Minimizing the Societal Cost of Credit Card Fraud with Limited and Imbalanced Data

Samuel Showalter, Zhixin Wu

Department of Mathematics, DePauw University, Greencastle, IN 46135, USA

Abstract

Machine learning has automated much of financial fraud detection, notifying firms of – or even blocking – questionable transactions instantly. However, data imbalance starves traditionally trained models of the content necessary to detect fraud. This study examines three separate factors of credit card fraud detection via machine learning. First, it assesses the potential for different sampling methods – undersampling and Synthetic Minority Oversampling Technique (SMOTE) – to improve algorithm performance in data-starved environments. Additionally, five industry-practical machine learning algorithms are evaluated on total fraud cost savings in addition to traditional statistical metrics. Finally, an ensemble of individual models is trained with a genetic algorithm to attempt to generate higher cost efficiency than its components. Monte Carlo performance distributions discerned random undersampling outperformed SMOTE in lowering fraud costs, and that an ensemble was unable to outperform its individual parts. Most notably, the F_1 Score, a traditional metric often used to measure performance with imbalanced data, was uncorrelated with derived cost efficiency. Assuming a realistic cost structure can be derived, cost-based metrics provide an essential supplement to objective statistical evaluation.

Keywords: Fraud, Linear SVC, Random Forest, Principal Component Analysis, Bayesian Theory, Logistic Regression, Monte Carlo, Genetic Algorithm, Ensemble, Cost Matrix, Cost Based Evaluation, AUPRC, AUROC

1. Introduction

Since the birth of e-commerce in the early 1990s, Internet purchasing and credit card fraud have proliferated across the globe. Moreover, the host of challenges inherent in detecting credit card fraud has made the field one of the most explored [1]. Outlined in [2], individual purchasing behavior evolves over time, as do the methods with which people commit fraud. Credit card transactions are both high in volume and imbalanced [3], with few genuinely fraudulent transactions. High volume also delays transaction feedback, prolonging the time needed to adequately verify and categorize questionable activity [2].

Originally, manual oversight and client feedback were the only defenses against fraud attacks. Today, however, a multitude of rule-based and machine learning algorithms exist that constantly scan for contentious credit card purchases [1, 4, 5, 6]. Despite these preventative measures, a 2016 LexisNexis survey [7] of hundreds of corporations approximated fraud cost to be billions annually. On average, businesses estimated their annual cost of fraud to be 1.47% of total revenues. Furthermore, fraud occurrence is substantially higher for online retailers [7], and increased domination of mobile and e-pay options was the single most cited concern companies had about preventing fraud. Over 48% of online fraudulent transactions were made with a credit card in 2016, far higher than debit card transactions [7] and other methods.

Dozens endeavor to optimize the detection of fraud [8]. While credit card fraud is particularly well researched, it is only one of many documented types spanning multiple industries. Even so, its impact is disproportionately large. Credit card fraud can either take the form of *application* or *behavioral* fraud. Application fraud is the act of procuring a credit card using falsified or stolen information, and can be thought of as a type of identity theft. Behavioral fraud, a much more common occurrence, is the act of purchasing goods with stolen credit card information. While this can occur at Point of Sale (POS) locations, it is more commonly seen in Internet transactions [7]. Many sources of information are leveraged to predict behavioral fraud, but profiling the purchasing activity of individuals provides a functional benchmark from which future transactions can be compared [9]. Temporal and location-based information is also useful in this regard.

However, if purchasing behavior is sporadic or scant, profiling is less useful. Consistently insightful transaction traits are difficult to find, as is procuring data to test different Fraud Detection Systems (FDS). Credit card transactions are a form of Personally Identifiable Information (*PII*), dictating many features must be hashed or encoded. Additionally, banks and retailers alike are extremely secretive about fraud [1] due to its sensitive and unsavory nature. The lack of fraud transparency in the public and private sectors is a consistent point of criticism.

In spite of data scarcity, countless Fraud Detection Systems have been created in the past decade. Generally, these are categorized as either supervised or unsupervised [4]. Supervised machine learning models face a number of challenges, namely an inability to identify new types of fraud as well as a perpetual delay of data in what is known as *verification latency* [2]. Questionable activity is often manually checked, and the result-

ing delay is often ignored by fraud detection studies [2]. Conversely, unsupervised algorithms tend to cluster transactions based on similarity. This enables a FDS to potentially categorize new types of fraud, though human intervention is needed to characterize each cluster [5]. In general, fraud detection has not yet evolved enough to detect fraud independently. Rather, anomaly detector may be a more adequate characterization of these techniques.

Further complicating the matter, fraudulent activity exists as a minute percentage of total credit card transactions, at times less than 0.2% of available data. The predictive power of fraud systems is heavily impacted by the composition of its training data. To combat classification bias, alternative methods re-balance sample composition in favor of minority classification. One of the most promising of these methods is undersampling, which randomly chooses a subsample of the majority class, placing emphasis on the minority class. Conversely, oversampling seeks to accomplish the same goal by sampling with replacement from a minority class up to a certain threshold [10].

Likewise, a novel sampling technique developed by [10] incorporates facets of both undersampling and oversampling. Synthetic Minority Oversampling Technique (SMOTE) can be described as randomly undersampling the majority class as well as generating synthetic minority records. While SMOTE's technique is novel, Blagus et. al. [11] finds it does not outperform undersampling when applied to gene expression data. According to their empirical results, synthetic records beyond a given threshold confound the classification boundary between the majority and minority classes.

Lastly, an ongoing debate examines the ideal method of evaluating fraud detection algorithms. High data imbalance causes accuracy and other traditional metrics to be misleading, as ignoring the minority class entirely could yield an accuracy well over 99%. Many have posed alternative metrics [12], including cost-based indicators and ranking systems.

2. Contribution

This paper has three objectives, all of which pertain to optimizing fraud detection in data-starved environments. The dataset implemented to test these theories is public, anonymized, and has been transformed by Principal Component Analysis [13]. Therefore, longitudinal data is absent and temporal and location-based profiling techniques are not possible. Accordingly, this study compares the efficacy of undersampling and SMOTE in training machine learning models. Since standard sampling is ineffective with imbalanced data [14], it is crucial to understand the predictive potential of alternative methods.

Additionally, by tuning and testing a variety of commonly used machine learning classifiers, this paper explores the correlation between traditional performance metrics (Positive Predictive Value (PPV), True Positive Rate (TPR), F_1 Score) and cost efficiency. Though previously examined by [15] [16] [17], economic cost evaluation is relatively unexplored despite promising initial findings [18]. In the same vein, examining fraud cost in situations with little data is particularly crucial as new forms of fraud may not be documented. Our findings serve as a contemporary exploration of previous research, but with a greater focus on societal cost ease of industrial adoption.

Finally, genetic algorithms train an ensemble classifier comprised of independent algorithms. Preeminent models from previous experiments are combined and assigned weights. Iteratively, a genetic algorithm optimizes these weights within specified threshold bounds. Ultimately, this experiment discerns the feasibility of improving fraud detection through a multi-model voting system. Such a practice could prove useful when little information can be gleaned from transaction history alone.

3. Procuring Publicly Available Credit Card Data

The paper utilizes a dataset comprised of 284,807 credit card transactions occurring over two days. Published by unnamed European bank in September 2013 [13], only 0.172% (492 records) of this data was fraudulent. As one of the few publicly released fraud datasets, the features of these transactions were anonymized (as $V_1, V_2, ...V_n$) and transformed using Principal Component Analysis (PCA) [19] before being posted publicly. PCA de-dimensionalizes data such that the only the most distinguishable traits remain, implementing orthogonal transformation to convert a set of observations with correlated variables into a linearly uncorrelated set. Only the *amount* and *time* variables were not transformed.

As a result, connecting transactions to a specific user is impossible. Even if data was not anonymized, transaction history over a two day window is likely unhelpful with profile based classification. Therefore, the central focus of this paper is to identify optimal sampling methods and model training practices in the absence of situational context or historical data.

4. Sampling Methods

4.1. Simple Random Sampling

Traditionally, machine learning models are trained on a simple random sample of population data. Aside from preprocessing, often all that is necessary to prepare data with popular machine learning software [20] is a test ratio, or the percentage of data to be used for testing classifier performance. However, it is commonly found that [10, 13, 14] simple random sampling is unhelpful with problems of severe data imbalance. In turn, alternative methods that artificially rebalance data are almost exclusively used in industry and academia.

4.2. Undersampling

Liu et. al. [21] describes the process of undersampling as using a "subset of the majority class to train [a] classifier". In this manner, classifiers are more directly influenced by the characteristics of the minority class. In practice, [22] confirms that undersampling outperforms random oversampling, a method that, instead of collecting a subset of the majority, generates a sample of minority records by selecting with replacement. Applications of oversampling tend to suffer from an ill-defined minority classification boundary. With-replacement samples of

the minority class, regardless of size, often cannot generalize trends beyond those present in training data [14]. However, undersampling suffers from its own pitfalls. If the chosen subset of majority class data is not prescriptive enough to identify the majority population, the classifier will be unable to generalize majority class identification. Even so, [21] finds that implementing a bagging ensemble trained with different majority subsets assuages the risks of information omission in training. This innovation, coupled with ancillary techniques, have made undersampling a staple of imbalanced data analysis.

4.3. SMOTE: Synthetic Minority Oversampling Technique

A combination of undersampling and oversampling is known as Synthetic Minority Oversampling. SMOTE endeavors to compensate for undersampling's weaknesses by allowing for larger majority sample sizes while maintaining a specified ratio of minority records. It is also thought to assist in generalizing a minority feature space with few records [10]. To do so, a minority record is randomly chosen, as are its most similar k "neighbors" using euclidean distance or an equivalent similarity measure. A second minority record is then randomly chosen from the k-neighbor subsample. Features X_{1-n} for the synthetic record are then derived as a linear combination of the original record α_i and its difference with selected neighbor α_k, multiplied by a random scalar c chosen from a uniform distribution.

$$X_i = \alpha_i + c(\alpha_i - \alpha_k) \qquad i = 1, 2, \ldots n$$
$$c \sim U(0, 1)$$

Chawla (2002) [10] provides evidence that supports the validity of SMOTE's aforementioned merits. Asserting that synthetically generating records prevents "overfitting on the multiple copies of minority class examples", Chawla (2002) maintains SMOTE causes the "decision region of the minority class to become more general." At the same time, increasing the number of minority records through synthetic creation allows for a greater number of majority records to be included in the training data while maintaining the desired minority-majority ratio. Ramezankhani et. al. (2016) [23] also found boosted performance with SMOTE in lipid and glucose experimentation.

5. Calculating Model Performance with Data Imbalance

As traditional sampling techniques falter with imbalanced optimization problems [11], so do traditional evaluation strategies. Other than accuracy measures, a common practice in binary classification problems is to define performance in terms of *true positive*, *false positive*, *true negative*, and *false negative* rates. As seen in the *confusion matrix* below, true positives and true negatives are correctly classified fraudulent and non-fraudulent records, respectively. By contrast, a false negative occurs when fraud goes undetected and a false positive characterizes a false alarm, when a transaction is flagged but not fraudulent. Segmenting classifier performance into these four groups affords researchers the freedom to weight different outcomes independently [2].

		Actual Transaction	
		Fraud	Not Fraud
Prediction	Fraud	True Positive (TP)	False Positive (FP)
	Not Fraud	False Negative (FN)	True Negative (TN)

Table 1: Confusion matrix for credit card fraud detection

Subsequently, many assessment metrics are derived from table 1. Listed in equations 1, 2, 3, and 4 are the eight basic indicators derived from confusion matrices: true and false positive and negative rates (TPR, FPR, TNR, FNR), positive and negative predictive value (PPV, NPV), false discovery rate (FDR), and false omission rate (FOR). Visually, all equations in the left column focus on successful classification, while the right column focuses on failure rates.

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN} \tag{1}$$

$$TNR = \frac{TN}{TN + FP} \qquad FNR = \frac{FN}{FN + TP} \tag{2}$$

$$PPV = \frac{TP}{TP + FP} \qquad FDR = \frac{FP}{TP + FP} \tag{3}$$

$$NPV = \frac{TN}{TN + FN} \qquad FOR = \frac{FN}{TN + FN} \tag{4}$$

Conceptually, TPR and TNR determine how many fraud records were appropriately classified. Both of these metrics are direct evaluations of model performance by class, and ask the question: *how many fraud records were classified correctly?* Conversely, PPV and NPV can be described as a *prediction centric* evaluation of the confusion matrix, conditionally examining results given a specific prediction. That is, *for all records predicted to be fraudulent, how many actually were?*

One way these indicators can discern classifier performance is with the Area Under the Reciever Operating Characteristic (AUROC) [24]. Implemented to increase emphasis on detecting the minority class, AUROC maps an algorithm's TPR and FPR classification ability at different thresholds. However, as noted in [25], some confusion matrix measures are biased for imbalanced datasets due to a disproportionally large number of true negative records. False Positive Rates (FPR) are held artificially low by this imbalance, over-estimating fraud detection efficacy.

To garner a more realistic picture of performance in an imbalanced setting, researchers derived the Area Under the Precision Recall (AUPRC) curve. Subtly distinct from AUROC,

AUPRC maps TPR and PPV, also known as recall and precision. AUPRC is thought to assuage issues of FPR bias by considering the *success rate* of minority classification rather than its failure rate, correcting biases caused by the majority class. Indeed, Davis & Goadrich (2006) [26] assert that a "curve dominates in ROC space if and only if it dominates in PR space." That is, AUPRC measurements embody a more robust method of testing for algorithmic optimality.

Nevertheless, some still take issue with AUPRC as the definitive performance indicator as the ultimate pursuit of fraud detection is to minimize the cost to society [16]. As noted by [7], this cost takes different forms. Implications beyond the de-facto cost of the fraudulent transaction include the time-loss on behalf of the bank or retailer in identifying the issue, loss of trust (and potentially business) of clients and customers, and loss of productivity on behalf of the victim who may need to apply for a new card. The fraud multiplier of $2.40 [7] only covers cost incurred by the company, excluding cost to the individual victim or tertiary party involved. Thus, [15], [18], and others propose a derived cost function.

		Actual Transaction Cost	
		Fraud	Not Fraud
Prediction Cost	Fraud	$-T_c + C_f$	C_e
	Not Fraud	$F_m(T_c) + C_l$	0

Table 2: Cost matrix for evaluating credit card fraud detection algorithms.

Cost matrices allow the user to dynamically weight the importance of different outcomes. Table 2 represents a unique cost matrix similar to those proposed by [15] and [18]. Identifying and preventing a fraudulent transaction equates to saving the cost of the transaction $-T_c$ as well as the time cost C_f of resolving the event. Similarly, false alarms (FP) cost the amount of time needed to correct the error, C_e. Most importantly, failing to detect fraud increases the cost of the transaction by the fraud multiplier F_m, plus the cost of addressing the loss, C_l.

The sensibility of cost-based evaluation has been questioned since its inception despite its practicality [15]. Simply put by [27], "the cost of a fraud is not easy to define." Indeed, contemporary Fraud Detection Systems have multiple layers. Rule-based credential authorization offers a first layer of defense, followed by algorithmic classification and terminating at human oversight. Derivation of company-specific cost models is hampered by this complexity. Additionally, [2] asserts that cost-based metrics ignore the disruptive nature of fraud alerts and company costs. However, such an argument also applies to many traditional metrics, including AUPRC. Debate continues and traditional metrics are still implemented as performance indicators [1].

6. Evolution of Fraud Detection Systems

In the past two decades, roughly a dozen articles summarize hundreds of experiments conducted on different types of fraud. Many of these dedicate large portions of their articles to credit card fraud [9, 4, 8, 1]. Of note, [8] identify outlier detection via supervised learning to be the most common, though it is implemented in many different forms.

In particular, neural networks are included in a plurality of Fraud Detection Systems. In fact, neural network applications have been utilized since the proliferation of e-commerce. Compensating for data imbalance with Fisher's linear discriminant analysis, [28] created a neural network classifier that stood as a cornerstone achievement alongside CARDWATCH [6]. Today, some networks make use of unsupervised clustering models to identify unprecedented fraud methods. Even with this advantage, *verification latency* stands as a major drawback for all supervised models, particularly neural networks that require significant time to train.

The additional challenge of *concept drift* and *verification latency*, along with the increasing cost of fraud, have led to the creation of innovative new techniques spanning multiple disciplines. Graph-based anomaly detection and inductive logic are only a few new implementations [1]. Multi-faceted ranking, searching, and scoring ensemble methods have also spread in the past decade [27], often coupled with cost-based evaluation and Genetic Algorithms [29].

Unfortunately, systems derived in academia do not consistently transfer to a corporate setting. While contemporary models continue to increase in complexity, Phua et. al. (2010) [1] laments that many fraud-plagued industries are "dependent on the practical issues of operational requirements, resource constraints, and management commitment towards reduction of fraud." Such constraints are rarely, if ever, imposed on models in a research setting. Despite the fact that experiments on computationally efficient algorithms [30] are continually conducted and published, their industrial practicality is seldom emphasized. In turn, the following experiment will only make use of relatively lightweight, scalable classification methods.

7. Research Methods

7.1. Automating Fraud Detection Experiments

To examine a variety of samples and models, as well as train an ensemble with genetic algorithm (GA) optimization, a modular testing platform was created. Encapsulated in $Test$ objects, a specified sample and model are run repeatedly via Monte Carlo simulations, as shown in Figure 1. Results are stored after each testing cycle in a *Logger* object as a result log before the sample is re-created and the process repeats. If an ensemble is specified, performance will be stored in the same location. At the end of the execution, a master log of all execution parameters, average results, and meta-data is sent to the *Logger*.

