Journal of Big Data

METHODOLOGY Open Access

Synthesizing class labels for highly imbalanced credit card fraud detection data

Robert K. L. Kennedy[1*], Flavio Villanustre[2], Taghi M. Khoshgoftaar[1] and Zahra Salekshahrezaee[1]

*Correspondence:
rkennedy@fau.edu

[1] College of Engineering & Computer Science, Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431, USA
[2] LexisNexis Business Information Solutions, 245 Peachtree Center Avenue, Atlanta, GA 30303, USA

Abstract
Acquiring labeled datasets often incurs substantial costs primarily due to the requirement of expert human intervention to produce accurate and reliable class labels. In the modern data landscape, an overwhelming proportion of newly generated data is unlabeled. This paradigm is especially evident in domains such as fraud detection and datasets for credit card fraud detection. These types of data have their own difficulties associated with being highly class imbalanced, which poses its own challenges to machine learning and classification. Our research addresses these challenges by extensively evaluating a novel methodology for synthesizing class labels for highly imbalanced credit card fraud data. The methodology uses an autoencoder as its underlying learner to effectively learn from dataset features to produce an error metric for use in creating new binary class labels. The methodology aims to automatically produce new labels with minimal expert input. These class labels are then used to train supervised classifiers for fraud detection. Our empirical results show that the synthesized labels are of high enough quality to produce classifiers that significantly outperform a baseline learner comparison when using area under the precision-recall curve (AUPRC). We also present results of varying levels of positive-labeled instances and their effect on classifier performance. Results show that AUPRC performance improves as more instances are labeled positive and belong to the minority class. Our methodology thereby effectively addresses the concerns of high class imbalance in machine learning by creating new and effective class labels.

Keywords: Label synthesis, Label generation, Unsupervised learning, Credit card fraud detection, Class imbalance

Introduction

Data labeling for classification modeling is often a resource-intensive task that has prohibitive costs and is highly susceptible to errors and inconsistencies [1]. When using labeled datasets in machine learning, label quality is very important. Noisy or inaccurately labeled data can drastically impact the effectiveness and usability of classification models [2]. However, a significant proportion of newly created datasets are unlabeled by default [3]. Interestingly, the vast amounts of newly generated raw data, paradoxically, presents both an advantage and challenge for machine learning. Large amounts of data often yield improved performance in machine learning [4], but the scarcity of class labels, in domains such as medical image diagnosis or fraud detection, significantly decreases the viability of supervised models as they require labels to function effectively. Domains such as these have additional labeling challenges resulting from privacy concerns or the required human experts to effectively label fraud [5]. Unsupervised learning becomes a clear choice when working with unlabeled data. These types of models learn from just the dataset features and class labels are entirely unused in the training process. Though it is possible to use unsupervised learners for classification, they often underperform the supervised alternative [6]. There is an important need for advances in machine learning research that addresses the problems with unlabeled data.

Another significant challenge in machine learning is the issue of class imbalance. This issue arises prominently in datasets where the class representation is significantly skewed, i.e., the number of observations in one class are significantly outnumbered by the number of instances in the other classes. While class imbalance can occur in datasets with more than two classes, our work focuses on the binary class problem. However, all the research presented can be adapted to the multi-class problem through class decomposition [7]. A dataset can be considered imbalanced if the ratio of one class to another is anything other than balanced, but it is often only considered imbalanced when the ratio of one class to another starts to exceed 1:4 [8]. In these cases, it is important to note that class imbalance does not guarantee a worse performance, provided that each class has sufficient representation. More extreme cases of class imbalance, or high class imbalance, are denoted when the class ratio starts to exceed 1:1000, or more. High class imbalance introduces additional levels of complexity and challenges, which necessitates additional considerations for effective model training and classification [9]. Our work seeks to improve upon the existing research in strategies and methodologies specifically designed to combat the challenges of the combination of unlabeled data and high class imbalance. This is especially important for anomaly detection fraud detection since these problems are inherently highly imbalanced, i.e., fraud and anomalies are very infrequent.

To evaluate our methodology, we apply it to the credit card fraud detection dataset [10]. This is a freely and publicly available dataset [11] that consists of anonymized credit card transactions used for credit card fraud detection. It is a binary labeled dataset where the fraudulently labeled instances represent only 0.172% of all instances. This dataset was chosen because it consists of real-world credit card transactions, as opposed to simulated data, its high class imbalance and potential to serve as a realistic benchmark for finding credit card fraud [12]. Importantly, to the best of our knowledge, this dataset is the only publicly available large data for credit card fraud analysis that consists of real world labeled transactions.

Leveraging machine learning techniques to analyze financial data used to find fraudulent activity, such as the credit card fraud detection dataset, is critical for several reasons. In focusing on credit card fraud, the scale of fraudulent activities is enormous. Huge strain is levied on the global financial system due to the billions of U.S. dollars lost to fraud globally and increases the cost for all other card holders [13]. Using advancements in machine learning to combat fraud can help alleviate the imposed financial burdens. Additionally, quickly and accurately finding fraudulent transactions can stand as a large deterrent to potential fraudsters as well as bolster the integrity of the credit card industry as a whole. Machine learning models are not only faster than expert human analysis, but they have the potential to just as quickly find trends or anomalies that lead to fraud that may go unnoticed by manual inspections. Furthermore, deploying machine learning models to find fraud can optimize the entire investigative process.

In this study, we extensively evaluate a novel approach for synthesizing new binary class labels in the context of highly imbalanced large data. The methodology effectively addresses the challenges of both class imbalance and the challenges associated with unlabeled data. Given an unlabeled dataset, the method uses a neural network (NN) to learn from the dataset features and it calculates an error metric for each instance which, in turn, is used to label instances as either positive or negative. For the scope of our work, positive instances represent fraudulent activity and negative instances indicate no fraud. Additionally, the method identifies majority instances near the class boundary and reassigns their label. For the scope of this work, instances that are either fraudulent or legitimate are separated by a class boundary. Empirical results show that this improves the quality of the data for supervised learning and subsequently improves performance. Our experimental results show that the synthesized labels are effective in training supervised classifiers for fraud detection with and without the additional positive labels.

The remainder of the paper is organized as follows. The "Related works" section provides a review of related works in the context of automatically labeling data and highlights how our research is novel in its field. The "Methodology" section provides an in-depth detail of the methodology, including its inputs and outputs. In "Results and analysis", we detail the dataset which we apply our method to, detail the supervised classifiers used to evaluate and validate the method's newly synthesized class labels, and present our empirical results and analysis. The "Conclusion" section concludes the paper and discusses potential areas of future work.

Related works

After our literature review, it was clear that our methodology is pioneering research in the field of addressing automated binary class labeling of highly imbalanced large or big data. Given the novelty of our work, the available related works are only loosely related and may not provide a basis for direct comparisons. However, we feel including tangentially related works provides a relevant context for our paper and further highlights the uniqueness of our work.

Baek et. al [14] aim to develop a method for detecting network anomalies in a supervised manner without needing intensive network traffic analysis by experts. Their approach consists of three key steps. First, they estimate labels for the training data using a clustering technique. Second, they train supervised models using the estimated labels and third, they use this supervised model to classify individual data points as either normal network behavior or anomalous network behavior. During the first step, they use K-means clustering to cluster their data. The main premise for labeling clusters as anomalous is twofold. The first rule to label clusters as anomalous is if a cluster is small or sparse. However, relying only on this produces unsatisfactory results; thus, they aim to improve this by introducing an additional rule when labeling clusters. The second rule for labeling clusters as anomalous is if a cluster is dense, it must be anomalous. During their work, they observed that several network attack types have similar patterns with only minor differences. For many instances, one type of attack in the dataset exhibits very similar feature values which creates dense clusters. Using this knowledge, they were able to improve the estimated labels during the clustering phase, which in turn, increased the performance of supervised learners using them. This differs from our work in a significant way as their work required finding details in features that correlated to their target label manually and ours does not. This makes their clustering approach entirely dependent on finding such a pattern, if one even exists, and dependent on an intensive investigation of the dataset features with prior knowledge. This approach is unable to find new patterns that would be a signal for an anomalous instance. Additionally, this approach would not work with a dataset that has anonymized features, as is the case with the dataset used in this work.

