Johnson and Khoshgoftaar *J Big Data* (2019) 6:63
https://doi.org/10.1186/s40537-019-0225-0

Journal of Big Data

RESEARCH Open Access

Medicare fraud detection using neural networks

Justin M. Johnson* and Taghi M. Khoshgoftaar

*Correspondence:
jjohn273@fau.edu
Florida Atlantic University,
777 Glades Road, Boca Raton,
FL, USA

Abstract

Access to affordable healthcare is a nationwide concern that impacts a large majority of the United States population. Medicare is a Federal Government healthcare program that provides affordable health insurance to the elderly population and individuals with select disabilities. Unfortunately, there is a significant amount of fraud, waste, and abuse within the Medicare system that costs taxpayers billions of dollars and puts beneficiaries' health and welfare at risk. Previous work has shown that publicly available Medicare claims data can be leveraged to construct machine learning models capable of automating fraud detection, but challenges associated with class-imbalanced big data hinder performance. With a minority class size of 0.03% and an opportunity to improve existing results, we use the Medicare fraud detection task to compare six deep learning methods designed to address the class imbalance problem. Data-level techniques used in this study include random over-sampling (ROS), random under-sampling (RUS), and a hybrid ROS–RUS. The algorithm-level techniques evaluated include a cost-sensitive loss function, the *Focal Loss*, and the *Mean False Error Loss*. A range of class ratios are tested by varying sample rates and desirable class-wise performance is achieved by identifying optimal decision thresholds for each model. Neural networks are evaluated on a 20% holdout test set, and results are reported using the area under the receiver operating characteristic curve (AUC). Results show that ROS and ROS–RUS perform significantly better than baseline and algorithm-level methods with average AUC scores of 0.8505 and 0.8509, while ROS–RUS maximizes efficiency with a 4× speedup in training time. Plain RUS outperforms baseline methods with up to 30× improvements in training time, and all algorithm-level methods are found to produce more stable decision boundaries than baseline methods. Thresholding results suggest that the decision threshold always be optimized using a validation set, as we observe a strong linear relationship between the minority class size and the optimal threshold. To the best of our knowledge, this is the first study to compare multiple data-level and algorithm-level deep learning methods across a range of class distributions. Additional contributions include a unique analysis of the relationship between minority class size and optimal decision threshold and state-of-the-art performance on the given Medicare fraud detection task.

Keywords: Class imbalance, Big data, Thresholding, Deep learning, Medicare, CMS, LEIE, Fraud detection

Introduction

Medicare is a United States (U.S.) healthcare program established and funded by the Federal Government that provides affordable health insurance to individuals 65 years and older, and other select individuals with permanent disabilities [1]. According to the 2018 Medicare Trustees Report [2], in 2017 Medicare provided coverage to 58.4 million beneficiaries and exceeded $710 billion in total expenditures. Medicare enrollment has grown to 60.6 million as of February 2019 [3]. There are many factors that drive the costs of healthcare and health insurance, including fraud, waste, and abuse (FWA) within the healthcare system. The Federal Bureau of Investigation (FBI) estimates that fraud accounts for 3–10% of all billings [4], and the Coalition Against Insurance Fraud [5] estimates that fraud costs all lines of insurance roughly $80 billion per year. Based on these estimates, Medicare is losing between $21 and $71 billion per year to FWA. Examples of fraud include billing for appointments that the patient did not keep, billing for services more complex than those performed, or billing for services not provided. Abusive practice is practice inconsistent with providing patients medically necessary services according to recognized standards, e.g. billing for unnecessary medical services or misusing billing codes for personal gain. Federal laws are in place to govern Medicare fraud and abuse, for example the *False Claims Act* (FCA) and *Anti-Kickback Statute* [6].

One way to improve the cost, efficiency, and quality of Medicare services is to reduce the amount of FWA. Manually auditing and investigating all Medicare claims data for fraud is very tedious and inefficient when compared to machine learning and data mining approaches [7]. As of 2017, 86% of office-based physicians and more than 96% of reported hospitals have adopted electronic health record (EHR) systems in accordance with the Health Information Technology for Economic and Clinical Health Act of 2009 and the Federal Health IT Strategic Plan [8, 9]. This explosion in healthcare-related data encourages the use of data mining and machine learning for detecting patterns and making predictions. The *Centers for Medicare and Medicaid Services* (CMS) joined this data-driven effort by making Medicare data sets publicly available, stating that bad actors intent on abusing federal health care programs cost taxpayers billions of dollars and risks the well-being of beneficiaries [6].

Fraud detection using CMS Medicare data presents several challenges. The problem is characterized by the four Vs of big data: volume, variety, velocity, and veracity [10, 11]. The 9 million records released by CMS each year satisfies both high volume and velocity. Variety arises from the mixed-type high-dimensional features and the combining of multiple data sources. These data sets also exhibit veracity, or trustworthiness, as they are provided by reputable government resources with transparent quality controls and detailed documentation [12, 13]. The processing of big data often exceeds the capabilities of traditional systems and demands specialized architectures or distributed systems [14]. Another challenge is that the positive class of interest makes up just 0.03% of all records, creating a severe class-imbalanced distribution. Learning from such distributions can be very difficult, and standard machine learning algorithms will typically over-predict the majority class [15]. This paper expresses the level of class imbalance within a given data distribution as N_{neg}:N_{pos}, where N_{neg} and N_{pos} correspond to the percentage of samples in the negative and positive classes, respectively.

We believe that deep learning is an important area of research that will play a critical role in the future of modeling class-imbalanced big data. Over the last 10 years, deep learning methods have grown in popularity as they have improved the state-of-the-art in speech recognition, computer vision, and other domains [16]. Their recent success can be attributed to an increased availability of data, improvements in hardware and software [17–21], and various algorithmic breakthroughs that speed up training and improve generalization to new data [22]. Deep learning is a sub-field of machine learning that uses the *artificial neural network* (ANN) with two or more hidden layers to approximate some function f^*, where f^* can be used to map input data to new representations or make predictions [23]. The ANN, inspired by the biological neural network, is a set of interconnected neurons, or nodes, where connections are weighted and each neuron transforms its input into a single output by applying a non-linear activation function to the sum of its weighted inputs. In a feedforward network, input data propagates through the network in a forward pass, each hidden layer receiving its input from the previous layer's output, producing a final output that is dependent on the input data, the choice of activation function, and the weight parameters [24]. Gradient descent optimization adjusts the network's weight parameters in order to minimize the loss function, i.e. the error between expected output and actual output. Composing multiple non-linear transformations creates hierarchical representations of the input data, increasing the level of abstraction through each transformation. The deep learning architecture, i.e. *deep neural network* (DNN), achieves its power through this composition of increasingly complex abstract representations [23]. Despite the success of DNN models in various domains, there is limited research that evaluates the use of deep learning for addressing class imbalance [25].

This study compares six deep learning methods for addressing class imbalance and assesses the importance of identifying optimal decision thresholds when training data is imbalanced. We expand upon existing Medicare fraud detection work [26] using the *Medicare Provider Utilization and Payment Data: Physician and Other Supplier Public Use File* provided by CMS, as it provides a firm baseline with traditional machine learning methods. This data set, referred to as *Part B* data hereafter, provides information on the services and procedures provided to Medicare beneficiaries and is currently available on the CMS website for years 2012–2016 [27]. The Part B data set includes both provider-level and procedure-level attributes, including the amounts charged for procedures, the number of beneficiaries receiving the procedure, and the payment reimbursed by Medicare. To enable supervised learning, fraud labels are mapped to the Part B claims data using the *List of Excluded Individuals and Entities* (LEIE) [13]. Since we are most interested in detecting fraud, we refer to the group of fraudulent samples as the positive class and the group of non-fraudulent samples as the negative class. The LEIE is maintained by the *Office of Inspector General* (OIG), and its monthly releases list providers that are prohibited from participating in Federal healthcare programs. Under the *Exclusion Statute* [28], the OIG must exclude providers convicted of program-related crimes, patient abuse, and healthcare fraud.

With three data-level and three algorithm-level methods for addressing class imbalance, multiple configurations for each method, and two network architectures, we evaluate the performance of 42 distinct DNN models. Data-level methods for addressing

class imbalance include random over-sampling (ROS), random under-sampling (RUS), and combinations of random over-sampling and random under-sampling (ROS–RUS). Multiple class distributions are tested for each method, i.e. 40:60, 50:50, 60:40, 80:20, and 99:1. Algorithm-level methods include cost-sensitive learning and two loss functions specifically designed to increase the impact of the minority class during training, i.e. *Mean False Error Loss* (MFE) [29] and *Focal Loss* (FL) [30]. To further offset the bias towards the majority class, we calculate an optimal decision threshold for each method using a validation set. For each method configuration, we train 30 models and report the average *area under the receiver operating characteristic curve* (ROC AUC) [31] score on a 20% holdout test set. *Analysis of variance* (ANOVA) [32] and *Tukey's HSD* (honestly significant difference) [33] tests are used to estimate the significance of the results. The mean optimal decision threshold, *true positive rate* (TPR), *true negative rate* (TNR), *geometric mean*, and training time are also reported for each method.

Results indicate that eliminating class imbalance from the training data through ROS or ROS–RUS produces significantly better AUC scores than all other methods, i.e. 0.8509 and 0.8505. While ROS methods perform best using the 50:50 class ratio, plain RUS outperforms baseline methods and achieves its highest AUC score with a 99:1 class ratio. Tukey's HSD test shows that there is no significant difference between AUC scores of algorithm-level methods and baseline models, but we show that the algorithm-level methods yield more stable decision boundaries than the baseline models. Analysis of training times further suggests combining ROS and RUS when working with big data and class imbalance, as the balanced training distribution yields superior results and the under-sampling component improves efficiency. Results also show that the optimal decision threshold is highly correlated with the minority class size. Hence, we suggest that the decision threshold always be optimized with a validation set when training data is imbalanced. To the best of our knowledge, this is the first study to compare DNN loss functions designed for addressing class imbalance with random sampling methods that consider multiple class distributions. Additional contributions include a unique thresholding assessment that stresses the importance of optimizing classification decision thresholds and state-of-the-art performance on the given Medicare Part B fraud detection task.

The remainder of this paper is outlined as follows. The "Related works" section discusses other works related to CMS Medicare data, fraud detection, and deep learning with class-imbalanced data. The CMS and LEIE data sets are described in full, including all pre-processing steps, in the "Data sets" section. The "Methodology" section explains the experiment framework, hyperparameter tuning, class imbalance methods, and performance criteria. Results are presented in the "Results and discussion" section, and the "Conclusion" section concludes the study with areas for future works.

Related works

Since CMS released the Public Use Files (PUF) in 2014, a number of studies relating to Medicare anomaly and fraud detection have been conducted. We have selected this data set to evaluate deep learning methods for addressing class imbalance because it exhibits severe class imbalance (99.97:0.03) and previous work has left an opportunity for improvement. This section discusses the fraud-related works performed by our research

group and others and outlines studies that consider the effects of class imbalance on deep learning.