Sampling: All executions begin with a command sent to the sampling engine that randomly selects 20% of the entire dataset to be used for sample creation. The remaining records are set aside for testing. Fed approximately 100 and 59,900

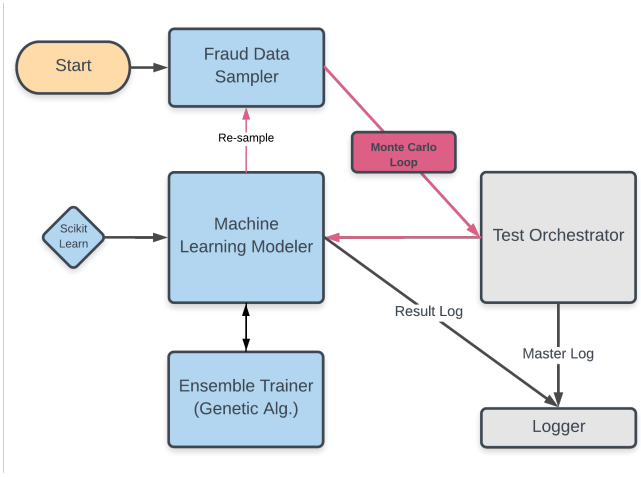

Figure 1: Scalable automation platform for testing fraud detection algorithms

fraud and non-fraud records, respectively, the sample engine then launches an orchestration package dependent on sample method, sample size, and fraud ratio parameters. Undersamping samples ignore the desired sample size, as its maximum size is constrained by the desired ratio of fraud records. Conversely, SMOTE samples includes all fraudulent records, and then synthetically adds to the minority subsample until the specified fraud threshold is reached. Any records not used in the final sample are added to the testing dataset. Due to PCA transformations of the source data, no preprocessing is considered aside from sample generation.

Modeling: Provided a training sample, the machine learning modeler enters a Monte Carlo loop with a specified number of iterations. Implemented with Sci-kit Learn, the modeler tests Logistic Regression, Linear Support Vector, Random Forest, K-Nearest Neighbors, and Gaussian Naive Baye's classifiers. KNN serves as the control due to its notoriously poor performance with imbalanced datasets [31]. Each model is trained and tested on the same sample, with performance metrics logged ad-hoc. After the execution terminates, the modeler re-instantiates the sample object, which re-partitions the entire dataset and creates a new sample. Once the Monte Carlo loop completes, performance records for each model are averaged and stored in the master log.

Ensemble: If little information can be gleaned from a dataset, then coordinating multiple predictions may yield a better result. Just as a governing body operates on popular vote, so too does the fraud detection ensemble. By assigning weights to different fraud predictors and then optimizing them with genetic algorithms, ensemble performance may exceed that of its components.

To test this theory, the modeler engine is provided an ensemble evolution object. If desired, this object is instantiated and called as a part of the Monte-Carlo cycle. Feedback during evolution is provided by partitioning the original sample into a training and testing subsets. All individual models are trained with the training subsample, and their individual predictions are recorded.

These individual predictions are then assigned random weights, and genetic evolution commences. A random population of weights is generated, and those with the highest fitness (lowest cost) receive a increased probability of being chosen to populate the following generation. New generations of weights inherit traits from their parents through bit string crossover. Fitness is then calculated for the new offspring, and the process continues until the execution times out. After evolution terminates, individual models generate predictions for the testing sample. Optimized weights are applied to these transactions, giving a weighted average probability of fraud.

7.2. Cost-Based and Statistical Evaluation Practices

During executions, TP, FN, TN, and FP rates are stored for each model, from which many additional metrics (TPR, FNR, PPV, NPV) are derived. Furthermore, general *Precision* and *Recall* metrics shown in equation 5 provide holistic measure of proficiency by class (*Recall*) as well as by prediction (*Precision*). These are composite competency metrics, not to be confused with PPV and TPR, sometimes referred to as precision and recall. Contextually, *Precision* here is defined as the mean prediction precision: what percentage of each prediction type were correct? Similarly, *Recall* discerns the mean percentage of each class that were categorized correctly.

$$Precision = \frac{PPV + NPV}{2} \qquad Recall = \frac{TPR + TNR}{2} \tag{5}$$

Likewise, the F_1 Score (also known as the $F - Measure$) encapsulates the holistic predictive power of an algorithm with a focus on the minority class. Contemporary researchers including [13] regard the F_1 Score to be a suitable measure of predictability, even with imbalanced datasets. Much like AUPRC, the F_1 Score combines PPV and TPR to avoid bias introduced by disproportionately large TN counts.

$$F_1 = 2\left(\frac{PPV * TPR}{PPV + TPR}\right) = \frac{2TP}{2TP + FP + FN} \tag{6}$$

Lastly, cost scores are generated using the framework outlined in Table 2. Since no company data is available for deriving explicit cost metrics, the cost of correct fraud detection C_f is set at \$10, while fraud error correction is set at C_e. The cost of not discovering fraud C_l is also set at \$10. The fraud multiplier is set at \$2.40, cited from the 2016 LexisNexis Survey. The total cost of fraud F_c for this study can be calculated as

$$F_c = (TP + FN)C_{fl} + (FP)C_e + \sum_{i=1}^{n} F_m(T_{ic}|T_{i=FN}) - (T_{ic}|T_{i=TP})$$

where $C_{fl} = C_l = C_f$, TP is the sum of true positives, FP is the total of false positives, and T_i represents the i^{th} transaction in a testing set of size n. These variables are not based on empirical data in this study, and addressing a false positive is assumed to be less costly compared to addressing a genuinely fraudulent

transaction. More importantly, this study is merely outlining a process by which companies could improve classification scoring. We recommend tailoring cost matrices to meet specific business needs.

7.3. Implementing Practically Feasible Classifiers

Acknowledging the challenges cited in [1] of applying academic models to industry, we selected classifiers based on a combination of implementation practicality and past success detecting fraud. Higher-order profiling is impossible in this dataset, and its size inhibits the presence of concept drift. Therefore, these models strike a balance between time and spatial complexity as well as predictive accuracy.

Logistic Regression (LOG): One of the most commonly implemented binary classification analyses [32], Logistic Regression is a discriminant Bayesian model that approaches binary classification through direct calculation of Bayesian posterior $P(y|x)$ of the joint probability distribution $P(x, y)$ [33]. Logistic regressions assume little correlation (if any) exists between predictors in the feature space, as well as that few outliers and minimal skew are present. While characteristics of the provided data are not available in raw form, it can be assumed that PCA transformation approximately satisfies these requirements [19]. Prevailing research also finds discriminative calculation – even without computational or data quality considerations – to be generally preferred [33]. The empirical findings of this model will be compared with Gaussian Naive Bayes predictors to test this hypothesis.

Linear Support Vector Classifier (SVC): Vapnik (1995) [34] first proposed Support Vector Machines (SVM) for pattern recognition by categorizing the problem as fitting an optimal separating hyperplane in $\mathbb{R}^n$ feature space, where n is the number of features. By treating observations as *support vectors*, Vapnik characterizes the classifier as a Lagrangian optimization of the form:

$$y(x) = sign\left[\sum_{i=1}^{N} \alpha_k y_k \Psi(x, x_k) + b\right] \tag{7}$$

"for a training set of N data points $\{y_k, x_k\}_{k=1}^{N}$ where $x^k \in \mathbb{R}^n$ is the k^{th} input pattern and $y_k \in \mathbb{R}$ is the k^{th} output pattern." [35]. α_k are positive, real constants, as is b. Notably, SVM makes use of kernel function Ψ, which transforms data to be optimal for hyperplane separation. Since Vapnik discovered Support Vector Machine's efficacy at pattern recognition, variant classifiers have become common due to the flexibility afforded by kernel functions. For example, Suykens (1999) [35] determined a popular method for implementing SVM with least squares optimization. Kernel research continues to derive new applications for different disciplines.

However, traditional Support Vector Machines are computationally expensive, even for Linear SVMs that utilize a kernel of the form $\Psi(x, x_k) = x_k^t x$. To assuage resource constraints for classifying large datasets, [36] developed a method for optimizing Linear Support Vector Classifiers without a kernel function. Instead, $L1$ and $L2$ regularization functions were employed.

$$L1 = argmin_w \sum_{i=1}^{n} \left[y_i - \sum_{j=0}^{m} w_j x_{ij}\right]^2 + \lambda \sum_{j=0}^{m} |w_j| \tag{8}$$

$$L2 = argmin_w \sum_{i=1}^{n} \left[y_i - \sum_{j=0}^{m} w_j x_{ij}\right]^2 + \lambda \sum_{j=0}^{m} w_j^2 \tag{9}$$

In machine learning, $L1$ loss is also known as Least Absolute Deviations (LAD) and included as part of regularization functions that classifiers seek to minimize. $L2$ is similar to $L1$ loss with the exception that it seeks to minimize the Least Squares Error (LSE). Both of these regularization functions are categorized for weights w, output label y, and prediction x in equations 8 and 9. C, another parameter tuned for Logistic Regression and Linear SVC, is a term for inverse regularization strength $\left(\frac{1}{\lambda}\right)$. This characterizes how harshly a model's complexity should be penalized during training.

Random Forests (RF): Breiman (2001) [37] defines Random Forests as "a combination of tree predictors such that each tree depends on the values of a random vector." An ensemble algorithm, Random Forest classifiers derive their efficiency from a sufficiently large number of decision trees. Each decision tree in a group may suffer from high generalization error and overfitting. But, if taken together as a random forest voting system, the ensemble has been proven to produce a "limiting ... generalization error" [37].

When applied as a fraud detection algorithm, Random Forests have also been deemed empirically robust against the challenges of data imbalance [30]. Moreover, Random Forest classification of credit card fraud was found by [38] to be a sufficient means of classification, even with incomplete or imbalanced data. Random Forest's ensemble may be stochastically robust against the lack of information present in data employed by this study.

Gaussian Naive Bayes (GNB): As opposed to the discriminant Bayesian implementation employed by Logistic Regression [33], Gaussian Naive Bayes classification generates the joint probability distribution $P(x, y)$ through learning and then uses Bayes Theorem to calculate $P(y|x)$. Favored for its simple and efficienct implementation, Bayesian models consistently perform well across a variety of applications [4]. The high transaction volume for credit cards favors Bayesian models for their fast prediction generation [1], and many variant techniques like Bayes Minimum Risk (BMR) and Bayesian Neural Networks [39] have found success in credit card fraud [25]. Outlined by [40], Gaussian Naive Bayesian models assume features are independent and model a normal distribution.

$$f(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{10}$$

By using an empirically discovered μ and σ for each feature, a Gaussian probability distribution can be derived, as shown in

equation 10. This information is then be fed into record classification in equation 11, where $P(A|B)$ is equal to the product of the probabilities of each feature B_i belonging to the normal distribution of that feature $N(\mu_{B_i}, \sigma_{B_i})$ derived from training data.

$$P(A|B) = P(A)\prod_{i=1}^{n} P(B_i|A) \tag{11}$$

K-Nearest Neighbors (KNN): The control classifier, K-Nearest Neighbors (KNN) is a rudimentary machine learning classifier. Though highly dependent on the dataset and application, KNN can be an effective means of prediction. A test prediction is generated by finding the K most similar records in the training data, and returning the most common class label in the set. Similarity d of a training record x relative to test record y is often determined using Euclidean distance (12) function defined for a dataset with n features as

$$d(x) = \sum_{i=1}^{n} \sqrt{(x_i - y_i)^2} \tag{12}$$

where x_i is the i^{th}feature of the training record. KNN has been shown to be an effective algorithm for some classification problems, but it is commonly known [38] to struggle with imbalanced data. Traditionally, an increase in the number of neighbors would correspond to a more general consideration of class characteristics and a better prediction. Imbalanced data prevents this improvement. As the number of neighbors increases, minority class representation in the K-subsample decreases, reducing algorithm performance.

Ensemble Classification and Genetic Algorithms Ensemble methods have found significant success across many fields of applied machine learning [30]. The ensemble implemented in this study takes the highest performing individual classifiers and weights them to produce a classification in the form of a weighted average. Ensemble studies in outlier detection and other disciplines [41] have determined genetic algorithms to be a suitable method for finding optimal parameters in high dimensional feature space.

The algorithm begins with an array of random weights, one for each algorithm. These weights are chosen from the uniform distribution $U \sim [1, 2^{40})$ so integer weights are defined by 40 bits or fewer. This allows the genetic algorithm to treat the binary representation of these numbers as "genes" and vary them throughout the evolution process. For each generation, 50 arrays are generated, each with a probability of succession $P(s)$ determined by its fitness f. To determine this probability, fitness measurements (fraud costs) are first converted to positive values f_p by the transformation in equation 13

$$f_{p_i} = f_i + f_{min} + 1 \tag{13}$$

where f_{pi} is the positive fitness of the i^{th} weight array and f_{min} is the lowest fitness in the generation. It is important to note that since cost is used as a fitness metric, low fitness scores are desired. Therefore, f_p values are inverted before probabilities are taken. This calculation takes the form of equation 14

$$P(x) = \frac{f_{p_{max}} + 1 - f_{p_i}}{\sum_{i=1}^{n}(f_{p_{max}} + 1 - f_{p_i})} \tag{14}$$

where $f_{p_{max}}$ is the maximum (worst) fitness score. In turn, all of the weights in the array sum to one. With these probability scores, the following generations are populated by repeatedly choosing two weight arrays from the existing population and generating a child array with bit-manipulation crossover. Bit flipping is also used to randomly mutate arrays in a population, though this is set to occur rarely. To prevent the ensemble from over-weighting an individual model, a ceiling weight w_{max} is set at 0.49. Each ensemble optimization via genetic algorithm is given a specific runtime of 1 minute.

8. Empirical Findings

8.1. Undersampling generates superior model performance relative to SMOTE

Sample size and fraud ratio combinations are listed in Table 3, with the best sample parameters for each method noted as well. Since undersampling did not depend on a set sample size, we were able to computationally search the entire sample-model feature space. That is, for each fraud ratio derived undersample, every model was tuned and tested for all parameter combinations listed in Table 4.

Table 3: Tested Sampling Combinations

Sample Method	Sample Size (*Thous.*)	Fraud Ratio	Best Sample (*size*, *ratio*)
SMOTE	1, 2, 3, 5, 10	0.1, 0.2, ... 0.5	(1, 0.5)
Under	-	0.1, 0.2, ... 0.5	(- , 0.3)

By contrast, SMOTE's dependency on both a specific sample size and fraud ratio made it computationally infeasible to search the entire model-sample feature space outlined in Table 4. Therefore, a bootstrapping method was implemented to indirectly converge on the optimal sample size and fraud ratio. First, all samples were tested with default parameters assigned to each model. Default parameters are provided by Scikit Learn, and are listed in Table 5. After testing all SMOTE sample sizes and fraud ratio combinations, the sample with the lowest average cost across all models is discerned and used to test all parameters. With sampling parameters held constant, optimal parameters are identified independently for each algorithm.

Classifier parameters are then fed back in to the model and held constant as sample-ratio combinations are tested again. This process continues until empirical results converge on a single sample-ratio combination. Random model parameters and sample-ratio combinations were also tested to independently verify a local optimum. If independent findings were inconsistent with the current results, bootstrapping continued using the random parameters aforementioned. All iterations ran 100 Monte Carlo simulations for every unique sample-ratio-model combination.

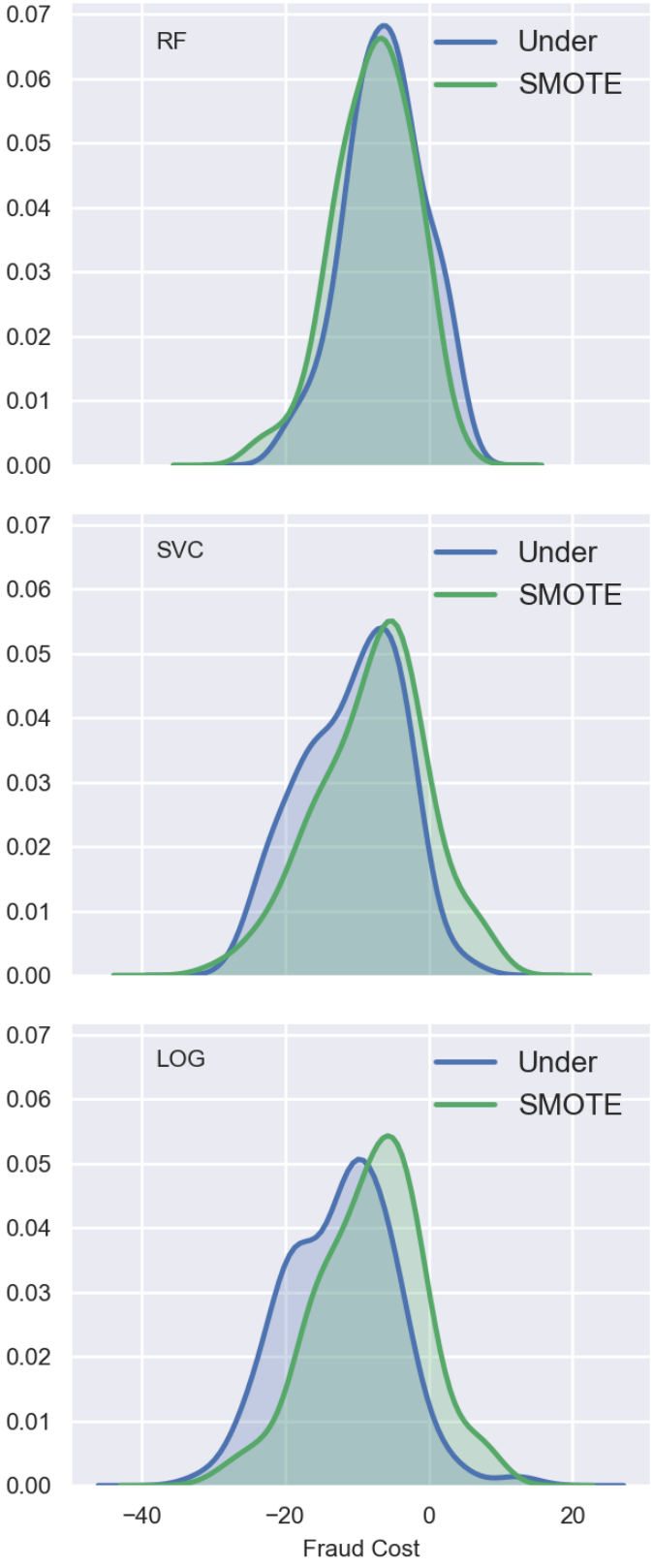

Figure 2: Comparison of SMOTE and Undersampling methods as they pertain to training RF, SVC, LOG classifiers to minimize fraud cost. All figures are cumulative frequency histograms. Fraud cost given in USD - Thousands.