Moslehi et. al [15] use a clustering approach and an algorithm designed to label the clusters. They use K-means clustering to improve the clustering labeling process, and their work's premise is that labeling the full dataset might not be possible. Instead, they suggest labeling a representative portion of the data based on the clusters formed in this subset. Their aim is to improve the quality of label assignment to clusters. As such, they use a secondary dataset as the source of their labels. The instances in their dataset have statistics generated by their clustering technique. These statistics are then matched to the same type of statistics of instances in another dataset, which already have an assigned label. Thus, they are using a secondary, labeled dataset as a look up table to assign labels to their data. This differs from our work significantly in that they do not create new class labels for their dataset. They are essentially assigning class labels from an existing source based on a similarity measurement, i.e., their work requires a secondary labeled dataset to assign labels to their unlabeled dataset. Additionally, their dataset is very small, with only 385 samples, and is significantly smaller than the dataset we used and evaluated in this paper.

Maqbool et. al [16] present an automated approach for labeling clusters based on keyword identifiers in their dataset. Their keywords are ranked using two ranking schemes: frequency and inverse frequency and the work is in the domain of legacy software systems. Given this, the need for an automated approach is clear since without appropriate labels being assigned to clusters, the clusters may not be readily interpretable. Their experimental results demonstrate that their labeling is an improvement over the existing work. Similarly, Rauber [17] introduces an approach for labeling self-organizing maps [18] applied to relatively small text datasets. This work also assigns labels to the clusters that are pulled from a set of words contained in the features of the dataset. Though our work aims to be automated, similar to [16], it is significantly different. The dataset in [16] has keywords in the features, which are ranked according to their two schemas. These keywords are then, in turn, used as cluster labels, i.e., the labels used are directly derived from the features of the dataset. Our dataset does not contain keywords from which we can generate labels. Instead, our approach calculates an error statistic using an autoencoder trained on only numeric features in an unsupervised fashion. The binary class label that is generated by our approach is primarily determined by this error statistic.

These tangentially related works differ from our work in four significant and key ways. First, the clustering approaches reviewed do not adequately address the challenges posed by highly imbalanced datasets, and the existing research does not consider this type of data. For example, [14] uses a dataset that is class balanced. A second important difference is that these works all use datasets significantly smaller than the size for which our method is designed. For instance, a dataset with 385 samples is used in [15]. Third, our work does not directly use aspects of the dataset features as the source for the class labels. Keywords are used as labels in [16], whereas our work uses data that only has numeric values as features. Fourth, and most importantly, our approach is automated by nature. It does not require intensive human intervention and is appropriate for highly imbalanced large and big datasets. This is not the case for other works, such as in [14–16].

Methodology

The methodology presented in this paper synthesizes binary class labels in the context of highly imbalanced large data. To the best of our knowledge, this is the first work in this area and the first method of its kind. The approach uses an underlying learner to effectively learn from just the dataset features and calculates an error metric for each instance after training. The instances are then sorted, from highest to lowest error. Instances with higher error are considered more likely to be in the positive class, in our empirical results positive represents fraudulent activity, and instances with lower error are considered more likely to be in the majority class, or non-fraud. The data is now in a state such that there exists a gradient from most likely positive to most likely negative and there exists an area in between the two ends that instances are equally likely to be in one class or the other.

The instances that the learner is uncertain about are in a gray area. Additionally, a threshold must be chosen so that instances above it are labeled as positive and negative if below. A portion of instances below the threshold but are near it are added to the minority class. Since these instances are nearest to the threshold they aren't strongly assigned to the majority. Labeling some of these instances as positive aims to add diversity to the minority instance group which is intended to improve model generalization and performance. These instances near the boundary can be considered as potential noisy instances and the alternative to placing them in the minority would be to remove them. This, however, would remove potentially valuable information from the training data. Various levels of increasing the number of positive instances are explored. Our results show that adding these uncertain instances, that are near the class boundary, to the positive class improves supervised learning and resulting classification performance on the original ground truth labels.

Underlying learner

The first step in synthesizing new class labels is to learn from the dataset features using a fully connected autoencoder. An autoencoder is a type of artificial neural network that is typically used for unsupervised learning. These architectures are an effective learner in the context of high class imbalance [19–21], making it a good fit for our target dataset. Autoencoders, fully connected or otherwise, were designed to encode and decode data often by using hidden layers [22]. Other types of layers, used in other NN architectures as well, can be used in, such as convolutional, recurrent, fully connected layers, or a combination of different layer types. As their name suggests, autoencoders are made up of two main components, an encoder and a decoder, and aim to automatically encode and decode, or reconstruct data. First, the encoder segment of the autoencoder encodes the input data, which is at a relatively high-dimensionality, by transforming it into a compressed, lower-dimensional representation. This compressed representation is then used as an input to the next part, the decoder. The decoder then uncompresses the lower dimensional data back into the same higher dimensionality of the original input data, i.e., this part of the autoencoder attempts to reverse the compression while maintaining the information contained in the data.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

To measure and quantify the difference between the input data and the output of the architecture, a reconstruction error metric is used. The higher the reconstruction error, the more different the input data and the autoencoder's representation output are. A lower reconstruction error indicates the autoencoder is better at reconstructing the compressed input data. For the autoencoder used in this work, the reconstruction error is measured as mean squared error (MSE). The metric, MSE, calculates the average of the squares of errors between the actual and predicted values, or the input data and the reconstructed data. MSE is defined in Eq. 1, where n is the total number of observations, y_i are the actual values, and $\hat{y}_i$ are the predicted values. During the training process, the optimization algorithm attempts to minimize the reconstruction error. A high reconstruction error for an instance, among other instances with a lower error, might indicate that instance is somehow significantly different, i.e., it can be considered an outlier or anomaly. As such, researchers have successfully used autoencoders in various different domains such as dimensionality reduction, image and video processing, and anomaly detection, among others.

In general, autoencoders are trained in a similar fashion to other NN. The weights between neurons are incrementally changed to minimize an objective function. We use backpropagation to train the autoencoder by minimizing the reconstruction error. The autoencoder used to generate class labels uses fully connected layers where the input layer has the same number of neurons as the number of features in our data and is illustrated in Fig. 1. From the left, the encoder portion has a single 100-neuron layer fully connected to the second hidden layer with 50 neurons. They both use the ReLu activation function. This layer is the end of the encoder portion and is connected to the decoder portion. The decoder's architecture mirrors the encoder. A 50-neuron layer fully connected to a 100-neuron layer. All layers in the decoder use the Tanh activation function and the final output layer uses ReLu. We train using a learning rate set to 0.0001, a batch size of 256, Adam optimizer function, and MSE as the loss metric, and a validation set size of 20%. We train using the EarlyStopping function to monitor training loss with the patience set to 25 epochs and 250 epochs as a maximum. Keras [23], version 2.8.0, was used to define, train, and make predictions with the autoencoder.