Medicare fraud detection

Our research group has performed extensive research on detecting anomalous provider behavior using the CMS PUF data. In [34], Bauder and Khoshgoftaar proposed an outlier detection method based on Bayesian inference that detected fraud within Medicare. This study used a small subset of the 2012–2014 Medicare Part B data by selecting dermatology and optometry claims from Florida office clinics for analysis. The authors demonstrated the model's ability to identify outliers with credibility intervals, and successfully validated the model using claims data from a known Florida provider that was under criminal investigation for excessive billing. In another study [35], Bauder and Khoshgoftaar use a subset of the 2012–2013 Medicare Part B data, i.e. Florida claims only, to model expected amounts paid to providers for services rendered to patients. Claims data is grouped by provider type, and five different regression models are used to model expected payment amounts. Actual payment amount deviations from the expected payment amounts are then used to flag potential fraudulent providers. Of the five regression methods tested, the *multivariate adaptive regression splines* [36] model is shown to outperform others in most cases, but the authors state that model selection varies between provider types. In [37], Bauder et al. used a Naive Bayes classifier to predict provider specialty types, suggesting that providers practicing outside their specialty norm warrant further investigation. This study also used a Florida-only subset of 2013 Medicare Part B claims data, but it included all 82 provider types, or classes, yielding 40,490 unique physicians and 2789 unique procedure codes. Recall, precision, and F1-scores were used to evaluate the model, showing that 7 of 82 classes scored very highly (F1-score > 0.90), and 18 classes scored reasonably ($0.5 <$ F1-score < 0.90). The authors conclude that specialties with unique billing procedures, e.g. audiologist or chiropractic, are able to be classified with high precision and recall. Herland et al. [38] expanded upon the work from [37] by incorporating 2014 Medicare Part B data and real-world fraud labels defined by the LEIE data set. Providers are labeled as fraudulent when the Naive Bayes model misclassifies the provider's specialty type, and LEIE ground truth fraud labels are used to evaluate performance. They found that removing specialty types that have many overlapping procedures improves overall performance, e.g. Internal Medicine and Family Practice. Similarly, the authors showed that grouping like specialties improves performance further still, yielding an overall accuracy of 67%. In a later study, Bauder and Khoshgoftaar [39] merge 2012–2015 Medicare Part B data sets, map fraud labels using LEIE data, and compare multiple learners on all available data. Rather than focus on Florida-specific claims, like earlier reports, this study includes all available data for the given years, yielding 37,147,213 instances. Class imbalance is addressed with RUS, and various class distributions are generated to identify the optimal imbalance ratio for training. ANOVA is used to evaluate the statistical significance of ROC AUC scores, and the C4.5 decision tree and logistic regression (LR) learners are shown to significantly outperform the support vector machine (SVM). The 80:20 class distribution outperformed all other distributions tested, i.e. 50:50, 65:35, and 75:25. These studies

jointly show that Medicare Part B claims data contains sufficient variability to detect bad actors and that the LEIE data set can be reliably used for ground truth fraud labels.

Our study is most closely related to the work performed by Herland et al. in [26], which uses three different 2012–2015 CMS Medicare PUF data sets, i.e. Part B, Part D [40], and DMEPOS [41]. Part B, Part D, and DMEPOS claims data are used independently to perform cross-validation with LR, random forest (RF), and Gradient Boosted Tree (GBT) learners. The authors also construct a combined data set by merging Part B, Part D, and DMEPOS and assess the performance of each learner to determine if models should be trained on individual sectors of Medicare claims or all available claims data. The combined and Part B data sets scored the best on ROC AUC, and the LR learner was shown to perform significantly better than GBT and RF with a maximum ROC AUC score 0.816. We follow the same protocol in preparing data for supervised learning, described in the "Data sets" section, so that we may compare deep learning results to those of traditional learners.

A number of other research groups have explored the use of CMS Medicare and LEIE data for the purpose of identifying patterns, anomalies, and potentially fraudulent activity. Feldman and Chawla [42] explored the relationship between medical school training and the procedures performed by physicians in practice in order to detect anomalies. The 2012 Medicare Part B data set was linked with provider-level medical school data obtained through the CMS physician compare data set [43]. Significant procedures for schools were used to evaluate school similarities and present a geographical analysis of procedure charges and payment distributions. Ko et al. [44] used the 2012 CMS data to analyze the variability of service utilization and payments, and found that the number of patient visits was strongly correlated with Medicare reimbursement. They also found that in terms of services per visit there was a high utilization variability and a possible 9% savings within the field of Urology. Chandola et al. [45] use healthcare claims and fraudulent provider labels provided by the Texas Office of Inspector General's exclusion database to detect anomalies and bad actors. They employ social network analysis, text mining, and temporal analysis to show that typical treatment profiles can be used to compare providers and highlight abuse. A weighted LR model was used to classify bad actors, and experimental results showed that the inclusion of the provider specialty attribute increases the ROC AUC score from 0.716 to 0.814. Branting et al. [46] propose a graph-based method for estimating healthcare fraud risk within the 2012–2014 CMS PUF and LEIE data sets. Since the LEIE data set contains many missing NPI values, the authors use the National Plan and Provider Enumeration System (NPPES) [47] data set from 2015 to identify additional NPIs within the LEIE data set. This allowed the authors to increase the total positive fraudulent provider count to 12,000, which they then combined with 12,000 randomly selected non-fraudulent providers. Features are constructed from behavioral similarity between known fraudulent providers and non-fraudulent providers and risk propagation through geospatial collocation, i.e. shared addresses. A J48 decision tree learner was used to classify fraud with tenfold cross-validation, yielding a mean ROC AUC of 0.96. We believe this high AUC score is misleading, however, as the class-balanced context created by the authors is not representative of the naturally imbalanced population.

Deep learning with class imbalance

In a recent paper [25], we surveyed deep learning methods for addressing class imbalance. Despite advances in deep learning, and its increasing popularity, many researchers agree that the subject of deep learning with class-imbalanced data is understudied [29, 48–52]. For the purpose of this study, we have selected a subset of data-level and algorithm-level methods for addressing class imbalance to be applied to Medicare fraud detection.

Anand et al. [53] studied the effects of class imbalance on the backpropagation algorithm in shallow networks. The authors show that when training networks with class-imbalanced data, the length of the majority class's gradient component that is responsible for updating network weights dominates the component derived by the minority class. This often reduces the error of the majority group very quickly during early iterations while consequently increasing the error of the minority group, causing the network to get stuck in a slow convergence mode. The authors of the related works in this section apply class imbalance methods to counter this effect and improve the classification of imbalanced data with neural networks.

The related works in this section often use Eq. (1) to describe the maximum between-class imbalance level, i.e. the size of the largest class divided by the size of the smallest class. C_i is a set of examples in class i, and $max_i\{|C_i|\}$ and $min_i\{|C_i|\}$ return the maximum and minimum class size over all i classes, respectively. This can be used interchangeably with our notation, e.g. a class distribution of 80:20 can be denoted by $\rho = 4$.

$$\rho = \frac{max_i\{|C_i|\}}{min_i\{|C_i|\}} \tag{1}$$

Data-level methods

Hensman and Masko [54] explored the effects of ROS on class imbalanced image data generated from the *CIFAR-10* [55] data set. The authors generated ten imbalanced distributions by varying levels of imbalance across classes, testing a maximum imbalance ratio of $\rho = 2.3$. The ROS method duplicates minority class examples until all classes are balanced, where any class whose size is less than that of the largest is considered to be a minority. This increases the size of the training data, therefore increasing training time, and has also been shown to cause over-fitting in traditional machine learning models [56]. Applying ROS until class imbalance was eliminated succeeded in restoring model performance, and achieved results comparable to the baseline model that was trained on the original balanced data set.

Buda et al. [52] presented similar results and showed that ROS generally outperforms RUS and two-phase learning. The RUS method used by Buda et al. randomly removes samples from the majority group until all classes are of equal size, where any class larger than the smallest class is treated as a majority class. If the data is highly imbalanced, under-sampling until class balance is achieved may result in discarding many samples. This can be problematic with high capacity neural network learners, as more training data is one of the most effective ways to improve performance on the test set [23]. Two-phase learning addresses this issue by first training a model on a balanced data set, generated through ROS or RUS, and then fine-tuning the model on the complete data set.

By simulating class imbalance ratios in the range $\rho \in [10, 100]$ on three popular image benchmarks, it was shown that applying ROS until class imbalance is eliminated outperforms both RUS and two-phase learning in nearly all cases. Results from Buda et al. discourage the use of RUS, as it generally performs the worst. Experiments by Dong et al. [57] support these findings and show that over-sampling outperforms under-sampling on the CelebA data set [58] with a max imbalance ratio of $\rho = 49$.

In this paper, we explore the use of ROS and RUS to address class imbalance at the data-level. Due to time constraints, we leave two-phase learning and other advanced data sampling strategies for future works, e.g. dynamic sampling [50]. The related works listed here conclude that over-sampling the minority class until the imbalance is eliminated from the training data yields the best results. They do not, however, consider big data problems exhibiting class rarity, i.e. where very few positive examples exist. Contrary to these related works, comprehensive experiments by Van Hulse et al. [15] suggest that RUS outperforms ROS when using traditional machine learning algorithms, i.e. non-deep learning. We believe that RUS will play an important role in training deep models with big data. We extend these related works by testing various levels of class imbalance and combining ROS with RUS to generate class-balanced training data. This is the first study to compare ROS, RUS, and ROS–RUS deep learning methods across a range of class distributions.

Algorithm-level methods

Wang et al. [59] employed a cost-sensitive deep neural network (CSDNN) method to detect hospital readmissions, a class imbalanced problem where a small percentage of patients are readmitted to a hospital shortly after their original visit. The authors used a *Weighted CE* loss (Eq. 2) to incorporate misclassification costs directly into the training process, where p_i is the model output activation that denotes the estimated probability of observing the ground truth label y_i.

$$\textit{Weighted CE loss} = -\sum_{i}^{C} w_i \cdot y_i \cdot log(p_i) \tag{2}$$

The weighted CE loss multiplies the loss for each class $i \in C$, i.e. $y_i \cdot log(p_i)$, by the corresponding class weight w_i. They show that increasing the weight of the minority class to $1.5\times$ and $2\times$ that of the majority class improves classification results and outperforms several baselines, e.g. decision trees, SVM, and a baseline ANN. Incorporating the cost matrix into the CE loss is a minor implementation detail that is often built into modern deep learning frameworks, making the selection of an optimal cost matrix the most difficult task. Cost matrices can be defined by empirical work, domain knowledge, class priors, or through a search process that tests a range of values while monitoring performance on a validation set.

Wang et al. [29] present the novel *Mean False Error* (MFE) loss function and compare it to the *Mean Squared Error* (MSE) loss function using imbalanced text and image data. They constructed eight imbalanced data sets with $\rho \in [5, 20]$ by sampling the CIFAR-100 [60] and 20 Newsgroup [61] image and text data sets. After demonstrating how the MSE loss is dominated by the majority class, and their models fail

to converge, they proposed the MFE loss to increase the sensitivity of errors in the minority class. The proposed loss function was derived by splitting the MSE loss into two components, *Mean False Positive Error* (FPE) and *Mean False Negative Error* (FNE). The FPE (Eq. 3) and FNE (Eq. 4) values are combined to define the total system loss, MFE (Eq. 5), as the sum of the mean errors from each class. The proposed MFE loss function, and its *Mean Squared False Error* (MSFE) (Eq. 6) variant, are shown to outperform the MSE loss in nearly all cases. Improvements over the baseline MSE loss are most apparent when class imbalance is greatest, i.e. imbalance levels of 95:5. For example, the MSFE loss improved the classification of Household image data and increased the F1-score from 0.1143 to 0.2353 when compared to MSE.

$$FPE = \frac{1}{N}\sum_{i=1}^{N}\sum_{n}\frac{1}{2}(d_n^{(i)} - y_n^{(i)})^2 \tag{3}$$

$$FNE = \frac{1}{P}\sum_{i=1}^{P}\sum_{n}\frac{1}{2}(d_n^{(i)} - y_n^{(i)})^2 \tag{4}$$

$$MFE = FPE + FNE \tag{5}$$

$$MSFE = FPE^2 + FNE^2 \tag{6}$$

It is unclear if Wang et al. performed mini-batch stochastic gradient descent (SGD) or standard batch gradient descent. We feel that this should be considered in future works, because when class imbalance levels are high, mini-batch gradient descent may contain many batches with no positive samples. This leads to many weight updates uninfluenced by the positive class of interest. Larger mini-batches or batch gradient descent will alleviate this problem, but the benefits of smaller mini-batches may prove more valuable [23].