Table 4: Model Tuning Tested Parameters

	Parameters		
Model Name	Penalty	C	Trees (RF) or K (KNN)
LOG	l1, l2	0.5, 1, 5, 10, 20	
SVC	l1, l2	0.5, 1, 5, 10, 20	
RF			10, 20 ... 100
KNN			10, 20 ... 100

While searching for the ideal SMOTE sample, models with outlier performance were detected and dropped. For a classifier to be dropped from the remainder of the study, its fraud cost had to be significantly higher than the average of all models in both undersampling and SMOTE samples.

As noted in Tables 5 and 6, the fraud cost of the K-Nearest Neighbors algorithm was orders of magnitude higher than other models, regardless of parameter tuning. Therefore, its performance was not considered during the bootstrapping process. Gaussian Naive Bayes, though lower in cost than KNN, was also dropped for the same reason (see Figure 3). Additionally, Logistic Regression and Linear SVC Classifiers optimized with $L2$ regularization functions were dropped as the difference in performance between $L1$ and $L2$ regularized functions was significantly large (see Figures 6 and 7).

In general, a SMOTE samples fraud ratio was the most indicative of its ability to train highly efficient, low cost algorithms. Though fraud costs varied by iteration, most algorithms experienced improved performance as the fraud ratio increased up to 50%. After a threshold of 30%, fraud cost differences in many models were not significant enough to be distinguished from variance in performance. In the same vein, many models (Table 6) depicted a slight positive relationship between rising cost and rising sample size.

Undersampling did not possess a consistent trend between fraud ratio and model performance. Gaussian Naive Bayes and Random Forest Classifiers continuously improved as the fraud ratio increased. Conversely, regression-based models Linear SVC and Logistic classification possessed a quadratic relationship with a fraud ratio that peaked at 30%, decreasing thereafter. Logistic Regression and Linear SVC also had higher average performance for a fraud ratio of 0.2, but this was overshadowed by Random Forest's increase in performance as it moved from a fraud ratio of 0.2 to 0.3. Repeatedly sampling verifies all of these trends with relative certainty despite the stochastic nature of sampling methods used in this analysis. Thus, the optimal fraud ratio was set a 0.3 to embody the best average performance for the three best classifiers. Using the best holistic sample, model parameters were tuned to minimize the cost of fraud.

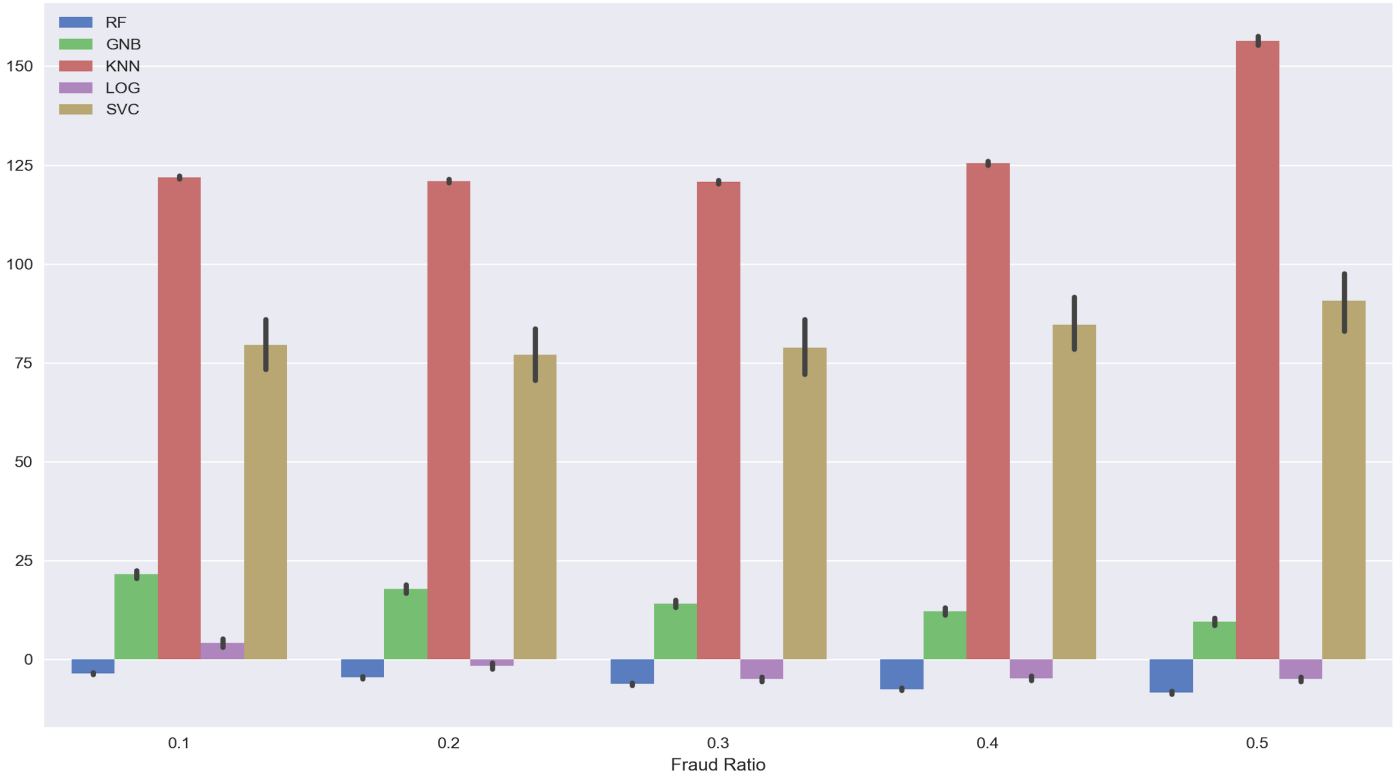

Figure 3: Mean performance of all FDSs, all parameters. The y-axis represents fraud cost in USD - Thousands, and black lines signify sample σ.

8.2. *Optimal model parameters consistent across sample types*

Optimal size and fraud ratios for SMOTE and undersampling were held constant as different parameters for Fraud Detection Systems were tested. With the exception of Random Forest Classifiers, no algorithm proved to be robust across all of its parameter combinations.

Table 5: Parameter Legend

	SMOTE		Under
Model (*Params.*)	Default Param.	Best. Param.	Best. Param.
LOG (*Pen. C*)	l2, 1	l1, 0.5	l1, 0.5
SVC (*Pen. C*)	l2, 1	l1, 0.5	l1, 0.5
RF (# *Trees*)	10	80	80
KNN (K)	5	10	10

K-Nearest Neighbors achieved its highest predictive power with a K of 10. Further increases in the number of neighbors gradually decreased the F_1 Score and increased the fraud cost of the algorithm. While the root cause of this trend is unknown, it is possible that the minority-majority class boundary in the training data is not well defined. KNN's predictive accuracy is dependent on a well-defined boundary between majority and minority data. Credit card fraud detection can be categorized as an outlier detection problem, and therefore may not have enough minority class information to generalize a boundary in the training data. Concept drift exacerbates this issue, potentially causing minority records in training data to not be representative of the testing sample. Thus, KNN was dropped as a suitable fraud detection system.

Gaussian Naive Bayes does not have parameters with which it may be tuned. Consistently, the fraud cost of GNB decreases as the fraud ratio of the sample data increases. Even so, data outlined in Table 6 denotes GNB was unable to consistently generate an average fraud cost below $0 for undersamples. With no additional tuning possible, the GNB Fraud System is impossible to improve. Therefore, it was also dropped from the analysis.

Random Forest classifiers performed best in situations where the minority class was represented most in sample. Cost savings for these Fraud Detection Systems was the lowest for samples with a fraud ratio of 50% across both undersampling and SMOTE. Also, models optimized best with the smallest SMOTE samples with 1000 records, holding fraud ratios constant. Perhaps decision trees built primarily on synthetic records are over-fitting to the training dataset and unable to generalize rules for identifying fraud. Synthetic records do not greatly deviate from the originals from which they were derived and may not allow classification boundaries to generalize.

Table 6: Under Sample Tuning Summary: Best Parameters

	Model Performance									
Fraud Ratio	LOG F_1 (%)	LOG Cost ($)	SVC F_1 (%)	SVC Cost ($)	RF F_1 (%)	RF Cost ($)	GNB F_1 (%)	GNB Cost ($)	KNN F_1 (%)	KNN Cost ($)
0.1	33.50	−4992	36.95	−4392	53.80	−3898	17.08	21,513	2.01	116,711
0.2	18.28	−9134	17.05	−8916	32.24	−5214	15.78	17,897	1.28	111,311
0.3	11.60	−12,318	9.77	−10,744	19.43	−6365	14.41	14,082	0.87	113,065
0.4	7.62	−11,011	6.00	−9520	12.49	−7969	13.56	12,159	0.61	121,524
0.5	4.94	−8888	3.95	−5942	8.93	−8894	12.37	9508	0.46	135,647

Table 7: SMOTE Sample Tuning Summary: Best Parameters

		Model Performance					
Sample Size	Fraud Ratio	LOG F_1 (%)	LOG Cost ($)	SVC F_1 (%)	SVC Cost ($)	RF F_1 (%)	RF Cost ($)
1,000	0.2	21.40	−8416	21.59	−6931	40.94	−4738
	0.3	15.88	−9286	15.43	−9289	29.58	−4663
	0.4	11.80	−10,685	11.78	−9499	25.05	−7329
	0.5	8.49	−10,012	8.22	−9681	18.76	−8290
2,000	0.1	38.82	−5571	44.05	−4587	50.80	−4024
	0.2	23.65	−8671	25.19	−6925	44.39	−5898
	0.3	18.83	−7854	20.09	−7249	38.30	−5492
	0.4	12.57	−7979	12.94	−7471	32.56	−5706
	0.5	9.32	−9331	9.61	−9157	26.83	−7708
3,000	0.1	39.65	−4672	46.36	−3463	52.71	−4410
	0.2	25.48	−6647	26.97	−5555	49.19	−4894
	0.3	18.88	−8997	20.10	−7866	42.20	−5726
	0.4	15.12	−8690	15.68	−7935	39.49	−5036
	0.5	10.59	−8499	10.77	−7333	32.85	−6091
5,000	0.1	43.43	−3039	51.36	−2358	56.17	−4218
	0.2	27.79	−6011	29.79	−4831	52.83	−3749
	0.3	20.09	−6956	21.71	−6391	50.00	−4366
	0.4	15.05	−7907	15.91	−7233	46.10	−4805
	0.5	11.26	−8008	11.71	−7098	42.54	−4671
10,000	0.1	43.25	−4299	51.76	−3172	58.21	−4225
	0.2	28.53	−7590	31.20	−6424	56.17	−5368
	0.3	21.56	−7575	23.30	−7080	55.15	−4191
	0.4	16.22	−8727	17.25	−8035	52.81	−4147
	0.5	12.97	−9276	13.43	−8236	50.71	−4740

Regardless, Random Forest Classifiers trained on both samples performed optimally with the maximum of 100 decision trees. Larger tree counts may have performed even better, but experimentation was practically limited by sample size constraints. An undersample with a 50% fraud ratio may only contain 160 records.

Outlined in Figure 4, Random Forest detection of fraud is particularly robust. Average fraud cost starkly shifts as models went from 10 to 20 trees, as did fraud cost variation. Cost continued to drop as more trees were added, but only slightly compared to initial improvements. Near 90 and 100 neighbors, changes in mean performance are too minute to differentiate from sample variation. However, it appears that the distribution skews left as more trees are added. As expected, Random Forest classification was one of the most robust models to concept drift, data imbalance, and even its own meta-parameters.

Logistic Regression was the most efficient model at minimizing the cost of fraud. Each classifier tested was tuned by two parameters: a regularization function (*penalty*) and the strength term C. A scalar coefficient that calibrates regularization strength (see Table 4), fraud detection models worked best with a low C value, or a high penalty λ for regularization. Contextually, a high λ value may be necessary to create a functional FDS. Little fraud data is available, leading many alternative samples to be small. Low regularization penalties may quickly lead to over-fitting and generalization error.

Cost-based performance with Linear SVC algorithms also improves as regularization strength λ increases. Fraud cost trends witnessed during model tuning for Logistic Regression are directly paralleled by Linear SVC. As noted in the next section, both were heavily influenced by different regularization functions.

8.3. Regularization central to LOG and SVC performance

Articulated by equations 8 and 9, two regularization functions were utilized to fit both Logistic and Linear SVC classifiers to fraud data. While implementations are similar, resulting fraud costs for each type were quite different. These differences are visualized with the illustrations in Figure 6 and 7 for Linear SVC and Logistic Regression, respectively.

Mapped on a feature space of F_1 performance and fraud cost, a performance distribution. The darkness of each section correlates to the density of observations found therein. Also note that each subplot in Figures 6 and 7 has a unique $y - range$, though all share the same $x - axis$. The purpose of this is to compare the F_1 scores while simultaneously underscoring the stark differences in fraud cost between $L1$ and $L2$ optimized Fraud Detection Systems.

Figures 3 and 5 for articulate the stark impact regularization had on performance for Linear SVC classification. Across all regularization strength parameters, Linear SVC classifiers optimized with Least Square Error regularization had an average fraud cost of $169,694 compared to a cost of -$5196 for Least Absolute Deviation, a difference of $174,890 over two days. Annually, the difference in cost savings for these classifiers would amount to millions, highlighting the central importance of model tuning.

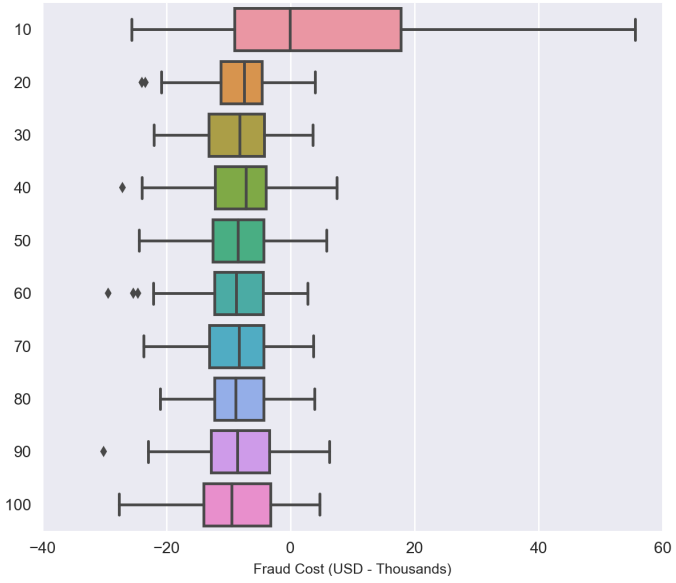

Figure 4: Fraud cost of RF classification using different numbers of decision trees ($y - axis$) in ensemble. Fraud cost represented in USD - Thousands.

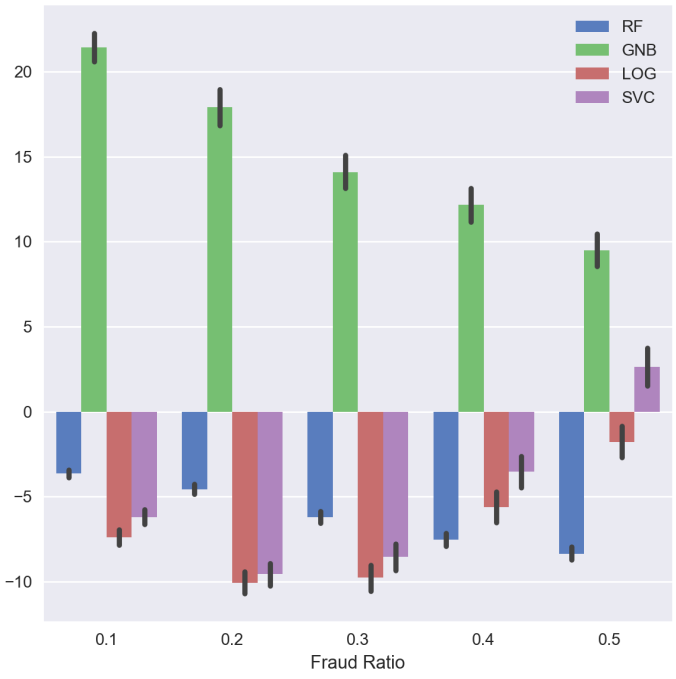

Figure 5: Model performance without outliers. LOG and SVC models trained with $L2$ regularization have been dropped. The y-axis represents fraud cost in USD - Thousands, and black lines signify sample σ.

Figure 6 provides further detail on Linear SVC cost discrepancies. The consistency of performance for $L1$ models is depicted by the unimodal distribution around a cost of -$10,000 and a F_1 Score of 0.1. Conversely, the same models tuned with $L2$ loss possess a polymodal distribution centered on an F_1 Score of approximately 0.03. The $L2$ trained performance distribution appears to achieve costs in excess of $200,000 fairly consistently.