It is important to reiterate that the autoencoder is trained in an unsupervised manner. This is necessary because the method is synthesizing new class labels and in practice, this would be done using unlabeled data. Once the autoencoder is fully trained,

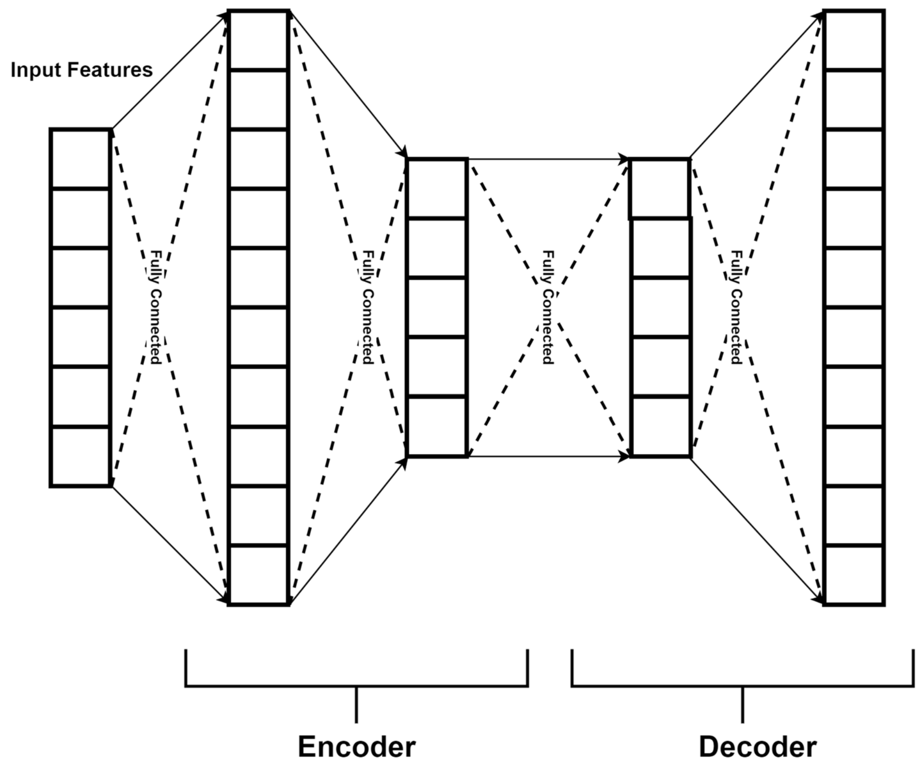

Fig. 1 Autoencoder visualization of the encoder and decoder components

we apply the autoencoder to the data to get a reconstruction error for each instance. The instances are then sorted from greatest to least MSE. Since our target dataset is a binary fraud detection dataset, and its domain has high class imbalance, instances in the minority class are more likely to have a high error and are ranked at the top of the list and instances in the majority class are more likely to have a lower error and are at the bottom of the list.

Input parameters

For the methodology to automatically synthesize class labels, it requires a minimal amount of input parameters. One parameter is the desired number of instances to be targeted as the positive class. This means our methodology utilizes domain expertise as a critical input component. In comparison with a strictly data-driven approach, which primarily uses only patterns present within the data, the addition of domain expertise allows for the methodology to be easily tailored for specific domain applications with minimal change and is not limited by patterns present in only one domain or dataset. Consider the domain of financial fraud detection. Datasets in this domain, such as the credit card fraud detection dataset used in our experiments, significant amount of expert human intervention is required to accurately detect fraudulent cases. Additionally, there are significant privacy concerns associated with this since detailed personal financial details are required to be inspected. When human inspection is required, it is reasonable to assume this sensitive information could not be obfuscated, and thus, there would be fewer possible people allowed to do this. However, it is reasonable to assume that domain experts would know of or easily be able to determine an expected, or acceptable, amount of fraud to identify. Using domain expertise also offers additional speed advantages. Though training and consuming the autoencoder is the most computationally demanding part of the overall methodology, it is domain agnostic. Only after this are the class labels considered, using input from the domain expertise. Once the NN is trained and the reconstruction error is calculated for all instances, the process of assigning labels is computationally trivial. Thus, another advantage to using domain expertise is if the expected level of fraud, or imbalance, changes. Adapting to this change and synthesizing new labels for the same data would happen quickly.

The other input parameter, though not strictly required, is the number of majority instances nearest to the class boundary to be labeled as positive, *P*. Consider the sorted list of instances that are not labeled as positive. Since they are sorted from greatest reconstruction error to least, the instances at the bottom are more likely to be in the negative class than the ones at the top and the instances at the top are not only least likely to be in the majority class, but they are also nearest to the class boundary, as defined by the reconstruction error. Instances near this class boundary, or ones the methodology is least confident in labeling as positive or negative, if labeled incorrectly, can contribute to the noise when training supervised classifiers and reduce classification performance. An input parameter defines what number of these instances, the ones not labeled positive, to be added to the majority class before the remaining ones are labeled as negative. This number happens in a top-down fashion. The method does this to improve classification performance by aiming to increase the number of instances, actual fraudulent instances, in the minority class for model training. It is possible to set this input parameter to 0% and it will consider all majority instances as such. Section "Results and analysis" presents the classification results after using several different percentages and shows its effect on classification performance.

Methodology output

The output of the methodology is a result of the input parameters, the desired number of positive-labeled instances and the features of the dataset. The methodology automatically produces a dataset that has the same features as the input dataset and the newly synthesized binary class labels and the total size of the dataset is the same size as the input. This dataset can be used to train supervised classifiers and then used to get classification predictions on unseen test data. In the case of our experiments, we synthesize class labels on a credit card fraud detection dataset, and, in practice, the subsequently trained supervised classifiers would then be used for fraud detection on new, unseen future data.

Results and analysis

We designed our experiments to measure the quality and efficacy of our method's newly synthesized binary class labels for highly imbalanced data. To achieve this, we apply our method to a widely used labeled fraud detection dataset, using only the features, to generate new and independent class labels. The method produces an imbalanced set of positive and negative class labels, where there are significantly more instances in the negative, or non-fraud, class. This imbalance is expected in this type of dataset. We then train supervised classifiers on the newly created class labels entirely and measure their classification performance using the original class labels, used as ground truth labels. Additionally, we selectively label instances in the majority, that are nearest the class boundary, and label them as positive. This effectively adds instances to the positive-labeled ones. The premise behind this is these instances were least confidently assigned to the majority by the methodology and adding additional instances to the positives would capture more of the actual fraud instances in the positive class in the training data. Increasing the number of positives by using the instances that were least separated during the unsupervised learning process, aims to improve classification performance.

Dataset for labeling

The dataset we consider for our experiments is the publicly available [11] and widely referenced credit card fraud detection dataset, originally introduced in [10]. A collaborative effort between Worldline, a large payment processing company, and the Université Libre de Bruxelles created this dataset. It contains over 280,000 credit card transactions, each of which have 30 independent features that have been anonymized using principal component analysis (PCA). They are European credit card transactions recorded between September 1st through September 30th, 2013. This makes this dataset unique in that it is the only publicly available credit card transaction data that represents a snapshot of real-world credit card usage and patterns.

This dataset is a fraud detection dataset where the binary class labels signify the transaction is either a genuine credit card usage or fraudulent credit card usage. It is important to note that the provided class labels are only used as ground truth labels in calculating the classification performance of supervised learner. They are completely ignored by our method during the synthesizing of the new class labels. As is expected with fraud detection datasets, this dataset is highly class imbalanced. The number of genuine transactions significantly outnumber the fraudulent ones. A detailed breakdown of the dataset and its imbalance is presented in Table 1. There are originally 30 independent features, "Amount", "Time", and "V1" through "V28". The "V" features are the anonymized features generated by PCA by the researchers who originally created the data. We omit the "Time" feature for the entirety of our study since previous work as shown that this feature contributes to noise and does not have any meaningful predictive value [24].