Lin et al. [30] proposed the FL (Eq. 7) function to address the class imbalance inherent to object detection problems, where positive foreground samples are heavily outnumbered by negative background samples. The FL reshapes the CE loss in order to reduce the impact that easily classified samples have on the loss by multiplying the CE loss by a modulating factor, $\alpha_t(1-p_t)^\gamma$. Hyper parameter $\gamma \geq 0$ adjusts the rate at which easy examples are down-weighted, and $\alpha_t \geq 0$ is a class-wise weight that is used to increase the importance of the minority class. Easily classified examples, where $p_t \to 1$, cause the modulating factor to approach 0 and reduce the sample's impact on the loss.

$$FL(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t) \tag{7}$$

The proposed one-stage FL model, *RetinaNet*, is evaluated against several state-of-the-art one-stage and two-stage detectors on the COCO [62] data set. It outscores the runner-up one-stage detector (DSSD513 [63]) and the best two-stage detector (Faster R-CNN with TDM [64]) by 7.6-point and 4.0-point precision gains, respectively. When compared to several *Online Hard Example Mining* (OHEM) [65] methods, RetinaNet outscores the best method with an increase in AP from 32.8 to 36.0. By down-weighting the loss of easily learned samples, the FL lends itself to not just class imbalanced

problems, but also hard-sample problems. Nemoto et al. [66] later used the FL for the automated detection of rare building changes, e.g. new construction, and concluded that FL improves problems related to class imbalance and over-fitting. The results provided by Nemoto et al. are difficult to interpret, however, because the FL and baseline experiments are conducted on different data distributions.

The final algorithm-level technique we consider is output thresholding, i.e. adjusting the output decision threshold that computes class labels from the model's output activation. Buda et al. applied output thresholding to all experiments by dividing network outputs for each class by their estimated priors and found that combining ROS with thresholding worked especially well. Appropriate decision thresholds can also be identified by varying the threshold and comparing results on a validation set. While it is rarely discussed in related deep learning work, we believe that thresholding is a critical component of training neural networks with class imbalanced data.

In this study, we evaluate the use of cost-sensitive learning, MFE loss, FL, and output thresholding. The first three methods are modifications to the loss function that influence network weight updates by increasing the impact of the minority class during training. Output thresholding, on the other hand, does not affect training and only changes the cutoff threshold that is used for determining class labels from output scores. Unlike data-level methods, these algorithm-level methods do not change the data distribution and should only have a marginal effect on training times. One disadvantage is that two of the three algorithm-level methods increase the number of tunable hyperparameters, making the process of searching for appropriate hyperparameters more time consuming.

Data sets

In this paper we use the Medicare Part B data set provided by CMS [27] for years 2012–2016, namely the *Medicare Provider Utilization and Payment Data: Physician and Other Supplier PUF*. To enable supervised learning, a second data set, the *List of Excluded Individuals and Entities* (LEIE) [13], is used to label providers within the Medicare Part B data set as fraudulent or non-fraudulent. In this section, we describe these data sets in detail and discuss the pre-processing steps that we take to create the final labeled data set. This process follows the procedures outlined by Herland et al. [26].

Medicare Part B data

The Medicare Part B claims data set describes the services and procedures that healthcare professionals provide to Medicare's Fee-For-Service beneficiaries. Records within the data set contain various provider-level attributes, e.g. National Provider Identifier (NPI), first and last name, gender, credentials, and address. The NPI is a unique 10-digit identification number for healthcare providers [67]. In addition to provider-level details, records contain claims information that describe a provider's activity within Medicare over a single year. Examples of claims data include the procedure performed, the average charge amount submitted to Medicare, the average amount paid by Medicare, and the place of service. The procedures rendered are encoded using the Healthcare Common Procedures Coding System (HCPCS) [68]. For example, HCPCS codes 99219 and 88346 are used to bill for hospital observation care and antibody evaluation, respectively. Also

included in the claims data is the provider type, a categorical value describing the provider's specialty that is derived from the original claim.

For each annual release, CMS aggregates the data over: (1) provider NPI, (2) HCPCS code, and (3) place of service. This produces multiple records for each provider, with one record for each HCPCS code and place of service combination. CMS decided to separate claims data by place of service, i.e. facility versus non-facility, because procedure fees will vary depending on where the service was performed [12]. An example of the 2016 Part B data set is presented in Table 1.

With 28 attributes describing the claims submitted by Medicare providers, and over 9 million rows per year, this publicly available data set is an excellent candidate for data analysis and machine learning. Unfortunately, it is not readily prepared for fraud detection. In the next section, we introduce a second data set that is used for the purpose of mapping fraud labels to the providers listed in the Medicare Part B data set.

LEIE data

Real-world Medicare provider fraud labels are identified using the publicly available LEIE data. The LEIE is maintained by the OIG in accordance with Sections 1128 and 1156 of the Social Security Act [69] and is updated on a monthly basis. The OIG has the authority to exclude providers from Federally funded health care programs for a variety of reasons. Excluded individuals are unable to receive payment from Federal healthcare programs for any services, and must apply for reinstatement once their exclusion period has been satisfied. The current LEIE data format contains 18 attributes that describe the provider and the reason for their exclusion. Table 2 provides a sample of the February 2019 LEIE data set. Some additional attributes not listed include first and last name, date of birth, address, and the provider's reinstation date.

The LEIE exclusion type attribute is a categorical value that describes the offense and its severity. Following the work by Bauder and Khoshgoftaar [35], a subset of

Table 1 Sample of Part B data set

NPI	Provider type	Place of service	HCPCS code	Number of services	Avg. submitted charge
1003000142	Anesthesiology	F	20611	15	137.20
1003000142	Anesthesiology	F	62311	88	145.00
1003000142	Anesthesiology	O	99205	11	305.00
1003000142	Anesthesiology	O	99213	65	109.00
1003000142	Anesthesiology	F	77003	95	48.00

Table 2 Sample of February 2019 LEIE data set

Specialty	NPI	City	Excltype	Excldate
Podiatry practice	1598041998	Foresthills	1128a1	20190320
Pharmacy	1275750374	Lynbrook	1128a1	20190320
Transporation Co	0	Phoenix	1128a1	20190320
Adult Day Care Facil	0	Santa Rosa	1128a1	20190320

Table 3 Fraud related LEIE rules [69]

Social Security Act	Description	Minimum exclusion period
1128(a)(1)	Conviction of program-related crimes	5 years
1128(a)(2)	Conviction relating to patient abuse or neglect	5 years
1128(a)(3)	Felony conviction relating to health care fraud	5 years
1128(b)(4)	License revocation, suspension, or surrender	State dependent
1128(b)(7)	Fraud, kickbacks, and other prohibited activities	None
1128(c)(3)(G)(i)	Conviction of second mandatory exclusion offenses	10 years
1128(c)(3)(g)(ii)	Conviction of third mandatory exclusion offenses	Permanent exclusion

exclusion rules that are most indicative of fraud is selected for labeling Medicare providers. Table 3 lists the exclusion rules used in this paper along with their mandatory exclusion periods. We use the NPI numbers of excluded individuals that have been convicted under one of these rules to identify fraudulent providers within the Medicare Part B data set. For these providers in the Medicare Part B data set, whose NPI number matches those of the LEIE data set, claims that are dated prior to the provider's exclusion date are labeled as fraudulent. In doing so, we are making the assumption that a provider's claims activity prior to the date that they were excluded from Medicare reflects fraudulent activity, as they were soon after convicted.

The LEIE data set is incomplete and contains missing values, e.g. missing NPI numbers denoted by 0 values. In addition, there are likely many more Medicare providers guilty of malpractice that have not been convicted and are therefore not included in the LEIE. Since this is the only mechanism by which we label Medicare providers as fraudulent, there will be a number of fraudulent providers that are mislabeled as non-fraudulent, i.e. class noise. This is taken into consideration when evaluating results.

Fraud labeling

Fraudulent provider labels are generated by matching the NPI numbers of excluded individuals from the LEIE data set to the Medicare Part B data set. By matching on NPI numbers only, we can be fairly confident that we are not incorrectly labeling providers as fraudulent. One shortcoming to this approach is that the LEIE data set only lists NPI numbers for a small fraction of the excluded individuals, e.g. 25% in February of 2019. We believe that we can increase the total number of fraudulent labels by looking up missing NPI numbers in the NPPES registry, similar to [46], and we leave this for future work.

Since Medicare PUF and LEIE data have different release schedules, Herland et al. decided to round exclusion end dates to the nearest year. Under certain circumstances, the OIG has the right to waive an exclusion and allow providers to continue practicing, and this is denoted by the waiver date attribute. Taking the exclusion end date as the minimum of the exclusion date and the waiver date, providers are labeled as fraudulent for a given year if they are on the exclusion list for the majority of that year. For example, if a provider has an exclusion end date of September 2015, their exclusion end date will be rounded to 2016 because they were listed as fraudulent for

more than half of 2015. On the other hand, if a provider has an exclusion end date of February 2015, their exclusion end date will be rounded to 2015.

If a provider has an exclusion end date of 2016, then all claims activity prior to 2016 are labeled as fraudulent for this given provider. As another example, a provider with an exclusion date of 2016-02-20 will have a newly defined exclusion end date of 2016. This provider is matched in the Part B data on NPI and all claims submitted prior to the year 2016 are labeled as fraudulent, i.e. 2012–2015 claims. This fraud labeling process accounts for both fraudulent claims activity and Medicare providers practicing during their mandatory exclusion period.

Data processing

All Medicare Part B data provided by CMS to date is utilized by combining years 2012–2016. We apply feature matching to remove those features not available across all 5 years. Consequently, standard deviation and standardized payment attributes were removed from years 2012–2013 and 2014–2016, respectively. Missing data was handled by removing Part B records with missing NPI numbers and missing HCPCS codes. Rows containing HCPCS codes corresponding to prescription drugs were also removed, limiting the data set to medical procedures. Unlike medical procedures, whose line_srvc_cnt feature quantifies the number of procedures performed, the line_srvc_cnt feature of prescription-related records quantifies the volume of a specific drug, and the removal of prescription-related claims eliminates this inconsistency.

As we are primarily interested in detecting fraud through claims activity, many of the provider-level attributes are removed from the Medicare data, e.g. name and address. Of the Medicare data's original 28 attributes, we keep six procedure-level attributes along with the provider's NPI and gender. The resulting feature set and their corresponding data types are outlined in Table 4.

We then organize the data by provider for each year by grouping records on year, NPI, provider type, and gender. These groups contain a unique provider's annual claims data, with one row for every combination of HCPCS code and place of service. The provider type is included in the grouping because it has been shown that providers sometimes list different specialties on different claims [26]. Each group is then aggregated, converting the multiple rows into a single row than contains summary attributes for each of the original numeric attributes, i.e. mean, sum, median,

Table 4 Description of features chosen from the Part B data set [26]

Feature	Description	Type
Npi	Unique provider identification number	Categorical
Provider_type	Medical provider's specialty (or practice)	Categorical
Nppes_provider_gender	Provider's gender	Categorical
Line_srvc_cnt	Number of procedures/services the provider performed	Numeric
Bene_unique_cnt	Number of distinct Medicare beneficiaries receiving the service	Numeric
Bene_day_srvc_cnt	Number of distinct Medicare beneficiary/per day services	Numeric
Average_submitted_chrg_amt	Average of the charges that the provider submitted for the service	Numeric
Average_medicare_payment_amt	Average payment made to a provider per claim for the service	Numeric
Exclusion	Fraud labels from the LEIE data set	Categorical

standard deviation, minimum, and maximum. Stratified random sampling [24] without replacement is used to create the 20% test set.

Categorical attributes, i.e. provider type and gender, must be properly encoded before they can be ingested by a neural network. Following the work of Herland et al. [26], we employ one-hot encoding, or one-of-K encoding, to convert these categorical attributes to numeric type. This process replaces the categorical variables with sparse one-hot vectors and increases the dimensionality of the data set from 34 to 126. A disadvantage of one-hot encoding is that it drastically increases dimensionality and fails to capture the relationship between similar providers, relationships that were shown to exist in previous works [38]. An alternative method that we leave for future work is to convert the categorical provider type variable to a dense, semantic embedding by following a learning procedure similar to Guo and Berkhahn [70] or Mikolov et al. [71].

