁵ "Banks are required to report suspicious activity that may involve money laundering, BSA violations, terrorist financing, and certain other crimes above prescribed dollar thresholds. However, banks are not obligated to investigate or confirm the underlying crime (e.g., terrorist financing, money laundering, tax evasion, identity theft, and various types of fraud). Investigation is the responsibility of law enforcement." (FFIEC 2014, pp. 67–68).

⁶ There could be a variety of measures of model performance. We use (alert-to-) SAR and (alert-to-) case indicators as two ways to measure model performance in this paper.

Table 1 Summary statistics

Variable	Mean	SD	% of 0	Min	25th Pctl.	50th Pctl.	75th Pctl.	Max
Analysis data (3922 observations)								
SAR	0.71%							
Case	5.48%							
x1	0.15							
x2	0.33							
x3	0.12							
x4	0.02							
x5	0.18							
x6	0.25	0.40	36	0	0	0.004	0.43	1
x7	0.15	0.33	69	0	0	0	0.03	1
x8	5.08	0.48	0	3.44	4.70	5.14	5.41	8.23
x9	0.75	1.55	80	0	0	0	0	6.19
x10	3.23	2.26	31	0	0	4.35	5.06	7.24
Test data (2157 observations)								
SAR	0.28%							
Case	4.13%							
x1	0.16							
x2	0.39							
x3	0.12							
x4	0.17							
x5	0.17							
x6	0.18	0.35	61	0	0	0	0.06	1
x7	0.11	0.29	80	0	0	0	0	1
x8	5.19	0.48	0	3.44	4.82	5.22	5.48	8.60
x9	0.78	1.59	79	0	0	0	0	5.83
x10	3.58	2.22	26	0	0	4.67	5.21	8.30

The data comprise a sample of alerts sourced from a U.S. financial institution's business for 13 months. The first 8 months of data are used for analysis, while the remaining 5 months of data provide an out-of-time test

are continuous variables. Table 1 shows that the distributions of the ten variables are relatively stable between the analysis and test data.

3 Methodology

This section discusses the six models used in this paper, sampling schemes proposed to address the rare event problem, criteria used to measure model performance, and model parameter optimization. The analysis is performed using the *R* Statistical Computing environment (R Core Team 2016).

3.1 Models

3.1.1 Maximum Likelihood Logistic Regression

The Maximum Likelihood Logistic Regression model (Hosmer and Lemeshow 2000) is the conventional method used to predict the probability of a binary event. It is a parametric statistical model that takes in inputs (i.e., explanatory or independent variables) and outputs a predicted probability of the output class (i.e., target or dependent variable), which is a continuous variable bounded between 0 and 1. The MLLR model achieves this through a logistic transformation of the weighted sum of the inputs.

Assume Y is the binary target variable that takes a value of 1 for an event, and 0 for a non-event; X is the vector of explanatory input variables, the response probability it is denoted by

$$\pi = \Pr(Y = 1|X). \quad (1)$$

The logistic regression model relates the logit of the response probability to a linear combination of the explanatory variables,

$$\text{logit}(\pi) = \log\left(\frac{\pi}{1 - \pi}\right) = \beta_0 + \beta'_X X. \quad (2)$$

The coefficients of the logistic regression can be estimated by maximizing the log-likelihood function

$$\begin{aligned} \log l(\beta) &= \log \left[\prod_{i=1}^n \pi(X_i)^{Y_i} \{1 - \pi(X_i)\}^{1-Y_i} \right] \\ &= \sum_{i=1}^n [Y_i \log\{\pi(X_i)\} + (1 - Y_i) \log\{1 - \pi(X_i)\}], \end{aligned} \quad (3)$$

where for each data-point i , X_i is the vector of features, and Y_i is the observed class; $i = 1, \dots, n$, with n being the number of observations. There is no closed-form solution to the objective function with regard to the coefficients, so the MLLR model is fitted using an iterative process (for example, the Newton–Raphson algorithm). Data describing rare events will commonly exhibit separation, where one or more predictors perfectly or near perfectly predict the outcome, the former termed complete separation and the latter quasi-complete separation. When complete separation occurs, the corresponding logistic regression coefficients mathematically estimated via maximum likelihood are infinite.⁷ Even under quasi-separation, the magnitude of coefficients can still be unreasonably large and unstable.

In this paper, MLLR is fitted in R with the *glm* function from the *stats* package, available in base R . By capturing model fit warning messages indicating separation

⁷ The fitting algorithm in R stops before achieving truly infinite maximum likelihood estimates. In this way it returns finite estimates with a warning regarding fitted probabilities being equal to 0 or 1.

in R , we are able to capture and report situations when separation occurs to assess its impact.

3.1.2 Bayes Logistic Regression

The Bayes Logistic Regression induces stability and finiteness in the coefficient estimates by imposing a prior distribution on the calculation of the coefficients. Therefore, BLR is expected to be advantageous over MLLR in the presence of separation in the data used to train the model. There are a number of priors to choose from: Firth (1993) used a non-informative Jeffreys prior, while Gelman et al. (2008) proposed a weakly informative Cauchy distribution as the prior. Gelman et al. (2008) showed that the Cauchy prior outperforms many other priors including the Firth's, Gaussian, and Laplace; moreover, it has a nice property to always produce estimates even when there is complete separation in logistic regression. We therefore use the Cauchy prior for BLR.