In the domain of fraud detection, obtaining accurate class labels presents unique and significant challenges, including labeling financial transactions as genuine or fraudulent. As an example, as a result of the financial aspect of datasets like the one we use, there are additional challenges presented by privacy concerns. This is why the credit card fraud detection dataset we use has had all personally identifiable information either anonymized or removed from the features. This allows these transactions to be a part of a publicly available dataset that represents real transactions, as opposed to simulated credit card transactions. These challenges substantially restrict the availability of a freely accessible credit card fraud

Table 1 Credit card fraud detection dataset class characteristics

Minority count	Majority count	Total count	Minority imblalance	Features
492	284,315	284,807	0.1727%	29

detection dataset, containing real word examples. Meaning that this dataset, to the best of our knowledge, is the only publicly available dataset for credit card fraud detection analysis. Thus, our experiments solely focus on this dataset. Another challenge in this dataset domain is it is inherently difficult to precisely pinpoint fraudulent credit card activity. To ensure accurate labeling of fraudulent instances, it often requires extensive corroborative manual investigation by financial experts. This is both costly and time-consuming, due to the required expert human involvement. Even with tedious expert financial analysis, some types of fraud are not always evident and can pose additional challenges to precise and timely fraud labeling.

Performance metrics

We present one performance metric when measuring the classification performance of the supervised learners, the area under the precision-recall curve (AUPRC). This performance metric is used to validate and evaluate the effectiveness of the synthesized class labels by summarizing the classification performance of supervised classifiers trained using the new labels and tested on original unseen labels. For binary classification problems, including fraud detection, it is conventional to use a confusion matrix illustrated in Table 2, to summarize the classification results. AUPRC values are derived from the TP, FP, FN, and TN values.

$$FPR = \frac{FP}{FP + TN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = TPR = \frac{TP}{TP + FN} \tag{4}$$

AUPRC is related to the widely used receiver operating characteristic curve (ROC) that was first introduced by Provost et al. [25]. It summarizes the true positive rate vs. the false positive rate. This curve illustrates the trade-off between incorrectly classified negative instances and correctly classified positive instances. Davis et al. [26] show that the ROC curves, and thus the area under the receiver operating characteristic curve (AUC-ROC or simply AUC), can show overly optimistic results for classification results of highly imbalanced data and can be misleading, especially with credit card fraud detection [27]. AUPRC is related to AUC in that like AUC, AUPRC is a single number that summarizes the classification performance of a model and is an area under a curve. Thus, the superior alternative is the precision-recall (PR) curves, and the area under this curve.

Table 2 Confusion matrix

	Actual positive	**Actual negative**
Predicted positive	True positive (TP)	False positive (FP)
Predicted negative	False negative (FN)	True negative (TN)

The AUPRC is the area under the PR curve and is a better measure of classification performance with highly imbalanced data [24, 28], and is the performance metric of interest presented in our empirical results. Further, Davis et. al [26] state that a curve can only dominate in the ROC space if and only if it dominates in the PR space. Thus, a classifier that has superior performance in the PR space would also have superior performance in the ROC space, over another model. PR summarizes a classifier's performance in terms of precision and recall, or TPR, while ROC summarizes a classifier's performance in terms of FPR and TPR. This makes AUC less sensitive to the change in FP as the number of negative instances grows, which is the case for highly imbalanced dataset (negative instances far outnumber the positives). The range of scores for AUPRC are between 0.0 and 1.0. A classifier with an AUPRC score of 1.0 is one that can identify all true positives with no misclassification and has perfect precision and recall across all thresholds. The minimum AUPRC is 0.0, though in practice it is the class imbalance ratio. For example, when working with a dataset that has a class imbalance ratio of 0.1727%, as is the case with our data, the minimum AUPRC score would be effectively 0.001727. Thus, a higher AUPRC score indicates a higher performing classifier.

Classifiers

Six supervised classifiers were used to evaluate our class label synthesizing method: decision tree, random forest, extra trees, logistic regression, Naïve Bayes, and a multilayer perceptron. decision trees (DT) are a widely used supervised method for classification and regression tasks. They consist of a hierarchical tree structure that has a root node, branches, internal nodes, and leaf nodes. The internal decision nodes are conditions or tests based on the data attributes and the leaf nodes are the final output, either a numeric value for regression or a class decision when used as a classifier. We use a DT as a classifier in our experiments. The goal is to create a model that uses simple decision rules derived from the features in the training dataset to predict a target value. To do so, a DT uses a divide and conquer strategy by conducting a greedy search to find optimal split points within a tree. This process is repeated in a recursive manner until all or most of the instances have been assigned under specific class labels. The depth, or size, of the DT has a large impact on its accuracy and performance. Many cases the leaves are pure nodes. When all data in a node belongs to a single class that node is considered pure. For example, in a fraud detection dataset where there are instances classified as fraud or non-fraud, a node is pure if all the sample data is either fraud or non-fraud. As trees grow in size, maintaining purity in the leaf nodes becomes more difficult and can often lead to overfitting. Pruning is often used to reduce tree size and avoid overfitting. This is a process that trims branches from the tree that split on features with low importance. One advantage to the DT is in its relative structural simplicity. This allows for ease of interpretability and can be easily visualized. Other more complex models such as neural networks, often referred to as black box models, are significantly harder to interpret and visualize. One large disadvantage to DT is a result of their simplicity. DT can underperform other models with similar data.

The second supervised classifier we use in our experiments is random forest (RF). RF is an ensemble method first introduced by Breiman et al. [29] and as its name implies, it consists of multiple tree classifiers. These trees are developed using a bootstrap sampling method of the training data, i.e., sampling with replacement. RF are built with an ensemble of DTs to promote diversity among the individual trees to improve the disadvantages of DT, such as reducing the tendency to overfit. For every split in each tree only a random subset of the features is considered, as opposed to DT where all of them can be used. The ensemble of trees that individually consider only a subset of the data produces a method that is robust to overfitting. RF is able to use datasets that have high dimensionality without the need for feature selection or elimination of features. Additionally, RF is easily interpretable, like DT. RF provides a measure of feature significance that highlights which features have the most importance for classification. Since each tree is a DT, they are each easily interpretable, but it is the combination of them all that gives RF its robustness. Though, as the number of trees in the forest increases, the requirements for computing power and memory go up. When RF is making predictions, each tree in the forest provides its own classification prediction and RF uses a majority voting mechanism to make its final prediction.

The third classifier we use to evaluate our synthesized labels is Extra trees (ET), or Extremely Randomized Trees, introduced by Geurts et al. [30]. ET is an extension of the RF algorithm and differs in two main ways. First, when the ET algorithm splits the components of the tree it is done in a completely randomized way as opposed to RF, which computes optimal split points, using each feature that is considered, for tree splitting. This inherent randomness of ET contributes to a more diversified ensemble of trees. It also reduces the computational requirements by not calculating optimal split points during ET training. Secondly, in ET, each of the individual trees are trained using all the training samples in the training dataset. RF uses random subsets for each individual tree. ET's differences from RF produce a classifier that is less prone to overfitting. ET uses majority voting of its forest trees to make final predictions, like RF.

The fourth supervised classifier used in our study is a widely known and used statistical method that is used primarily for binary classification, namely logistic regression (LR). LR estimates the probability that an input instance belongs to a class by fitting the data to the logistic function curve or sigmoid function. The dependent variable can be categorical and the relationship between the input features and dependent variable is non-linear but can be defined by log-odds. The sigmoid function, useful for probability estimates, is an S-shaped curve that accepts any real number and will map it to a value between 0 and 1. To calculate a probability, LR takes a linear combination of the input features then applies the logistic function. A threshold, typically 0.5, is used to convert the numeric probability to a classification output. Instances above the threshold are in the positive class and instances below are in the negative class. LR is advantageous in that it is easily interpretable and easily implemented. The coefficients of a trained LR model can be used to provide insights into features and their importance. LR has been used across many domains and is a foundational model and maintains its relevance in statistics and machine learning applications.