Data normalization is a critical step that speeds up training and influences model performance [72]. In this study, min–max scaling is used to map numeric input values to the range $[0, 1]$ in order to preserve outlier relationships. The normalizer is fit to the training data and then used to scale both train and test sets.

Once all data processing steps are completed, the final Medicare Part B data set contains 4,692,370 samples, 125 predictors, and a fraud label. The details of the training and test sets, including class imbalance levels, are detailed in Table 5.

As illustrated in Table 5, the Medicare data set combines the challenges of both big data and class rarity. The total number of fraudulent samples available for training (1206) will be further reduced to just 1085 after holding out 10% for validation. In the next section, we will discuss the methods used in this case study to address these challenges.

Methodology

Deep learning methods for addressing class imbalance are evaluated on the Medicare Part B data set by fitting models on the 80% training data and evaluating performance on the 20% test set. First, a validation phase holds out 10% of the training data to evaluate model performance and tune hyperparameters. Once optimal network settings are defined, models are fitted to the full training set and then applied to the test set. This protocol ensures that the test set does not influence hyperparameter tuning.

This section begins by discussing the runtime environment and deep learning frameworks used to carry out experiments. We then describe the DNN architectures and hyperparameters used throughout the experiments, as well as the class imbalance methods employed and their implementation details. Finally, we discuss the performance metrics and statistical analysis that are used to evaluate results.

Table 5 Training and test set details

Data set	Total samples	Fraudulent samples	% fraudulent
Training data	3,753,896	1206	0.032
Test data	938,474	302	0.032

Runtime environment

All experiments are conducted using a high-performance computing (HPC) environment running Scientific Linux 7.4 (Nitrogen) [73]. Jobs are dispatched onto CPU nodes with 20 Intel(R) Xeon(R) CPU E5-2660 v3 2.60GHz processors and 128GB of RAM. Neural networks are implemented using the Keras [19] open-source deep learning library written in Python with its default backend, i.e. TensorFlow [17]. Advantages of the Keras library include a simplistic Python API that enables fast prototyping, seamless compatibility with multiple backends and extensibility that allows for custom components, e.g. loss functions. The specific library implementations used in this study are the default configurations of *Keras 2.1.6-tf* and *TensorFlow 1.10.0*.

Baseline models

The baseline neural network architecture and its hyperparameters are discovered through a random search procedure that evaluates models on a validation set. The number of hidden layers, the number of neurons per layer, and regularization techniques are the primary focus of hyperparameter tuning in this study. With the 80% of Medicare Part B data set aside for training, stratified random sampling is used to holdout 10% validation sets for the purpose of identifying optimal network settings. Each set of hyperparameters is evaluated by training ten models, using a new random 10% holdout set to validate each model. We compare the effectiveness of each hyperparameter set by averaging the ROC AUC and loss scores of the ten models and visualizing results across 100 epochs. Experiments are restricted to deep neural networks, i.e. networks containing two or more hidden layers. Preliminary experiments sought a model with sufficient capacity to learn the training data, while successive experiments aimed to reduce overfitting and improve generalization to new data.

We use mini-batch stochastic gradient descent (SGD) with a mini-batch size of 256. Mini-batch SGD approximates the gradient of the loss function by computing the loss over a subset of examples. This is preferred over batch gradient descent because it is computationally expensive to compute the loss over the entire data set, and increasing the number of samples that contribute to the gradient provides less than linear returns [23]. It has also been suggested that smaller batch sizes offer a regularization effect by introducing noise into the learning process [74]. We employ an advanced form of SGD that adapts parameter-specific learning rates through training, i.e. the Adam optimizer, as it has been shown to outperform other popular optimizers [75]. The default learning rate of 0.001 is used along with default moment estimate decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The rectified linear unit (ReLU) activation function is used in all hidden layer neurons, and the sigmoid activation function is used at the output layer to estimate posterior probabilities [76]. The non-saturating ReLU activation function has been shown to alleviate the vanishing gradient problem and allow for faster training [22]. These settings yielded the most desirable validation results during preliminary experiments, and we therefore use these across all experiments.

The network topology was defined by first iteratively increasing architecture depth and width while monitoring training and validation performance. We determined that two hidden layers containing 32 neurons per layer provided sufficient capacity to overfit the

model to the training data. This was indicated by observing near-perfect performance on the training set and an increasing error rate on the validation set. We then explored regularization techniques to eliminate overfitting and improve validation performance. One way to reduce overfitting is to reduce the total number of learnable parameters, i.e. reducing network depth or width. L^1 or L^2 regularization methods, or weight decay, add parameter penalties to the objective function that constrain the network's weights to lie within a region that is defined by a coefficient α [23]. Dropout simulates the ensembling of many models by randomly disabling non-output neurons with a probability $P \in [0, 1]$ during each iteration, preventing neurons from co-adapting and forcing the model to learn more robust features [77]. Although originally designed to address internal covariate shift and speed up training, batch normalization has also been shown to add regularizing effects to neural networks [78]. Batch normalization is similar to normalizing input data to have a fixed mean and variance, except that it normalizes the inputs to hidden layers across each batch. Through monitoring validation results, we determine that dropout with probability $P = 0.5$ combined with batch normalization is most effective. Batch normalization is applied before the activation function in each hidden unit.

Table 6 describes the baseline architecture that we have selected for Medicare fraud detection. This multilayer neural network contains two hidden layers and 5249 trainable parameters. All experiments conducted in this study use this multilayer architecture with the learning parameters that have been defined in this section. To determine if network depth affects performance, we extend this model to four hidden layers following the same pattern, i.e. using 32 neurons, batch normalization, ReLU activations, and dropout in each hidden layer. For all class imbalance methods tested, we report results for both two-hidden-layer networks and four-hidden-layer networks across all experiments.

Class imbalance methods

We explore both data-level and algorithm-level methods for addressing class imbalance. The data methods that we use consist of altering the training data distribution with ROS and RUS. The algorithm-level methods modify the loss function to strengthen the impact of the minority class on model training. We also identify optimal decision thresholds for each model to improve overall class-wise performance.

Table 6 Baseline architecture

Layer type	# of neurons	# of parameters
Input	125	0
Dense	32	4032
Batch normalization	32	128
ReLU activation	32	0
Dropout $P = 0.5$	32	0
Dense	32	1056
Batch normalization	32	128
ReLU activation	32	0
Dropout $P = 0.5$	32	0
Dense	1	33
Sigmoid activation	1	0

Data-level methods

Data-level methods explored in this paper include ROS, RUS, and combinations of ROS and RUS (ROS–RUS). For each method, we adjust the sampling rates and create distributions with varying levels of class imbalance to better understand how class imbalance levels affect model training and classification performance. These distributions are listed in Table 7. The first row describes the training data prior to data sampling, and the remaining rows provide the size of the positive and negative classes after applying data sampling. $N_{train} = n_{neg} + n_{pos}$ denotes the total number of samples in the training set, where n_{neg} and n_{pos} correspond to the total number of negative and positive samples, respectively. The level of class imbalance within each experiment's training data is represented as the ratio of total negative samples to total positive samples, i.e. N_{neg}:N_{pos}. Of the 3,378,506 total Medicare claims available for model training, a mere 0.03% are labeled positive for fraud, i.e. 1085 fraudulent samples. This combination of big data and class rarity poses data sampling challenges that are often ignored in related works.

The RUS procedure employed in this paper consists of randomly sampling from the majority class without replacement. The sampled majority class is combined with all minority samples to create the training set. Hence, class imbalance levels within the training data are strictly determined by the size of the sample that is selected from the majority class. Creating a 50:50 class-balanced training set with RUS requires combining all positive samples from the training set with a randomly selected subset of 1085 negative samples. We vary the size of the sampled negative class to create class ratios of 99:1, 80:20, 60:40, 50:50, and 40:60. These distributions effectively cover minority class sizes between 1 and 50%, and then proceed to test what happens when the minority becomes the majority, i.e. 40:60.

One advantage of applying RUS is that the resulting training set size is reduced significantly, drastically decreasing the time required to train a model. This feature decreases turnaround time and allows for fast prototyping and hyperparameter tuning. Unfortunately, due to the extreme level of class imbalance in the Medicare data, very high reduction rates are required to create semi-balanced and balanced data sets. To create a class ratio of 99:1, which is still highly imbalanced, RUS-1 combines

Table 7 Varying levels of class imbalance with ROS and RUS

Method	n_{neg}	n_{pos}	N_{train}	n_{neg}:n_{pos}
–	3,377,421	1085	3,378,506	99.97:0.03
RUS-1	107,402	1085	108,487	99:1
RUS-2	4390	1085	5475	80:20
RUS-3	1620	1085	2705	60:40
RUS-4	1085	1085	2170	50:50
RUS-5	710	1085	1795	40:60
ROS-1	3,377,421	33,635	3,411,046	99:1
ROS-2	3,377,421	844,130	4,221,551	80:20
ROS-3	3,377,421	2,251,375	5,628,796	60:40
ROS-4	3,377,421	3,377,421	6,754,842	50:50
ROS-5	3,377,421	5,064,780	8,442,201	40:60
ROS–RUS-1	1,688,710	1,688,710	3,377,420	50:50
ROS–RUS-2	844,355	844,355	1,688,710	50:50
ROS–RUS-3	337,742	337,742	675,484	50:50

the positive group with a negative class sample that is just 3.18% of the original negative group. This reduces the size of the negative class training set from 3,377,421 to just 107,402, discarding millions of samples and likely depriving the model of valuable training data. As outlined in Table 7, the size of the negative class decreases as the levels of class imbalance decrease in methods RUS-2 through RUS-5. We expect to see high variance in performance when using RUS because the representation of the negative class in the training data will likely be very different from one run to the next. RUS results may be unpredictable and difficult to reproduce, as each model trained will learn a different subset of the majority class.

The ROS method employed in this paper consists of duplicating minority class samples until the desired level of class imbalance is achieved. Since there are many more non-fraud cases than there are fraud, the fraud cases must be over-sampled at high rates in order to balance out the class distributions. For example, creating a 50:50 class-balanced training set with ROS requires sampling the minority class at a rate of 3.112%. In other words, 3112 positive samples are created for every single positive instance, increasing the size of the minority class from 1085 samples up to 3,377,421 and approximately doubling the size of the training data set. The added data may improve model generalization, but at the cost of increased training times. This is especially exacerbated by big data and class rarity.

Finally, we combine ROS and RUS (ROS–RUS) to produce three class-balanced training sets. We test three ROS–RUS distributions, reducing the majority class by 90%, 75%, and 50% while simultaneously over-sampling the minority class until class balance is achieved. Higher reduction rates have the advantage of decreasing the size of the training set and improving efficiency. On the other hand, lower reduction rates preserve valuable information and provide a better representation of the majority class. For example, ROS–RUS-1 reduces the size of the majority class by 50% and ROS–RUS-3 reduces the size of the majority class by 90%, producing training set sizes of 3,777,420 samples and 675,484 samples, respectively. We expect ROS–RUS-1 to outperform ROS–RUS-3, as both experiments have 50:50 class-balanced distributions and ROS–RUS-1 has 5× more unique training samples than ROS–RUS-3.

Unlike plain RUS, the ROS–RUS method allows us to keep a greater percentage of the majority class, reducing the risk of discarding too many negative samples and under-representing the majority class. Since the majority group has been decreased in size through RUS, the over-sampling rate required to balance the classes is going to be less than would be required if using plain ROS. As shown in Table 7, the largest ROS–RUS training set has 3,377,420 samples, which is still smaller than the original training set. We find these methods most favorable when working with big data and class rarity, as they simultaneously maximize efficiency and performance.