The modeling of BLR analysis in this paper is performed using the *arm* package in R (Gelman and Su 2016). Gelman et al. (2008) recommended to standardize input variables before model estimation, where real-valued variables are normalized to have mean 0 and standard deviation 0.5, and binary variables are centered; we follow this practice in fitting BLR. The default setting of Gelman's BLR assumes a Cauchy distribution with center 0 and scale 2.5. The scale regulates how strong the prior is: the smaller the scale, the more condensed the prior.

3.1.3 Decision Tree

Decision tree (Breiman et al. 1984; Kass 1980; Quinlan 1987, 1993) is a non-parametric algorithm that recursively partitions the entire sample space into mutually exclusive and exhaustive subsets (or strata). Each subset is defined by the set of decision rules that lead to its creation, separating it from all other subsets. Decision tree can be used for both categorical and continuous target variables.

There are two major components of a DT algorithm: the selection criterion and the stopping rules. For a categorical target, the selection criterion can be misclassification error, Gini index, or information entropy; for a continuous target, the selection criterion is generally the least squared error. Starting with the whole training dataset, DT searches over all predictor variables at all possible split points to find the optimal splits that satisfy the selection criterion. This process is repeated recursively until the stopping rules take effect.

As a non-parametric approach, DT does not make any assumptions as to distribution, nor does it require a functional form for the predictors. Compared with parametric methods such as logistic regression, its flexible form makes it easier to capture non-linearity and handle missing values and outliers. However, DT is often subject to over-fitting, which is a common issue for non-parametric methods. To ensure that the algorithm is capturing patterns instead of noises in the data, some level of supervision is needed. Users can create stopping rules limiting the maximum number of branches

from a node, the maximum depth of a tree, the minimum leaf⁸ size; or to first grow a full tree, then prune the tree back using a complexity cost function.

Our analysis uses the C5.0 model (Quinlan 1993) to build a single tree, leveraging the *C50* package in *R* (Kuhn et al. 2015). The criterion used by C5.0 to evaluate splits is information entropy. The C5.0 model employs several additional measures to guide the fitting of the model. It offers the ability to *winnow* the predictors, which is to remove predictor variables that offer little improvement in performance to the model, and to specify a minimum number of observations (*minCase*) in each terminal leaf node. The C5.0 model additionally employs a pruning mechanism to prune back the tree to make it generalize better to new data. This post-growth pruning is guided by a parameter, the confidence factor *cf*, bound on the unit interval.

3.1.4 Random Forest

Decision Tree is inherently unstable in that a small change to the training dataset can have an extremely large impact on the structure of the grown tree. Breiman (2001) proposed RF, arguing that by using the outcomes of multiple trees, the instability can be used to explore more signals in the data than a single tree could do alone. As RF is built from multiple models acting in concert, it is termed an *ensemble* model. Compared with a single tree, RF attempts to be more robust as it leverages multiple trees built on different samples, which have been made significantly more random to reduce the effect of correlation between samples or between trees.

In this paper, we use the *randomForest* package in *R* (Liaw and Wiener 2002). The model uses a scoring function similar to the Gini Index to evaluate splits. The algorithm works by generating an ensemble of unpruned trees of dimension n_{tree} , each created using a bootstrap sample of the training data and random feature selection at each node in building the tree. For each node, an attempt at a split is made on a subset of covariates (or features) of the dimension m_{try} . The resulting trees grown from this method are not pruned so as to allow for more randomness. For a classification problem, prediction is made by aggregating (via majority vote or averaging class probabilities) the predictions of the ensemble.

3.1.5 Support Vector Machine

Soon after it was proposed (Cortes and Vapnik 1995), SVM gained significant popularity in the data mining and machine learning fields. Hyperplanes are used to linearly separate points mapped onto a higher-dimensional space. The collections of data points used to construct these hyperplanes are called support vectors. The key philosophy of SVM is to achieve maximum margin data separation, which in itself contains two objectives: to maximize the minimum margin, and to separate the dependent variable by hyperplanes. The margin is the distance between the hyperplane and the observations. However, it is not always possible or could be very complicated to find hyperplanes that completely separate the response. Therefore soft instead of hard hyperplanes are used to allow a small percentage of misclassifications.

⁸ The strata at the bottom (or final) level of a decision tree are called leaves.

A high dimensional feature space (number of explanatory variables and their derived terms) is more descriptive. However, if the dimension of the feature space is large compared to the number of observations, solving an objective function would be challenging. To address this, SVM typically employs a kernel trick, which uses a kernel function to implicitly map data to another feature space. To describe this feature space, it is sufficient to know the pairwise inner products of observations in the data. Training the SVM then involves choosing the particular observations, or vectors, that will support the hyperplane. Cecchnin et al. (2010) showed that using SVM with financial kernel is a useful method in discriminating between fraudulent and nonfraudulent companies.