The fifth supervised classifier used is Naïve Bayes (NB). NB is a widely known, relatively simple, probabilistic classifier based on the Bayes' theorem. It makes the naïve assumption that there is conditional independence between the features given a class label. Though they are one of the simplest of their kind, which makes them train quickly, they can achieve good performance in practice and have been widely used in areas such as text classification, sentiment analysis, and spam detection, a type of outlier or fraud detection. NB calculates the posterior probabilities of each class given a set of features and assigned the class with the highest probability to the input instance. It often only requires one pass through the data for estimation, which contributes to its speed and efficiency. Different variations of the NB classifier exist, such as Multinomial Naïve Bayes, Bernoulli Naïve Bayes, and Gaussian Naïve Bayes, among others. Multinomal NB is well suited for data with discrete features, Bernoulli NB is well suited for datasets with Boolean features, and Gaussian NB, the type we use in our experiments, is well suited to datasets with continuous numeric features and assumes the features have a Gaussian distribution. Another advantage to NB is that it is inherently interpretable, like DT. NB does have its disadvantages. One significant disadvantage is the strong conditional independence of the features. This may not always reflect real-world data and can potentially negatively affect classification performance. NB was chosen for our study due to its suitability to a large array of machine learning tasks and because it is a widely understood and known classifier.

The sixth, and final, supervised classifier we use with the synthesized class labels is a multilayer perceptron (MLP). An MLP is a type of artificial neural network that consists of at least three layers: an input layer, one or more hidden layers, and an output layer. It is the most basic architecture type for NNs. Each node, also referred to as a neuron or perceptron, in a layer is connected to one or more nodes in the next layer. A fully connected network, the MLP we use is fully connected, is where every node in one layer is individually connected to every node in the next. Thes types of NN are known as feed forward networks. Input data moves from the input layer through the hidden layers and finally to the output layer, where the final numeric prediction or classification is produced. An MLP is capable of approximating non-linearly separable data. This makes it well suited for complex machine learning tasks. The neurons in the hidden layer use a non-linear activation function, such as a sigmoid function or a rectified linear unit (ReLu) function. The connections between neurons are a combination of numeric weights and are incrementally changed during the training process when the MLP is fitted to the training data. How specifically the weights are modified is dependent on the type of optimization algorithm used during training. We use backpropagation and a gradient descent optimization method to train the MLP used in our experiments. Backpropagation computes the gradient of the loss function, or error, for a given input–output pair. The gradients are used to update the weights to minimize the loss function. This process is repeated for all instances in the training data and is repeated for several epochs. An epoch is completed after one pass through the training data is completed. A sufficient size and number of hidden layers is what gives an MLP its performance. However, excessive number of neurons or layers, relative to the complexity of the classification task or training data, can lead to overfitting.

To more completely evaluate the performance of the supervised classifiers trained on our method’s synthetized labels, we compare the classification performance of an unsupervised anomaly detection method as a baseline. This baseline represents the expected performance in finding fraudulent instances when learning from new data that does not have class labels. We use a popular outlier or anomaly detection method as our baseline model, namely isolation forest (IF), originally introduced by Liu et al. in

[31]. Like many of our supervised learners, IF is a tree-based algorithm and as its name suggests, it attempts to isolate anomalies from the rest of the dataset. IF accomplishes this by recursively partitioning the training data into smaller and smaller sets. The main premise behind this is anomalies are inherently "few and different" [31]. Thus, anomalous instances have a higher probability of being isolated by the algorithm than non-anomalies. During the recursive process, data-induced random trees are created until all instances are isolated. Anomalies then have tree paths than non-anomalies [31] for two main reasons: (1) anomalous datasets are inherently class imbalanced and anomalies are fewer in number than normal ones; and (2) instances are more likely to be separated earlier in the recursive process if they have separable attributes. The IF algorithm creates a forest of trees. Instances that have shorter paths across many trees are more likely to be anomalous. IF is a good baseline to compare to since it is widely used and has been shown to outperform RF and local outlier factor on a highly imbalanced Medicare fraud detection dataset [32]. When IF determines an instance is anomalous, that is equivalent to classifying that instance to the minority class. Thus, for our experiments, we train an IF using only the dataset features and measure its classification performance using the original labels of the dataset. IF does not use the method's newly synthesized labels. For our experimental analysis, we only consider model performance. We don't consider other factors such as required computational time, model interpretability, or other non-classification-performance measures.

For all six classifiers used to evaluate our synthesized labels, as well as the baseline model we compare them to, IF, we use the implementation provided in Scikit-learn [33], version 1.3.0. We use defaults values for DT. This consists of using the Gini criterion for impurity and best splits are chosen (as opposed to random splits). The DT does not have a predetermined maximum tree depth which results in nodes that are expanded until all leaf nodes are pure or until all leaf nodes contain less than two samples. The minimum samples per leaf node is set to 1 and the minimum number of samples required to split an internal node is set to 2. The DT will use all consider all available features when looking for the best split. Our RF used consists of 100 trees in the forest. It uses the Gini criterion, a minimum of 2 samples to split an internal node, and the minimum number of samples per leaf node is 1, like the configuration for DT. The difference between the individual trees in our RF and the DT is the RF consists of trees no larger than 4. The ET configuration used also has 100 trees in the forest. It also uses the Gini configuration, minimum samples per leaf node set to 1, and minimum samples to split set to 2, however the maximum depth is set to 8. The parameters for LR are all kept to the library defaults. For NB, we use the *GaussianNB* variant in the library and set all parameters to their defaults. The MLP architecture consists of one hidden layer with 100 neurons, uses the ReLu activation function [34], and the *Adam* algorithm for weigh optimization, which is type of stochastic gradient descent [35]. The log-loss function is used as the loss function. The MLP is trained using a batch size of 200, and a constant learning rate, for 300 epochs with default settings for early stopping. IF has only one parameter [31], namely the contamination rate. This value was set to the dataset's expected fraud rate, provided by domain expertise.

Any other parameters not explicitly stated were left as the library defaults. This was chosen for ease of comparison. Additionally, we aim to evaluate the quality of the synthesized labels and their classification performance across several different supervised learners. Finding the optimal settings for a given learner is out of scope for this paper, hence, we do not perform any hyperparameter tuning on for the classifiers. We use 5-fold cross validation, where the training splits have the synthesized labels and the test splits use the original class label, when evaluating the classifiers. We repeat this for 10 rounds each to provide a basis for statistical comparison and to avoid any potential abnormally high or low classification performance. Any AUPRC value shown in a table is the average value across the fifty trials.

Experimental results

We first apply the methodology to the credit card fraud detection dataset in order to measure the effectiveness of the newly synthesized labels. First, we train the autoencoder on the dataset to calculate the reconstruction error for every row and sort from greatest to least. Next, we set the method's first input to produce 500 instances in the positive class, i.e., it labels 500 instances as fraud. The 500 instances at the top, which have the highest reconstruction error, are labeled. Labeling 500 as positive was selected by domain expertise. The other method's other input parameter is then set to determine what number of instances in the majority, that the method is most uncertain of, are to be labeled with the positives. For our experiments and results in the tables, P is the total number of positive-labeled instances for that experiment. For example, when $P = 700$, this means a total of 700 instances are labeled as positive which is comprised of 500 set by the first parameter, and an additional 200 instances from the majority that were added to the positive class. For our experiments, we set P to: 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000, 2100, 2200, 2300, 2400, and 2500. During experimentation, we observed that the AUPRC performance of the learners started to reach their peak at around 1500 positive-labeled instances. We chose to include up to $P = 2500$ since for most learners, the AUPRC started to drop off after 1500 and including up to 2500 shows a more complete picture of how the number of synthetic positive labels affects supervised classification performance. Increasing P beyond 2500 does not contribute to a better understanding of the quality of the new labels and it would show unnecessary results. In total, 21 P values are examined, each repeated for 10 replications, which results in 210 experimental trials.

