Journal of Big Data

METHODOLOGY Open Access

Iterative cleaning and learning of big highly-imbalanced fraud data using unsupervised learning

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

Fraud datasets often times lack consistent and accurate labels, and are characterized by having high class imbalance where the number of fraudulent examples are far fewer than those of normal ones. Machine learning designed for effectively detecting fraud is an important task since fraudulent behavior can have significant financial or health consequences, but is presented with significant challenges due to the class imbalance and availability of reliable labels. This paper presents an unsupervised fraud detection method that uses an iterative cleaning process for effective fraud detection. We measure our method performance using a newly created Medicare fraud big dataset and a widely used credit card fraud dataset. Additionally, we detail the process of creating the highly-imbalanced Medicare dataset from multiple publicly available sources, how additional trainable features were added, and how fraudulent labels were assigned for final model performance measurements. The results are compared with two popular unsupervised learners and show that our method outperforms both models in both datasets. Our work achieves a higher AUPRC with relatively few iterations across both domains.

Keywords: Unsupervised learning, Fraud detection, High class-imbalance, Credit card fraud, Medicare fraud, Big data

Introduction

Fraud detection is the task of distinguishing fraudulent behavior from normal behavior. It is a critical aspect in many real-world applications such as fraudulent credit card transactions, identity fraud such as identity theft, or healthcare insurance fraud. Fraud detection is a type of anomaly detection and, fundamentally, an anomaly is something that is dissimilar to the rest of the data and occurs at a significantly smaller rate. In general, fraud datasets can lack consistent and accurate labels and are class imbalanced.

An example of a real word fraud detection task would be in Medicare fraud detection. Medicare is a federal health insurance for people 65 years or older in the US. The Federal Bureau of Investigation (FBI) has estimated that 10% of all Medicare related billings are fraudulent [1]. For instance, healthcare billing fraud is when a medical provider bills the insurance for a more complex, and costly, procedure than what was performed [2]. This type of fraud has obvious financial consequences but can also pose additional risk to the patients if medically unnecessary procedures are completed in order to fraudulently bill insurance. Addressing the issue of Medicare fraud faces challenges. Healthcare data has big data challenges [3], feature dimensionality challenges [4], and has a high degree of class imbalance, which poses its own challenges [5]. We present a new Medicare Part D big dataset based on several publicly available sources. We enrich the dataset by increasing the number of trainable features and prepare it with class labels for fraud detection. To the best of our knowledge, this is the first time a Medicare dataset has been prepared in this fashion and has been used in a study.

In the binary classification problem, class imbalance occurs when one class contains significantly more instances than the other. Fraud datasets fall into this category since fraudulent, or minority, examples are few and far between. In many cases these are of the most interest [6–8]. However, learners tend to over fit to the majority group and over classify it as a result of the majority class's increased prior probability. Furthermore, the performance metric used needs careful consideration. Some metrics, such as the Area Under the Receiver Operating Characteristics (ROC) Curve (AUC) and accuracy, can be misleading when working with highly imbalanced data [9, 10]. Class labels themselves also present some challenges. For example, labeling requires additional effort beyond recording subject data, such as human labeling, which is relative slow, costly, and can be error prone. Class label noise is another issue presenting its own challenges [11]. An unsupervised approach is immune to these since it entirely relies on the data features and does not use or need class labels.

Our approach presented in this paper is motivated by the fact that we do not want to use labels when looking for fraud and want to iteratively clean the training data. The procedure iteratively cleans a training dataset using an anomaly score by re-training an underlying learner each time. By iteratively cleaning the data of the fraudulent class, the goal is to produce a trained model that efficiently learns the distribution of non-fraudulent data. This can then in turn be used to identify fraudulent data effectively because a fraudulent example would produce a relatively high error. Our approach presented in this work uses an autoencoder as the underlying learner, so a fraudulent observation passed into a trained model would have high reconstruction error and a non-fraudulent instance would have a low reconstruction error. From here we can label which instances in the unseen test data are likely to be fraud or otherwise. This allows our approach to wholly adhere to the unsupervised paradigm. We use two big datasets in this work, a Medicare fraud dataset and a credit card fraud dataset. The credit card dataset consists of labeled credit card transactions where the majority of examples represent normal credit card usage, making it class imbalanced. Though both of these datasets are labeled, the training process is entirely unsupervised. Labels are only used to measure the effectiveness of our approach. We compare our method to two widely used unsupervised learners, namely Isolation Forest and Copula-Based Outlier Detector (COPOD). Our results show that we outperform the existing learners. To the best of our knowledge, our work is the first to combine an unsupervised autoencoder with an error based iterative cleaning process for highly-imbalanced anomaly or outlier detection.

The remainder of this paper is organized as follows. "Related works" section provides a review of related works in the context of anomaly or outlier detection and highlights where there are gaps in the existing research and how our research fits in. "Methodology" section is where we detail the steps for our methodology and gives an overview of the different machine learning models used in this work. "Datasets" section gives details on how the newly prepared Medicare Part D big dataset was created and overviews the existing dataset we use. "Results" section discusses the performance metrics presented and details our experimental results on both datasets. The last section concludes the paper and discusses potential avenues for future work.

Related works

There are various existing categories of approaches for machine learning in the context of anomaly or outlier detection: supervised learning, unsupervised learning, semi-supervised learning, and hybrid approaches. Supervised learning consists of learning from labeled examples in the dataset that represent input–output pairs. A significant challenge that arises when using supervised classifiers is created by class imbalance. Class label imbalance, especially when is highly class imbalanced, significantly degrades the performance of machine learning [12, 13]. Unsupervised approaches rely on learning patterns and distributions in the features of the datasets without using any class label. Such a machine learning model is immune to the challenges derived from class labels, such as high class-imbalance or class label noise. Semi-supervised learning approaches utilize small set of labeled instances and a comparatively large number of unlabeled samples. With respect to binary labeled data, the labeled set can include positive and negative labels, only negative labels, or only positive labels. Hybrid approaches use a combination of any of the above, such as combining an unsupervised and supervised techniques. Carcillo et al. [14] has shown a combination of unsupervised and supervised techniques can be used for fraud detection.