SVM performs well in classifying non-linearly separable groups; they do not require large training datasets. Conversely, its calculation could be burdensome for large datasets. More relevantly, SVM seems to suffer performance loss while classifying rare events. This is potentially caused by observed events lying far from the ideal boundary, weakness of the soft margin, or imbalanced support vector ratio (Akbani et al. 2004; Wu and Chang 2003). We expect sampling methods used to increase the event rate will improve the model performance for SVM.

We use ν -classification from the *e1071* package in *R* (Meyer et al. 2017), which provides an interface from *R* to the *libsvm* program developed by Chang and Lin (2011). For our analysis we fit SVM with the polynomial kernel function. We tried other kernel functions, notably radial basis and sigmoid, and the polynomial kernel provided the best performance. This model's tunable parameters include γ which describes how far the influence of an individual training example reaches; ν which balances the number of support vectors and margin errors; *degree* which is the degree of the polynomial kernel function; and *coef0*, an independent term in the kernel projection that essentially creates a lower bound on the distance between points in the inner-product space.

3.1.6 Artificial Neural Network

First introduced in McCulloch and Pitts (1943), ANN was developed with the intent to mimic the neurophysiology of the human brain by combining many simple computing units (alternatively called perceptrons, neurons, or nodes) into a highly interconnected system. The most popular form of neural network architecture is the single-hidden layer feedforward network (SLFN), which we adopt for conducting the neural network analysis. The SLFN can be expressed mathematically as

$$y = G \left(a_0 + \sum_{j=1}^h a_j v_j \right), \quad (4)$$

$$v_j = F \left(\beta_{0j} + \sum_{i=1}^k \beta_{ij} x_i \right), \quad (5)$$

where $x_i, i = 1, \dots, k$ are the input units and $v_j, j = 1, \dots, h$ are the hidden units. The term β_{ij} is the weight of the connection between the input-layer unit x_i and the hidden-layer unit v_j ; a_j is the weight of the connection between the hidden-layer unit v_j and the output y ; β_{0j} is the bias of hidden-layer unit v_j ; a_0 is the bias of

the output y . $F(\cdot)$ is the activation function of the hidden-layer units, and $G(\cdot)$ is the activation function of the output. The activation functions are frequently sigmoid functions: s-shaped functions such as the logistic and hyperbolic tangent that produce bounded values within a range of $[0, 1]$ or $[-1, 1]$. Sigmoid functions are preferred as activation functions in that they are differentiable, which allows using Hessian-reliant optimization algorithms to solve for the weights.

Artificial Neural Network is known as the universal approximator (Hornik et al. 1989; Cybenko 1989; Haykin 1998), as with enough hidden units (i.e., large enough k), it can achieve predictions with very small error terms. However, focusing on minimizing errors to near-perfection on the training data may lead to a model that does not generalize well to other data. Our analysis makes use of the *nnet* package in *R* (Venables and Ripley 2002). In *nnet*, controlling the training of the model such that it generalizes to new data can be accomplished by limiting the complexity of the fitted model. This is achieved in two ways: limiting k , the number of hidden units, and during the training process iteratively shrinking the weights towards 0 through a weight decay parameter ω . The logistic function is used as the activation function for both the hidden units and the output unit. To improve the convergence of the weights, input variables have been normalized to have a mean of 0 and standard deviation of 1.

3.2 Sampling Schemes

There are two implications associated with rare event data. First, the rate of occurrence of events to non-events is highly imbalanced, with non-events dominating events. Because of this, the event is alternatively called the minority class, and non-event the majority class. Second, with a very low event rate, it takes a large dataset to obtain a sufficient number of events for modeling.

Two commonly used sampling methods are under-sampling and over-sampling. Under-sampling forms a more balanced dataset by keeping all the events and taking a smaller-than-population random sample of the non-events. Over-sampling increases the event rate by keeping all non-events and events, and additionally sampling from the events with replacement to increase the number of events. As both under- and over-sampling are conducted conditional on the target variable, they are also termed as choice-based sampling, endogenous stratified sampling, or case-control design. Compared to the original data, under-sampling generates a significantly smaller dataset while over-sampling will increase the size for data with rare events. The main disadvantage of over-sampling is that it tends to over-fit as it makes exact copies of existing events. Additionally, with the increased data size given over-sampling, it can be computationally burdensome to train complicated algorithms. On the contrary, under-sampling reduces data size, easing the computational burden for training complex algorithms. Nonetheless, under-sampling might discard useful information as it only keeps a portion of the non-events. In the interest of comparison, we use both under-sampling and over-sampling in our analysis.

We define a sampling scheme as a combination of the sampling method and the target event rate. With 28 events out of 3922 total observations, the analysis dataset has a raw event rate of 0.71%. Imposing the sampling approach of over or under-

sampling, we study whether a model responds to various scenarios of increased event rates, which is meant to address the rare event issue. Specifically, we test boosted event rates from 1% through 40% incrementing by 1%. In total for each sampling approach, there are 41 event rate scenarios with 1 raw event rate and 40 increased event rates.