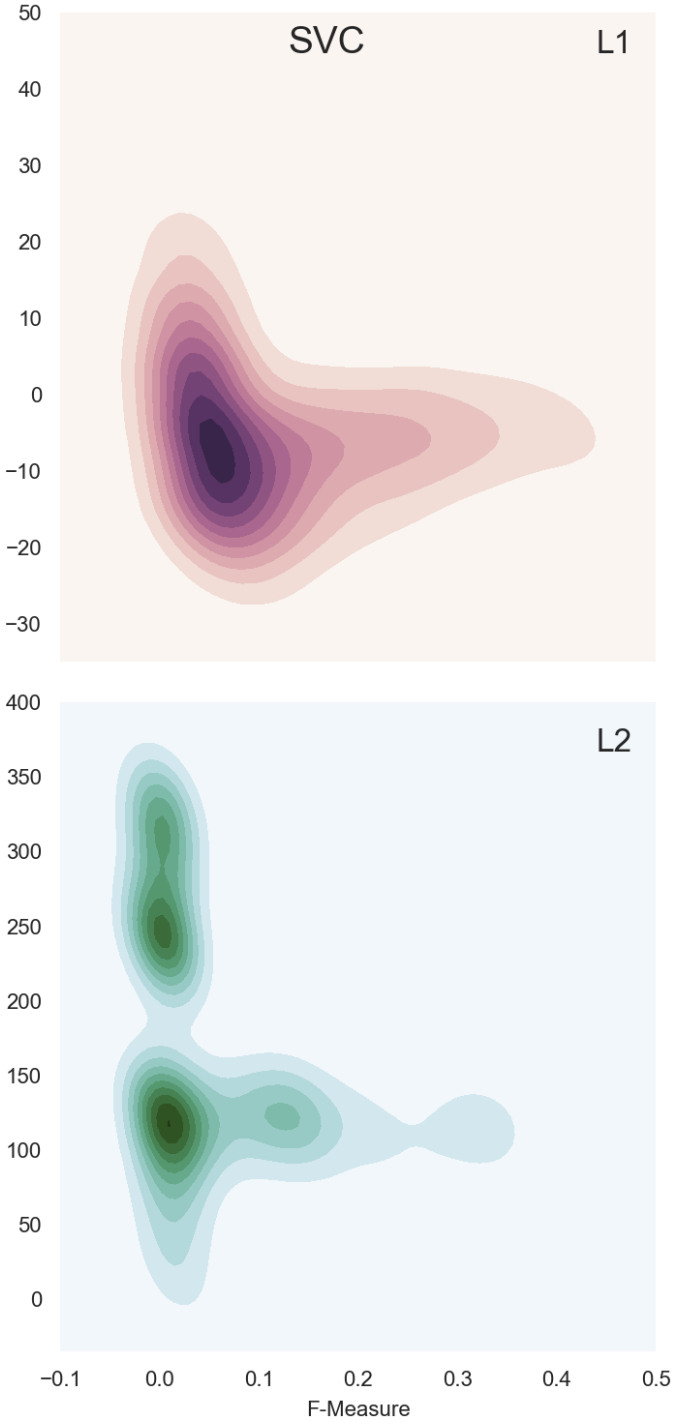

Figure 6: Linear SVC comparison of L1 (LAD) and L2 (LSE) regularization functions. The $y-axis$ represents fraud cost in USD - thousands.

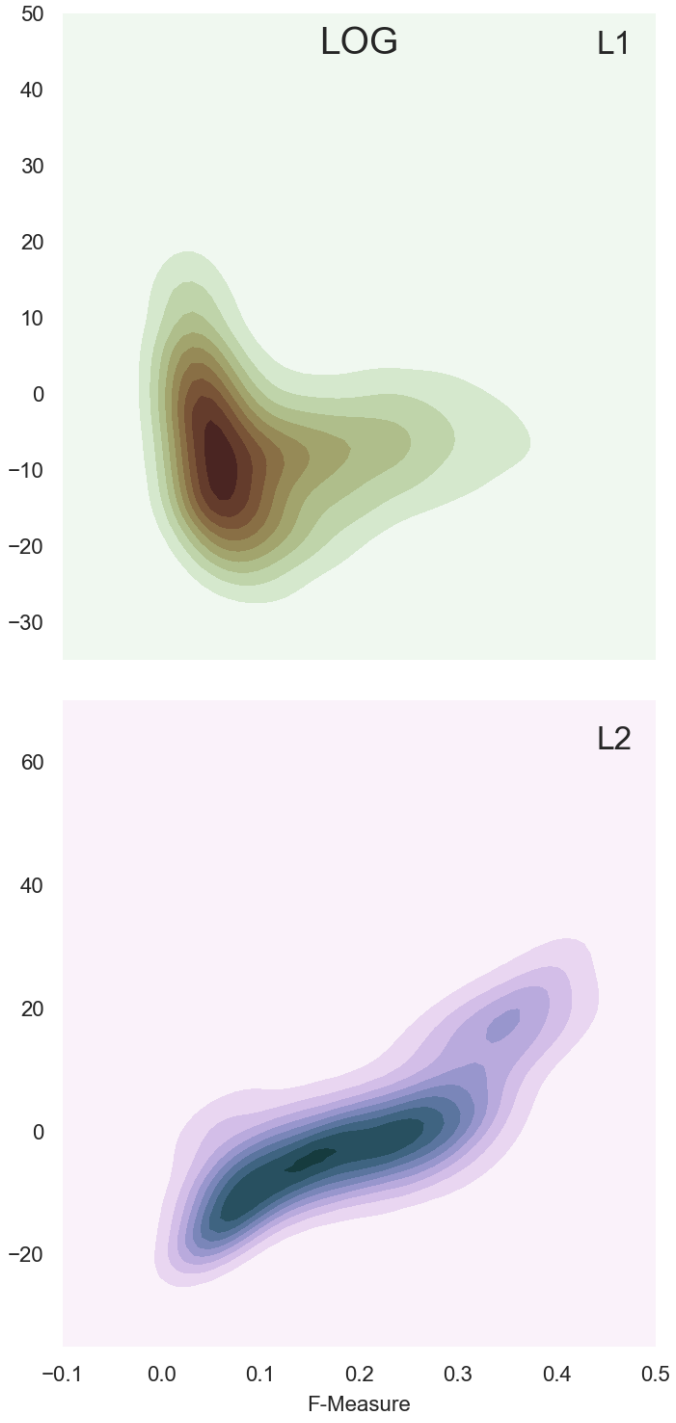

Figure 7: Logistic Regression comparison of L1 (LAD) and L2 (LSE) regularization functions. The $y-axis$ represents fraud cost in USD - thousands.

Logistic regression maintains the same cost differences between regularization practices, though the effect is far less pronounced. On average, the performance of $L2$ models across all C parameters is approximately equal to $2521. Though a respectable fraud cost, $L1$ trained algorithms generate an average cost of $6988 under the same circumstances. Additionally, Figure 7 describes the differences in performance distributions for Logistic Regression on both traditional and cost-based metrics.

Again, performance distribution for $L2$ trained models is much less concentrated for Logistic Regression. Trending in the plot seems to indicate a slight positive relationship between F_1 Score and higher cost. This positive correlation contradicts the notion that a higher F_1 Score should correspond to lower cost. Visually, the SVC performance distribution appears nearly identical to that of Logistic Regression when trained with $L1$ regularization, implying $L1$ is effective for imbalanced data.

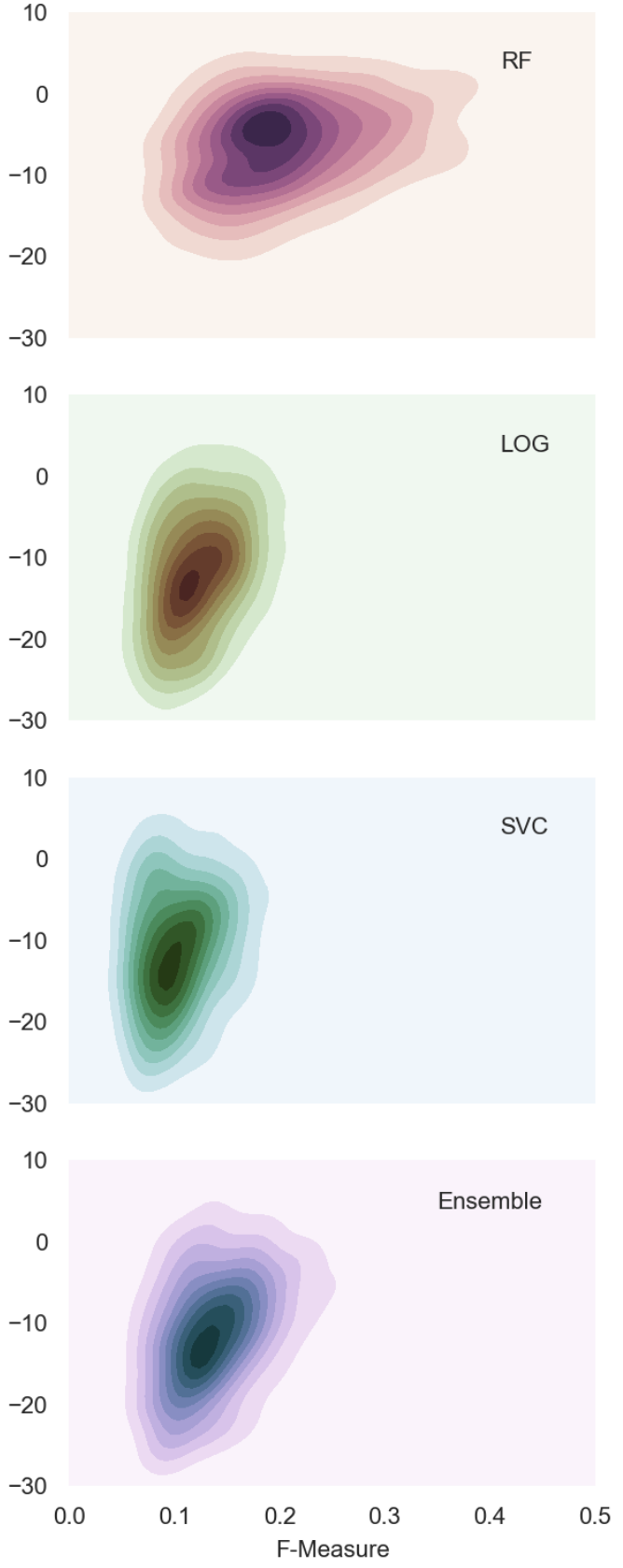

Figure 8: Ensemble performance relative to its components based on F_1 score and fraud cost. All models have identical axes, and the $y-axis$ represents fraud cost in USD - thousands.

8.4. Ensemble unable to outperform individual components.

The top three performing models by fraud cost: Random Forest, Linear SVC, and Logistic Regression, were chosen as the components of the ensemble. Due to the stochastic nature of genetic algorithms, ensemble performance, and random sampling, Monte Carlo tests for ensemble efficacy were run 1000 times. For the majority of these iterations, at least one weight converged to the ceiling threshold of 49%.

Undersampling was yielded the most cost-effective Fraud Detection Systems, and therefore it was used to test ensemble performance. Undersamples with a fraud ratio of 0.3 were taken an then split into 60% and 40% training and testing sets, respectively. Optimal model parameters (Table 5) were also utilized. Interestingly, these subsets were roughly comprised of 100 records each, an incredibly small percentage of the sampling population. Yet, as depicted by Figure 8, all algorithms were able to achieve costs far lower than those built with larger samples.

In particular, Random Forest proved to be the most consistent model regarding fraud cost, but the most variable regarding F_1 scores as denoted by the width of its distribution plot. Logistic and Linear SVC classifiers were far more variable in their cost output despite consistent F_1 scores. Ensemble performance in the bottom plot embodies both of these distributions, but was unable to attain a lower average cost performance that its components.

With an average fraud cost of -$10,847, the ensemble algorithm outperformed all other models except Logistic Regression, which had an mean cost of -$11,168. Yet, it had a lower cost variance than Logistic Regression and Linear SVC, as seen in Figure 9. Though sub-optimal when considered solely for its cost saving potential, ensemble consistency may be particularly valuable if high variation is present in existing transaction activity. Striking a balance between performance and variation is central to long-term fraud detection success.

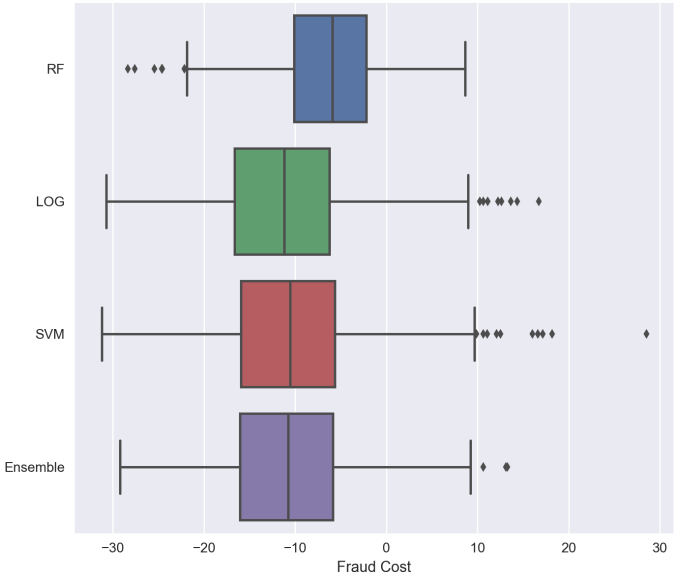

Figure 9: Box plot of ensemble and component fraud cost (USD - thousands)

Table 8: Ensemble Performance Relative to Components

	Fraud Counts				Model Performance								
Model (*Param.*)	TP (#)	FN (#)	TN (#)	FP (#)	*TPR* (%)	*TNR* (%)	*PPV* (%)	*NPV* (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
SVC (*Pen.*, *C*) l1, 0.5	348	45	277,542	6544	88.50	97.70	5.05	99.98	93.10	52.52	97.68	9.56	−10,548
RF (*Trees*) 80	341	53	281,175	2911	86.61	98.98	10.48	99.98	92.79	55.23	98.96	18.70	−6353
LOG (*Pen.*, *C*) l1, 0.5	349	45	278,807	5278	88.60	98.14	6.20	99.98	93.37	53.09	98.13	11.58	−11,168
Ensemble	348	46	279,411	4675	88.36	98.35	6.92	99.98	93.36	53.45	98.34	12.84	−10,847

More sophisticated ensemble classifiers may take greater advantage of the balance between low fraud cost and consistent performance. Avenues for further research may include refining the ensemble optimization process or assigned specific models to detect fraud and non-fraud transactions. Currently, all models seek to accomplish the same goal of preventing fraud by categorizing all transactions correctly. If fraud systems are trained to identify fraud or non-fraud transactions, then the ensemble may specialize weights more effectively and holistically improve.

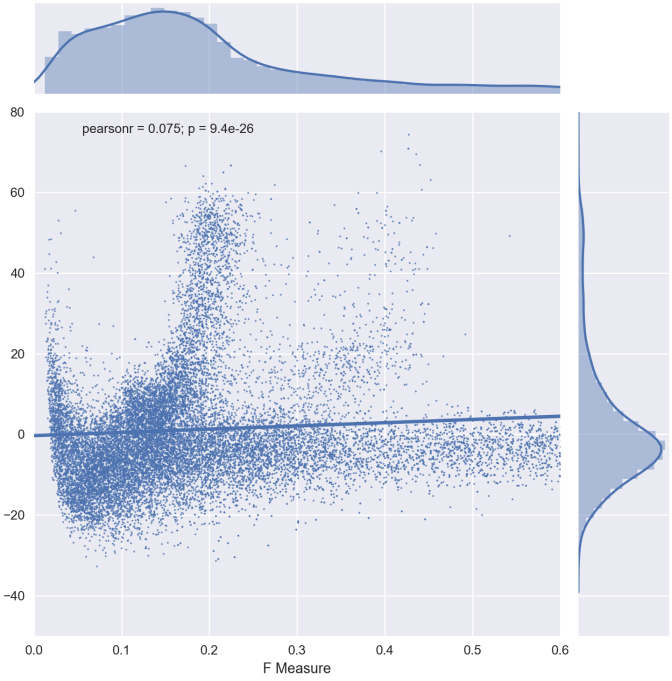

Figure 10: Scatter plot of F_1 Score against fraud cost to identify relationship between performance indicators. The $y-axis$ represents fraud cost in USD - thousands.

8.5. F_1 Score uncorrelated with fraud cost performance

While examining ensemble performance, it became clear that the F_1 Score did not depict any clear relationship with cost performance. However, many researchers including [13] and [42] cite the F_1 Score as a commonly used evaluation metric with imbalanced data problems. To garner a better perspective of the relationship between the F_1 Score to other metrics in this study, a scatter plot mapping it to fraud cost for all algorithms is shown in Figure 10. Since a higher F_1 Score indicates better predictive power, we expected to find a negative relationship between F_1 and fraud cost.

Instead, a slight positive relationship was found between F_1 and fraud cost. Little trending could be found using a linear model, and the r value of the function mapping the plot is 0.071. At the same time, Figure 8 outlines how a single F_1 score often corresponds a wide range of costs. This is likely explained by the disproportionate weighting of TPR over PPV. True positive rate considers how often fraud transactions are classified correctly and is calculated with TP and FN, the two most heavily weighted detection outputs. Since F_1 combines both TPR and PPV, a low TPR and high PPV can generate the same F_1 as a high TPR and a low PPV.

Why, then, include PPV in FDS evaluation at all? Without PPV, a model ceases to consider the occurrence of false positives (false alarms). In this study, FP was weighted extremely low, as it only represents the cost of addressing a false alarm. However, it may cost a great deal more through its intangible effects on business operations. For example, a credit card user who consistently has their transactions blocked unnecessarily will likely stop using their card, losing companies revenue and perhaps tarnishing its public image. From another perspective, employees will be overwhelmed by false alarm transactions until potentially fraudulent transactions can be verified automatically. Thus, F_1 necessarily considers TPR and PPV, but appears unable to appropriately weight their influence for credit card fraud detection.

Another valid question is how to proceed with evaluating Fraud Detection Systems. It would be a mistake to propose quantifying performance solely with cost-based algorithms. The financial effects of fraud, particularly credit card fraud, are difficult to quantify in general, let alone for specific firms. An approximation-based cost matrix may add value to discerning optimal FDS practices, but ultimately it should be supplemented by other metrics. As with ensemble predictions, evaluation metrics are most effective when used in conjunction.

9. Discussion

The objectives of this analysis were to determine optimal approaches for addressing credit card fraud in online channels. As technology evolves, innovate methods of payment will continue to arise, potentially using blockchains or other systems. Therefore, it is imperative to develop a structure for detecting fraudulent purchases where little or no precedent exists and traditional approaches like profiling purchasing behavior are unavailable.