Srivastava et al. [15] demonstrated in their study the effectiveness of using a Hidden Markov Model (HMM) to identify credit card fraud. They trained the HMM using a dataset that consisted of only normal credit card behavior, i.e. non-fraudulent, and subsequently used it to detect fraud in new credit card transactions. This is similar to one class classifiers (OCC) in that it trains on data that has been filtered on having a non-fraud class label or trained on a dataset that is assumed to belong to one class [16]. Requiring large assumptions about the labels, or using them for filtering the training set, does not make this an unsupervised method from start to finish.

Liu et al. introduced EasyEnsemble (EE) in [17, 18], which they applied to an imbalanced fraud detection problem. Although EE is trained in an unsupervised manner and then used for undersampling imbalanced data, our method differs in that EE requires a supervised preprocessing step in which they sample exclusively from the non-fraudulent class. Our approach does not require any supervised step, pre-processing, or otherwise.

Zong et al. [19] employed a combination of deep autoencoding and a Gaussian mixture model to detect anomalies in datasets that have significantly less class imbalance than the datasets we use in this paper. Like Liu et al. [17], Zong et al. filtered their training dataset by class label before training, assuming their data is clean. They use random undersampling in a preprocessing step to generate a balanced dataset that their models are directly trained on. Their study demonstrated that the autoencoder projects instances into a low-dimensional space while still preserving the necessary information for effective anomaly detection. Using a network attack dataset, Pu et al. [20] introduced an unsupervised anomaly detection method combining sub-space clustering and a one-class support vector machine (SVM). Similar to [19], Pu also utilized class labels in a pre-processing step to divide their data into various subsets. As such, this work is not completely unsupervised from start to finish.

In another study, Maleki et al. [21] used a long short-term memory (LSTM) autoencoder for unsupervised anomaly detection. Their work focused on an industrial gas turbine dataset and a dataset describing CPU utilization of Amazon EC2 instances, both time series datasets. Their methodology made the assumption that a substantial portion of the initial training data is free of anomalies. Specifically, the two types of temporal datasets represent systems that have been operating for a sufficient amount of time with relatively few anomalies. Their work shows that an LSTM autoencoder can be effective in anomaly detection.

Pang et al. [22] used an iterative learning via self-training. Their work aimed to iteratively improve their anomaly detection model; however, their work used a semi-supervised learning method. In their iterations they train a model using a small, labeled subset of the data and then apply their trained model on the rest of the unlabeled data to generate more reliable labels. Beggel et al. [23] used Adversarial Autoencoders for anomaly detection in images. In their work, they refine their training data using an iterative step. However, they are using a one-class SVM to determine potential anomalies by applying it to the autoencoder's lower dimensional representations of the input images. For this to work, either one of the following is required: the one-class SVM requires a hyperparameter that represents the expected upper bounds of the expected fraction of anomalies, or the assumption that the instances the one-class SVM is working with are from one class.

It is important to note that our approach wholly adheres to the definition of unsupervised training from start to finish. Some of the related works above assume the training data used for anomaly detection is anomaly or outlier free. Others stated the model training is done in an unsupervised fashion on a dataset that was filtered in a supervised way. Our work differs than the works above in that we neither prefilter nor make large assumptions of the anomalous qualities of our datasets. Additionally, datasets used in our work have a significantly higher level of class imbalanced than the works reviewed.

Methodology

Our goal is to produce an effective fraud detection model that learns from unlabeled highly-imbalanced big data. The defining characteristic of our approach is it learns from the unlabeled data through an iterative training process where our procedure incrementally cleans the training data by sequentially training randomly initialized learners. This work uses a fully connected Autoencoder as the underlying learner, however, our approach is flexible, and it is possible that any unsupervised machine learning model that outputs an anomaly score can be used.

To begin, a randomly initialized autoencoder is trained on the entire training data and, after the training process is completed, an anomaly score is calculated for each instance. Using these values, we calculate an error threshold (detailed in "Error thresholding" section). Values above this threshold have high anomaly scores and values below this threshold have low anomaly scores. High anomaly scores are considered anomalous or fraudulent, and low anomaly scores are considered normal or non-fraudulent. At the end of each iteration, the training data is cleaned of fraudulent instances so that a slightly smaller set of data can be used in the next step. Then, using the cleaned training data a new, a randomly initialized autoencoder is trained, a new error threshold is calculated, and the process repeats itself for a desired number of iterations, as shown in Algorithm 1. Each iteration is training a new, though architecturally identical, autoencoder on increasingly clean data. After each passing iteration, a new autoencoder learns the distribution of increasingly clean data. The final iteration's autoencoder, and error threshold value, is the end product and is used for fraud detection on unseen data. We manually set the number of iterations to 10.

Algorithm 1 Unsupervised Anomaly Detection

```
1:  for Each fold in 5-fold do
2:      for Each iteration do
3:          Train Learner on train data
4:          $\lambda \leftarrow$ computed error threshold
5:          for all rows in train do
6:              $e \leftarrow GetError(Learner, row)$
7:              Remove instance if instance $e > \lambda$
8:          end for
9:      end for
10:     for all rows in test do
11:         $e \leftarrow GetError(Learner, row)$
12:         if instance $e > \lambda$ then
13:             Assign row to minority class
14:         else
15:             Assign row to majority class
16:         end if
17:     end for
18: end for
19: Average training and test metrics across all folds
```

Algorithm 2 Get Mean Squared Error

```
1: procedure GetError($Learner, Instance$)
2:     Calculate $LearnerMSE$ for $Instance$
3:     Return $MSE$
4: end procedure
```

Autoencoder

Our methodology uses a fully connected autoencoder as its underlying learner. An autoencoder is a type of artificial neural network used for unsupervised learning that learns to encode and decode data using a set of fully connected hidden layers [24]. A layer is fully connected to the next when each of the first layer's neurons is connected to every neuron in the following layer. Autoencoders are made up of two main components: an encoder and decoder. The encoder component maps the input data into a lower-dimensional space then the decoder component maps the lower-dimensional representation back into the high-dimensional space. The training objective of the autoencoder is to minimize the reconstruction error between the input and the output. This allows an autoencoder to be used for several different areas such as dimensionality reduction, image and video processing, natural language processing, and anomaly detection. We use the autoencoder for fraud detection, a type of anomaly detection by using tabular data.

The training process of an autoencoder is similar to that of other neural networks in that during training the parameters, or weights, between neurons are optimized to minimize an objective function. Here the reconstruction error between the input and output layers is minimized with backpropagation, a widely used algorithm that employs gradient methods, such as gradient descent or stochastic gradient descent. An error metric, such as mean squared error (MSE), is used to quantify the reconstruction error between the input and output. The objective is to minimize this error metric. There are various different algorithms that can be used to minimize such as the Adam optimizer or stochastic gradient descent.