3.3 Performance Evaluation Measure

We measure the model performance by discrimination, which is quantified by Area Under the Curve (AUC). The curve here refers to the Receiver Operating Characteristic (ROC) curve.⁹ AUC is considered a statistically consistent and more discriminating measure than prediction accuracy (or $1 - \text{misclassification rate}$) (Bradley 1997; Ling et al. 2003). However as we are exploring the modeling of rare events, the ROC curve is often plagued by abrupt kinks, resulting in large variations in AUC even with a minor change in the data. To obtain a comprehensive picture of model performance, we employ a bootstrap algorithm to produce multiple samples of data and consequently AUCs. A bootstrap sample is a random sample of a dataset, taken with replacement, of size equal to or lesser than the original dataset. We perform bootstrap with equal size on the provided analysis dataset and the resulting bootstrap samples are used for training models.

One by-product of bootstrap is the holdout sample, which consists of observations that are not selected by the bootstrap. If a bootstrap sample is equal in size to the original dataset, then the probability of an individual observation being selected in the bootstrap sample is approximately 63.2% (Larose and Larose 2015). Therefore by bootstrapping the analysis dataset provided by the financial institution, two datasets can be created: the training and holdout datasets, representing approximately 63.2% and 36.8% of the original analysis data, respectively.

Due to the rare event nature of the data, we use stratified bootstrap sampling to ensure that the data have sufficient events. As a matter of fact, we started with un-stratified bootstrap and often ended up with samples with very few or even no events, which prompted us to consider stratified bootstrap sampling. Stratified bootstrap sampling is carried out as follows: the analysis dataset is separated into events and non-events; bootstrap samples are taken from the two strata separately, and then combined to form the training dataset. The holdout dataset is formed from those observations that are not selected into the training dataset during the stratified bootstrap.

Given the following initial conditions:

- Analysis dataset A (provided by the financial institution),
- Test dataset T (provided by the financial institution),
- Model $M \in \{\text{MLLR, BLR, DT, RF, SVM, ANN}\}$,
- Sampling approach $s \in \{\text{over-sampling, under-sampling}\}$,
- Target event rate $e \in \{0.71\%, 1\%, 2\%, \dots, 39\%, 40\%\}$,

⁹ A ROC curve illustrates the classification performance of a binary target. Its x-axis measures the false positive rate ($1 - \text{specificity}$) and the y-axis represents the true positive rate (sensitivity). A perfect prediction yields a point in the top left corner where the true positive rate is 1 and the false positive rate is 0. The prediction produced by a random guess follows the diagonal line with equal chance of false positive and true positive rates.

the process to generate the performance measure for each target event rate e can be outlined in the following steps:

1. Conduct stratified bootstrap on A to yield TR_i , the bootstrap training sample, and H_i , the bootstrap holdout sample, for $i = 1, 2, \dots, 100$.
2. For each sample i :
 - a. Run sampling method s to event rate e on TR_i to yield TR_{ise} .
 - b. Train model M on TR_{ise} to yield M_{ise} .
 - c. Calculate AUC θ for M_{ise} on TR_{ise} , H_i , and T to yield $\theta_{M_{ise}|TR_{ise}}$, $\theta_{M_{ise}|H_i}$, and $\theta_{M_{ise}|T}$.

This process in generating bootstrap samples largely resembles that used by Steyerberg et al. (2001). As some of the non-parametric algorithms studied here will have the tendency to overfit on the training sample, we rely on the holdout sample rather than the original analysis dataset to perform internal testing. For the combination of 6 model algorithms, 2 sampling methods, 41 event rates, and 100 bootstrap samples, there are 49,200 models to be built. For each model, 3 AUCs are calculated for the training, holdout, and test samples, generating 147,600 AUCs. We use the AUCs to study the interactions between models and sampling schemes for rare events. The sampling algorithm is written as described using base R .

3.4 Model Parameter Optimization

In the R packages used, all models except MLLR have parameters that can be calibrated to affect the fit of the model. We employ a grid search method to optimize these parameters for each model. First, a calibration range is determined to cover the potential values of the parameters. Next, following the performance evaluation algorithm previously described, for each possible parameter combination we create 100 bootstrap samples of the analysis data, over- and under-sample the bootstrapped data to a modest event rate of 10%, and build models on the adjusted bootstrap samples. We evaluate the models on the holdout datasets, searching for optimal parameter combinations that exhibited high average AUC and low variance in AUC. If, during optimization, an optimal value is found at an edge of the grid, then the grid is extended in that direction and optimization continues. Table 2 describes the final grid search ranges and optimal values found. These optimal values are then used in building the models for comparison.

4 Results

We present the estimation results in this section. For each of the models, we detail its performance through the under- and over-sampling methods using plots of the average AUC versus the event rate. Furthermore, we conduct regression analysis of AUCs to quantify the efficacy of models and sampling schemes in predicting rare events.