Though the method exclusively uses the dataset features to generate class labels, it is important to note that the original class labels, as provided by the original dataset, are maintained for each row. These original labels, or ground truth labels, are used only to calculate the AUPRC score for the supervised classifiers trained on using the synthesized labels. This is implemented in the K-fold cross validation steps. We use 5-fold cross validation which results in 5 splits so that there can be 5 different folds using 80% of the data for training and 20% for the test split. We used the stratified K-fold method which attempts to maintain the class imbalance ratio between the splits using the original class labels. This is an important consideration since the unseen test split should have the same distribution and characteristics as the overall dataset. This ensures a test split that is representative of the overall population of the dataset. The data in the training folds use the synthesized labels and the test folds have the original labels. This is repeated for all six supervised classifiers. We use 5-fold cross validation when measuring the AUPRC for IF and all six supervised classifiers. However, in the case of IF, we do not use the synthesized labels since IF is an unsupervised anomaly detection method and does not use labels during training but it does use the ground truth labels for the test split and performance evaluation.

We use one-factor analysis of variance (ANOVA) as a test to determine if there are statistical differences between the various metrics we compare. We compare the mean AUPRC score vs. all levels of positive-labeled instances, or *P*, the one factor. As can be observed in Table 3, for all six classifiers the number of positive instances significantly influences AUPRC, as indicated by the low p-value of less than *2e–16* in most cases. Therefore, we can reject the null hypothesis and conclude that there are statistically significant differences between the mean AUPRC values. We also perform Tukey's honestly significant difference (HSD) test to rank our results as needed [36, 37]. The Tukey HSD test is a widely used pairwise comparison test used following ANOVA to determine which groups' means differ from each other. Here, the mean is the average AUPRC value across the 5-folds and 10 replications, i.e., the mean across 50 different folds. We apply ANOVA and the Tukey HSD test to the AUPRC score across the various different levels of *P* to measure if adding some of the most uncertain instances to the positive instances changes classification performance. The outcome of a Tukey test on a pair is a range of confidence intervals and if the interval includes zero, it would suggest that the two respective groups do not significantly differ from each other. As an example, if one mean belongs to group "a" and another belongs to group "b", they are statistically significantly different. However, if one mean is in group "a" and another, yet different, mean is also in group "a", they are not statistically significantly different. Further, if there are multiple overlapping letters, groups that have the same letter in them do not differ statistically but groups that do not share similar letters are statistically different. For example, if there are three groups, "a", "ab", and "b", the first group and second group are not statistically different, and the second and third group are not statistically different, however, the first and third group are statistically different. We provide an in-depth analysis of the AUPRC performance across varying levels of P. Note that even though we provide comparisons across supervised learners using our labeling approach, this paper's focus is on the efficacy of our novel method and the synthesized labels, not an in-depth analysis of the supervised methods themselves.

We examine the AUPRC scores for all models and across all levels of positive labels. As mentioned previously, AUPRC scores are the main performance indicators that

Table 3 ANOVA table for AUPRC across all levels of *P*

Classifier	Df	Sum Sq	Mean SQ	F value	Pr(>F)	Df	Sum Sq	Mean SQ
	Positive labels					Residuals		
DT	20	1.0176	0.05088	63.52	< 2e−16	1029	0.8242	0.0008
RF	20	0.2861	0.014306	7.771	< 2e−16	1029	1.8943	0.001841
ET	20	0.1707	0.008534	4.782	2.99E−11	1029	1.8362	0.001784
LR	20	0.8621	0.0431	24.44	< 2e−16	1029	1.8149	0.00176
NB	20	0.0288	0.001441	3.251	1.96E−06	1029	0.4561	0.000443
MLP	20	0.9528	0.04764	27.41	< 2e−16	1029	1.7883	0.00174

Table 4 Isolation forest performance results

Isolation forest	
P	**AUPRC**
N/A	0.2414

Table 5 Average classifier AUPRC performance results with respect to number of positive-labeled instances *P*

AUPRC						
P	**DT**	**RF**	**ET**	**LR**	**NB**	**MLP**
500	0.2935	0.3394	0.3357	0.2301	0.3693	0.1904
600	0.3008	0.3404	0.3491	0.2355	0.3717	0.1935
700	0.3029	0.3493	0.3569	0.2607	0.3735	0.1962
800	0.3111	0.3470	0.3574	0.2610	0.3779	0.2090
900	0.3146	0.3607	0.3649	0.2759	0.3791	0.2198
1000	0.3181	0.3529	0.3670	0.2793	0.3814	0.2345
1100	0.3292	0.3523	0.3675	0.2818	0.3814	0.2476
1200	0.3363	0.3487	0.3613	0.2831	0.3826	0.2409
1300	0.3542	0.3602	0.3795	0.3074	0.3842	0.2576
1400	0.3565	0.3608	0.3806	0.3108	0.3837	0.2517
1500	0.3714	0.3596	0.3842	0.3226	0.3833	0.2692
1600	0.3751	0.3615	0.3832	0.3241	0.3838	0.2685
1700	0.3774	0.3597	0.3859	0.3256	0.3852	0.2899
1800	0.3801	0.3623	0.3834	0.3219	0.3851	0.2763
1900	0.3818	0.3547	0.3825	0.3220	0.3861	0.2825
2000	0.3764	0.3491	0.3741	0.3118	0.3860	0.2766
2100	0.3737	0.3393	0.3712	0.3108	0.3860	0.2735
2200	0.3767	0.3328	0.3683	0.3117	0.3866	0.2688
2300	0.3758	0.3221	0.3650	0.3106	0.3869	0.2672
2400	0.3763	0.3125	0.3592	0.3093	0.3879	0.2651
2500	0.3813	0.3009	0.3616	0.3191	0.3893	0.2659

should be primarily considered. AUPRC provides a better metric in the context of high class imbalance. Table 5 shows the AUPRC scores for all *P* values. We can compare that to the IF's AUPRC score of 0.2414, shown in Table 4. When P=500, or when no additional instances are added to the positive class, our method outperforms IF for DT, RF, ET, and NB. The LR and MLP classifiers underperform the baseline IF by a small amount with AUPRC scores of 0.2301 and 0.1904, respectively. As the *P* increases, all 6 learners show very similar trends; AUPRC scores increase as *P* starts to increase. Once there are 700 total positive instances, LR outperforms IF by a significant margin with 0.2607 AUPRC. For the MLP classifier, it takes longer for it to outperform IF. For this learner, it requires 1100 positive instances in its training data for it to first outperform IF with an AUPRC of 0.2476. It then achieves its best AUPRC of 0.2899 at 1700 positive instances. This makes MLP the worst performing supervised classifier in our work. Though with enough positive labels, our method produces labeled data that is of sufficient quality to outperform the baseline.

We used three different tree-based classifiers in our experiments, DT, RF, and ET. DT is the simplest classifier of these three and is a building block used by the forests in RF and ET. When learning from 500 positive labels, DT has an AUPRC of 0.2935 and underperforms both RF and ET, 0.3394 and 0.3357 AUPRC respectively. This shows that the added complexity of the RF and ET classifiers are better able to capture the patterns in the data for fraud detection when the fewest number of positives are used. All three of these significantly outperform IF. However, when *P* increases so do the AUPRC scores for DT, RF and ET. This pattern can be seen in both Tables 5 and 6. In general, the AUPRC increases with *P* up until a certain point, and then the AUPRC values level off and increasing *P* no longer increases AUPRC. When increasing *P* to a minimum of 1700 positives, DT performs nearly as good as the more complex ET model and outperforms the RF model. RF and ET aren't computationally prohibitive, but they do take more time and resources to fully train than the simpler DT classifier. Using less computational resources can suggest that our method used with DT would be a wholistically better choice, but we don't use overall training time as a factor. Thus, we would say the model with the outright highest performance metric is the best choice. Even though in practice, other factors may be of consideration.