Specifically, the architecture of our method's underlying autoencoder stays the same for each iteration and dataset. The input layer has the same number of neurons as the number of features in the dataset it is being applied to. Described in "Datasets" section, the input has a size of 29 and 328 when used for credit card fraud detection and Medicare Part D fraud detection, respectively. The encoder component of the autoencoder consists of two fully connected layers. The first layer has 100 neurons and the second has 50 neurons. Both of these each use the ReLu activation function (rectified linear unit). These are then connected to the decoder components of the architecture. This mirrors the encoder where the next fully connected layer has 50 neurons and the next one after that has 100 fully connected neurons. The layers in the decoder section both use the Tanh activation function. These are followed by the final output layer which uses the ReLu activation function. A visualization of the architecture is shown in Fig. 1. For both datasets in this work, the autoencoder is trained for 50 epochs for each iteration on the 80% training split. We use a learning rate of 0.0001, batch size of 512, the Adam optimizer function, and MSE as our loss function. We chose a

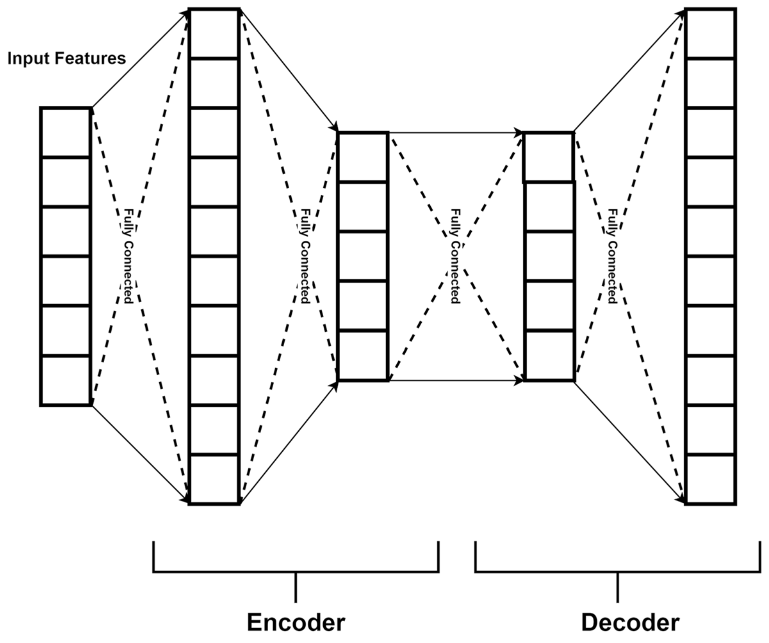

Fig. 1 Autoencoder visualization of the encoder and decoder components

fixed number of epochs after observing at 50 epochs both the training and validation loss had reached an asymptote. Keras, version 2.8.0, was used [25]. One limitation of our approach is it has relatively high training times, as compared to the other two baseline learners. Since we use a fixed number of epochs, this slow training can be mostly attributed to the number of training epochs we perform. Though we observed an asymptote of the training loss, and thus a relative minimum, evaluating different early stopping techniques may prove effective in reducing the overall training time while not negatively affecting performance.

Class imbalance

Class imbalance is a common issue for datasets when the distribution of the classes is imbalanced. Though this definition is applicable to multi-class datasets, we focus on the binary class datasets. However, the concepts still apply to the multi-class problem since it is possible to convert the multi-class problem into a set of binary class problems via class decomposition [26]. In many cases, a dataset can be considered class imbalanced when the ratio of the classes starts from 1:4 up to 1:100 [27]. If each class is well represented, class imbalance does not necessarily reduce model performance. High class-imbalance, a more severe case of class imbalance, is defined as datasets with an imbalance ratio of 1:1000 or greater [28]. These highly class imbalanced datasets pose additional challenges due to the relatively small number of minority samples and the large number of majority samples. Japkowicz et al. [29] show that as the problem complexity increases, its sensitivity to class imbalance also increases.

The experiments presented in this paper are conducted on two imbalanced fraud datasets. In the fraud datasets, the vast majority of the instances represent, and are labeled accordingly, non-fraudulent or normal behavior. There are significantly fewer labeled instances of fraudulent behavior since it can be difficult to accurately identify and label the fraudulent samples. This characteristic occurs in other domains as well such as in detecting oil spills [7]. Learning from highly imbalanced datasets is important because it is the rare events that we are interested in. For instance, we are interested in identifying credit card fraud or Medicare insurance fraud out of the very large number of transactions or claims.

Error thresholding

Our approach uses an error threshold as a cutoff point throughout the methodology. It is used in both the training process and iterative cleaning steps, as well as in the evaluation of the performance. The error threshold is two standard deviations from the mean (λ_{2SD}). $\lambda_{2SD} = \epsilon + 2 \cdot \sigma$, where σ is the standard deviation of the errors of the current iteration, and ϵ is the mean error for the current training iteration. The set of errors are recalculated after each of our method's iterations and thus the λ_{2SD} only changes at each iterative cleaning step. When evaluating our method on an unseen holdout set, at the end of each iteration the same λ_{2SD} is used to categorize the test data as fraud or non-fraud as was calculated in the training step of the same iteration. We use λ_{2SD} because it has been shown to be an effective error thresholding measure [30, 31].

COPOD

We compare our iterative cleaning approach to existing unsupervised anomaly detection methods. One such model is called Copula-Based Outlier Detector (COPOD), first introduced by Li et al. [32]. COPOD, as the name suggests, is an outlier/anomaly detection algorithm that is inspired by copulas for modeling multivariate data distributions. It is parameter-free and is computationally efficient. Copulas are multivariate cumulative distribution functions that enable the COPOD model to separate marginal distributions from a given data distribution. This gives COPOD the flexibility to be used in high dimensional datasets. More specifically, COPOD is based on fitting empirical cumulative distribution functions, called an empirical copula. This allows the COPOD model to be nonparametric. This does however reduce the researcher's ability to fine tune the COPOD model, whereas many other models have one or more tunable parameters. We use the implementation of COPOD which is included in the PyOD python library [33], version 1.0.7. We train on the 80% split, from our fivefold cross validation splits, and use the trained COPOD to predict on our 20% test split. The output from the COPOD test function is binary and indicates whether an instance is predicted to be an outlier or otherwise. We compare to the actual label to measure the model performance for comparison.