For some like [10], the issue presents itself as a matter of sampling. Overcoming data imbalance is one of the major hurdles for efficiently preventing fraud. Therefore, we explicitly analyzed the benefits and drawbacks of undersampling and SMOTE. While it appeared that SMOTE's creation of synthetic records ultimately caused over-fitting to the minority class, researchers continue to add innovative adaptations [43]. Particularly with credit card fraud, the composition of training data is far more essential for success than size.

For others, credit card fraud detection is approached by finding and tuning the most effective model. While dozens of Fraud Detection Systems have been proposed and tested, few operate under the resource constraints found in industry [1]. Therefore, we implemented five machine learning algorithms that are not reliant on computationally expensive procedures. The control model, K-Nearest Neighbors, embodies all of the pitfalls a traditional machine learning model can suffer from when applied to credit card fraud. Data imbalance tarnished KNN's predictive accuracy, even with re-balanced samples.

Fortunately, three models consistently generated low cost metrics. All models were scored on a cost matrix that emphasized the societal cost of fraud by transaction as opposed to generalizing a static cost for each fraud occurrence [29]. Logistic regression proved to be the most efficient FDS and Random Forest was the most consistent across different samples. Implemented in another e-commerce fraud detection experiment, Logistic Regression's performance in [44] found similar predictive success to its results in this study. Moreover, SVC classification was the second most effect FDS, despite its lack of a kernel function. Notably, it was able to do so in a fraction of the time necessary to run a full Support Vector Machine.

Moreover, ensemble techniques and genetic algorithms are applied to fraud detection, because of their versatility in broad solution spaces and general ability to converge on an optimal solution. Duman & Ozcelik (2011) [45] found relative success detecting bank fraud with an ensemble of scatter search heuristics and genetic algorithm optimization. However, they score the performance of these algorithms solely on statistical metrics like AUROC, hit rate, and gini index. While these metrics are particularly useful for cross-study comparisons, they provide little information to companies that wish to minimize cost.

In 1997, Stolfo et. al. [46] proclaimed that "a cost function that takes into account both the True and False Positive rates should be used to compare ... classifiers." Many studies echo this sentiment, but some continue to publish innovative Fraud Detection Systems without contextualizing cost benefits. While the difficulty of quantifying the cost of fraud is a noted and worthy criticism, it does not reduce the potential benefits an approximated cost structure can provide as a supplemental metric for existing methods. F_1 scores were found to be uncorrelated with cost-based performance in this study. While it is likely no one would score a fraud detection algorithm on F_1 alone, the problems the metric suffers from are endemic to other measures.

Criticism of AUROC is often borne from its inclusion of TN rates that may hold FPR artificially low. Conversely, AUPRC metrics are commended for eliminating some of the effects of data imbalance by not considering TN. Yet, it is intuitively clear that unnoticed fraud transactions (FN) are far more detrimental to a company than false alarms (FP). Yet, AUPRC, a combination of these metrics, weights TPR and PPV equally. Shouldn't the area gained by improving TPR hold a greater significance than area gained by PPV? We maintain it should.

With that said, the difficulty in discerning fraud cost mandates that it be supplemented by traditional metrics for several reasons. Most importantly, standardized metrics unrelated to cost allow for an unbiased categorization of Fraud Detection Systems from an academic standpoint. Furthermore, just as purchasing behavior changes over time, so do the costs of fraud to a firm. Fraud cost matrices for companies have a shelf life, and should only be used to supplement independent statistical measurements. In this way, an algorithm can be characterized independently and benchmarked against its peers, as well as provide contextual information to firms that may wish to implement it. The problem of transferring Fraud Detection Systems from academia to industry is continues today.

Few implementation practices are considered empirically standard in the field of credit card fraud detection. Innovative Fraud Detection Systems continue to be discovered at an accelerating rate. To ensure these innovations positively impact the public, its performance must be clear to all parties involved. No matter how creative, detection algorithms that rely on only cost-based or statistical metrics [39] often do not provide enough perspective to companies or researchers to quantify their impact. Evaluating fraud models in a holistic manner facilitates the adoption of Fraud Detection Systems worldwide, increasing the probability that preventative technology outpaces the proliferation of credit card fraud as spreads to new channels.

10. References

[1] C. Phua, V. Lee, K. Smith, R. Gayler, A Comprehensive Survey of Data Mining-based Fraud Detection Research, Technical Report arXiv:1009.6119, 2010. Comments: 14 pages.

[2] A. D. Pozzolo, G. Boracchi, O. Caelen, C. Alippi, G. Bontempi, Credit card fraud detection: A realistic modeling and a novel learning strategy, IEEE Transactions on Neural Networks and Learning Systems PP (2018) 1–14.

[3] A. D. Pozzolo, R. Johnson, O. Caelen, S. Waterschoot, N. V. Chawla, G. Bontempi, Using hddt to avoid instances propagation in unbalanced and evolving data streams, in: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 588–594.

[4] Y. Kou, C.-T. Lu, S. Sirwongwattana, Y.-P. Huang, Survey of fraud detection techniques, in: IEEE International Conference on Networking, Sensing and Control, 2004, volume 2, pp. 749–754 Vol.2.

[5] R. J. Bolton, D. J. Hand, Statistical fraud detection: A review, Statistical Science 17 (2002) 235–249.

[6] E. Aleskerov, B. Freisleben, B. Rao, Cardwatch: a neural network based database mining system for credit card fraud detection, in: Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr), pp. 220–226.

[7] The true cost of fraud, LexisNexis Solutions (2016).

[8] E. Ngai, Y. Hu, Y. Wong, Y. Chen, X. Sun, The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature, Decision Support Systems 50 (2011) 559 – 569. On quantitative methods for detection of financial fraud.

[9] L. Delamaire, H. Abdou, J. Pointon, Credit card fraud and detection techniques: a review, Banks and Bank systems 4 (2009) 57–68.

[10] K. W. Bowyer, N. V. Chawla, L. O. Hall, W. P. Kegelmeyer, Smote: Synthetic minority over-sampling technique, Journal of artificial intelligence research 16 (2011) 321 – 357.

[11] R. Blagus, L. Lusa, Smote for high-dimensional class-imbalanced data, BMC Bioinformatics 14 (2013) 106.

[12] V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akoglu, M. Snoeck, B. Baesens, Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions, Decision Support Systems 75 (2015) 38–48.

[13] A. D. Pozzolo, O. Caelen, R. A. Johnson, G. Bontempi, Calibrating probability with undersampling for unbalanced classification, in: 2015 IEEE Symposium Series on Computational Intelligence, pp. 159–166.

[14] N. V. Chawla, Data Mining for Imbalanced Datasets: An Overview, Springer US, Boston, MA, pp. 875–886.

[15] P. K. Chan, W. Fan, A. L. Prodromidis, S. J. Stolfo, Distributed data mining in credit card fraud detection, IEEE Intelligent Systems and their Applications 14 (1999) 67–74.

[16] C. Elkan, The foundations of cost-sensitive learning, in: International joint conference on artificial intelligence, volume 17, Lawrence Erlbaum Associates Ltd, pp. 973–978.

[17] C. Phua, D. Alahakoon, V. Lee, Minority report in fraud detection: Classification of skewed data, SIGKDD Explor. Newsl. 6 (2004) 50–59.

[18] A. C. Bahnsen, A. Stojanovic, D. Aouada, B. Ottersten, Cost sensitive credit card fraud detection using bayes minimum risk, in: 2013 12th International Conference on Machine Learning and Applications, volume 1, pp. 333–338.

[19] I. T. Jolliffe, Principal component analysis and factor analysis, in: Principal component analysis, Springer, 1986, pp. 115–128.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, Journal of machine learning research 12 (2011) 2825–2830.

[21] X.-Y. Liu, J. Wu, Z.-H. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39 (2009) 539–550.

[22] C. Drummond, R. C. Holte, et al., C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling, in: Workshop on learning from imbalanced datasets II, volume 11, Citeseer, pp. 1–8.

[23] A. Ramezankhani, O. Pournik, J. Shahrabi, F. Azizi, F. Hadaegh, D. Khalili, The impact of oversampling with smote on the performance of 3 classifiers in prediction of type 2 diabetes, Medical Decision Making 36 (2016) 137–144. PMID: 25449060.

[24] V. Bewick, L. Cheek, J. Ball, Statistics review 13: receiver operating characteristic curves, Critical care 8 (2004) 508.

[25] A. C. Bahnsen, D. Aouada, A. Stojanovic, B. Ottersten, Feature engineering strategies for credit card fraud detection, Expert Systems with Applications 51 (2016) 134 – 142.

[26] J. Davis, M. Goadrich, The relationship between precision-recall and roc curves, in: Proceedings of the 23rd international conference on Machine learning, ACM, pp. 233–240.

[27] A. D. Pozzolo, O. Caelen, Y.-A. L. Borgne, S. Waterschoot, G. Bontempi, Learned lessons in credit card fraud detection from a practitioner perspective, Expert Systems with Applications 41 (2014) 4915 – 4928.

[28] J. R. Dorronsoro, F. Ginel, C. Sgnchez, C. Cruz, Neural fraud detection in credit card operations, IEEE transactions on neural networks 8 (1997) 827–834.

[29] M. F. A. Gadi, X. Wang, A. P. do Lago, Credit card fraud detection with artificial immune system, in: International Conference on Artificial Immune Systems, Springer, pp. 119–131.

[30] S. Bhattacharyya, S. Jha, K. Tharakunnel, J. C. Westland, Data mining for credit card fraud: A comparative study, Decision Support Systems 50 (2011) 602 – 613. On quantitative methods for detection of financial fraud.

[31] R. J. Bolton, D. J. Hand, D. J. H, Unsupervised profiling methods for fraud detection, in: Proc. Credit Scoring and Credit Control VII, pp. 5–7.

[32] D. W. Hosmer Jr, S. Lemeshow, R. X. Sturdivant, Applied logistic regression, volume 398, John Wiley & Sons, 2013.

[33] A. Y. Ng, M. I. Jordan, On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes, in: Advances in neural information processing systems, pp. 841–848.

[34] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (1995) 273–297.

[35] J. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Processing Letters 9 (1999) 293–300.

[36] C.-H. Ho, C.-J. Lin, Large-scale linear support vector regression, Journal of Machine Learning Research 13 (2012) 3323–3348.

[37] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32.

[38] C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, N. M. Adams, Transaction aggregation as a strategy for credit card fraud detection, Data Mining and Knowledge Discovery 18 (2009) 30–55.

[39] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, P. K. Chan, Cost-based modeling for fraud and intrusion detection: results from the jam project, in: DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings, volume 2, pp. 130–144 vol.2.

[40] D. D. Lewis, Naive (bayes) at forty: The independence assumption in information retrieval, in: C. Nédellec, C. Rouveirol (Eds.), Machine Learning: ECML-98, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 4–15.

[41] C. C. Aggarwal, P. S. Yu, Outlier detection for high dimensional data, in: ACM Sigmod Record, volume 30, ACM, pp. 37–46.

[42] S.-J. Yen, Y.-S. Lee, Cluster-based under-sampling approaches for imbalanced data distributions, Expert Systems with Applications 36 (2009) 5718 – 5727.

[43] H. Han, W.-Y. Wang, B.-H. Mao, Borderline-smote: a new over-sampling method in imbalanced data sets learning, in: International Conference on Intelligent Computing, Springer, pp. 878–887.

[44] R. Maranzato, A. Pereira, A. P. do Lago, M. Neubert, Fraud detection in reputation systems in e-markets using logistic regression, in: Proceedings of the 2010 ACM symposium on applied computing, ACM, pp. 1454–1455.

[45] E. Duman, M. H. Ozcelik, Detecting credit card fraud by genetic algorithm and scatter search, Expert Systems with Applications 38 (2011) 13057 – 13063.

[46] S. Stolfo, D. W. Fan, W. Lee, A. Prodromidis, P. Chan, Credit card fraud detection using meta-learning: Issues and initial results, in: AAAI-97 Workshop on Fraud Detection and Risk Management.

Appendix A. SMOTE Sample Tuning, Default Parameters

Table A.9: SMOTE Sample Tuning with Default Parameters: Linear SVC

		Fraud Counts				Model Performance								
Sample Size	Fraud Ratio	TP (#)	FN (#)	TN (#)	FP (#)	TPR (%)	TNR (%)	PPV (%)	NPV (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
1,000	0.1	126	267	222,536	60,929	32.14	78.51	0.21	99.88	55.32	50.04	78.44	0.41	161,243
	0.2	149	246	212,994	70,571	37.69	75.11	0.21	99.88	56.40	50.05	75.06	0.42	166,737
	0.3	163	232	204,240	79,425	41.26	72.00	0.20	99.89	56.63	50.05	71.96	0.41	177,688
	0.4	197	194	176,046	107,719	50.33	62.04	0.18	99.89	56.19	50.04	62.02	0.36	189,600
	0.5	184	211	186,008	97,157	46.54	65.69	0.19	99.89	56.11	50.04	65.69	0.38	192,107
2,000	0.1	119	275	235,060	47,555	30.10	83.17	0.25	99.88	56.64	50.07	83.10	0.49	157,440
	0.2	158	235	215,023	67,792	40.22	76.03	0.23	99.89	58.12	50.06	75.98	0.46	165,654
	0.3	196	199	189,895	93,120	49.59	67.10	0.21	99.90	58.34	50.05	67.08	0.42	184,780
	0.4	196	198	178,906	104,309	49.81	63.17	0.19	99.89	56.49	50.04	63.15	0.37	191,962
	0.5	179	213	189,466	92,999	45.72	67.08	0.19	99.89	56.40	50.04	67.11	0.38	182,824
3,000	0.1	124	270	239,126	42,639	31.54	84.87	0.29	99.89	58.21	50.09	84.80	0.58	150,816
	0.2	173	222	215,037	67,028	43.84	76.24	0.26	99.90	60.04	50.08	76.19	0.51	163,258
	0.3	188	205	202,972	79,393	47.76	71.88	0.24	99.90	59.82	50.07	71.85	0.47	178,327
	0.4	214	179	182,170	100,495	54.56	64.45	0.21	99.90	59.50	50.06	64.43	0.42	188,824
	0.5	175	218	211,552	69,763	44.53	75.20	0.25	99.90	59.86	50.07	75.23	0.50	170,848
5,000	0.1	154	240	235,655	44,410	39.17	84.14	0.35	99.90	61.66	50.12	84.08	0.69	151,794
	0.2	178	216	223,691	56,874	45.22	79.73	0.31	99.90	62.48	50.11	79.68	0.62	158,982
	0.3	206	187	198,210	82,855	52.49	70.52	0.25	99.91	61.51	50.08	70.50	0.49	176,899
	0.4	203	190	201,116	80,449	51.68	71.43	0.25	99.91	61.56	50.08	71.40	0.50	173,130
	0.5	188	205	215,945	62,620	47.75	77.52	0.30	99.91	62.63	50.10	77.60	0.59	161,995
10,000	0.1	159	234	242,743	33,072	40.49	88.01	0.48	99.90	64.25	50.19	87.96	0.95	139,995
	0.2	203	191	222,777	54,038	51.61	80.48	0.37	99.91	66.04	50.14	80.45	0.74	148,821
	0.3	223	170	209,389	68,426	56.66	75.37	0.32	99.92	66.02	50.12	75.36	0.64	161,385
	0.4	238	156	189,477	89,338	60.34	67.96	0.27	99.92	64.15	50.09	67.96	0.53	176,008
	0.5	130	68	88,614	51,044	65.47	63.45	0.25	99.92	64.46	50.09	31.73	0.50	93,526

Table A.10: SMOTE Sample Tuning with Default Parameters: RF

		Fraud Counts				Model Performance								
Sample Size	Fraud Ratio	TP (#)	FN (#)	TN (#)	FP (#)	TPR (%)	TNR (%)	PPV (%)	NPV (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
1,000	0.1	334	60	282,695	770	84.82	99.73	30.24	99.98	92.27	65.11	99.71	44.58	−3587
	0.2	337	149	282,409	1156	69.41	99.59	22.57	99.95	84.50	61.26	99.57	34.06	−4723
	0.3	341	163	281,973	1692	67.65	99.40	16.76	99.94	83.53	58.35	99.39	26.86	−5681
	0.4	341	197	281,349	2416	63.36	99.15	12.36	99.93	81.25	56.15	99.13	20.69	−6917
	0.5	337	184	281,725	1440	64.75	99.49	18.97	99.93	82.12	59.45	99.47	29.34	−4739
2,000	0.1	334	119	282,156	459	73.83	99.84	42.16	99.96	86.83	71.06	99.82	53.67	−4979
	0.2	336	158	282,145	670	67.96	99.76	33.39	99.94	83.86	66.67	99.74	44.78	−4569
	0.3	338	196	282,101	914	63.32	99.68	26.97	99.93	81.50	63.45	99.66	37.83	−5041
	0.4	338	196	281,915	1300	63.30	99.54	20.66	99.93	81.42	60.29	99.52	31.15	−5159
	0.5	335	179	281,606	859	65.14	99.70	28.06	99.94	82.42	64.00	99.68	39.23	−5709
3,000	0.1	334	124	281,413	352	72.90	99.88	48.72	99.96	86.39	74.34	99.85	58.41	−4222
	0.2	337	173	281,556	509	66.04	99.82	39.81	99.94	82.93	69.87	99.80	49.67	−4850
	0.3	336	188	281,686	679	64.16	99.76	33.11	99.93	81.96	66.52	99.74	43.68	−5569
	0.4	337	214	281,733	932	61.14	99.67	26.57	99.92	80.41	63.25	99.65	37.04	−5104
	0.5	334	175	280,639	676	65.63	99.76	33.07	99.94	82.69	66.50	99.74	43.98	−5356
5,000	0.1	332	154	279,808	257	68.29	99.91	56.37	99.94	84.10	78.16	99.89	61.76	−3862
	0.2	333	178	280,242	323	65.15	99.88	50.77	99.94	82.52	75.35	99.86	57.07	−4347
	0.3	334	206	280,624	441	61.80	99.84	43.08	99.93	80.82	71.50	99.82	50.77	−4709
	0.4	334	203	280,975	590	62.26	99.79	36.17	99.93	81.02	68.05	99.77	45.76	−4981
	0.5	330	188	278,187	378	63.75	99.86	46.59	99.93	81.81	73.26	99.84	53.84	−3465
10,000	0.1	329	159	275,646	169	67.39	99.94	66.07	99.94	83.67	83.01	99.92	66.73	−3137
	0.2	330	203	276,612	203	61.91	99.93	61.96	99.93	80.92	80.95	99.90	61.94	−4475
	0.3	328	223	277,588	227	59.59	99.92	59.13	99.92	79.75	79.53	99.90	59.36	−3784
	0.4	331	238	278,532	283	58.14	99.90	53.84	99.91	79.02	76.88	99.88	55.91	−4011
	0.5	166	130	139,505	152	56.20	99.89	52.16	99.91	78.04	76.03	49.93	54.10	−2020