As can be seen in Table 6 when DT uses 1700, 1800, 1900, and 2500 positive instances, the model achieves its highest AUPRC performance. All three of these belong in Group 'a', which indicates their AUPRC values are not statistically different. This is also the case for RF and ET. For both of these classifiers, using 1700 positive instances produces an

Table 6 Tukey HSD test for decision tree, random forest, and extra trees, ordered by AUPRC

DT			RF			ET		
P	AUPRC	Group	*P*	AUPRC	Group	*P*	AUPRC	Group
1900	0.3818	a	1800	0.3623	a	1700	0.3859	a
2500	0.3813	a	1600	0.3615	a	1500	0.3842	a
1800	0.3801	a	1400	0.3608	a	1800	0.3834	a
1700	0.3774	a	900	0.3607	a	1600	0.3832	a
2200	0.3767	ab	1300	0.3602	a	1900	0.3825	a
2000	0.3764	ab	1700	0.3597	a	1400	0.3806	a
2400	0.3763	ab	1500	0.3596	a	1300	0.3795	a
2300	0.3758	ab	1900	0.3547	a	2000	0.3741	ab
1600	0.3751	ab	1000	0.3529	a	2100	0.3712	ab
2100	0.3737	abc	1100	0.3523	ab	2200	0.3683	ab
1500	0.3714	abc	700	0.3493	ab	1100	0.3675	ab
1400	0.3565	bcd	2000	0.3491	ab	1000	0.3670	ab
1300	0.3542	cd	1200	0.3487	ab	2300	0.3650	abc
1200	0.3363	de	800	0.3470	ab	900	0.3649	abc
1100	0.3292	ef	600	0.3404	abc	2500	0.3616	abc
1000	0.3181	efg	500	0.3394	abc	1200	0.3613	abc
900	0.3146	fg	2100	0.3393	abc	2400	0.3592	abc
800	0.3111	fgh	2200	0.3328	abc	800	0.3574	abc
700	0.3029	gh	2300	0.3221	bcd	700	0.3569	abc
600	0.3008	gh	2400	0.3125	cd	600	0.3491	bc
500	0.2935	h	2500	0.3009	d	500	0.3357	c

optimal AUPRC score. Similar to DT, 1700 is in the highest Group 'a'. It is important to note that in the Tukey HSD Table 6, the results are sorted from highest AUPRC score to lowest. Unlike DT, RF achieves an optimal AUPRC score with as few as 900 positive instances but is a lower AUPRC than DT. ET achieves an optimal AUPRC score with as few as 1300 positive instances and it is a higher AUPRC score than RF and DT. This shows that the combination of the ET algorithm and our method with a minimum of 1300 positive instances produces the best AUPRC score for the tree-based classifiers in our study.

Similar patterns can be observed for the other three learners, LR, NB, and MLP. Generally, as P increases, so do their respective AUPRC scores. Additionally, when training with 1700 positive instances, LR, NB, and MLP achieve optimal AUPRC scores. This can be seen in Table 7 where LR and MLP have the highest AUPRC score with 1700 positives and NB's AUPRC for 1700 is in Group 'abc'. Group 'a' and Group 'abc' both contain the letter 'a'. Thus, they are not statistically different groups. I.e., though NB achieves its highest AUPRC score of 0.3893 with 2500 positive instances, its performance isn't dissimilar to its AUPRC of 0.3852 it gets with 1700 positive labels. NB does exhibit similar patterns as the others, where its AUPRC generally increases with P, but its rate of increase is slower than the others. Additionally, its lowest AUPRC score, training with 500 positive instances, is higher than the best scores from RF, LR, and MLP. Since the rate of AUPRC increase is less than the others, between $P = 500$ and $P = 2500$, we can determine that increasing moving instances from the majority to the minority has a

Table 7 Tukey HSD test for logistic regression, Naïve Bayes, and MLP, ordered by AUPRC

LR			NB			MLP		
P	AUPRC	Group	*P*	AUPRC	Group	*P*	AUPRC	Group
1700	0.3256	a	2500	0.3893	a	1700	0.2899	a
1600	0.3241	a	2400	0.3879	ab	1900	0.2825	ab
1500	0.3226	a	2300	0.3869	ab	2000	0.2766	abc
1900	0.3220	a	2200	0.3866	abc	1800	0.2763	abc
1800	0.3219	a	1900	0.3861	abc	2100	0.2735	abc
2500	0.3191	a	2100	0.3860	abc	1500	0.2692	abcd
2000	0.3118	ab	2000	0.3860	abc	2200	0.2688	abcd
2200	0.3117	ab	1700	0.3852	abc	1600	0.2685	abcd
2100	0.3108	ab	1800	0.3851	abc	2300	0.2672	abcd
1400	0.3108	ab	1300	0.3842	abcd	2500	0.2659	abcd
2300	0.3106	ab	1600	0.3838	abcd	2400	0.2651	abcd
2400	0.3093	abc	1400	0.3837	abcd	1300	0.2576	bcde
1300	0.3074	abc	1500	0.3833	abcd	1400	0.2517	cde
1200	0.2831	bcd	1200	0.3826	abcd	1100	0.2476	cdef
1100	0.2818	bcd	1000	0.3814	abcd	1200	0.2409	def
1000	0.2793	cd	1100	0.3814	abcd	1000	0.2345	efg
900	0.2759	d	900	0.3791	abcd	900	0.2198	fgh
800	0.2610	de	800	0.3779	abcd	800	0.2090	gh
700	0.2607	de	700	0.3735	bcd	700	0.1962	h
600	0.2355	ef	600	0.3717	cd	600	0.1935	h
500	0.2301	f	500	0.3693	d	500	0.1904	h

smaller effect than it does with all other classifiers. Additionally, NB achieves the highest AUPRC score of 0.3893 which is marginally higher than ET's at 0.3859. We can conclude that all supervised classifiers effectively train on our method synthesized labels. Most classifiers outperform the baseline IF learner without any additional positive labels, but all six have increasing AUPRC performance up until the 1700 mark. At that level of *P*, all six classifiers have an optimal AUPRC performance and any additional increase in *P* doesn't increase performance in a statistically significant way.

Conclusion

This study presents a detailed evaluation of a novel method for synthesizing class labels for highly imbalanced credit card fraud detection data. Large data is commonly without labels and the datasets that do contain labels, can often come at a large financial cost, time cost, and are prone to noise and inaccuracies due to their required human creation. Another significant challenge in machine learning is when data has a high class imbalance which can significantly degrade machine learning classification performance.

This methodology effectively addresses the challenges of large unlabeled data as well as the challenges associated with high class imbalance. To evaluate the method, it is applied to a widely used, publicly available credit card fraud detection dataset. This highly imbalanced dataset has new binary class labels synthesized by our unique method, exclusively leveraging the feature values in an unsupervised manner. Varying levels of *P*, number of positive-labeled instances created by the method, were applied to capture more actual fraud instances in the training data. These new binary class labels are then used to train supervised classifiers for fraud detection. Our empirical results show that the synthesized labels are effective, and are of high enough quality, to enable supervised learning, on otherwise unlabeled data, that achieves higher classification performance than a widely used unsupervised anomaly detection method. The results show that all six supervised classifiers in our study, at most all levels of *P*, significantly outperform IF. Further, the results show that increasing *P* captures more trainable information and is reflected in the significant improvement of the supervised classifiers trained on the synthesized labels, as measured by AUPRC. Increasing the *P* is not strictly necessary to outperform a baseline for all supervised classifiers, however, it has its clear advantages in improving classification performance. We can conclude that this method effectively learns from imbalanced dataset features to produce usable class labels for credit card fraud detection. Future work includes evaluating the method with other large and highly imbalanced datasets in other domains and exploring algorithmic ways to choose *P*.