Isolation Forest

The second baseline learner we compare our method to is called the Isolation Forest (IF), originally introduced by Liu et al. in [34]. It is a popular unsupervised machine learning model for anomaly or outlier detection. It is a tree-based algorithm that aims to isolate anomalies from the rest of the data by partitioning the data into smaller and smaller subsets. Liu et al. [34] defines isolation to mean "separating an instance from the rest of the instances". Their reasoning was that anomalies are "few and different" and therefore they are more prone to isolation than non-anomalies or outliers. Specifically, an IF is a data-induced random tree where the instances are recursively partitioned until all instances are isolated. They show that the random partitioning produces a shorter path for anomalies (longer paths for non-anomalies) because of two reasons: (1) anomalies are fewer in number by nature and thus result in a smaller number of partitions; and (2) instances that have separable attribute values are more likely to be separated early in the partitioning process. At the end of the algorithm, there is a forest of random trees. Instances that have shorter paths across many of the trees, it is highly likely to be an anomaly. The average path length, for a given instance, converges as the number of random trees in the forest increases.

Additionally, IF has previously been shown to be an effective unsupervised method for the high-imbalanced Medicare fraud detection [35], outperforming other approaches such as the deep learning framework called ORCA, local outlier factor (LOF), and random forests when comparing AUC and processing time. We use the IF implementation provided in Scikit-learn [36], version 0.23.1, for our experiments. The IF used in this work is trained using the same 6 rounds of fivefold cross validation as the others, but it is trained independently. One parameter, namely the contamination rate, was chosen for our experiments, all other parameters were kept as the default. This parameter is the amount of anomalous contamination in the dataset. It is the percentage of the minority class compared to the majority class. We set that to the level of imbalance for our datasets used in this study. This was calculated using labels; however, it is reasonable to assume that this piece of information would fall under domain expertise and would be available in practice. Additionally, we found that when this contamination rate was set, it produces an IF that had higher performance as measured by the area under the precision-recall curve (AUPRC). Thus, giving the IF the best case scenario results to compare our method to. It is important to note that this domain knowledge, the expected class imbalance, is unused in our approach.

Datasets

We present a newly prepared Medicare fraud big dataset. It is a high-imbalanced dataset derived from several publicly available Medicare datasets. Medicare fraud datasets have been used in other works [5, 37] but the one we present here is a significant improvement in that the dataset is built from additional sources and has a significant increase in the number of trainable features. The Medicare program provides health insurance to individuals 65 years and older, and other individuals with approved disabilities [38]. In 2020, there were 62 million Medicare beneficiaries and had a total expenditure greater than $926 billion US dollars [39]. The scale of the Medicare insurance program lends itself to being a target for fraudulent activity [40]. This data, in part, is publicly released by the Centers for Medicare and Medicaid Services (CMS), a United States federal agency, for analysis. The results in this paper are limited to just using data derived from the original Part D data. The original Medicare Part D fraud data is enriched with an additional 51 features for final use.

We use Medicare Part D data from 2013 through 2019, originally called the Medicare Part D Summary by Provider and Drug. It describes a medical provider's prescription drug activity as it pertains to the Medicare program, for a given year and drug name. Each year CMS releases 1 year worth of data but lags by a couple of years; our dataset contains data from 2013 through 2019. The original Part D spanning the 7 years has roughly 172 million records with 22 features. This makes it the largest of the Medicare datasets released thus far by CMS. The features include provider-level and claims-level information. The provider-level attributes include the healthcare provider's national provider identifier (NPI), their specialty, gender, name, medical credentials, and geographic details. The data is the yearly aggregate of the NPI and prescription drug name. The claims-level attributes include the number of beneficiaries for the drug, cost, and number of prescriptions created.

First, the raw Part D data is cleaned and preprocessed to merge the years of data, normalize columns, fill in missing values, and removing of duplicates. Missing values are rectified by using dictionary files provided by CMS. For example, instances with missing provider gender values are imputed with a third gender "U" for unknown. Another example is for the *Tot_Bene*, or total beneficiaries for a given drug and NPI, contain missing values but CMS states that this is done when the number is less than 10. A median value of 5 is filled in for any rows that are missing the *Tot_Bene* feature value. Other features that have missing values, such as provider name, are removed from entirely since it treated as personally identifiable information. Many other features require little preprocessing since they are consistently inputted and adhere to the dataset's schema.

However, the drug cost feature can have string values representing floating point numbers and need to be processed. The provider type feature, a categorical feature, needs to be cleaned. There are some identical types with different inputted names that need to be combined. This reduces the cardinality for the provider type from 269 down to 249 after combining. The provider type and NPI are used for the data preprocessing steps but is omitted during the training and test phases of this work.

Next, the preprocessed dataset now is modified with data aggregation steps. The large number of instances in the Medicare data, especially Part D, drastically increases the computation and storage requirements and significantly increases the challenges of its class imbalanced nature [3]. A compressed representation of the data is created by data aggregation and aims to significantly reduce the size and dimensionality of the Part D data. It is aggregated over the NPI, year, and provider type. The drug name attribute is removed, and the reaming numeric attributes are converted to their summary statistics: minimum, maximum, median, mean, sum, and standard deviation. The summary statistics are for a given provider NPI. For example, the feature *Tot_Clms* (total claims) is replaced by *Tot_clms_min*, *Tot_clms_max*, *Tot_clms_median*, *Tot_clms_mean*, *Tot_clms_sum*, *Tot_clms_std*. This aggregation step reduces the number of instances by a significant amount while creating new features that represent the provider's billing behaviors. The number of instances in Part D went from roughly 172 million records with 7 features to roughly 6 million rows with 31 features, an 80% reduction in required computational memory.