Table A.11: SMOTE Sample Tuning with Default Parameters: GNB

		Fraud Counts				Model Performance								
Sample Size	Fraud Ratio	TP (#)	FN (#)	TN (#)	FP (#)	TPR (%)	TNR (%)	PPV (%)	NPV (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
1,000	0.1	283	110	280,870	2595	71.96	99.08	9.83	99.96	85.52	54.90	99.708	17.30	18,466
	0.2	297	97	280,907	2658	75.35	99.06	10.05	99.97	87.21	55.01	99.573	17.74	13,984
	0.3	302	92	281,052	2613	76.59	99.08	10.37	99.97	87.84	55.17	99.385	18.27	15,068
	0.4	303	88	280,884	2881	77.40	98.98	9.52	99.97	88.19	54.74	99.132	16.95	12,376
	0.5	294	100	280,533	2632	74.55	99.07	10.05	99.96	86.81	55.00	99.473	17.70	15,943
2,000	0.1	292	102	280,202	2413	74.05	99.15	10.78	99.96	86.60	55.37	99.817	18.83	18,207
	0.2	299	95	280,378	2437	75.87	99.14	10.92	99.97	87.50	55.44	99.743	19.09	15,133
	0.3	303	91	280,474	2541	76.84	99.10	10.66	99.97	87.97	55.31	99.657	18.72	15,051
	0.4	306	88	280,483	2732	77.60	99.04	10.06	99.97	88.32	55.01	99.522	17.81	13,480
	0.5	293	99	280,013	2452	74.78	99.13	10.68	99.96	86.96	55.32	99.677	18.70	18,486
3,000	0.1	292	102	279,517	2248	73.99	99.20	11.48	99.96	86.60	55.72	99.854	19.88	21,390
	0.2	301	93	279,731	2334	76.34	99.17	11.44	99.97	87.76	55.70	99.799	19.89	17,596
	0.3	305	88	279,941	2424	77.56	99.14	11.17	99.97	88.35	55.57	99.740	19.53	14,837
	0.4	307	86	280,019	2646	78.17	99.06	10.40	99.97	88.62	55.19	99.651	18.36	12,417
	0.5	297	96	278,945	2370	75.66	99.16	11.14	99.97	87.41	55.55	99.740	19.42	17,186
5,000	0.1	296	98	277,860	2205	75.13	99.21	11.83	99.96	87.17	55.90	99.886	20.44	17,410
	0.2	301	94	278,223	2342	76.28	99.17	11.38	99.97	87.72	55.67	99.863	19.80	16,408
	0.3	305	88	278,657	2408	77.68	99.14	11.25	99.97	88.41	55.61	99.822	19.66	13,605
	0.4	308	85	278,890	2675	78.41	99.05	10.32	99.97	88.73	55.14	99.770	18.23	11,700
	0.5	297	96	276,263	2302	75.52	99.17	11.42	99.97	87.35	55.69	99.843	19.84	18,147
10,000	0.1	294	99	273,718	2097	74.82	99.24	12.31	99.96	87.03	56.14	99.916	21.14	20,675
	0.2	302	92	274,562	2253	76.66	99.19	11.81	99.97	87.92	55.89	99.904	20.47	16,220
	0.3	304	89	275,518	2297	77.37	99.17	11.69	99.97	88.27	55.83	99.895	20.31	16,073
	0.4	308	87	276,300	2515	77.98	99.10	10.90	99.97	88.54	55.43	99.876	19.12	13,182
	0.5	156	42	138,390	1267	78.82	99.09	10.96	99.97	88.96	55.46	49.934	19.24	6720

Table A.12: SMOTE Sample Tuning with Default Parameters: KNN

		Fraud Counts				Model Performance								
Sample Size	Fraud Ratio	TP (#)	FN (#)	TN (#)	FP (#)	TPR (%)	TNR (%)	PPV (%)	NPV (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
1,000	0.1	78	315	271,057	12,408	19.85	95.62	0.63	99.88	57.74	50.25	95.52	1.21	110,409
	0.2	118	297	254,413	29,152	28.42	89.72	0.40	99.88	59.07	50.14	89.64	0.79	110,844
	0.3	159	302	228,912	54,753	34.42	80.70	0.29	99.87	57.56	50.08	80.64	0.57	121,235
	0.4	203	303	196,787	86,978	40.06	69.35	0.23	99.85	54.70	50.04	69.33	0.46	131,542
	0.5	139	294	227,740	55,425	32.14	80.43	0.25	99.87	56.28	50.06	80.40	0.50	126,707
2,000	0.1	73	292	271,197	11,418	19.99	95.96	0.63	99.89	57.97	50.26	95.85	1.23	110,551
	0.2	110	299	255,103	27,712	26.98	90.20	0.40	99.88	58.59	50.14	90.12	0.78	111,486
	0.3	153	303	230,828	52,187	33.53	81.56	0.29	99.87	57.55	50.08	81.50	0.58	116,494
	0.4	196	306	198,899	84,316	39.05	70.23	0.23	99.85	54.64	50.04	70.20	0.46	129,691
	0.5	135	293	229,633	52,832	31.47	81.30	0.25	99.87	56.38	50.06	81.28	0.50	122,097
3,000	0.1	69	292	270,928	10,837	19.09	96.15	0.63	99.89	57.62	50.26	96.04	1.22	110,946
	0.2	110	301	255,128	26,937	26.73	90.45	0.41	99.88	58.59	50.14	90.36	0.80	109,574
	0.3	151	305	231,089	51,276	33.05	81.84	0.29	99.87	57.45	50.08	81.78	0.58	115,790
	0.4	192	307	201,000	81,665	38.48	71.11	0.23	99.85	54.80	50.04	71.08	0.47	123,963
	0.5	134	297	229,783	51,532	31.00	81.68	0.26	99.87	56.34	50.06	81.70	0.51	119,985
5,000	0.1	68	296	269,599	10,466	18.75	96.26	0.65	99.89	57.51	50.27	96.15	1.25	109,132
	0.2	108	301	255,429	25,136	26.45	91.04	0.43	99.88	58.74	50.16	90.95	0.84	105,485
	0.3	149	305	232,970	48,095	32.80	82.89	0.31	99.87	57.84	50.09	82.83	0.61	106,630
	0.4	192	308	203,243	78,322	38.44	72.18	0.24	99.85	55.31	50.05	72.16	0.49	116,092
	0.5	133	297	229,134	49,431	30.96	82.26	0.27	99.87	56.61	50.07	82.37	0.53	117,062
10,000	0.1	72	294	266,407	9408	19.57	96.59	0.76	99.89	58.08	50.32	96.48	1.45	106,847
	0.2	109	302	253,553	23,262	26.51	91.60	0.47	99.88	59.05	50.17	91.51	0.92	101,005
	0.3	150	304	233,765	44,050	33.09	84.14	0.34	99.87	58.62	50.11	84.09	0.67	101,155
	0.4	193	308	206,900	71,915	38.62	74.21	0.27	99.85	56.41	50.06	74.18	0.53	111,481
	0.5	108	156	95,671	43,986	40.84	68.50	0.24	99.84	54.67	50.04	34.24	0.49	59,567

Table A.13: SMOTE Sample Tuning with Default Parameters: LOG

		Fraud Counts				Model Performance								
Sample Size	Fraud Ratio	TP (#)	FN (#)	TN (#)	FP (#)	TPR (%)	TNR (%)	PPV (%)	NPV (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
1,000	0.1	340	53	281,244	2221	86.47	99.22	13.28	99.98	92.84	56.63	99.20	23.02	−8118
	0.2	344	51	280,257	3308	87.13	98.83	9.41	99.98	92.98	54.69	98.82	16.98	−8069
	0.3	348	46	278,916	4749	88.28	98.33	6.84	99.98	93.30	53.41	98.31	12.69	−9401
	0.4	350	42	276,938	6827	89.29	97.59	4.87	99.98	93.44	52.43	97.58	9.24	−9329
	0.5	343	51	278,747	4418	86.98	98.44	7.20	99.98	92.71	53.59	98.43	13.30	−6429
2,000	0.1	340	54	280,875	1740	86.31	99.38	16.34	99.98	92.85	58.16	99.37	27.48	−6840
	0.2	344	50	280,056	2759	87.24	99.02	11.07	99.98	93.13	55.53	99.01	19.65	−7475
	0.3	346	49	279,234	3781	87.67	98.66	8.38	99.98	93.16	54.18	98.65	15.30	−7468
	0.4	349	45	277,827	5388	88.59	98.10	6.08	99.98	93.34	53.03	98.08	11.38	−8547
	0.5	342	50	278,718	3747	87.24	98.67	8.37	99.98	92.96	54.18	98.66	15.27	−7289
3,000	0.1	339	56	280,201	1564	85.91	99.44	17.80	99.98	92.68	58.89	99.43	29.48	−6228
	0.2	343	52	279,544	2521	86.94	99.11	11.98	99.98	93.02	55.98	99.09	21.06	−6868
	0.3	345	48	278,737	3628	87.77	98.72	8.68	99.98	93.24	54.33	98.70	15.81	−9460
	0.4	346	47	277,716	4949	88.16	98.25	6.54	99.98	93.21	53.26	98.24	12.18	−7671
	0.5	342	51	277,724	3591	87.01	98.72	8.69	99.98	92.87	54.34	98.71	15.80	−6997
5,000	0.1	338	56	278,622	1443	85.73	99.48	18.96	99.98	92.61	59.47	99.47	31.05	−5300
	0.2	342	52	278,281	2284	86.72	99.19	13.02	99.98	92.95	56.50	99.17	22.64	−6671
	0.3	344	49	277,845	3220	87.49	98.85	9.65	99.98	93.17	54.82	98.84	17.38	−8301
	0.4	346	47	276,660	4905	88.14	98.26	6.59	99.98	93.20	53.28	98.24	12.26	−6983
	0.5	339	54	275,618	2947	86.28	98.94	10.32	99.98	92.61	55.15	98.93	18.43	−5463
10,000	0.1	336	57	274,609	1206	85.46	99.56	21.81	99.98	92.51	60.89	99.54	34.75	−4139
	0.2	340	54	274,815	2000	86.39	99.28	14.54	99.98	92.84	57.26	99.26	24.89	−6554
	0.3	343	50	274,840	2975	87.25	98.93	10.33	99.98	93.09	55.16	98.91	18.47	−6941
	0.4	347	47	274,399	4416	88.01	98.42	7.29	99.98	93.22	53.64	98.40	13.46	−7977
	0.5	175	23	137,042	2615	88.35	98.13	6.27	99.98	93.24	53.13	49.06	11.70	−3548

Appendix B. Undersample and Model Tuning

Table B.14: Under Sample and Model Tuning: Linear SVC

		Fraud Counts				Model Performance								
Fraud Ratio	*Pen.*, *C*	TP (#)	FN (#)	TN (#)	FP (#)	*TPR* (%)	*TNR* (%)	*PPV* (%)	*NPV* (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
0.1	l1, 0.5	332	62	282,368	1071	84.17	99.62	23.67	99.98	91.89	61.82	99.60	36.95	−4392
	l1, 1	335	60	281,908	1532	84.84	99.46	17.94	99.98	92.15	58.96	99.44	29.62	−6023
	l1, 5	334	60	281,801	1633	84.69	99.42	16.97	99.98	92.06	58.47	99.40	28.27	−5848
	l1, 10	335	59	281,444	1988	85.10	99.30	14.43	99.98	92.20	57.21	99.28	24.68	−6843
	l1, 20	334	59	281,693	1736	84.97	99.39	16.15	99.98	92.18	58.06	99.37	27.14	−6945
	l2, 0.5	120	272	219,266	64,149	30.52	77.37	0.19	99.88	53.94	50.03	77.30	0.37	161,766
	l2, 1	140	252	201,837	81,585	35.75	71.21	0.17	99.88	53.48	50.02	71.16	0.34	177,896
	l2, 5	145	249	214,173	69,254	36.78	75.57	0.21	99.88	56.17	50.05	75.51	0.41	165,758
	l2, 10	130	265	214,919	68,528	32.91	75.82	0.19	99.88	54.37	50.03	75.76	0.38	169,688
	l2, 20	103	290	235,151	48,272	26.19	82.97	0.21	99.88	54.58	50.04	82.89	0.42	150,780
0.2	l1, 0.5	342	52	280,647	3274	86.86	98.85	9.45	99.98	92.86	54.72	98.83	17.05	−8916
	l1, 1	342	51	279,997	3923	86.90	98.62	8.01	99.98	92.76	54.00	98.60	14.67	−9495
	l1, 5	343	50	278,867	5053	87.33	98.22	6.36	99.98	92.77	53.17	98.21	11.86	−9808
	l1, 10	343	49	278,593	5324	87.46	98.12	6.06	99.98	92.79	53.02	98.11	11.33	−9729
	l1, 20	344	48	278,135	5782	87.74	97.96	5.62	99.98	92.85	52.80	97.95	10.56	−9834
	l2, 0.5	147	247	205,615	78,306	37.32	72.42	0.19	99.88	54.87	50.03	72.37	0.37	170,694
	l2, 1	129	263	212,718	71,198	33.00	74.92	0.18	99.88	53.96	50.03	74.87	0.36	158,470
	l2, 5	129	265	213,306	70,616	32.72	75.13	0.18	99.88	53.92	50.03	75.07	0.36	164,528
	l2, 10	122	271	216,129	67,792	31.09	76.12	0.18	99.87	53.61	50.03	76.06	0.36	166,878
	l2, 20	123	271	220,741	63,181	31.29	77.75	0.19	99.88	54.52	50.04	77.68	0.39	158,129
0.3	l1, 0.5	348	46	277,704	6383	88.38	97.75	5.17	99.98	93.06	52.58	97.74	9.77	−10,744
	l1, 1	350	45	276,277	7811	88.58	97.25	4.28	99.98	92.91	52.13	97.24	8.17	−10,182
	l1, 5	348	44	273,455	10,627	88.82	96.26	3.17	99.98	92.54	51.58	96.25	6.13	−7780
	l1, 10	349	43	272,550	11,531	89.01	95.94	2.94	99.98	92.48	51.46	95.93	5.68	−6991
	l1, 20	349	45	272,153	11,932	88.68	95.80	2.84	99.98	92.24	51.41	95.79	5.50	−7013
	l2, 0.5	127	266	218,192	65,894	32.29	76.80	0.19	99.88	54.55	50.04	76.74	0.38	157,186
	l2, 1	174	219	180,877	103,206	44.28	63.67	0.17	99.88	53.98	50.02	63.64	0.34	165,541
	l2, 5	157	236	196,028	88,058	39.95	69.00	0.18	99.88	54.48	50.03	68.96	0.36	164,723
	l2, 10	160	234	192,593	91,493	40.50	67.79	0.17	99.88	54.15	50.03	67.75	0.35	170,509
	l2, 20	146	247	194,563	89,521	37.06	68.49	0.16	99.87	52.77	50.02	68.44	0.32	173,311
0.4	l1, 0.5	352	41	273,161	11,007	89.50	96.13	3.10	99.98	92.82	51.54	96.12	6.00	−9520
	l1, 1	354	40	270,766	13,402	89.74	95.28	2.57	99.99	92.51	51.28	95.28	5.00	−7763
	l1, 5	353	40	265,665	18,502	89.86	93.49	1.87	99.98	91.67	50.93	93.48	3.67	−1831
	l1, 10	353	41	264,563	19,605	89.69	93.10	1.77	99.98	91.40	50.88	93.10	3.47	−205
	l1, 20	353	41	263,639	20,529	89.63	92.78	1.69	99.98	91.20	50.84	92.77	3.32	1616
	l2, 0.5	165	229	189,455	94,712	41.87	66.67	0.17	99.88	54.27	50.03	66.64	0.35	159,595
	l2, 1	189	205	166,379	117,789	48.04	58.55	0.16	99.88	53.30	50.02	58.53	0.32	183,423
	l2, 5	162	230	187,775	96,391	41.33	66.08	0.17	99.88	53.70	50.02	66.05	0.33	171,267
	l2, 10	164	229	184,028	100,139	41.80	64.76	0.16	99.88	53.28	50.02	64.73	0.33	162,443
	l2, 20	187	208	164,220	119,950	47.27	57.79	0.16	99.87	52.53	50.01	57.77	0.31	187,240
0.5	l1, 0.5	358	36	266,850	17,366	90.89	93.89	2.02	99.99	92.39	51.00	93.89	3.95	−5942
	l1, 1	357	37	262,766	21,451	90.56	92.45	1.64	99.99	91.51	50.81	92.45	3.22	−2029
	l1, 5	356	37	256,716	27,500	90.54	90.32	1.28	99.99	90.43	50.63	90.32	2.52	5511
	l1, 10	355	37	255,123	29,093	90.48	89.76	1.21	99.99	90.12	50.60	89.76	2.38	7101
	l1, 20	356	36	252,662	31,554	90.76	88.90	1.12	99.99	89.83	50.55	88.90	2.21	8569
	l2, 0.5	190	204	163,853	120,364	48.32	57.65	0.16	99.88	52.98	50.02	57.64	0.31	170,638
	l2, 1	234	159	131,239	152,977	59.48	46.18	0.15	99.88	52.83	50.02	46.19	0.30	188,965
	l2, 5	186	205	162,082	122,132	47.59	57.03	0.15	99.87	52.31	50.01	57.02	0.30	165,009
	l2, 10	203	191	154,348	129,869	51.48	54.31	0.16	99.88	52.89	50.02	54.30	0.31	184,784
	l2, 20	208	185	148,676	135,540	53.01	52.31	0.15	99.88	52.66	50.01	52.31	0.31	184,930