Abbreviations

ANOVA	Analysis of variance
AUC	Area under the receiver operating characteristic curve
AUPRC	Area under the precision-recall curve
DT	Decision tree
ET	Extra trees
FN	False negative
FP	False positive
FPR	False positive rate
HSD	Tukey's honestly significant difference
IF	Isolation forest
LR	Logistic regression
MLP	Multilayer perceptron
MSE	Mean squared error

NB	Naïve Bayes
NN	Neural network
PCA	Principal component analysis
PR	Precision-recall
ReLu	Rectified linear unit
RF	Random forest
ROC	Receiver operating characteristic curve
TN	True negative
TP	True positive
TPR	True positive rate

Acknowledgements
The authors would like to thank the reviewers in the Data Mining and Machine Learning Laboratory at Florida Atlantic University.

Author contributions
RKLK carried out the concept and design of the research, performed the implementation and experimentation, and drafted the manuscript. FV, TMK and ZS provided reviews on the manuscript. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 26 December 2023 Accepted: 19 February 2024
Published online: 09 March 2024

References

1. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. Imagenet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE. 2009. p. 248–55.
2. Zhang C, Bengio S, Hardt M, Recht B, Vinyals O. Understanding deep learning (still) requires rethinking generalization. Commun ACM. 2021;64(3):107–15.
3. Halevy A, Norvig P, Pereira F. The unreasonable effectiveness of data. IEEE Intell Syst. 2009;24(2):8–12.
4. Sun C, Shrivastava A, Singh S, Gupta A. Revisiting unreasonable effectiveness of data in deep learning era. In: Proceedings of the IEEE International Conference on Computer Vision, 2017. p. 843–52.
5. Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, Van Der Laak JA, Van Ginneken B, Sánchez CI. A survey on deep learning in medical image analysis. Med Image Anal. 2017;42:60–88.
6. Xie J, Girshick R, Farhadi A. Unsupervised deep embedding for clustering analysis. In: International Conference on Machine Learning, PMLR. 2016. p. 478–87.
7. Wang S, Yao X. Multiclass imbalance problems: analysis and potential solutions. IEEE Trans Syst Man Cybern Part B. 2012;42(4):1119–30.
8. Krawczyk B. Learning from imbalanced data: open challenges and future directions. Prog Artif Intell. 2016;5(4):221–32.
9. Khoshgoftaar TM, Seiffert C, Van Hulse J, Napolitano A, Folleco A. Learning with limited minority class data. In: Sixth International Conference on Machine Learning and Applications, 2007. ICMLA. IEEE. 2007. p. 348–53.
10. Dal Pozzolo A, Caelen O, Johnson RA, Bontempi G. Calibrating probability with undersampling for unbalanced classification. In: 2015 IEEE Symposium Series on Computational Intelligence, IEEE. 2015. p. 159–66.
11. Kaggle: Credit Card Fraud Detection. 2018. https://www.kaggle.com/mlg-ulb/creditcardfraud. Accessed 15 Nov 2023.
12. Leevy JL, Johnson JM, Hancock J, Khoshgoftaar TM. Threshold optimization and random undersampling for imbalanced credit card data. J Big Data. 2023;10(1):58.
13. Leevy JL, Hancock J, Khoshgoftaar TM, Abdollah Zadeh A. Investigating the effectiveness of one-class and binary classification for fraud detection. J Big Data. 2023. https://doi.org/10.1186/s40537-023-00825-1.
14. Baek S, Kwon D, Suh SC, Kim H, Kim I, Kim J. Clustering-based label estimation for network anomaly detection. Digit Commun Netw. 2021;7(1):37–44.

15. Moslehi F, Haeri A, Gholamian MR. A novel selective clustering framework for appropriate labeling of clusters based on k-means algorithm. Sci Iran. 2020;27(5):2621–34.
16. Maqbool O, Babri HA. Automated software clustering: an insight using cluster labels. J Syst Softw. 2006;79(11):1632–48.
17. Rauber A. Labelsom: On the labeling of self-organizing maps. In: IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339), vol. 5, IEEE. 1999. p. 3527–32.
18. Kohonen T. The self-organizing map. Proc IEEE. 1990;78(9):1464–80.
19. Kennedy RK, Salekshahrezaee Z, Khoshgoftaar TM. A novel approach for unsupervised learning of highly-imbalanced data. In: 2022 IEEE 4th International Conference on Cognitive Machine Intelligence (CogMI), IEEE. 2022. p. 52–8.
20. Wan Z, Zhang Y, He H. Variational autoencoder based synthetic data generation for imbalanced learning. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE. 2017. p. 1–7.
21. Kennedy RK, Salekshahrezaee Z, Khoshgoftaar TM. Unsupervised anomaly detection of class imbalanced cognition data using an iterative cleaning method. In: 2023 IEEE 24th International Conference on Information Reuse and Integration for Data Science (IRI), IEEE. 2023. p. 303–8.
22. Ng A, et al. Sparse autoencoder. CS294A Lecture notes 72(2011), 2011. p. 1–19.
23. Chollet F, et al. Keras. 2015. https://keras.io. Accessed 21 Dec 2021.
24. Leevy JL, Khoshgoftaar TM, Hancock J. Evaluating performance metrics for credit card fraud classification. In: 2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE. 2022. p. 1336–41.
25. Provost FJ, Fawcett T, et al. Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. KDD. 1997;97:43–8.
26. Davis J, Goadrich M. The relationship between precision-recall and roc curves. In: Proceedings of the 23rd International Conference on Machine Learning, 2006. p. 233–40.
27. Wang H, Liang Q, Hancock JT, Khoshgoftaar TM. Enhancing credit card fraud detection through a novel ensemble feature selection technique. In: 2023 IEEE 24th International Conference on Information Reuse and Integration for Data Science (IRI), IEEE. 2023. p. 121–6.
28. Hancock JT, Khoshgoftaar TM, Johnson JM. Evaluating classifier performance with highly imbalanced big data. J Big Data. 2023;10(1):42.
29. Breiman L. Random forests. Mach Learn. 2001;45:5–32.
30. Geurts P, Ernst D, Wehenkel L. Extremely randomized trees. Mach Learn. 2006;63:3–42.
31. Liu FT, Ting KM, Zhou Z-H. Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining, IEEE. 2008. p. 413–22.
32. Bauder RA, da Rosa R, Khoshgoftaar TM. Identifying medicare provider fraud with unsupervised machine learning. In: 2018 IEEE International Conference on Information Reuse and Integration (IRI), IEEE. 2018. p. 285–92.
33. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
34. Fukushima K. Cognitron: a self-organizing multilayered neural network. Biol Cybern. 1975;20(3–4):121–36.
35. Kingma DP, Ba J. Adam: a method for stochastic optimization. 2014. arXiv:1412.6980.
36. Abdi H, Williams LJ. Tukey's honestly significant difference (HSD) test. In: Salkind Neil, editor. Encyclopedia of research design. Thousand Oaks: Sage; 2010. p. 1–5.
37. Berenson M, Levine D, Goldstein M. Intermediate statistical methods and applications: a computer package approach. Englewood Cliffs: Prentice-Hall; 1983.

Publisher's Note