Next, the preprocessed version of the Medicare Part D Summary by Provider and Drug with the CMS provided Medicare Part D Summary by Provider, a second and distinct data source, is used for enrichment. Similar to the previous CMS data, this too is from the years 2013 through 2019. This data does not include statistics at the drug level, but rather summary data that describes a medical provider's services and beneficiaries over a given year. Like the pre-processed steps outlined above, this second data source is one record for each NPI per year. The aggregated data above is then enriched with this second source to produce the aggregated-enriched Medicare Part D data. We refer to this as just the Part D data in the rest of this paper. This enrichment steps adds 51 new features for a total of 82 features. These include beneficiary summary statistics, such as number of beneficiaries prescribed for per age group, gender, and Medicare or Medicaid membership. These statistics are divided into different categories: opiate drug claims, long-acting opiate drug claims, antibiotic drug claims, and anti-psychotic drug claims. The aggregated data and the Summary by Provider and Drug data set is joined on the NPI and year columns using an inner join. This results in a loss of roughly 1 million rows and the addition of the 51 features. One categorical feature, namely prescriber type, is one-hot encoded. This results in a total of 328 features for use in the experiments, summarized in Table 1.

We have only discussed the features of the Medicare Part D thus far. Though we use an entirely unsupervised approach in this work, fraudulent labels are required for us to accurately measure the effectiveness of our approach. The data from CMS does not include fraudulent labels. The Part D data is labeled using the publicly available List of Excluded Individuals and Entities (LEIE) [41]. The LEIE data is maintained by the Office of Inspector General (OIG) and is a monthly updated list. The OIG can add providers

Table 1 Dataset class characteristics

Dataset	Minority count	Majority count	Total count	Minority imbalance (%)	Features
Medicare part D	3700	5,340,406	5,344,106	0.0692	328
Credit card	492	284,315	284,807	0.1727	28

to the LEIE and they are excluded from being able to receive payment from any federal healthcare programs. This is a result of an individual or entity being convicted in a court of law for a verity of reasons such as conviction of program-related crimes, or conviction of fraud, kickbacks, and other prohibited activities [42]. The exclusion type ranges in offense and severity. The Part D data is labeled in the same manner as Bauder et al. [43]. They found a subset of exclusion rules that indicate fraud. The NPI number is used to match the fraudulent instances in the LEIE and label those in the Part D dataset. Summarized in Table 1, 3700 instances are labeled fraudulent which makes this dataset highly class imbalanced.

The smaller of the two datasets used in this study is a credit card fraud dataset [44]. Originally collected as a result of a research partnership between Worldline and the Université Libre de Bruxelles. There are a total of 284,807 instances that represent real word credit card transactions. There are 30 independent features, and each have a binary class label. The class label is only used in the performance metric calculations. The card transactions were made by European credit card holders between September 1st through September 30th, 2013. Summarized in Table 1, the dataset is highly-imbalanced with only 492 labeled fraudulent transactions. We drop one feature, namely *time*, and use the remaining 29 features for our learners. This dropped feature has been shown to not have any predictive value, contributes to noise and thus, was not used.

Results

We conduct experiments comparing our method to two baseline unsupervised learners on two different fraud datasets in different domains. We evaluate our method, using an autoencoder as the underlying learner, against two unsupervised outlier detection methods: Isolation Forest and COPOD. The first dataset is a new Medicare Part D dataset that we prepared, using multiple public sources, for fraud detection. The other is a widely used credit card fraud dataset.

For each domain, we conduct 6 rounds of fivefold cross validation producing 30 sets of 80/20 train/test splits and corresponding results. The AUC and AUPRC values shown in the result tables are the average value across the 30 replications [9]. The primary performance metric is the AUPRC values since it is a better than AUC as an indicator of model performance in a highly imbalanced setting.

Performance metrics and statistical analysis

One main challenge for measuring model performance of fraud detection methods is that this task is class imbalanced. Our datasets used in this work are considered highly class imbalanced and thus further increase the negative effects of imbalance. The performance metric used needs to be chosen with good reason. We use the area under the precision-recall curve (AUPRC) as our main performance metric when comparing our results. In addition, we also provide the area under the receiver operating curve (AUC) value since it is a widely used performance metric. However, AUPRC is a superior metric when comparing models in a high class imbalanced scenario since AUC has been shown to be a misleading performance metric in the presence of a severe class imbalance as is the case with the datasets used in this paper [9].

Fraud detection is a binary classification problem, and it is conventional to use a confusion matrix, illustrated in Table 2, to summarize the binary classification results. AUPRC and AUC values can be derived from the TP, FP, FN, and TN values.

$$FPR = \frac{FP}{FP + TN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = TPR = \frac{TP}{TP + FN} \tag{3}$$

The receiver operating characteristic curve (ROC), first introduced by Provost et al. [45], is a widely used performance metric that summarizes the true positive rate vs. the false positive rate. This curve visualizes the trade-off between correctly classified positive instances and incorrectly classified negative instances, or correctly identified fraudulent instances and fraudulent instances that were missed, respectively. The AUC is the area under the ROC and is a single number that summarizes these performance measurements.

Davis et al. [9] show that the ROC curves, and thus the AUC values, can present overly optimistic results in the presence of highly imbalanced data. As a better alternative, precision-recall (PR) curves, and thus the AUPRC values, provide a better performance metric. The authors state that a curve can only dominate in ROC space if and only if it dominates in PR space. ROC summarizes a model's performance in terms of true positive rate (TPR) and false positive rate (FPR), while the PR summarizes a model's performance in terms of precision and recall (also known as TPR). This makes AUC less sensitive to the changes in false positives as the size of the negative, or majority class grows, as it does in highly imbalanced datasets.

We use one-factor ANOVA (analysis of variance) as a statistical test to determine if there exist statistical differences between various metrics we compare. Additionally we use the Tukey's honestly significant difference (HSD) test to rank our results where needed [46, 47]. The Tukey HSD test ranks the metrics being tested from highest to lowest and assigns each row to a letter-group. The letter-groups indicate statistical

Table 2 Confusion matrix

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

differences between the rows. For example, Table 10 shows the Tukey HSD results for the credit card datasets for each of our method's iterations. The test ranks the AUPRC from greatest to least showing that the highest AUPRC was from iteration 9 and is statistically similar in value to iteration 8, 7, and 6 because they all belong to *Group* "a". Another example, Table 12, shows the AUPRC values of our approach outperform the ones of COPOD which outperform the ones of IF.

Result analysis

For each of the datasets, we run our methodology for 10 iterations for each of the 6 rounds of fivefold cross validation. The IF and COPOD models are run as designed, thus they do not have any iterative results. A breakdown of the class characteristics is shown in Table 1. Both the datasets are highly imbalanced, but the Medicare Part D is both significantly larger and more class imbalanced than the credit card dataset.