Table 2 Parameter optimization grid search

Model	Parameter	Calibration range	Optimal value
BLR	<i>scale</i>	0.5–2.5 (by 0.5)	1
DT	<i>minCase</i>	1–4 (by 1)	2
	<i>cf</i>	0.05–0.95 (by 0.05)	0.85
	<i>winnow</i>	True, false	False
RF	<i>mtry</i>	1–5 (by 1)	1
	<i>ntree</i>	20–30,000 (irregular)	5000
SVM	γ	10^{-4} – 10^3 (by $\times 10$)	10^{-3}
	ν	10^{-4} – 10^0 (by $\times 10$)	10^{-2}
	<i>degree</i>	1–4 (by 1)	4
	<i>coef0</i>	0–1 (by 0.5)	0.5
ANN	<i>k</i>	1–10 (by 1)	6
	ω	2^{-4} – 2^8 (by $\times 2$)	32

For each model, a grid search was performed along all dimensions of configurable parameters using 100 bootstrap samples of the analysis dataset, over- and under-sampled to a modest 10% event rate. The optimal values are determined based on model performance on the holdout datasets

4.1 Graphic Illustration

Figure 1 plots the performance of the six models with the under- and over-sampling methods across the training, holdout, and test datasets. Corresponding to an event rate with under or over-sampling, the AUCs obtained from the 100 bootstrap samples are averaged and plotted. We draw our conclusion based on the patterns revealed by the holdout and test samples. Without sampling (at the stock event rate), model performance ranked from best to worst is ANN, BLR, MLLR, RF, DT, and SVM. The models are clustered in two groups with the first four having an average AUC above 0.74 and the remaining two having an average AUC below 0.60.

The rank-ordering of the model performance changes with event rate. The ANN model is resilient to the effect of artificially increasing the event rate as shown by the essentially constant value of average AUC. The two logistic models remain to be the second best despite a mild downward trend in average AUC on the under-sampling test sample. The performance of RF improves with the event rate and then stabilizes at a level similar to the logistic models starting event rate of approximately 4%. Being the worst performer at the stock event rate, SVM benefits the most from sampling. Its average AUC ramps up continuously and surpasses logistic and RF in the event rate range of 15–20% to be the second best performer with under-sampling, and even overtake ANN to be the best performer in the high event regimes with over-sampling. Sampling also improves the model performance of DT but the improvement is not as substantial as it does to SVM. DT has the lowest AUC overall. As for differences in sampling method, it is observed that the pattern of AUC change with event rate is different for DT and SVM. The average AUC increases faster and to a higher level with over-sampling than with under-sampling.

Fig. 1 Average AUC. The figure plots average AUCs of 100 bootstrap samples for 3 datasets (training, holdout, and test), 6 model algorithms (MLLR, BLR, DT, RF, SVM, and ANN), 2 sampling methods (over- and under-sampling), at 41 event rates, using 147,600 AUCs in total

As mentioned previously, we also track the occurrence of data separation and evaluate its impact on the performance of the logistic regressions. The average rate of separation in the bootstrap samples of our data is 4.4% for under-sampling and 1.2% for over-sampling. Moreover, for under-sampling, the separation rate increases with the event rate: at the 40% event rate, 13 of the 100 bootstrap samples experienced separation. For MLLR, while separation may increase the value of AUC on the training sample, the resulting non-robust model leads to inferior model performance on the

test sample. The MLLR AUC curve shows a downward trend on the under-sampling test sample.

We perform the same analysis using the case indicator as the target variable. The rank-ordering of the model performance maintains a similar order as that using the SAR indicator.¹⁰

4.2 Regression Analysis

Graphs based on average AUC provide a visual illustration of the model performance in predicting rare events and the effect of sampling. However they do not reveal the variation in AUCs given the model, sampling approach, and event rate. We conduct an ordinary least squares (OLS) regression analysis to statistically test the efficacy of models and sampling schemes in predicting money laundering. The AUC is considered as the dependent variable. Given the dataset (holdout or test¹¹), the model (MLLR vs. the competing model), sampling approach (under- vs. over-sampling), and event rate (41 levels) are analyzed as the three main factors in the regression. Therefore each regression dataset consists of 16,400 AUCs with 100 bootstrap samples for each combination of model, sampling approach, and event rate. The squared and cubed terms of the event rate are introduced to account for its potential nonlinear relationship with AUC. The regression is set up to include two- and three-way interaction terms of the main effects. To summarize, the independent variables are an indicator for the competing model (CM), an indicator for under-sampling (UnSm), the event rate as a continuous variable (ER), squared and cubed event rates (ER^2 and ER^3), and two- and three-way interactions of CM, UnSm, and ER (including ER^2 and ER^3).

The modeling analysis is performed using *SAS Proc Reg* and the corresponding results are reported in Table 3. Two panels are constructed, with one for holdout and the other for test. Five models (BLR, DT, RF, SVM, and ANN) are listed as competing models for MLLR. For each competing model, the table reports the regression coefficients (scaled up by 100) and statistical significance. The R^2 value is also provided to show the overall goodness-of-fit of the regression.

The terms without CM describe the relationship of MLLR model performance with sampling method and event rate. As each regression uses the same MLLR model as the benchmark, these terms have the same coefficients across regressions in Table 3. Their statistical significances vary slightly with the change of competing model. As indicated by the intercept coefficient, the average AUC of MLLR is 77.70% on the holdout and 86.04% on the test dataset. The coefficients on UnSm, ER, ER^2 , ER^3 , UnSm $\times$ ER, UnSm $\times$ ER^2 , and UnSm $\times$ ER^3 are statistically insignificant on the holdout dataset, which suggests that AUC does not vary with either sampling method or event rate for MLLR. On the test dataset, the event rates are marginally significant with a negative first order effect in some regressions, which corroborates the subtle downward trend of AUC curve observed in Fig. 1 for MLLR.