Table B.15: Undersample and Model Tuning: RF

		Fraud Counts				Model Performance								
Fraud Ratio	*Trees* (#)	TP (#)	FN (#)	TN (#)	FP (#)	*TPR* (%)	*TNR* (%)	*PPV* (%)	*NPV* (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
0.1	10	329	66	282,775	664	83.40	99.77	33.15	99.98	91.58	66.56	99.74	47.44	−2832
	20	332	63	282,842	598	84.05	99.79	35.69	99.98	91.92	67.83	99.77	50.10	−3295
	30	332	62	282,916	518	84.16	99.82	39.04	99.98	91.99	69.51	99.80	53.34	−2848
	40	332	62	282,841	591	84.38	99.79	35.98	99.98	92.09	67.98	99.77	50.45	−4190
	50	332	62	282,933	496	84.33	99.83	40.10	99.98	92.08	70.04	99.80	54.35	−4055
	60	331	61	282,924	491	84.46	99.83	40.27	99.98	92.14	70.12	99.81	54.54	−3175
	70	332	61	282,931	491	84.57	99.83	40.36	99.98	92.20	70.17	99.81	54.64	−4537
	80	332	61	282,918	508	84.38	99.82	39.49	99.98	92.10	69.73	99.80	53.80	−3898
	90	334	62	282,947	500	84.34	99.82	40.01	99.98	92.08	70.00	99.80	54.28	−2965
	100	332	61	282,941	482	84.54	99.83	40.81	99.98	92.18	70.40	99.81	55.05	−3869
0.2	10	334	60	282,079	1842	84.85	99.35	15.35	99.98	92.10	57.66	99.33	25.99	−4546
	20	335	59	282,366	1554	85.09	99.45	17.72	99.98	92.27	58.85	99.43	29.33	−4213
	30	336	57	282,457	1462	85.48	99.49	18.69	99.98	92.48	59.33	99.47	30.67	−4656
	40	336	57	282,367	1551	85.60	99.45	17.81	99.98	92.53	58.90	99.43	29.49	−5326
	50	336	56	282,291	1625	85.70	99.43	17.14	99.98	92.56	58.56	99.41	28.57	−4941
	60	335	58	282,549	1372	85.14	99.52	19.62	99.98	92.33	59.80	99.50	31.90	−3959
	70	334	58	282,419	1498	85.23	99.47	18.26	99.98	92.35	59.12	99.45	30.07	−3837
	80	336	58	282,568	1355	85.33	99.52	19.87	99.98	92.43	59.93	99.50	32.24	−5214
	90	336	57	282,523	1398	85.46	99.51	19.40	99.98	92.49	59.69	99.49	31.62	−4384
	100	337	57	282,621	1302	85.54	99.54	20.56	99.98	92.54	60.27	99.52	33.15	−4597
0.3	10	338	56	280,508	3578	85.75	98.74	8.62	99.98	92.25	54.30	98.72	15.67	−4844
	20	340	55	281,055	3034	86.16	98.93	10.08	99.98	92.54	55.03	98.91	18.05	−5578
	30	339	53	280,918	3165	86.58	98.89	9.69	99.98	92.73	54.83	98.87	17.43	−6636
	40	339	52	280,934	3148	86.63	98.89	9.73	99.98	92.76	54.86	98.88	17.50	−5693
	50	341	53	280,918	3167	86.65	98.89	9.71	99.98	92.77	54.85	98.87	17.47	−6359
	60	341	52	280,967	3119	86.73	98.90	9.86	99.98	92.82	54.92	98.89	17.71	−5956
	70	341	51	280,705	3379	86.96	98.81	9.18	99.98	92.89	54.58	98.79	16.60	−7528
	80	341	53	281,312	2774	86.54	99.02	10.94	99.98	92.78	55.46	99.01	19.43	−6365
	90	342	52	281,054	3033	86.71	98.93	10.12	99.98	92.82	55.05	98.92	18.13	−6952
	100	340	53	281,282	2802	86.46	99.01	10.81	99.98	92.74	55.40	99.00	19.22	−6167
0.4	10	342	52	278,723	5445	86.81	98.08	5.91	99.98	92.45	52.94	98.07	11.06	−4523
	20	345	49	278,932	5237	87.51	98.16	6.18	99.98	92.83	53.08	98.14	11.55	−7540
	30	345	48	279,076	5091	87.82	98.21	6.35	99.98	93.02	53.17	98.19	11.84	−7392
	40	345	49	279,329	4840	87.59	98.30	6.66	99.98	92.94	53.32	98.28	12.37	−6828
	50	346	48	278,928	5240	87.91	98.16	6.19	99.98	93.03	53.09	98.14	11.57	−8228
	60	345	48	279,413	4754	87.68	98.33	6.76	99.98	93.00	53.37	98.31	12.56	−7918
	70	346	48	279,006	5163	87.93	98.18	6.29	99.98	93.05	53.13	98.17	11.73	−8307
	80	345	48	279,380	4786	87.83	98.32	6.72	99.98	93.07	53.35	98.30	12.49	−7969
	90	345	48	279,220	4947	87.78	98.26	6.52	99.98	93.02	53.25	98.24	12.14	−7655
	100	348	47	278,894	5276	88.01	98.14	6.18	99.98	93.08	53.08	98.13	11.55	−8792
0.5	10	348	45	274,702	9514	88.45	96.65	3.53	99.98	92.55	51.76	96.64	6.79	−5358
	20	350	45	275,837	8381	88.71	97.05	4.01	99.98	92.88	52.00	97.04	7.67	−7950
	30	350	43	276,053	8163	89.00	97.13	4.11	99.98	93.06	52.05	97.12	7.85	−8666
	40	348	45	276,577	7639	88.56	97.31	4.35	99.98	92.93	52.17	97.30	8.30	−8224
	50	350	43	276,424	7791	89.05	97.26	4.30	99.98	93.15	52.14	97.25	8.20	−8511
	60	351	43	276,656	7561	89.00	97.34	4.43	99.98	93.17	52.21	97.33	8.44	−8823
	70	350	43	276,459	7758	88.96	97.27	4.32	99.98	93.11	52.15	97.26	8.23	−9035
	80	347	45	277,171	7044	88.64	97.52	4.70	99.98	93.08	52.34	97.51	8.93	−8894
	90	348	45	276,921	7295	88.48	97.42	4.56	99.98	92.95	52.27	97.42	8.67	−8796
	100	352	42	276,209	8007	89.40	97.18	4.21	99.98	93.29	52.10	97.17	8.03	−9278

Table B.16: Undersample and Model Tuning: KNN

		Fraud Counts				Model Performance								
Fraud Ratio	K (#)	TP (#)	FN (#)	TN (#)	FP (#)	TPR (%)	TNR (%)	PPV (%)	NPV (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
0.1	10	29	366	280,976	2462	7.34	99.13	1.16	99.87	53.24	50.52	99.00	2.01	116,711
	20	6	389	282,613	827	1.55	99.71	0.73	99.86	50.63	50.30	99.57	1.00	122,378
	30	1	394	283,242	192	0.14	99.93	0.30	99.86	50.04	50.08	99.79	0.19	123,166
	40	0	394	283,419	13	0.02	100.00	0.53	99.86	50.01	50.19	99.86	0.03	122,700
	50	0	394	283,429	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	122,756
	60	0	392	283,415	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	121,795
	70	0	393	283,422	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	121,509
	80	0	393	283,427	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	122,425
	90	0	396	283,447	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	123,016
	100	0	393	283,423	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	122,348
0.2	10	67	327	273,977	9944	16.91	96.50	0.66	99.88	56.70	50.27	96.39	1.28	111,311
	20	31	362	277,481	6438	7.97	97.73	0.48	99.87	52.85	50.18	97.61	0.91	121,305
	30	16	377	281,246	2673	3.99	99.06	0.58	99.87	51.52	50.22	98.93	1.02	121,376
	40	5	388	283,024	894	1.29	99.69	0.56	99.86	50.49	50.21	99.55	0.78	121,370
	50	1	392	283,754	162	0.18	99.94	0.43	99.86	50.06	50.15	99.81	0.25	122,517
	60	0	394	283,921	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	122,918
	70	0	392	283,916	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	121,633
	80	0	394	283,923	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	122,935
	90	0	393	283,921	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	122,065
	100	0	394	283,922	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	122,496
0.3	10	107	287	259,836	24,250	27.20	91.46	0.44	99.89	59.33	50.16	91.37	0.87	113,065
	20	80	315	264,308	19,780	20.30	93.04	0.40	99.88	56.67	50.14	92.94	0.79	119,726
	30	51	341	272,723	11,359	12.95	96.00	0.44	99.88	54.47	50.16	95.89	0.86	118,210
	40	27	365	277,708	6373	6.83	97.76	0.42	99.87	52.29	50.14	97.63	0.79	120,832
	50	7	386	282,182	1903	1.80	99.33	0.37	99.86	50.56	50.12	99.20	0.61	122,587
	60	4	390	282,888	1197	0.89	99.58	0.29	99.86	50.23	50.08	99.44	0.44	122,407
	70	3	390	283,161	923	0.69	99.68	0.29	99.86	50.18	50.08	99.54	0.41	122,411
	80	2	392	283,485	601	0.45	99.79	0.30	99.86	50.12	50.08	99.65	0.36	122,660
	90	0	394	284,086	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	122,564
	100	0	393	284,084	0	0.00	100.00	0.00	99.86	50.00	49.93	99.86	0.00	122,860
0.4	10	146	247	236,471	47,697	37.19	83.22	0.31	99.90	60.20	50.10	83.15	0.61	121,524
	20	128	266	242,274	41,894	32.50	85.26	0.30	99.89	58.88	50.10	85.18	0.60	125,666
	30	105	288	248,513	35,654	26.78	87.45	0.29	99.88	57.12	50.09	87.37	0.58	124,725
	40	77	317	255,481	28,687	19.51	89.90	0.27	99.88	54.71	50.07	89.81	0.53	127,577
	50	68	326	258,848	25,320	17.16	91.09	0.27	99.87	54.13	50.07	90.99	0.52	125,706
	60	50	344	265,360	18,807	12.62	93.38	0.26	99.87	53.00	50.07	93.27	0.52	124,662
	70	44	350	264,978	19,190	11.27	93.25	0.23	99.87	52.26	50.05	93.13	0.45	127,007
	80	28	364	269,882	14,284	7.22	94.97	0.20	99.87	51.10	50.03	94.85	0.39	127,901
	90	18	375	275,851	8316	4.58	97.07	0.22	99.86	50.83	50.04	96.95	0.41	124,369
	100	4	391	281,214	2956	1.10	98.96	0.15	99.86	50.03	50.00	98.82	0.26	125,100
0.5	10	196	198	200,589	83,628	49.67	70.58	0.23	99.90	60.12	50.07	70.55	0.46	135,647
	20	201	194	193,380	90,837	50.87	68.04	0.22	99.90	59.46	50.06	68.02	0.44	143,860
	30	202	191	188,045	96,171	51.48	66.16	0.21	99.90	58.82	50.05	66.14	0.42	147,056
	40	190	202	185,035	99,181	48.51	65.10	0.19	99.89	56.81	50.04	65.08	0.38	153,828
	50	191	201	181,048	103,167	48.71	63.70	0.19	99.89	56.20	50.04	63.68	0.37	156,212
	60	186	208	178,158	106,059	47.31	62.68	0.18	99.88	55.00	50.03	62.66	0.35	161,166
	70	182	212	174,396	109,821	46.15	61.36	0.17	99.88	53.76	50.02	61.34	0.33	163,997
	80	180	211	171,706	112,509	46.05	60.41	0.16	99.88	53.23	50.02	60.39	0.32	165,464
	90	191	203	161,723	122,494	48.50	56.90	0.16	99.87	52.70	50.02	56.89	0.31	168,178
	100	192	201	160,010	124,206	48.93	56.30	0.15	99.87	52.62	50.01	56.29	0.31	168,590

Table B.17: Undersample and Model Tuning: LOG

		Fraud Counts				Model Performance								
Fraud Ratio	*Pen.*, *C*	TP (#)	FN (#)	TN (#)	FP (#)	*TPR* (%)	*TNR* (%)	*PPV* (%)	*NPV* (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
0.1	l1, 0.5	335	59	282,166	1272	85.00	99.55	20.86	99.98	92.28	60.42	99.53	33.50	−4992
	l1, 1	338	57	281,835	1605	85.59	99.43	17.39	99.98	92.51	58.69	99.41	28.91	−6414
	l1, 5	337	57	281,396	2038	85.56	99.28	14.20	99.98	92.42	57.09	99.26	24.35	−7319
	l1, 10	338	56	280,998	2434	85.85	99.14	12.20	99.98	92.49	56.09	99.12	21.36	−8171
	l1, 20	338	56	281,027	2403	85.79	99.15	12.32	99.98	92.47	56.15	99.13	21.55	−8849
	l2, 0.5	295	97	282,290	1125	75.19	99.60	20.76	99.97	87.40	60.36	99.57	32.54	16,482
	l2, 1	297	96	282,259	1163	75.68	99.59	20.36	99.97	87.63	60.16	99.56	32.09	14,911
	l2, 5	300	94	282,302	1125	76.20	99.60	21.04	99.97	87.90	60.50	99.57	32.98	13,417
	l2, 10	301	95	282,257	1190	76.00	99.58	20.16	99.97	87.79	60.06	99.55	31.87	17,061
	l2, 20	297	96	282,354	1069	75.59	99.62	21.74	99.97	87.61	60.85	99.59	33.77	15,904
0.2	l1, 0.5	344	50	280,900	3021	87.32	98.94	10.21	99.98	93.13	55.10	98.92	18.28	−9134
	l1, 1	343	50	280,446	3474	87.25	98.78	8.99	99.98	93.01	54.49	98.76	16.30	−9658
	l1, 5	345	48	278,670	5249	87.70	98.15	6.16	99.98	92.93	53.07	98.14	11.52	−10,677
	l1, 10	345	48	278,113	5804	87.78	97.96	5.60	99.98	92.87	52.79	97.94	10.54	−10,333
	l1, 20	346	47	277,263	6654	88.08	97.66	4.94	99.98	92.87	52.46	97.64	9.35	−10,596
	l2, 0.5	309	85	282,380	1541	78.45	99.46	16.69	99.97	88.96	58.33	99.43	27.53	8592
	l2, 1	313	80	282,103	1814	79.73	99.36	14.71	99.97	89.54	57.34	99.33	24.84	5472
	l2, 5	312	82	282,132	1791	79.28	99.37	14.85	99.97	89.33	57.41	99.34	25.02	5853
	l2, 10	311	83	282,208	1713	78.94	99.40	15.35	99.97	89.17	57.66	99.37	25.70	6149
	l2, 20	310	84	282,302	1620	78.63	99.43	16.05	99.97	89.03	58.01	99.40	26.65	8160
0.3	l1, 0.5	349	45	278,809	5277	88.68	98.14	6.21	99.98	93.41	53.10	98.13	11.60	−12,318
	l1, 1	350	45	277,953	6135	88.68	97.84	5.40	99.98	93.26	52.69	97.83	10.18	−11,441
	l1, 5	349	43	274,316	9766	89.06	96.56	3.45	99.98	92.81	51.72	96.55	6.65	−9888
	l1, 10	350	42	272,681	11,401	89.30	95.99	2.98	99.98	92.64	51.48	95.98	5.76	−7773
	l1, 20	349	44	271,722	12,363	88.84	95.65	2.75	99.98	92.25	51.37	95.64	5.33	−7425
	l2, 0.5	322	71	281,497	2588	81.95	99.09	11.08	99.97	90.52	55.53	99.07	19.52	1252
	l2, 1	322	70	281,225	2859	82.13	98.99	10.14	99.98	90.56	55.06	98.97	18.04	−1227
	l2, 5	322	72	281,429	2657	81.80	99.06	10.81	99.97	90.43	55.39	99.04	19.10	−16
	l2, 10	323	71	281,068	3018	82.02	98.94	9.67	99.97	90.48	54.82	98.91	17.30	−579
	l2, 20	322	71	281,388	2696	81.91	99.05	10.66	99.97	90.48	55.32	99.03	18.87	−531
0.4	l1, 0.5	352	42	275,673	8495	89.44	97.01	3.98	99.98	93.22	51.98	97.00	7.62	−11,011
	l1, 1	355	40	274,275	9894	89.92	96.52	3.46	99.99	93.22	51.72	96.51	6.66	−11,391
	l1, 5	354	39	268,196	15,971	90.05	94.38	2.17	99.99	92.22	51.08	94.37	4.23	−4706
	l1, 10	354	40	265,486	18,682	89.86	93.43	1.86	99.98	91.64	50.92	93.42	3.64	−1747
	l1, 20	353	40	263,500	20,668	89.80	92.73	1.68	99.98	91.26	50.83	92.72	3.30	854
	l2, 0.5	333	60	279,582	4585	84.78	98.39	6.78	99.98	91.58	53.38	98.37	12.56	−4454
	l2, 1	335	58	279,514	4654	85.17	98.36	6.72	99.98	91.77	53.35	98.34	12.46	−5216
	l2, 5	333	60	279,602	4565	84.74	98.39	6.79	99.98	91.57	53.39	98.37	12.58	−3368
	l2, 10	332	61	279,671	4496	84.50	98.42	6.88	99.98	91.46	53.43	98.40	12.72	−3001
	l2, 20	333	62	279,717	4452	84.31	98.43	6.96	99.98	91.37	53.47	98.41	12.85	−3324
0.5	l1, 0.5	358	36	270,469	13,748	90.96	95.16	2.54	99.99	93.06	51.26	95.16	4.94	−8888
	l1, 1	358	36	268,301	15,916	90.78	94.40	2.20	99.99	92.59	51.09	94.39	4.30	−7994
	l1, 5	357	36	260,539	23,677	90.84	91.67	1.49	99.99	91.26	50.74	91.67	2.92	−661
	l1, 10	356	37	258,410	25,805	90.59	90.92	1.36	99.99	90.76	50.67	90.92	2.68	2798
	l1, 20	357	36	254,355	29,861	90.80	89.49	1.18	99.99	90.15	50.58	89.50	2.33	5830
	l2, 0.5	353	41	273,595	10,622	89.67	96.26	3.22	99.99	92.97	51.60	96.25	6.21	−8451
	l2, 1	353	40	272,879	11,337	89.79	96.01	3.02	99.99	92.90	51.50	96.00	5.85	−8894
	l2, 5	351	41	272,789	11,426	89.57	95.98	2.98	99.99	92.78	51.48	95.97	5.77	−8922
	l2, 10	353	41	272,443	11,774	89.63	95.85	2.91	99.99	92.74	51.45	95.85	5.64	−7629
	l2, 20	355	39	270,786	13,430	90.20	95.27	2.57	99.99	92.74	51.28	95.27	5.00	−6982