In our approach, we split the data into 80% training and 20% test using fivefold cross validation, a newly initialized autoencoder is trained on the training data then λ_{2SD} is determined and used to remove instances with high anomaly scores. The then slightly smaller and cleaner dataset it used for training the next iteration's new initialized autoencoder. This process is repeated for 10 iterations. At the end of each iteration, we use the trained autoencoder to make predictions on the 20% unseen test split. We then calculate the AUPRC and AUC for each iteration. Neither this process nor its performance results affect any of the iterations that follow it. The unseen test split is used for all iteration results and is not changed during our method's iterative process.

Additionally, at the beginning of the first iteration, when none of the instances have been cleaned out, it is functionally similar to just using the autoencoder of the same architecture. Thus, we can compare the AUPRC and AUC for our approach against the IF, COPOD, and autoencoder models.

Medicare part D dataset

The new Medicare Part D dataset that we prepared has 5,344,106 total instances, with only 3700 instances in the fraudulent class, giving this a class imbalance of 0.0692%. As can be seen in Table 3, the AUPRC significantly increases after each iteration. Starting from an AUPRC value of 0.238 and climbing up to 0.2598 at the final iteration. The AUC values exhibit a similar effect for the first few iterations but start to plateau after iteration 4. The first few iterations the AUPRC increase is clear but towards the final interactions the AUPRC converges to the final value.

We calculate the ANOVA for each iteration and perform a Tukey HSD test, in Tables 4 and 5, respectively. The Tukey HSD results clearly show that the highest performing iteration is the final iteration and that the AUPRC does significantly continue to improve. The *Group* column shows that each iteration is in their own letter-group. Each letter-group indicates that it is significantly different than the next. Additionally, the Tukey test shows that our method only outperforms using an autoencoder by itself after iteration 2. An autoencoder alone, AE-0, produces an AUPRC of 0.0238 and after iteration 2, our method, AE-2, has an AUPRC score of 0.0736, a $3\times$ improvement. Similar trends are observed using the AUC metric. Due to the consistently increasing AUPRC, we can

Table 3 Medicare part D results comparison

Medicare part D		
Model	**AUC**	**AUPRC**
AE-0	0.5181	0.0238
AE-1	0.5276	0.0350
AE-2	0.5547	0.0736
AE-3	0.5791	0.1185
AE-4	0.5906	0.1516
AE-5	0.5975	0.1778
AE-6	0.6016	0.2010
AE-7	0.6033	0.2198
AE-8	0.6116	0.2468
AE-9	0.6093	0.2598
IF	0.5083	0.0176
COPOD	0.5668	0.1179

Table 4 Medicare part D iterations ANOVA

	Df	**Sum Sq**	**Mean Sq**	**F value**	**Pr(> F)**
Iteration	9	1.9691	0.2188	1095	$< 2e-16$
Residuals	290	0.0579	0.0002		

Table 5 Medicare part D autoencoder iteration comparison

Part D Tukey HSD test		
Iterations	**AUPRC**	**Group**
9	0.2598	a
8	0.2468	b
7	0.2198	c
6	0.2010	d
5	0.1778	e
4	0.1516	f
3	0.1185	g
2	0.0736	h
1	0.0350	i
0	0.0238	i

conclude that in practice, iterating for at least as many iterations as we completed produces the best performing approach.

Our approach has the highest AUPRC at iteration 9 when working with our Medicare Part D dataset. We compare this model, AE-9, with the two baseline unsupervised learners, IF and COPOD. The bottom of Table 3 clearly shows that the 0.2598 AUPRC of AE-9 outperforms both IF and COPOD by a significant amount which have AUPRC values of 0.0176 and 0.1176, respectively. For completeness, we present the results in Tables 6 and 7 that show the Tukey HSD test, and associated ANOVA, confirms our observations. Our approach outperforms both IF and COPOD at iteration 4, with an AUPRC

Table 6 Medicare part D models ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(> F)
Model	2	0.8889	0.4444	2931	< 2e−16
Residuals	87	0.0132	0.0002		

Table 7 Medicare part D overall Tukey HSD

Medicare part D comparison		
Model	**AUPRC**	**Group**
AE-9	0.2598	a
COPOD	0.1179	b
IF	0.0176	c

Table 8 Credit card fraud results

Credit card fraud		
Model	**AUC**	**AUPRC**
AE-0	0.7261	0.3073
AE-1	0.8169	0.3738
AE-2	0.9035	0.4412
AE-3	0.8955	0.4462
AE-4	0.8814	0.4534
AE-5	0.8649	0.4553
AE-6	0.8553	0.4605
AE-7	0.8466	0.4632
AE-8	0.8401	0.4659
AE-9	0.8339	0.4667
IF	0.6586	0.3148
COPOD	0.8891	0.4455

value of 0.1516. This shows the effectiveness of our approach as compared to existing unsupervised leaners when being used for fraud detection on a large, highly imbalanced dataset. This also highlights how effecting the iterative cleaning process is. Further analysis is needed to determine where AUPRC results stop improving. However, from these results, AE-9 would be used in practice since it is the highest performing model.

Credit card dataset

The credit card dataset has a total of 284,807 instances with only 492 instances in the fraudulent/minority class, giving this a class imbalance of 0.1727%. This is significantly more balanced than the Medicare Part D dataset, though still imbalanced. However, the number of fraud instances is significantly lower than the number of fraud instances in the Part D dataset, making these fraud cases significantly rarer.

Table 8 presents the AUPRC and AUC values for each of our methods iterations, IF, and COPOD. When looking at this dataset, our approach outperforms the baseline autoencoder, AE-0, after the first cleaning iteration as measured in both AUC and

Table 9 Credit card iterations ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(> F)
Iteration	9	0.7303	0.08114	283.3	< 2e−16
Residuals	290	0.0831	0.00029		

Table 10 Credit card autoencoder iteration comparison

CC iteration Tukey HSD		
Iterations	**AUPRC**	**Group**
9	0.4667	a
8	0.4659	a
7	0.4632	a
6	0.4605	a
5	0.4553	ab
4	0.4534	abc
3	0.4462	bc
2	0.4412	c
1	0.3738	d
0	0.3073	e

AUPRC. The AUPRC increases from 0.3073 to 0.3738 and the AUC increases from 0.7261 to 0.8169. In the previous dataset it took 2 iterations to measure an increase in AUPRC. Similar to the previous dataset, we want to find at which iteration does our method best perform, when measuring AUPRC. These values are closer together than the previous dataset, so we must perform a Tukey HSD test to determine this.