The terms with CM record the interplay of competing model performance with sampling scheme relative to the benchmark. The CM coefficient reveals the difference

¹⁰ Results are not provided here for brevity, but are available upon request.

¹¹ Here we exclude the training dataset so as to assess robust model performance.

Table 3 Regression analysis of model, sampling method, and event rate

	BLR	DT	RF	SVM	ANN
Holdout					
Intercept	77.70***	77.70***	77.70***	77.70***	77.70***
CM	1.06*	− 19.65***	− 3.43***	− 30.63***	4.30***
UnSm	0.57	0.57	0.57	0.57	0.57
ER	13.09	13.09	13.09	13.09	13.09
ER ²	− 34.12	− 34.12	− 34.12	− 34.12	− 34.12
ER ³	13.76	13.76	13.76	13.76	13.76
CM × UnSm	− 0.44	− 0.85	0.56	2.43**	− 0.13
CM × ER	− 16.33	271.40***	61.51***	384.97***	− 10.83
CM × ER ²	81.56	− 1280.23***	− 223.67***	− 1454.15***	− 7.04
CM × ER ³	− 135.19	1772.35***	236.59*	1771.87***	50.65
UnSm × ER	− 15.38	− 15.38	− 15.38	− 15.38	− 15.38
UnSm × ER ²	37.18	37.18	37.18	37.18	37.18
UnSm × ER ³	− 45.86	− 45.86	− 45.86	− 45.86	− 45.86
CM × UnSm × ER	27.56	− 182.17***	33.08*	− 74.62***	− 6.35
CM × UnSm × ER ²	− 119.50	1032.91***	− 189.22*	386.91***	91.63
CM × UnSm × ER ³	177.32	− 1482.69***	322.77*	− 520.08**	− 137.33
Adjusted R ²	0.01	0.29	0.05	0.37	0.06
Test					
Intercept	86.04***	86.04***	86.04***	86.04***	86.04***
CM	1.34**	− 29.04***	− 6.33***	− 52.26***	5.89***
UnSm	− 0.69	− 0.69	− 0.69	− 0.69	− 0.69
ER	− 15.68*	− 15.68	− 15.68*	− 15.68	− 15.68**
ER ²	111.67**	111.67	111.67**	111.67	111.67**
ER ³	− 179.59**	− 179.59	− 179.59**	− 179.59	− 179.59**
CM × UnSm	1.12	− 0.65	1.59**	4.58***	1.15*
CM × ER	− 5.76	427.21***	145.95***	767.64***	7.90
CM × ER ²	− 3.02	− 2047.95***	− 712.76***	− 3170.15***	− 100.14
CM × ER ³	17.95	2793.98***	930.30***	4065.14***	154.17
UnSm × ER	9.35	9.35	9.35	9.35	9.35
UnSm × ER ²	− 87.68	− 87.68	− 87.68	− 87.68	− 87.68
UnSm × ER ³	124.64	124.64	124.64	124.64	124.64
CM × UnSm × ER	− 15.50	− 345.22***	− 29.78*	− 112.86***	− 7.07
CM × UnSm × ER ²	76.36	2159.20***	168.42*	609.33***	109.04
CM × UnSm × ER ³	− 114.89	− 3271.87***	− 234.26	− 855.92***	− 153.03
Adjusted R ²	0.01	0.33	0.06	0.49	0.24

Each regression dataset consists of 16,400 AUCs with 100 bootstrap samples for each combination of model, sampling method, and event rate. The target variable is AUC; the independent variables are an indicator for competing model (CM), an indicator for under-sampling (UnSm), the event rate as a continuous variable (ER), squared and cubed event rates (ER² and ER³), two- and three-way interactions of CM, UnSm, and ER (including ER² and ER³). The regression is performed using OLS. ***, **, and * denotes statistical significance level of 1%, 5%, and 10%, respectively

in average AUC between the competing model and MLLR; the interactions of CM with UnSm and event rate show the effect of sampling method and event rate on competing model performance as compared with MLLR. The first column of Table 3 presents the results of BLR vs. MLLR. The CM coefficient is positive with a small magnitude and marginal statistical significance, indicating that BLR improves over MLLR but the improvement is not substantial. Other than the AUC level difference by model, the AUC curve shape is very similar between the two models as suggested by the insignificance of the terms with CM. This is confirmed by the low adjusted R^2 .

The C5.0 model is less predictive than MLLR as shown by the highly significant and negative coefficient of the CM factor and the large R^2 value in both holdout and test samples. Its large coefficients on the interactions of ER, ER^2 , and ER^3 with CM indicate that the event rate matters for DT, confirming that sampling is effective to address the rare event issue for DT, *ceteris paribus*. The positive sign of $CM \times ER$ suggests that AUC increases with event rate under DT. The three-way interactions are highly significant with a large magnitude, which is consistent with Fig. 1 in that the impact of event rate varies by sampling approach for DT.