Table B.18: Under Sample and Model Tuning: GNB

	Fraud Counts				Model Performance								
Fraud Ratio	TP (#)	FN (#)	TN (#)	FP (#)	TPR (%)	TNR (%)	PPV (%)	NPV (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
0.1	274	120	280,894	2536	69.49	99.11	9.74	99.96	84.30	54.85	99.06	17.08	21,513
0.2	279	114	281,060	2860	70.89	98.99	8.88	99.96	84.94	54.42	98.95	15.78	17,897
0.3	284	110	280,825	3260	72.13	98.85	8.00	99.96	85.49	53.98	98.82	14.41	14,082
0.4	288	106	280,601	3566	73.18	98.74	7.47	99.96	85.96	53.72	98.71	13.56	12,159
0.5	292	101	280,183	4033	74.19	98.58	6.75	99.96	86.39	53.35	98.55	12.37	9508

Appendix C. SMOTE Sample Tuning, Best Parameters

Table C.19: SMOTE Sample Tuning with Best Parameters: Linear SVC

		Fraud Counts				Model Performance								
Sample Size	Fraud Ratio	TP (#)	FN (#)	TN (#)	FP (#)	TPR (%)	TNR (%)	PPV (%)	NPV (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
1,000	0.2	339	53	281,108	2407	86.37	99.15	12.34	99.98	92.76	56.16	99.13	21.59	−6931
	0.3	344	49	279,889	3726	87.55	98.69	8.46	99.98	93.12	54.22	98.67	15.43	−9289
	0.4	348	46	278,553	5162	88.34	98.18	6.31	99.98	93.26	53.15	98.17	11.78	−9499
	0.5	352	42	275,985	7830	89.25	97.24	4.31	99.98	93.25	52.15	97.23	8.22	−9681
2,000	0.1	336	62	281,725	790	84.44	99.72	29.80	99.98	92.08	64.89	99.70	44.05	−4587
	0.2	340	54	280,748	1967	86.38	99.30	14.75	99.98	92.84	57.36	99.29	25.19	−6925
	0.3	344	50	280,226	2689	87.35	99.05	11.35	99.98	93.20	55.67	99.03	20.09	−7249
	0.4	346	47	278,501	4614	88.17	98.37	6.98	99.98	93.27	53.48	98.36	12.94	−7471
	0.5	351	43	276,760	6555	89.09	97.69	5.08	99.98	93.39	52.53	97.67	9.61	−9157
3,000	0.1	332	64	280,911	704	83.83	99.75	32.04	99.98	91.79	66.01	99.73	46.36	−3463
	0.2	340	54	280,126	1789	86.32	99.37	15.98	99.98	92.84	57.98	99.35	26.97	−5555
	0.3	342	51	279,543	2672	87.07	99.05	11.36	99.98	93.06	55.67	99.04	20.10	−7866
	0.4	346	48	278,844	3671	87.76	98.70	8.61	99.98	93.23	54.30	98.69	15.68	−7935
	0.5	348	46	277,094	5721	88.28	97.98	5.74	99.98	93.13	52.86	97.96	10.77	−7333
5,000	0.1	328	65	279,259	556	83.43	99.80	37.10	99.98	91.61	68.54	99.78	51.36	−2358
	0.2	339	55	278,771	1544	85.98	99.45	18.02	99.98	92.72	59.00	99.43	29.79	−4831
	0.3	341	50	278,407	2408	87.14	99.14	12.40	99.98	93.14	56.19	99.13	21.71	−6391
	0.4	345	48	277,712	3603	87.73	98.72	8.75	99.98	93.22	54.36	98.70	15.91	−7233
	0.5	347	46	276,632	5183	88.28	98.16	6.27	99.98	93.22	53.13	98.15	11.71	−7098
10,000	0.1	329	64	274,766	549	83.64	99.80	37.47	99.98	91.72	68.72	99.78	51.76	−3172
	0.2	336	56	274,886	1429	85.84	99.48	19.06	99.98	92.66	59.52	99.46	31.20	−6424
	0.3	341	54	275,124	2191	86.40	99.21	13.47	99.98	92.80	56.72	99.19	23.30	−7080
	0.4	343	48	275,077	3238	87.61	98.84	9.57	99.98	93.22	54.78	98.82	17.25	−8035
	0.5	346	48	274,911	4405	87.75	98.42	7.27	99.98	93.09	53.63	98.41	13.43	−8236

Table C.20: SMOTE Sample Tuning with Best Parameters: RF

		Fraud Counts				Model Performance								
Sample Size	Fraud Ratio	TP (#)	FN (#)	TN (#)	FP (#)	TPR (%)	TNR (%)	PPV (%)	NPV (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
1,000	0.2	335	57	282,605	910	85.42	99.68	26.92	99.98	92.55	63.45	99.66	40.94	−4738
	0.3	338	344	282,349	1266	49.54	99.55	21.08	99.88	74.55	60.48	99.53	29.58	−4663
	0.4	340	348	282,025	1690	49.47	99.40	16.77	99.88	74.44	58.32	99.39	25.05	−7329
	0.5	344	352	281,184	2631	49.43	99.07	11.58	99.87	74.25	55.73	99.06	18.76	−8290
2,000	0.1	336	336	282,200	315	50.03	99.89	51.60	99.88	74.96	75.74	99.87	50.80	−4024
	0.2	336	340	282,212	503	49.72	99.82	40.10	99.88	74.77	69.99	99.80	44.39	−5898
	0.3	338	344	282,170	745	49.55	99.74	31.21	99.88	74.64	65.55	99.72	38.30	−5492
	0.4	339	346	282,056	1059	49.48	99.63	24.26	99.88	74.55	62.07	99.61	32.56	−5706
	0.5	342	351	281,799	1516	49.38	99.47	18.42	99.88	74.42	59.15	99.45	26.83	−7708
3,000	0.1	334	332	281,348	267	50.13	99.91	55.56	99.88	75.02	77.72	99.88	52.71	−4410
	0.2	336	340	281,562	353	49.66	99.87	48.73	99.88	74.77	74.30	99.85	49.19	−4894
	0.3	336	342	281,636	579	49.57	99.79	36.74	99.88	74.68	68.31	99.77	42.20	−5726
	0.4	336	346	281,831	684	49.30	99.76	32.94	99.88	74.53	66.41	99.74	39.49	−5036
	0.5	339	348	281,778	1037	49.33	99.63	24.63	99.88	74.48	62.25	99.61	32.85	−6091
5,000	0.1	331	328	279,626	189	50.22	99.93	63.72	99.88	75.08	81.80	99.91	56.17	−4218
	0.2	333	339	280,059	256	49.55	99.91	56.58	99.88	74.73	78.23	99.89	52.83	−3749
	0.3	331	341	280,494	321	49.28	99.89	50.75	99.88	74.58	75.31	99.86	50.00	−4366
	0.4	335	345	280,877	438	49.24	99.84	43.35	99.88	74.54	71.61	99.82	46.10	−4805
	0.5	334	347	281,259	556	49.06	99.80	37.55	99.88	74.43	68.71	99.78	42.54	−4671
10,000	0.1	328	329	275,172	143	49.97	99.95	69.72	99.88	74.96	84.80	99.92	58.21	−4225
	0.2	331	336	276,135	180	49.58	99.93	64.77	99.88	74.76	82.32	99.91	56.17	−5368
	0.3	331	341	277,117	198	49.27	99.93	62.62	99.88	74.60	81.25	99.91	55.15	−4191
	0.4	329	343	278,069	246	49.00	99.91	57.26	99.88	74.46	78.57	99.89	52.81	−4147
	0.5	330	346	279,019	296	48.84	99.89	52.73	99.88	74.37	76.30	99.87	50.71	−4740

Table C.21: SMOTE Sample Tuning with Best Parameters: LOG

		Fraud Counts				Model Performance								
Sample Size	Fraud Ratio	TP (#)	FN (#)	TN (#)	FP (#)	TPR (%)	TNR (%)	PPV (%)	NPV (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
1,000	0.2	341	51	281,061	2454	86.96	99.13	12.20	99.98	93.05	56.09	99.12	21.40	−8416
	0.3	345	49	280,012	3603	87.62	98.73	8.73	99.98	93.17	54.36	98.71	15.88	−9286
	0.4	349	45	278,547	5168	88.64	98.18	6.32	99.98	93.41	53.15	98.17	11.80	−10,685
	0.5	353	42	276,250	7565	89.43	97.33	4.46	99.98	93.38	52.22	97.32	8.49	−10,012
2,000	0.1	339	58	281,503	1012	85.41	99.64	25.12	99.98	92.53	62.55	99.62	38.82	−5571
	0.2	342	52	280,557	2158	86.89	99.24	13.69	99.98	93.06	56.84	99.22	23.65	−8671
	0.3	345	49	279,986	2929	87.67	98.96	10.55	99.98	93.32	55.27	98.95	18.83	−7854
	0.4	347	46	278,328	4787	88.39	98.31	6.77	99.98	93.35	53.37	98.30	12.57	−7979
	0.5	352	42	276,505	6810	89.46	97.60	4.92	99.98	93.53	52.45	97.59	9.32	−9331
3,000	0.1	336	60	280,653	962	84.79	99.66	25.88	99.98	92.22	62.93	99.64	39.65	−4672
	0.2	342	52	279,967	1948	86.74	99.31	14.93	99.98	93.02	57.46	99.29	25.48	−6647
	0.3	344	49	279,307	2908	87.54	98.97	10.58	99.98	93.26	55.28	98.95	18.88	−8997
	0.4	347	47	278,666	3849	88.09	98.64	8.27	99.98	93.36	54.13	98.62	15.12	−8690
	0.5	350	44	276,945	5870	88.78	97.92	5.63	99.98	93.35	52.81	97.91	10.59	−8499
5,000	0.1	332	61	279,011	804	84.40	99.71	29.23	99.98	92.06	64.61	99.69	43.43	−3039
	0.2	342	53	278,594	1722	86.57	99.39	16.55	99.98	92.98	58.27	99.37	27.79	−6011
	0.3	342	49	278,142	2673	87.45	99.05	11.35	99.98	93.25	55.66	99.03	20.09	−6956
	0.4	347	47	277,450	3865	88.05	98.63	8.23	99.98	93.34	54.11	98.61	15.05	−7907
	0.5	348	45	276,372	5443	88.62	98.07	6.01	99.98	93.35	53.00	98.06	11.26	−8008
10,000	0.1	334	59	274,498	817	84.92	99.70	29.01	99.98	92.31	64.49	99.68	43.25	−4299
	0.2	338	54	274,673	1642	86.34	99.41	17.09	99.98	92.87	58.53	99.39	28.53	−7590
	0.3	342	52	274,876	2439	86.75	99.12	12.31	99.98	92.94	56.14	99.10	21.56	−7575
	0.4	344	47	274,811	3504	87.88	98.74	8.93	99.98	93.31	54.46	98.73	16.22	−8727
	0.5	347	47	274,710	4605	88.01	98.35	7.00	99.98	93.18	53.49	98.34	12.97	−9276

Appendix D. SMOTE Model Tuning, Best Sample

Table D.22: SMOTE Model Tuning with Best Sample

	Fraud Counts				Model Performance								
Model (*Param.*)	TP (#)	FN (#)	TN (#)	FP (#)	*TPR* (%)	*TNR* (%)	*PPV* (%)	*NPV* (%)	Rec. (%)	Prec. (%)	Acc. (%)	F_1 (%)	Cost ($)
SVC													
(*Pen.*, *C*)													
l1, 0.5	351	42	276,083	7732	89.27	97.28	4.34	99.98	93.27	52.16	97.26	8.28	−9825
l1, 1	350	43	275,850	7965	89.09	97.19	4.21	99.98	93.14	52.10	97.18	8.04	−7857
l1, 5	351	42	275,186	8629	89.28	96.96	3.91	99.98	93.12	51.95	96.95	7.48	−8510
l1, 10	349	43	275,630	8185	88.96	97.12	4.09	99.98	93.04	52.04	97.10	7.82	−8796
l1, 20	350	44	275,083	8732	88.89	96.92	3.85	99.98	92.91	51.92	96.91	7.39	−8328
l2, 0.5	210	183	158,519	125,296	53.51	55.85	0.17	99.88	54.68	50.03	55.85	0.33	188,099
l2, 1	213	180	170,740	113,075	54.14	60.16	0.19	99.89	57.15	50.04	60.15	0.37	189,205
l2, 5	197	198	172,909	110,906	49.79	60.92	0.18	99.89	55.36	50.03	60.91	0.35	195,084
l2, 10	188	206	181,296	102,519	47.68	63.88	0.18	99.89	55.78	50.03	63.86	0.36	182,116
l2, 20	214	180	156,695	127,120	54.34	55.21	0.17	99.89	54.78	50.03	55.21	0.33	198,737
RF													
(*Trees*)													
10	341	52	280,612	3203	86.82	98.87	9.63	99.98	92.85	54.81	98.85	17.34	−7188
20	342	51	280,923	2892	87.03	98.98	10.58	99.98	93.01	55.28	98.96	18.86	−6870
30	343	50	281,057	2758	87.25	99.03	11.06	99.98	93.14	55.52	99.01	19.63	−7769
40	341	51	281,202	2613	86.99	99.08	11.56	99.98	93.04	55.77	99.06	20.41	−6701
50	344	50	281,042	2773	87.27	99.02	11.03	99.98	93.15	55.51	99.01	19.58	−7782
60	343	51	281,202	2613	87.12	99.08	11.59	99.98	93.10	55.79	99.06	20.46	−7564
70	342	50	281,332	2483	87.21	99.13	12.12	99.98	93.17	56.05	99.11	21.28	−7641
80	344	51	281,416	2399	87.07	99.15	12.53	99.98	93.11	56.26	99.14	21.90	−7932
90	344	50	281,294	2521	87.21	99.11	12.00	99.98	93.16	55.99	99.10	21.10	−7628
100	343	50	281,288	2527	87.29	99.11	11.97	99.98	93.20	55.97	99.09	21.05	−7245
GNB	307	86	280,856	2959	78.10	98.96	9.41	99.97	88.53	54.69	98.93	16.79	9740
KNN													
(*K*)													
10	199	194	200,762	83,053	50.73	70.74	0.24	99.90	60.73	50.07	70.71	0.48	131,053
20	209	184	193,235	90,580	53.21	68.08	0.23	99.90	60.65	50.07	68.06	0.46	134,371
30	207	186	187,880	95,935	52.78	66.20	0.22	99.90	59.49	50.06	66.18	0.43	138,306
40	210	183	184,525	99,290	53.42	65.02	0.21	99.90	59.22	50.06	65.00	0.42	140,684
50	213	181	183,282	100,533	54.08	64.58	0.21	99.90	59.33	50.06	64.56	0.42	141,361
60	210	183	183,969	99,846	53.42	64.82	0.21	99.90	59.12	50.06	64.80	0.42	141,989
70	208	184	182,563	101,252	53.07	64.32	0.21	99.90	58.70	50.05	64.31	0.41	143,980
80	206	189	183,836	99,979	52.07	64.77	0.21	99.90	58.42	50.05	64.76	0.41	145,748
90	209	185	181,572	102,243	53.09	63.98	0.20	99.90	58.53	50.05	63.96	0.41	146,454
100	211	183	179,546	104,269	53.57	63.26	0.20	99.90	58.42	50.05	63.25	0.40	145,850
LOG													
(*Pen.*, *C*)													
l1, 0.5	352	41	276,389	7426	89.52	97.38	4.52	99.99	93.45	52.25	97.37	8.61	−10,260
l1, 1	351	42	276,059	7756	89.28	97.27	4.33	99.98	93.27	52.16	97.26	8.26	−8196
l1, 5	351	42	274,892	8923	89.27	96.86	3.78	99.98	93.07	51.88	96.85	7.26	−8062
l1, 10	349	44	275,014	8801	88.85	96.90	3.81	99.98	92.88	51.90	96.89	7.31	−8394
l1, 20	350	43	274,381	9434	88.97	96.68	3.58	99.98	92.82	51.78	96.67	6.89	−8188
l2, 0.5	348	46	277,298	6517	88.38	97.70	5.06	99.98	93.04	52.52	97.69	9.58	−6362
l2, 1	348	45	277,110	6705	88.62	97.64	4.93	99.98	93.13	52.46	97.62	9.34	−5323
l2, 5	351	44	276,518	7297	88.87	97.43	4.59	99.98	93.15	52.29	97.42	8.72	−7657
l2, 10	350	44	276,626	7189	88.84	97.47	4.65	99.98	93.15	52.31	97.45	8.83	−6294
l2, 20	350	44	276,548	7266	88.91	97.44	4.59	99.98	93.18	52.29	97.43	8.74	−6683