Tables 9 and 10 show the ANOVA and Tukey HSD test when comparing the AUPRC across each of our method's iterations, respectively. Starting from the bottom of Table 10, the results show that the AUPRC significantly improves after each iteration, as noted by group e, d, and c. Iterations that have shared letters in their group are not statistically different from each other. For example, iteration 2, 3, and 4 each have the letter "c" in it. This means the AUPRC value might be slightly higher as the iterations increase, but they are not significantly higher. Iterations that belong to the letter group "a" have the highest AUPRC value and are significantly higher than iterations not belonging to "a". Each additional iteration has a computational cost associated with it and is removing additional, potentially useful, instances from its training dataset. Thus, we can conclude that the overall best iteration is iteration 9 since it is produces the highest AUPRC of 0.4667 and belongs to the "a" group. Our approach shows slight AUPRC improvement from the fourth iteration to the ninth. However, the increase after the fourth is slight and is not statistically significant, as indicated by the Tukey HSD test. This shows clear diminishing returns after first several iterations.

We can compare AE-9, our approach's best performing iteration, to the baseline unsupervised fraud detection methods, IF, and COPOD. Unlike the results in the previous section, the AUPRC values are much closer together for AE-9, IF, and COPOD. Tables 11 and 12 show the ANOVA and Tukey HSD results, respectively, when comparing AE-9, IF, and COPOD using AUPRC. These tables confirm that AE-9 outperforms

Table 11 Credit card models ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(> F)
Model	2	0.3942	0.1971	310.6	< 2e−16
Residuals	87	0.0552	0.00063		

Table 12 Credit card results comparison

Overall CC Tukey HSD		
Model	**AUPRC**	**Group**
AE-9	0.4667	a
COPOD	0.4455	b
IF	0.3148	c

both COPOD and IF when using AUPRC as a performance metric, though it outperforms the next best model by a smaller degree when looking at the credit card data as opposed to the Medicare Part D data. AE-9 has an AUPRC of 0.4667, COPOD is 0.445, but both are significantly greater than the 0.3148 AUPRC for IF. In both datasets, our model outperforms COPOD while IF is always the least performant. When using our approach in practice, AE-9 would be chosen since it is the highest performing model. However, if significant time or computation constraints existed, choosing AE-4 would produce statistically similar results, albeit with slightly less AUPRC.

Conclusion

This study presents an unsupervised and iterative methodology for fraud detection of high-imbalanced data. Our approach uses a fully connected autoencoder as its baseline learner. We compare our unsupervised fraud detection approach with existing unsupervised fraud detection models, namely Isolation Forest and COPOD. Additionally, we present a new Medicare Part D dataset that uses several publicly available claims data from CMS. To the best of our knowledge, this is the first paper to utilize these CMS datasets to create a labeled fraud dataset for Medicare Part D claims. We also leverage the list of fraudulent Medicare providers from the LEIE to construct fraudulent labels.

We compare the AUPRC and AUC performance of our approach with IF and COPOD when using the presented Medicare Part D dataset as well as a widely used credit card fraud detection dataset. We present results and statistical analysis that show our methodology outperforms both IF and COPOD in both datasets when using AUPRC. Our iterative cleaning method outperforms the baseline autoencoder in as few as one iteration on the credit card dataset and as few as 2 iterations on the Part D dataset. Additionally, we can conclude that across both datasets, our approach outperforms the baseline autoencoder, IF, and COPOD at the fourth iteration. Additional iterations increased AUPRC performance in the Part D data but did not significantly improve when using the credit card data. Additional analysis is needed to determine the precise reasoning for this. Future work includes expanding these results to other application domains as well as other baseline learners. Additionally, future work on measuring the effectiveness of our approach on big datasets that do not have high class imbalance. Another possible avenue for future work is to explore potential early stopping techniques while training the autoencoders. This has the potential to reduce overall training time while not negatively affecting the approach's performance.

Abbreviations

ANOVA	Analysis of Variance
AUC	Area under the receiver operating curve
AUPRC	Area under the precision–recall curve
CMS	Center for Medicare and Medicaid Services
COPOD	Copula-based Outlier Detector
EE	EasyEnsemble
FBI	Federal bureau of investigation
FN	False negative
FP	False positive
FPR	False positive rate
HMM	Hidden Markov Model
HSD	Tukey's honestly significant difference
IF	Isolation Forest
LEIE	List of Excluded Individuals and Entities
LOF	Local outlier factor
LSTM	Long short-term memory
MSE	Mean squared error
NPI	National provider identifier
OCC	One-Class Classifier
OIG	Office of Inspector General
PR	Precision–recall
ReLu	Rectified linear unit
ROC	Receiver operating characteristic curve
SVM	Support vector machine
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 and ZS carried out the concept and design of the research, performed the implementation and experimentation, and drafted the manuscript. TMK and FV 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: 4 April 2023 Accepted: 8 May 2023
Published online: 19 June 2023

References

1. Morris L. Combating fraud in health care: an essential component of any cost containment strategy. Health Aff. 2009;28(5):1351–6.
2. Bauder RA, Khoshgoftaar TM, Seliya N. A survey on the state of healthcare upcoding fraud analysis and detection. Health Serv Outcomes Res Methodol. 2017;17:31–55.