The low R^2 and the small negative coefficient of CM suggest that RF performs slightly worse than MLLR overall. However with the significant and large coefficients on the interactions of ER, ER^2 , and ER^3 with CM, and the positive sign of $CM \times ER$, AUC increases with elevated event rates for RF. This pattern of RF is similar to that of DT, though the impact is smaller as the corresponding coefficients are of a smaller magnitude.

The adjusted R^2 of SVM is the highest among the five competing models, which indicates that SVM performs differently from MLLR. Despite a large and negative coefficient on CM, the interactions of CM with ER, ER^2 , and ER^3 are of the highest magnitude and the sign of $CM \times ER$ is positive, echoing its better performance than the benchmark model for high event rates. For SVM, under-sampling has a small positive effect on average AUC, but a much larger negative effect on the interaction of AUC with event rate, suggesting that the model performance improves more as event rate increases with over-sampling.

The ANN model performs better than MLLR as suggested by the CM coefficient. The advantage is more obvious in the test sample as indicated by the high R^2 . The remaining variables are either not or marginally significant, reinforcing the observation from Fig. 1 that ANN is resilient to changes in event rate and sampling approach.

Overall, AUC is rather robust to event rate and sampling approach under MLLR, BLR, and ANN. The ANN model consistently outperforms MLLR, and BLR is marginally better than MLLR. Despite inferior performance at the stock event rate, inflating the event rate by sampling helps improve model performance for DT, RF, and SVM, with SVM being the most sensitive and RF the least. Under DT and SVM, the model performance improvement is more sensitive to event rate increases with over-sampling than with under-sampling. In high-event regimes, SVM surpasses logistic regression and RF delivers comparable performance.

5 Conclusions

Transaction monitoring is vital to safeguard the U.S. financial system from money launderers. As financial institutions begin to adopt statistical and machine learning models in their automated transaction monitoring systems, understanding the benefits and drawbacks of each model for its specific application to money laundering is important.

We conduct a comprehensive study of major machine learning models using actual transaction data provided by a U.S. financial institution. As money laundering occurrences are rare, we introduce under-sampling and over-sampling techniques to improve model performance and evaluate the changes they bring. To induce variability so as to measure performance robustly, we bootstrap the data prior to using the sampling scheme. We observe the best performance under ANN, making ANN a strong contender for a general use rare event classification algorithm. The SVM and RF models react favorably to sampling, delivering better or comparable performance compared to conventional logistic regression models in high event regimes.

It is worth noting that there are many factors impacting model performance. As these models are supervised methods, it is important to have historical data with events accurately assigned and input variables properly captured. If biases or errors are left in the data without being treated, they will get passed on to the model building process, resulting in erroneous models. It is of paramount importance that model parameters be optimized for the problem at hand; blindly applying default parameters will likely lead to sub-optimal performance. In addition, an effective anti-money laundering program consists of many components, such as KYC/CDD, independent review and audit, and employee training. Models need to work together with all the other components to form a well-woven net to prevent penetration of money laundering activities.

For future work, we can further extend the work to other models and sampling schemes. For example, multi-layer ANN can be conducted to allow deep learning of the data. There are a host of other sampling schemes that could be leveraged.¹² We do not discount the possibility that other models may exhibit better performance after thorough optimization.

References

- Akbani, R., Kwek, S., & Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. In *European conference on machine learning* (pp. 39–50). Berlin: Springer.
- Altman, E. I., Marco, G., & Varetto, F. (1994). Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the Italian experience). *Journal of Banking & Finance*, 18, 505–529.
- Barandela, R., Valdovinos, R., Sanchez, J., & Ferri, F. (2004). The imbalanced training sample problem: Under or over sampling? In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)* (pp. 806–814).
- Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. *Statistical Science*, 17, 235–249.

¹² Examples of other sampling techniques include one-sided selection (Kubat and Matwin 1997), Wilson's Editing (Barandela et al. 2004), Synthetic Minority Over-sampling (SMOTE) (Chawla et al. 2002), cluster-based over-sampling (Jo and Japkowicz 2004).

- Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30, 1145–1159.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32.
- Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. Boca Raton: CRC Press.
- Butaru, F., Chen, Q., Clark, B., Das, S., Lo, A., & Siddique, A. (2016). Risk and risk management in the credit card industry. *Journal of Banking & Finance*, 72, 218–239.
- Cecchin, M., Aytug, H., Koehler, G. J., & Pathak, P. (2010). Detecting management fraud in public companies. *Management Science*, 56(7), 1146–1160.
- Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2, 27.
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357.
- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20, 273–297.
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4), 303–314.
- Federal Financial Institutions Examination Council. (2014). *Bank Secrecy Act anti-money laundering examination manual*. https://www.ffiec.gov/bsa_aml_infobase/documents/bsa_aml_man_2014.pdf. Accessed 4 Mar 2018.
- Firth, D. (1993). Bias reduction of maximum likelihood estimates. *Biometrika*, 80, 27–38.
- Gelman, A., Jakulin, A., Pittau, M. G., & Su, Y.-S. (2008). A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics*, 2, 1360–1383.
- Gelman, A., & Su, Y.-S. (2016). *arm: Data analysis using regression and multi-level/hierarchical models*. R package version 1.9-3. <https://CRAN.R-project.org/package=arm>. Accessed 4 Mar 2018.
- Haykin, S. (1998). *Neural networks: A comprehensive foundation* (2nd ed.). Upper Saddle River, NJ: Prentice Hall PTR.
- Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366.
- Hosmer, D. W., & Lemeshow, S. (2000). *Applied logistic regression*, A Wiley-Interscience Publication. New York: Wiley.
- Jo, T., & Japkowicz, N. (2004). Class imbalances versus small disjuncts. *ACM SIGKDD Explorations Newsletter*, 6, 40–49.
- Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29, 119–127.
- Khandani, A. E., Kim, A. J., & Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11), 2767–2787.
- King, C., Walker, C., & Gurulé, J. (Eds.). (2018). *The Palgrave Handbook of Criminal and Terrorism Financing Law*. Cham: Palgrave Macmillan.
- Kirkland, J. D., Senator, T. E., Hayden, J. J., Dybala, T., Goldberg, H. G., & Shyr, P. (1999). The NASD regulation advanced-detection system (ADS). *AI Magazine*, 20(1), 55–67.
- Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. In *Proceedings of the fourteenth international conference on machine learning*.
- Kuhn, M., Weston, S., Coulter, N., & Culp, M. (2015). *C50: C5.0 decision trees and rule-based models*. R package version 0.1.0-24, C code for C5.0 by R. Quinlan. <https://CRAN.R-project.org/package=C50>. Accessed 4 Mar 2018.
- Larose, D. T., & Larose, C. D. (2015). *Data mining and predictive analytics*. New York: Wiley.
- Le-Khac, N. A., Markos, S., & Kechadi, M. T. (2010). A data mining-based solution for detecting suspicious money laundering cases in an investment bank. In *Second international conference on advances in databases, knowledge, and data applications*.
- Liaw, A., & Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2, 18–22.
- Ling, C. X., Huang, J., & Zhang, H. (2003). AUC: A statistically consistent and more discriminating measure than accuracy. In *International joint conference on artificial intelligence* (Vol. 3, pp. 519–524).
- Liu, X., Zhang, P., & Zeng, D. (2008). Sequence matching for suspicious activity detection in anti-money laundering. In *Proceedings of the IEEE ISI 2008 PAISI, PACCF, and SOCO international workshops on intelligence and security informatics*.
- Lv, L.-T., Ji, N., & Zhang, J.-L. (2008). A RBF neural network model for anti-money laundering. In *Proceedings of the 2008 international conference on wavelet analysis and pattern recognition*.

- McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5, 115–133.
- Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2017). *e1071: Misc functions of the Department of Statistics, Probability Theory Group (Formerly: E1071)*, TU Wien. R package version 1.6-8. <https://CRAN.R-project.org/package=e1071>. Accessed 4 Mar 2018.
- Phua, C., Lee, V., Smith, K., & Gayler, R. (2010). A comprehensive survey of data mining-based fraud detection research. arXiv preprint [arXiv:1009.6119](https://arxiv.org/abs/1009.6119).
- Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, 27, 221–234.
- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning* (Vol. 1). Los Altos, CA: Morgan Kaufmann.
- R Core Team. (2016). *R: A language and environment for statistical computing*. Vienna: R Foundation for Statistical Computing. <https://www.R-project.org/>. Accessed 4 Mar 2018.
- Senator, T. E., Goldberg, H. G., & Wooton, J. (1995). The financial crimes enforcement network AI system (FAIS): Identifying potential money laundering from reports of large cash transactions. *AI Magazine*, 16(4), 21–39.
- Steyerberg, E. W., Harrell, F. E., Borsboom, G. J., Eijkemans, M., Vergouwe, Y., & Habbema, J. D. F. (2001). Internal validation of predictive models: Efficiency of some procedures for logistic regression analysis. *Journal of Clinical Epidemiology*, 54, 774–781.
- Sudjianto, A., Nair, S., Yuan, M., Zhang, A., Kern, D., & Cela-Díaz, F. (2010). Statistical methods for fighting financial crimes. *Technometrics*, 52, 5–19.
- Tam, K. Y., & Kiang, M. Y. (1992). Managerial applications of neural networks: The case of bank failure predictions. *Management Science*, 38(7), 926–947.
- Tang, J., & Yin, J. (2005). Developing an intelligent data discriminating system of anti-money laundering based on SVM. In *Proceedings of the fourth international conference on machine learning and cybernetics*.
- Venables, W. N., & Ripley, B. D. (2002). *Modern applied statistics with S* (4th ed.). New York: Springer.
- Villalobos, M. A., & Silva, E. (2017). A statistical and machine learning model to detect money laundering: An application. http://hddavii.eventos.cimat.mx/sites/hddavii/files/Miguel_Villalobos.pdf. Accessed 4 Mar 2018.
- Wang, S.-N., & Yang, J.-G. (2007). A money laundering risk evaluation method based on decision tree. In *Proceedings of the sixth international conference on machine learning and cybernetics*.
- Wu, G., & Chang, E. Y. (2003). Class-boundary alignment for imbalanced dataset learning. In *International conference on machine learning, Workshop on Learning from Imbalanced Data Sets* (Vol. II, pp. 49–56).