We naively implement data sampling by creating a new training set from the original, where the new training set contains randomly duplicated minority samples, a random subset of the majority group, or a combination of the two and the respective over-sampling and under-sampling rates are defined by the desired class distribution. Alternatively, the same effect can be achieved by building the data sampling component into the mini-batch SGD implementation. When over-sampling with big data and class rarity, the latter implementation can significantly reduce memory requirements during training.

Algorithm-level methods

We assess the use of three algorithm-level methods that address class imbalance by modifying the loss function. Cost-sensitive learning is used to increase the importance of the positive class by incorporating class weights into the CE loss. Two new loss functions designed for class-imbalanced data, MFE [29] and FL [30], are also evaluated. Unlike data-level methods for addressing class imbalance, these methods do not alter the underlying class distributions.

The weighted CE loss (Eq. 2) is used to create a cost-sensitive deep neural network (CoSen), similar to the work by Wang et al. [59]. We consider two approaches for balancing the loss contributions made by each class, increasing the contribution of the positive class and decreasing the contribution of the majority class. Both approaches are defined by the estimated class priors, and their corresponding cost matrices are listed in Table 8. The Keras deep learning library provides built-in support for class weights, allowing these costs to be supplied to the training step in the form of (label, weight) pairs.

Cost-sensitive learning is similar to ROS and RUS in the sense that both methods increase or decrease the loss contribution of a particular class. For example, in ROS-4 we balance the contribution to the network loss by showing the model 3112 copies of each positive sample. In the CoSen-1 method, on the other hand, we multiply the loss generated from one sample by 3112. In both cases, we are increasing the loss that one sample generates by 3.112%. Unlike CoSen-1, however, ROS-4 has the advantage of training the model with batches that contain an equal number of positive and negative samples. On the other hand, models trained with CoSen-1 will see many mini-batches with no positive samples in them, resulting in many weight updates that are not influenced by the positive class. For this reason, we expect the cost-sensitive learning progress to be less stable than the data sampling methods.

In our second set of algorithm-level experiments, we address class imbalance by replacing the CE loss with the MFE loss (Eq. 5) [29]. This loss function helps to balance class-wise loss contributions by computing the loss as the sum of the average positive class error and the average negative class error. We also consider the variant proposed by Wang et al., the MSFE loss (Eq. 6). We have not provided a table summarizing these experiments, MFE and MSFE, because they do not contain any tunable hyperparameters.

The final algorithm-level method explored in this paper is the FL (Eq. 7) that was proposed by Lin et al. [30]. We expect this loss to down-weight the easily classified negative samples, allowing the difficult positive samples to contribute more to the loss and better influence weight updates. One important detail provided by Lin et al. is the use of class priors in initializing output layer bias weights, a strategy explained further in [23]. This initialization proved very important to our FL experiments, as preliminary experiments with default bias initialization yielded poor results (AUC < 0.70). Through observing training and validation scores, we found initializing output bias weights with the prior $\pi = 0.01$

Table 8 Cost-sensitive learning experiments

Method	w_{pos}	w_{neg}
CoSen-1	3112	1
CoSen-2	0.9997	0.0003

Table 9 Focal loss experiments

Method	α	γ
FL-1	0.25	2
FL-2	0.25	3
FL-3	0.25	4
FL-4	0.25	5

Table 10 Confusion matrix

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

to be most effective, and we used this value for all FL experiments. We use a fixed value for the FL weight parameter, i.e. $\alpha = 0.25$ while varying the modulating factor in the range $\gamma \in [2, 5]$, as outlined in Table 9.

Performance metrics

This study utilizes multiple complementary evaluation metrics to provide a clear understanding of model performance and class-wise score trade-offs [79]. The confusion matrix (Table 10) is constructed by comparing predicted labels to ground truth labels, where the predicted labels are dependent on output scores and the decision threshold.

We report the *true positive rate* (TPR), *true negative rate* (TNR), and *geometric mean* (G-Mean) scores on all experiments. The TPR (Eq. 8), or *Recall*, measures the percentage of the positive group that was correctly predicted to be positive, while the TNR (Eq. 9) measures the percentage of the negative group correctly predicted to be negative. Since the TPR and TNR scores are each derived from just one class, i.e. the positive or negative class, they are insensitive to class imbalance. For this same reason, reporting one without the other would be misleading and incomplete. For example, baseline models always predicting the negative class will have a TNR of 100%, but this model is useless as it fails to capture any of the positive class. The G-Mean (Eq. 10) summarizes a model's total predictive power by combining TPR and TNR.

$$TPR = \frac{TP}{TP + FN} \tag{8}$$

$$TNR = \frac{TN}{TN + FP} \tag{9}$$

$$G\text{-}Mean = \sqrt{TPR \times TNR} \tag{10}$$

The performance metrics listed thus far are all dependent on the decision threshold that is used to assign labels to output probability estimates. In this study, we find that a default threshold of 0.5 causes baseline models to always predict the non-fraudulent label. Therefore, we rely on the threshold-agnostic ROC AUC score to determine how well a model can discriminate between the positive and negative class. The ROC curve is

constructed by plotting the TPR against the *false positive rate* (FPR) over a range of decision thresholds, and the AUC is the area under the ROC curve. By the very derivation of the AUC score, if a model outputs class probability scores that produce reasonable AUC scores (> 0.70), then there must exist a decision threshold that will yield reasonable TPR and TNR scores.

We have found that the level of class imbalance within the training data has a significant impact on the range of output probability scores produced by neural networks. Therefore, we believe that selecting an optimal decision threshold using a validation set is a critical component of learning from class-imbalanced data. In the next section, we explain how optimal decision thresholds are identified.

Threshold moving

Through monitoring ROC AUC scores on baseline models during training and validation, we observe reasonable ROC AUC scores (> 0.70). However, consistent TPR and TNR scores of 0.0 and 1.0 suggested that the default decision threshold of 0.5 was too high, causing the model to always predict the negative class. To improve overall accuracy and better illustrate the efficacy of DNNs in detecting Medicare fraud, we apply threshold moving to each method independently.

Selecting an optimal decision threshold should be driven by the problem definition and requirements. For example, a cancer detection system will usually maximize recall because false negatives are life-threatening. In our Medicare fraud detection system we prefer a high TPR over a high TNR, as detecting fraud is more important than detecting non-fraud. Additionally, we wish to approximately balance the TPR and TNR rates in order to maximize the model's total predictive power. We use these goals to construct a procedure (Algorithm 1) for identifying optimal decision boundaries using validation data. For every experiment, the optimal decision threshold is calculated for each of the ten validation models, averaged, and then applied to the test set.

```
input  : targets y, output activations p
output: optimal threshold
best_thresh ← curr_thresh ← max_gmean ← 0;
delta_thresh ← 0.0005;
while curr_thresh < 1.0 do
    ŷ ← ApplyThreshold(y, p, curr_thresh);
    tpr, tnr, gmean ← CalcPerformance(y, ŷ);
    if tpr < tnr then
        return best_thresh;
    end
    if gmean > max_gmean then
        max_gmean ← gmean;
        best_thresh ← curr_thresh;
    end
    curr_thresh ← curr_thresh + delta_thresh;
end
return best_thresh;
```

Algorithm 1: Optimal Decision Threshold

The TPR, TNR, and G-Mean results presented in this study are dependent on this threshold selection procedure. Class-wise scores can be adjusted to increase or decrease bias towards the positive class by defining a new threshold selection procedure.

Significance testing

One-way ANOVA and Tukey's HSD test are used to estimate the significance of ROC AUC results with a significance level of $\alpha = 0.05$. ANOVA calculates a p-value from the between-method variance, within-method variance, and degrees of freedom. If $p < \alpha$, we reject the null hypothesis that method means are statistically equal and conclude that there exists a significant difference between class imbalance methods based on ROC AUC scores with 95% confidence. Tukey's HSD test is a multiple comparison procedure that determines which method means are statistically different from each other by identifying differences that are greater than the expected standard error. Methods are assigned to alphabetic groups based on the statistical difference of AUC means, e.g. group a is significantly different from group b.

Results and discussion

We present the average ROC AUC, TPR, TNR, and G-Mean scores for each set of experiments, grouped by method type, e.g. ROS, RUS, cost-sensitive, etc. When discussing method results, -2 or -4 are appended to method names to distinguish between network architectures containing two and four hidden layers, respectively. Within each group, the highest average AUC score is listed in bold font. The best methods from each group are then selected for further analysis. The ANOVA and Tukey's HSD tests are used to estimate the statistical significance of the results of these methods and to identify the best method for this problem with a confidence greater than 95%. Finally, training times, decision boundaries, and G-Mean scores are compared across the best methods from each group.

Baseline model performance

Table 11 lists the results of the baseline DNNs defined in the "Baseline models" section. To better establish a firm baseline for the 2012–2016 Medicare Part B fraud detection problem, we have included scores of three traditional machine learning algorithms. Table 12 lists AUC scores for LR, RF, and GBT learners, averaged across 10 runs of fivefold cross-validation. No class imbalance methods are applied when obtaining these baseline results.

The DNN Baseline-2 performed second best with an average ROC AUC of 0.8058, runner up to the LR learner with an average ROC AUC of 0.8076. The LR and DNN learners all outperformed the two tree-based learners. Baseline-2 outperforms Baseline-4 based

Table 11 Average baseline DNN results (30 runs)

Method	Hidden layers	Decision threshold	ROC AUC	TPR	TNR	G-Mean
Baseline-2	2	0.0002	*0.8058*	0.8280	0.6099	0.7088
Baseline-4	4	0.0003	0.8018	0.7488	0.7135	0.7301

Italic font indicates the maximum ROC AUC score

Table 12 Average results of traditional learners (10 runs)

	Logistic regression	Random forest	Gradient Boosted Tree
Avg ROC AUC	*0.8076*	0.7937	0.7990

Italic font indicates the maximum ROC AUC score

on average AUC scores, suggesting that increasing network depth does not improve results.

We would like to stress the importance of the decision threshold, noting that it was not until the threshold was decreased to 0.0002 and 0.0003 that the baseline DNNs achieved reasonable TPR and TNR. Using a default threshold of 0.5 causes the model to predict the negative class (non-fraud) for all test samples. If ROC AUC was not monitored and thresholding was not applied, this model would appear to be useless since the default threshold would predict all new samples to be non-fraudulent. We also observe that the optimal decision threshold is approximately the same as the minority class size, i.e. 0.03%. This relationship is investigated further in the "Analysis of decision thresholds" section.

RUS performance

Table 13 lists the results obtained when using RUS to vary the level of class imbalance within the training data. RUS-1-2, with a 99:1 class distribution, scored the highest of the RUS methods and outperformed all baseline learners, with an average ROC AUC of 0.8124 and G-Mean of 0.7383. RUS-2-2, with an 80:20 class distribution, did not perform as well as RUS-1-2, but it does outperform the baseline DNN models.

Results show that the average performance decreases as the level of class imbalance decreases through RUS, i.e. the size of the negative class available for training decreases. Recall that the Medicare Part B data set exhibits both big data and class rarity, and that creating class-balanced training sets with RUS requires discarding millions of negative samples. These results suggest that maintaining a sufficient representation of the majority class is more important than reducing the level of class imbalance, and that undersampling until classes are balanced can degrade performance. We continue to observe that two-hidden-layer networks outperform four-hidden-layer networks and that there exists a strong relationship between the level of class imbalance and the optimal decision threshold.