3. Leevy JL, Khoshgoftaar TM, Bauder RA, Seliya N. A survey on addressing high-class imbalance in big data. J Big Data. 2018;5(1):42.
4. Johnson JM, Khoshgoftaar TM. Encoding techniques for high-cardinality features and ensemble learners. In: 2021 IEEE 22nd international conference on information reuse and integration for data science (IRI). IEEE; 2021. p. 355–61.
5. Bauder RA, Khoshgoftaar TM. The effects of varying class distribution on learner behavior for medicare fraud detection with imbalanced big data. Health Inf Sci Syst. 2018;6:1–14.
6. Wei W, Li J, Cao L, Ou Y, Chen J. Effective detection of sophisticated online banking fraud on extremely imbalanced data. World Wide Web. 2013;16(4):449–75.
7. Kubat M, Holte RC, Matwin S. Machine learning for the detection of oil spills in satellite radar images. Mach Learn. 1998;30(2):195–215.
8. Cieslak DA, Chawla NV, Striegel A. Combating imbalance in network intrusion datasets. In: GrC; 2006. p. 732–7.
9. 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.
10. Hancock JT, Khoshgoftaar TM, Johnson JM. Evaluating classifier performance with highly imbalanced big data. J Big Data. 2023;10(1):1–31.
11. Kennedy RK, Johnson JM, Khoshgoftaar TM. The effects of class label noise on highly-imbalanced big data. In: 2021 IEEE 33rd international conference on tools with artificial intelligence (ICTAI). IEEE; 2021. p. 1427–33.
12. Salekshahrezaee Z, Leevy JL, Khoshgoftaar TM. A class-imbalanced study with feature extraction via PCA and convolutional autoencoder. In: 2022 IEEE 23rd international conference on information reuse and integration for data science (IRI). IEEE; 2022. p. 63–8.
13. Hasanin T, Khoshgoftaar TM, Leevy JL, Seliya N. Examining characteristics of predictive models with imbalanced big data. J Big Data. 2019;6(1):1–21.
14. Carcillo F, Le Borgne Y-A, Caelen O, Kessaci Y, Oblé F, Bontempi G. Combining unsupervised and supervised learning in credit card fraud detection. Inf Sci. 2021;557:317–31.
15. Srivastava A, Kundu A, Sural S, Majumdar A. Credit card fraud detection using hidden Markov model. IEEE Trans Dependable Secure Comput. 2008;5(1):37–48.
16. Lee C-Y, Li C-L, Yoon J, Sohn K, Arik S, Pfister T. Self-supervise, refine, repeat: improving unsupervised anomaly detection. arXiv preprint. 2022. arXiv:2106.06115.
17. Liu X-Y, Wu J, Zhou Z-H. Exploratory undersampling for class-imbalance learning. IEEE Trans Syst Man Cybern B Cybern. 2008;39(2):539–50.
18. Liu T-Y. Easyensemble and feature selection for imbalance data sets. In: 2009 International joint conference on bioinformatics, systems biology and intelligent computing. IEEE; 2009. p. 517–20.
19. Zong B, Song Q, Min MR, Cheng W, Lumezanu C, Cho D, Chen H. Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. In: International conference on learning representations. 2018.
20. Pu G, Wang L, Shen J, Dong F. A hybrid unsupervised clustering-based anomaly detection method. Tsinghua Sci Technol. 2020;26(2):146–53.
21. Maleki S, Maleki S, Jennings NR. Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering. Appl Soft Comput. 2021;108: 107443.
22. Pang G, Yan C, Shen C, Hengel AVD, Bai X. Self-trained deep ordinal regression for end-to-end video anomaly detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020. p. 12173–82.
23. Beggel L, Pfeiffer M, Bischl B. Robust anomaly detection in images using adversarial autoencoders. In: Machine learning and knowledge discovery in databases: European conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I. Springer; 2020. p. 206–22.
24. Ng A, *et al.* Sparse autoencoder. CS294A Lecture notes. 2011;72(2011):1–19.
25. Chollet F, et al. Keras. 2015. https://keras.io.
26. Wang S, Yao X. Multiclass imbalance problems: analysis and potential solutions. IEEE Trans Syst Man Cybern B Cybern. 2012;42(4):1119–30.
27. Krawczyk B. Learning from imbalanced data: open challenges and future directions. Progr Artif Intell. 2016;5(4):221–32.
28. 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 (ICMLA 2007). IEEE; 2007. p. 348–53.
29. Japkowicz N. The class imbalance problem: significance and strategies. In: Proc. of the Int'l Conf. on artificial intelligence, vol. 56; 2000. p. 111–7.
30. Fang J, Xia S, Lin J, Xia Z, Liu X, Jiang Y. Alpha discovery neural network based on prior knowledge. 2019. arXiv preprint. arXiv:1912.11761.
31. Clark J, Liu Z, Japkowicz N. Adaptive threshold for outlier detection on data streams. In: 2018 IEEE 5th international conference on data science and advanced analytics (DSAA). IEEE; 2018. p. 41–9.
32. Li Z, Zhao Y, Botta N, Ionescu C, Hu X. COPOD: copula-based outlier detection. In: 2020 IEEE international conference on data mining (ICDM). IEEE; 2020. p. 1118–23.
33. Zhao Y, Nasrullah Z, Li Z. Pyod: a python toolbox for scalable outlier detection. J Mach Learn Res. 2019;20(96):1–7.
34. Liu FT, Ting KM, Zhou Z-H. Isolation forest. In: 2008 eighth IEEE international conference on data mining. IEEE; 2008. p. 413–22.
35. 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). 2018; IEEE. p. 285–92.
36. 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.
37. Johnson JM, Khoshgoftaar TM. Hcpcs2vec: healthcare procedure embeddings for medicare fraud prediction. In: 2020 IEEE 6th international conference on collaboration and internet computing (CIC). IEEE; 2020. p. 145–52.
38. U.S. Government. US Centers for Medicare & Medicaid Services: the official U.S. government site for medicare. https://www.medicare.gov/.

39. Centers for Medicare & Medicaid Services: trustees report & trust funds. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/ReportsTrustFunds/index.html.
40. van Capelleveen G, Poel M, Mueller RM, Thornton D, van Hillegersberg J. Outlier detection in healthcare fraud: a case study in the medicaid dental domain. Int J Acc Inf Syst. 2016;21:18–31.
41. U.S. Department of Health and Human Services Office of Inspector General: LEIE downloadable databases. https://oig.hhs.gov/exclusions/exclusions_list.asp.
42. U.S. Department of Health and Human Services Office of Inspector General: exclusion authorities. https://oig.hhs.gov/exclusions/authorities.asp.
43. Bauder RA, Khoshgoftaar TM. A novel method for fraudulent medicare claims detection from expected payment deviations (application paper). In: 2016 IEEE 17th international conference on information reuse and integration (IRI). 2016; IEEE. p. 11–19.
44. 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.
45. Provost FJ, Fawcett T, et al. Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In: KDD, vol. 97; 1997. p. 43–8.
46. Abdi H, Williams LJ. Tukey's honestly significant difference (HSD) test. Encycl Res Des. 2010;3:1–5.
47. Berenson M, Levine D, Goldstein M. Intermediate statistical methods and applications: a computer package approach. Englewood Cliffs: Prentice-Hall; 1983.

Publisher's Note