Since the AUC performance increases as the size of the majority class increases, two additional RUS experiments were conducted to determine if further increasing the majority class size will continue to increase performance. Following the same protocol as other experiments, class ratios of 99.5:0.5 and 99.9:0.1 are evaluated. Results in Fig. 1

Table 13 Average RUS results (30 runs)

Method	n_{neg}:n_{pos}	Hidden layers	Decision threshold	ROC AUC	TPR	TNR	G-Mean
RUS-1-2	99:1	2	0.0110	*0.8124*	0.7807	0.6987	0.7383
RUS-1-4		4	0.0145	0.8040	0.7581	0.7002	0.7265
RUS-2-2	80:20	2	0.2680	0.8076	0.7521	0.7163	0.7338
RUS-2-4		4	0.3520	0.7920	0.7674	0.6853	0.7228
RUS-3-2	60:40	2	0.4200	0.8043	0.7783	0.6700	0.7212
RUS-3-4		4	0.5370	0.7907	0.7978	0.6288	0.7021
RUS-4-2	50:50	2	0.4970	0.8027	0.7864	0.6601	0.7195
RUS-4-4		4	0.6078	0.7913	0.7778	0.6422	0.6966
RUS-5-2	40:60	2	0.5730	0.7994	0.7802	0.6588	0.7154
RUS-5-4		4	0.7060	0.7802	0.7226	0.6462	0.6412

Italic font indicates the maximum ROC AUC score

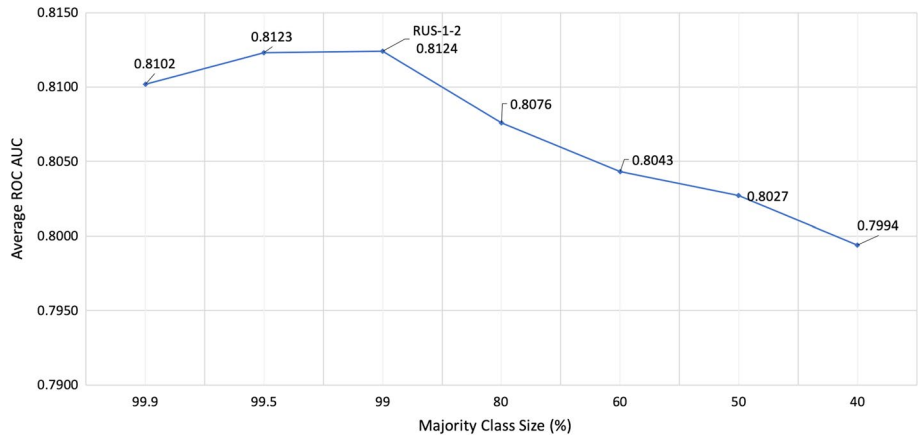

Fig. 1 RUS: majority class size vs average AUC (30 runs)

Table 14 Average ROS results (30 runs)

Method	n_{neg}:n_{pos}	Hidden layers	Decision threshold	ROC AUC	TPR	TNR	G-Mean
ROS-1-2	99:1	2	0.0110	0.8383	0.8572	0.6334	0.7338
ROS-1-4		4	0.0130	0.8325	0.8064	0.6857	0.7372
ROS-2-2	80:20	2	0.2410	0.8484	0.8282	0.6926	0.7549
ROS-2-4		4	0.3000	0.8440	0.8497	0.6165	0.7109
ROS-3-2	60:40	2	0.4080	0.8454	0.8056	0.7198	0.7582
ROS-3-4		4	0.4370	0.8438	0.8163	0.6820	0.7385
ROS-4-2	50:50	2	0.4530	*0.8505*	0.8084	0.7324	0.7692
ROS-4-4		4	0.4740	0.8389	0.8066	0.6861	0.7365
ROS-5-2	40:60	2	0.5630	0.8503	0.8163	0.7272	0.7701
ROS-5-4		4	0.5950	0.8423	0.8086	0.7023	0.7508

Italic font indicates the maximum ROC AUC score

show that increasing the size of the majority class beyond 99% does not improve performance beyond that of RUS-1-2. This suggests that both the class imbalance level and the representation of the majority class are important and that a validation set is required to find the best configuration.

ROS performance

Table 14 lists the results obtained by varying the training set's class distribution through ROS. Method ROS-4-2, with a 50:50 class distribution, performed the best with an average ROC AUC of 0.8505 and average G-Mean of 0.7692. ROS-4-2 also shows improvements to class-wise accuracy scores when compared to RUS, scoring a 0.8084 TPR and 0.7324 TNR. ROS-5-2 achieved similar results (0.8503 AUC) by over-sampling the minority class until there were more positive samples than negative samples, i.e. 60:40. ROS-1-4, with the highest level of class imbalance in its training set, performed the worst with an average ROC AUC of 0.8325, but still outperformed all RUS methods from Table 13.

Similar to related works by Hensman and Masko [54] and Buda et al. [52], our results suggest that over-sampling until class imbalance is eliminated from the training data is best for neural networks. We find that ROS outperforms RUS in all cases, which also supports related works [52, 57]. Networks with two hidden layers consistently outperform those with four hidden layers, and optimal decision thresholds increase near-linearly with the class imbalance level.

ROS–RUS performance

Results from the six ROS–RUS experiments are illustrated in Table 15. The Neg. Class Reduction column denotes the amount of the majority class that was discarded prior to applying over-sampling. For example, ROS–RUS-2 creates a 50:50 class distribution in the training data by first removing 75% of the negative class, and then over-sampling the positive class until they are balanced. All three ROS–RUS methods compete closely with the best ROS method and outperform both baseline and RUS learners. ROS–RUS-2-2 performs the best across all data-level methods with an average ROC AUC of 0.8509 and G-Mean of 0.7710. ROS-1-2 and ROS-3-2 perform nearly as well, with average AUC scores of 0.8500 and 0.8477, respectively. ROS–RUS-3, with the highest reduction rate, performs the worst of all the ROS–RUS methods.

ROS–RUS results continue to show that balanced training distributions yield better ROC AUC and G-Mean scores. Results also suggest that when working with big data containing millions of records, training with a sufficiently large random sample of the majority class will perform as well as the full majority class. For this data set, we observe that reducing the size of the majority class representation to just 10% (ROS–RUS-3) begins to degrade performance. The exact value of the reduction rate that will cause performance to degrade significantly is problem-specific, however, and will depend on various factors including class distributions and data redundancy.

One of the greatest achievements of the ROS–RUS methods is that they maintain model performance while drastically reducing training costs due to the reduced size of the training set. For example, ROS–RUS-2 methods train approximately 4× faster than ROS-4 methods. This allows for faster turnaround times during preliminary experiments and hyperparameter tuning and is particularly useful when working with big data.

Table 15 Average ROS–RUS results (30 runs)

Method	Neg. Class Reduction (%)	n_{neg}:n_{pos}	Hidden layers	Decision threshold	ROC AUC	TPR	TNR	G-Mean
ROS–RUS-1-2	50	50:50	2	0.5090	0.8500	0.8029	0.7354	0.7665
ROS–RUS-1-4			4	0.4820	0.8454	0.8064	0.7189	0.7597
ROS–RUS-2-2	75	50:50	2	0.5218	*0.8509*	0.7876	0.7553	0.7710
ROS–RUS-2-4			4	0.5140	0.8443	0.7992	0.7175	0.7526
ROS–RUS-3-2	90	50:50	2	0.4850	0.8477	0.8104	0.7209	0.7625
ROS–RUS-3-4			4	0.5020	0.8425	0.8063	0.7161	0.7585

Italic font indicates the maximum ROC AUC score

Table 16 Average cost-sensitive results (30 runs)

Method	w_{pos}	w_{neg}	Hidden layers	Decision threshold	ROC AUC	TPR	TNR	G-Mean
CoSen-1-2	3112	1	2	0.4760	*0.8075*	0.7574	0.7031	0.7283
CoSen-1-4			4	0.4480	0.8011	0.8008	0.6522	0.7213
CoSen-2-2	0.9997	0.0003	2	0.4780	0.8072	0.7529	0.7086	0.7291
CoSen-2-4			4	0.4825	0.8028	0.7265	0.7389	0.7318

Italic font indicates the maximum ROC AUC score

Table 17 Average MFE and MSFE loss results (30 runs)

Method	Hidden layers	Decision threshold	ROC AUC	TPR	TNR	G-Mean
MFE-2	2	0.0350	0.8041	0.7727	0.6828	0.7249
MFE-4	4	0.0290	0.8003	0.7741	0.6727	0.7200
MSFE-2	2	0.2890	*0.8065*	0.7862	0.6707	0.7251
MSFE-4	4	0.2980	0.8010	0.7715	0.6847	0.7257

Italic font indicates the maximum ROC AUC score

Cost-sensitive performance

Table 16 lists the average cost-sensitive learning results. Based on AUC scores, CoSen-1-2 slightly outperforms CoSen-2-2 with scores of 0.8075 and 0.8072, respectively. Based on G-Mean scores, however, CoSen-2-2 scores higher than CoSen-1-2 with scores of 0.7290 and 0.7283, respectively. There is similarly little difference between CoSen-1-2 and CoSen-2-2 TPR and TNR scores, and we conclude that these two methods perform approximately the same. These AUC scores are only marginally better than those of the baseline models. The networks comprised of two hidden layers continue to outperform those with four hidden layers.

One notable difference between the cost-sensitive method and the data-level methods is that the cost-sensitive method has produced an output decision boundary near 0.5. Similar to the ROS and RUS methods with 50:50 balanced distributions, the cost-sensitive method is completely balancing out the loss contributions made by the negative and positive class. This allows the model to receive weight updates equally from both classes and arrive at a default decision boundary of 0.5. We believe that this is dependent on the cost-matrices defined in Table 8 and that less balanced class-wise costs would produce a different decision boundary.

MFE and MSFE performance

Table 17 summarizes the MFE and MSFE loss function results. All four of the MFE and MSFE loss results have AUC scores in the range [0.8003, 0.8065] and G-Mean scores in the range [0.7200, 0.7257], i.e. there is very little difference in performance. MSFE-2 does perform the best with a AUC of 0.8065, and for both variants of the loss function the two-hidden-layer networks outperform their four-hidden-layer alternatives. Based on AUC scores, the CoSen-1-2 and LR learners perform better than the MFE and MSFE learners.

Table 18 Average focal loss results (30 runs)

Method	α	γ	Hidden layers	Decision threshold	ROC AUC	TPR	TNR	G-Mean
FL-1-2	0.25	2	2	0.0315	0.8015	0.8741	0.5202	0.6722
FL-1-4			4	0.0315	0.8019	0.8167	0.6264	0.7115
FL-2-2	0.25	3	2	0.0730	*0.8073*	0.7616	0.7019	0.7295
FL-2-4			4	0.0730	0.8020	0.7912	0.6595	0.7184
FL-3-2	0.25	4	2	0.1195	0.8071	0.7342	0.7309	0.7310
FL-3-4			4	0.1190	0.8025	0.7769	0.6781	0.7230
FL-4-2	0.25	5	2	0.1615	0.8072	0.7574	0.7018	0.7267
FL-4-4			4	0.1615	0.8030	0.7646	0.6952	0.7267

Italic font indicates the maximum ROC AUC score

Table 19 One-way ANOVA results (AUC)

Source	DF	Sum Sq	Variance	F-value	p-value
Between	6	0.0793	0.0132	1935	< 2.0e−16
Within	203	0.00138	7.00e−06		
Total	209	0.0807			

Despite the class imbalance levels in the training data, we observe that the optimal decision thresholds are significantly closer to 0.5 than those of the baseline models. This is expected, as these learners compute the loss as the sum of average false positive errors and average false negative errors, preventing one single class from dominating the training process.

Focal loss performance

Results in Table 18 show that the best FL score (AUC 0.8073) is achieved by FL-2-2 with $\gamma = 3$ and two hidden layers. The average AUC of FL-2-2 is nearly the same as that of CoSen-1-2, i.e. 0.8073 vs. 0.8075, and it shows a slight improvement over MSFE-2's AUC of 0.8065.

Adjusting the rate γ at which easy samples are down-weighted appears to have a minimal impact on the average performance, but increasing γ does move the optimal decision threshold closer to 0.5. If we were using a default decision threshold of 0.5, we would observe progressively better TPR and TNR results as γ increases. Unlike the results of Lin et al. [30], we do not observe the best results with parameters $\alpha = 0.25$ and $\gamma = 2$. We can see that these values do produce the strongest bias towards the positive class, however, as it yields very unbalanced TPR and TNR scores of 0.8741 and 0.5202, respectively. These results suggest that the use of FL will require additional hyperparameter tuning, as weighting parameters appear to be problem specific.

Statistical analysis

Area under the curve scores are used to select the best methods from each group for further analysis, i.e. Baseline-2, RUS-1-2, ROS-4-2, ROS–RUS-2-2, CoSen-1-2, MSFE-2, and FL-2-2. A one-way ANOVA test (Table 19) with a significance level of

Table 20 Tukey's HSD test results (AUC)

Method	Group	AUC	sd	Min	Max
ROS–RUS-2-2	a	0.8509	0.0038	0.8433	0.8591
ROS-4-2	a	0.8505	0.0038	0.8430	0.8594
RUS-1-2	b	0.8124	0.0030	0.8045	0.8170
CoSen-1-2	c	0.8075	0.0012	0.8048	0.8100
FL-2-2	c	0.8073	0.0013	0.8045	0.8102
MSFE-2	c	0.8065	0.0023	0.7972	0.8090
Baseline-2	c	0.8058	0.0013	0.8029	0.8080

Table 21 Average training time per epoch

Method	Time (s)	sd	N_{train}
RUS-1-2	1.9213	0.0294	108,487
ROS–RUS-2-2	31.0784	0.7683	1,688,710
MSFE-2	59.8842	1.2310	3,378,506
FL-2-2	62.3954	1.3034	3,378,506
Baseline-2	63.0775	1.8798	3,378,506
CoSen-1-2	65.4384	1.8057	3,378,506
ROS-4-2	128.2847	3.8514	6,754,842

$\alpha = 0.05$ is used to estimate the significance of the difference between method AUC scores. With $p < 2.0\text{e}{-6} < \alpha$, we can conclude that mean AUC results are significantly different between methods.

Tukey's HSD results (Table 20) further groups these select class imbalance methods into three distinct categories, i.e. a, b, and c. These groups are defined by the pairwise statistical differences between method AUC scores, and each group is statistically different from the other with a confidence of at least 95%.

ROS–RUS-2-2 and ROS-4-2 in group *a* obtain significantly higher scores than all other methods, with a mean AUC of 0.8509 and 0.8505, respectively. RUS-1-2, placed in group *b* with an average AUC score of 0.8124, performs significantly better than the baseline and algorithm-level methods. All algorithm-level methods perform approximately the same as the baseline DNNs, according to ROC AUC scores. Subsequent sections will compare these methods across additional criteria, as this method ranking is based solely on AUC scores.

Training time analysis

Table 21 lists the average time to complete one training epoch for each method, where averages are computed across 50 epochs. We have included the size of the training set, N_{train}, because this has the greatest influence on total training time. Other factors that will impact the total training time include network topology, activation functions, and loss functions, i.e. any hyperparameter that affects the total number of matrix

operations. Since all methods were trained for exactly 50 epochs, we can compare methods directly using the time to train one epoch.

Taking training times into consideration, we prefer ROS–RUS-2-2 over ROS-4-2 because AUC scores are statistically the same and ROS–RUS-2-2 trains approximately 4× faster. The speed up of ROS–RUS-2-2 is the result of the majority class being reduced by 75% before over-sampling the minority class. This produces a training set approximately 4× smaller than that of ROS-4-2 and 2× smaller than the baseline and algorithm-level methods. RUS-1-2 sees more than a 30× speed up in training when compared to baseline and algorithm-level methods.

Based on these timings and Tukey's HSD test, we find that combining ROS and RUS is very effective when training neural networks on big data with severe class imbalance. We suggest the use of plain RUS for preliminary experimentation and hyperparameter tuning, as RUS has been shown to outperform baseline and algorithm-level methods while providing significant improvements to turnaround times.

Analysis of decision thresholds

When comparing the ROS and RUS method results, and their varying class distributions, a relationship between the level of class imbalance within the training data and the optimal decision threshold is apparent. More specifically, when training networks comprised of two hidden layers with the cross-entropy loss, the learned decision boundary appears to fall near the minority class distribution size. For example, Baseline-2 has a minority class ratio of 0.0003, and the average optimal decision boundary calculated on the trained model is 0.0002. On the other hand, ROS-4-2 has a minority class ratio near 0.5, and the average optimal decision boundary was found to be 0.4530.

To add rigor to this observation, we fit linear models to this data with the *Ordinary Least Squares* [80] method. For each of the baseline, ROS, RUS, and ROS–RUS validation runs, i.e. 10 runs per method/architecture pair, the minority class ratio size is plotted against the calculated optimal decision threshold. Furthermore, we group this data by network topology to observe how architecture type impacts the learned decision boundary. These results are illustrated in Fig. 2 with 0.01 horizontal jitter and 95% confidence interval bands.

The two-hidden-layer networks show a strong linear relationship between the minority class size and the optimal decision threshold, with $r^2 = 0.987$ and p-value = 6.70e−132. The strength of this relationship is weakened slightly when the network depth is increased to four hidden layers, with an $r^2 = 0.964$ and p-value = 1.73e−102.

Figure 2 shows that the number of hidden layers impacts the learned decision boundary. Visually examining the optimal decision thresholds from the other methods, it is clear that the loss function also has a significant impact on the output decision boundary. For example, Baseline-2 and FL-2-2 are both fit to data with a minority size of 0.03%, but the FL-2-2 threshold is an order of magnitude larger than Baseline-2. In addition, we observed that some methods produce a larger between-class margin at the output layer. With a larger between-class margin at the output layer, the decision threshold will be more stable and the classifier will be more confident in its predictions.

We analyze the decision boundary margins further by calculating the range of acceptable decision thresholds for each method. For this problem, we loosely define this range

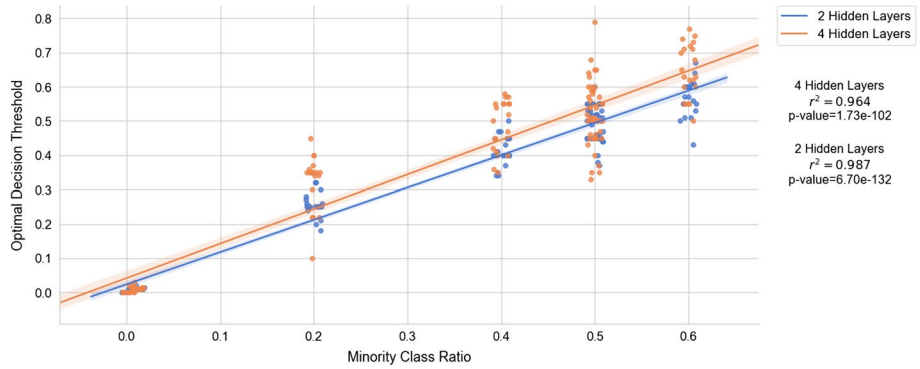

Fig. 2 Minority class ratio vs optimal decision threshold (CE loss)

as the difference between the minimum and maximum decision thresholds that satisfy $G\text{-}Mean > 0.7$. Using the best methods from each group, we calculate these thresholds using the decision threshold data from the validation step. We average the threshold range across each method's ten validation runs. Since the baseline learner's threshold is orders of magnitude smaller than those learned by the class imbalance methods, we present these results in Fig. 3 in logarithmic scale.

Through visualizing the approximate range of acceptable decision thresholds, we find that all three algorithm-level methods produce larger decision boundaries at the output layer than the baseline DNNs. We believe that these larger class-separating boundaries make the model more robust to threshold selection and should therefore improve class-wise performance scores. This is further supported by the average G-Mean scores presented in Fig. 4. We observe that algorithm-level methods yield higher average G-Mean scores when compared to the baseline models and that G-Mean scores improve overall as the decision boundary increases. The Baseline-2 model, which has the smallest margin, has the greatest G-Mean variance and lowest overall G-Mean scores. The methods with more balanced class distributions, i.e. RUS-1-2, ROS-4-2, and ROS–RUS-2-2, perform the best with the highest G-Mean averages and little variance. Of the three remaining algorithm-level methods, FL-2-2 performs the best, on average, based on the G-Mean scores. We conclude that, although average AUC scores between the baseline and algorithm-level methods are statistically the same, all three algorithm-level methods are preferred over the baseline as their decision boundary is more stable and should generalize better to new data.

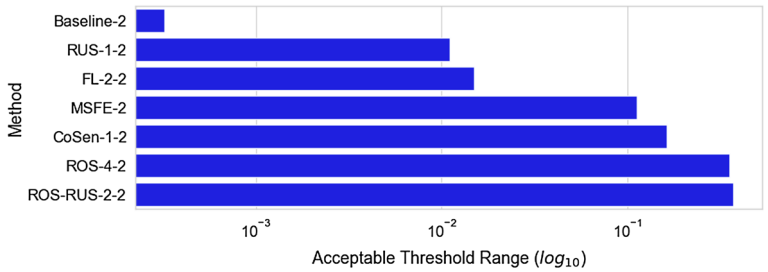

Fig. 3 Average decision threshold intervals (30 runs)

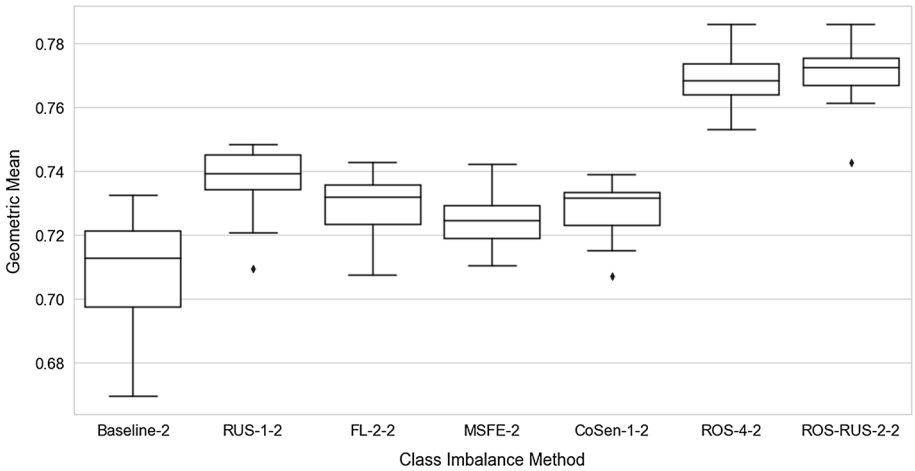

Fig. 4 Average G-Mean scores (30 runs)

Conclusion

The Medicare program provides affordable healthcare to more than 60 million U.S. residents. It has been estimated that Medicare loses between $20 and $70 billion per year to fraud, waste and abuse. This costs taxpayers billions of dollars and risks the well-being of its beneficiaries. In an effort to increase transparency and reduce fraud, CMS has made several Medicare data sets available to the public. Related works have shown that this big data is suitable for anomaly and fraud detection tasks, but that non-standard techniques are required to address the severe class imbalance. This study evaluates the performance of six deep learning methods for addressing class imbalance using CMS Medicare data with LEIE fraud labels. Additionally, we consider a range of class distributions and study the relationship between the minority class size and the optimal decision threshold. Through deep learning with methods for addressing class imbalance, we achieve the highest ROC AUC scores to date on the given CMS/LEIE data set.

Eliminating class imbalance from the training data through ROS or ROS–RUS outperforms all algorithm-level methods and baseline models, with average ROC AUC scores of 0.8505 and 0.8509. With 4× faster training times compared to baseline models, we conclude that deep learning with ROS–RUS is the preferred method for detecting fraud within the CMS Medicare data sets. RUS performs significantly better than algorithm-level and baseline methods using a class distribution of 99:1, but further decreasing imbalance levels with RUS degrades performance. Algorithm-level methods perform statistically the same as baseline methods, based on ROC AUC scores, but analysis of decision threshold intervals and G-Mean scores suggest that algorithm-level methods yield more stable decision boundaries than baseline models. A strong linear relationship is observed between the minority class size and the optimal decision threshold, suggesting that classification decision thresholds should always be optimized with a validation set when training neural networks with imbalanced data.

Future work in the area of class-imbalanced big data should reinforce these findings by comparing these methods across a variety of domains and data types. The ROS–RUS method can be improved by identifying more efficient techniques for determining

effective sample sizes. Regarding Medicare fraud detection, data quality can be improved by leveraging the NPPES registry to look up NPI numbers that are currently missing from the LEIE database. Provider specialty types can be converted from sparse one-hot vectors to dense embeddings that capture relationships between provider types, and HCPCS codes can be incorporated into the feature space in a similar manner. These latent semantic embeddings can be learned through various unsupervised deep learning methods [81]. Finally, several more advanced deep learning methods for addressing class imbalance can also be explored, e.g. dynamic sampling [50], LMLE [51], deep oversampling [82], and the class rectification loss [57].

Abbreviations

ANN: artificial neural network; ANOVA: analysis of variance; AUC: area under the curve; CSDNN: cost-sensitive deep neural network; DNN: deep neural network; FBI: Federal Bureau of Investigation; FCA: False Claims Act; FL: focal loss; FNE: false negative error; FPE: false positive error; FPR: false positive rate; FWA: fraud, waste, and abuse; GBT: Gradient Boosted Tree; LEIE: List of Excluded Individuals and Entities; LR: logistic regression; MFE: mean false error; OHEM: online hard example mining; OIG: Office of Inspector General; PUF: Public Use File; ReLU: rectified linear unit; RF: random forest; ROC: receiver operating characteristics; ROS: random over-sampling; RUS: random under-sampling; SGD: stochastic gradient descent; SVM: support vector machine; TNR: true negative rate; TPR: true positive rate; U.S.: United States.

Acknowledgements

We would like to thank the reviewers in the Data Mining and Machine Learning Laboratory at Florida Atlantic University. Additionally, we acknowledge partial support by the NSF (CNS-1427536). Opinions, findings, conclusions, or recommendations in this paper are solely of the authors' and do not reflect the views of the NSF.

Authors' contributions

JMJ performed the research and drafted the manuscript. TMK worked with JMJ to develop the article's framework and focus. TMK introduced this topic to JMJ. Both authors read and approved the final manuscript.

Funding

Not applicable.

Availability of data and materials

Not applicable.

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: 16 May 2019 Accepted: 7 July 2019
Published online: 18 July 2019

References

1. U.S. Government, U.S. Centers for Medicare & Medicaid Services. The Official U.S. Government Site for Medicare. https://www.medicare.gov/. Accessed 01 Feb 2019.
2. 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. Accessed 02 Feb 2019.
3. Centers for Medicare & Medicaid Services. Medicare enrollment dashboard. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Dashboard/Medicare-Enrollment/Enrollment%20Dashboard.html. Accessed 15 Mar 2019.
4. Morris L. Combating fraud in health care: an essential component of any cost containment strategy. Health Aff. 2009;28:1351–6. https://doi.org/10.1377/hlthaff.28.5.1351.
5. Coalition Against Insurance Fraud: by the numbers: Fraud Statistics. https://www.insurancefraud.org/statistics.htm. Accessed 02 Feb 2019.
6. Medicare fraud & abuse: prevention, detection, and reporting. Centers for Medicare & Medicaid Services. 2017. https://www.cms.gov/Outreach-and-Education/Medicare-Learning-Network-MLN/MLNProducts/Downloads/Fraud_and_Abuse.pdf. Accessed 20 Jan 2019.
7. Li J, Huang K-Y, Jin J, Shi J. A survey on statistical methods for health care fraud detection. Health Care Manag Sci. 2008;11:275–87. https://doi.org/10.1007/s10729-007-9045-4.
8. The Office of the National Coordinator for Health Information Technology: Office-based Physician Electronic Health Record Adoption. https://dashboard.healthit.gov/quickstats/quickstats.php. Accessed 03 Mar 2019.

9. The Office of the National Coordinator for Health Information Technology: Adoption of Electronic Health Record Systems Among U.S. Non-Federal Acute Care Hospitals: 2008–2015. https://dashboard.healthit.gov/evaluations/data-briefs/non-federal-acute-care-hospital-ehr-adoption-2008-2015.php. Accessed 03 Mar 2019.
10. Dumbill E. What is Big Data? : an introduction to the Big Data landscape. http://radar.oreilly.com/2012/01/what-is-big-data.html. Accessed 15 Nov 2018.
11. Ahmed SE. Perspectives on Big Data analysis: methodologies and applications. Providence: American Mathematical Society; 2014.
12. Centers For Medicare & Medicaid Services. Medicare fee-for-service provider utilization & payment data physician and other supplier public use file: a methodological overview. https://www.cms.gov/research-statistics-data-and-systems/statistics-trends-and-reports/medicare-provider-charge-data/physician-and-other-supplier.html. Accessed 20 Jan 2019.
13. Office of Inspector General. LEIE downloadable databases. https://oig.hhs.gov/exclusions/exclusions_list.asp. Accessed 20 Jan 2019.
14. 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. https://doi.org/10.1186/s40537-018-0151-6.
15. Van Hulse J, Khoshgoftaar TM, Napolitano A. Experimental perspectives on learning from imbalanced data. In: Proceedings of the 24th international conference on machine learning. ICML '07. ACM, New York, NY, USA. 2007. pp. 935–42. https://doi.org/10.1145/1273496.1273614
16. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521:436.
17. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. TensorFlow: large-scale machine learning on heterogeneous systems. 2015. http://tensorflow.org/. Accessed 01 Nov 2018.
18. Theano Development Team: Theano: a Python framework for fast computation of mathematical expressions. arXiv e-prints. 2016. arxiv:abs/1605.02688
19. Chollet F, et al. Keras. 2015. https://keras.io. Accessed 01 Nov 2018.
20. Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A Automatic differentiation in pytorch. In: NIPS-W. 2017.
21. Chetlur S, Woolley C, Vandermersch P, Cohen J, Tran J, Catanzaro B, Shelhamer E. cudnn: efficient primitives for deep learning. 2014.
22. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Neural Inf Process Syst. 2012. https://doi.org/10.1145/3065386
23. Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge: The MIT Press; 2016.
24. Witten IH, Frank E, Hall MA, Pal CJ. Data mining: practical machine learning tools and techniques. 4th ed. San Francisco: Morgan Kaufmann Publishers Inc.; 2016.
25. Johnson JM, Khoshgoftaar TM. Survey on deep learning with class imbalance. J Big Data. 2019;6(1):27. https://doi.org/10.1186/s40537-019-0192-5.
26. Herland M, Khoshgoftaar TM, Bauder RA. Big data fraud detection using multiple medicare data sources. J Big Data. 2018;5(1):29. https://doi.org/10.1186/s40537-018-0138-3.
27. Centers For Medicare & Medicaid Services. Medicare provider utilization and payment data: physician and other supplier. https://www.cms.gov/research-statistics-data-and-systems/statistics-trends-and-reports/medicare-provider-charge-data/physician-and-other-supplier.html. Accessed 20 Jan 2019.
28. United States Government Publishing Office: United States Code, Title 42—The Public Health and Welfare. https://www.govinfo.gov/content/pkg/USCODE-2016-title42/pdf/USCODE-2016-title42-chap7-subchapXI-partA-sec1320a-7.pdf. Accessed 02 Mar 2019.
29. Wang S, Liu W, Wu J, Cao L, Meng Q, Kennedy PJ. Training deep neural networks on imbalanced data sets. In: 2016 international joint conference on neural networks (IJCNN). 2016. pp. 4368–74. https://doi.org/10.1109/IJCNN.2016.7727770.
30. Lin T-Y, Goyal P, Girshick RB, He K, Dollár P. Focal loss for dense object detection. In: 2017 IEEE international conference on computer vision (ICCV). 2017. pp. 2999–3007.
31. Provost F, Fawcett T. Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In: Proceedings of the third international conference on knowledge discovery and data mining. 1999. pp. 43–8.
32. Gelman A. Analysis of variance: why it is more important than ever. Ann Stat. 2005;33(1):1–31.
33. Tukey JW. Comparing individual means in the analysis of variance. Biometrics. 1949;5(2):99–114.
34. Bauder RA, Khoshgoftaar TM. A probabilistic programming approach for outlier detection in healthcare claims. In: 2016 15th IEEE international conference on machine learning and applications (ICMLA). 2016. pp. 347–54. https://doi.org/10.1109/ICMLA.2016.0063.
35. 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. pp. 11–19. https://doi.org/10.1109/IRI.2016.11.
36. Friedman JH. Multivariate adaptive regression splines. Ann Stat. 1991;19(1):1–67.
37. Bauder RA, Khoshgoftaar TM, Richter A, Herland M. Predicting medical provider specialties to detect anomalous insurance claims. In: 2016 IEEE 28th international conference on tools with artificial intelligence (ICTAI). 2016. pp. 784–90. https://doi.org/10.1109/ICTAI.2016.0123.
38. Herland M, Bauder RA, Khoshgoftaar TM. Medical provider specialty predictions for the detection of anomalous medicare insurance claims. In: 2017 IEEE international conference on information reuse and integration (IRI). 2017. pp. 579–88. https://doi.org/10.1109/IRI.2017.29.
39. Bauder RA, Khoshgoftaar TM. The detection of medicare fraud using machine learning methods with excluded provider labels. In: FLAIRS conference. 2018.

40. Centers For Medicare & Medicaid Services. Medicare provider utilization and payment data: part D prescriber. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Part-D-Prescriber.html. Accessed 20 Jan 2019.
41. Centers For Medicare & Medicaid Services. Medicare provider utilization and payment data: referring durable medical equipment, prosthetics, orthotics and supplies. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/DME.html. Accessed 20 Jan 2019.
42. Feldman K, Chawla NV. Does medical school training relate to practice? evidence from big data. Big Data. 2015;3:103–13.
43. Centers for Medicare & Medicaid Services. Physician compare datasets. https://data.medicare.gov/data/physician-compare. Accessed 05 Feb 2019.
44. Ko J, Chalfin H, Trock B, Feng Z, Humphreys E, Park S-W, Carter B, Frick KD. Variability in medicare utilization and payment among urologists. Urology. 2015;85:1045–51. https://doi.org/10.1016/j.urology.2014.11.054.
45. Chandola V, Sukumar SR, Schryver JC. Knowledge discovery from massive healthcare claims data. In: KDD. 2013.
46. Branting LK, Reeder F, Gold J, Champney T. Graph analytics for healthcare fraud risk estimation. In: 2016 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM). 2016. pp. 845–51. https://doi.org/10.1109/ASONAM.2016.7752336.
47. National Plan & Provider Enumeration System. NPPES NPI Registry. https://npiregistry.cms.hhs.gov/registry/. Accessed 20 Jan 2019.
48. Khan SH, Hayat M, Bennamoun M, Sohel FA, Togneri R. Cost-sensitive learning of deep feature representations from imbalanced data. IEEE Trans Neural Netw Learn Syst. 2018;29:3573–87.
49. Lee H, Park M, Kim J. Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning. In: 2016 IEEE international conference on image processing (ICIP). 2016. pp. 3713–17. https://doi.org/10.1109/ICIP.2016.7533053.
50. Pouyanfar S, Tao Y, Mohan A, Tian H, Kaseb AS, Gauen K, Dailey R, Aghajanzadeh S, Lu Y, Chen S, Shyu M. Dynamic sampling in convolutional neural networks for imbalanced data classification. In: 2018 ieee conference on multimedia information processing and retrieval (MIPR). 2018. vol. 00. pp. 112–7. https://doi.org/10.1109/MIPR.2018.00027.
51. Huang C, Li Y, Loy CC, Tang X. Learning deep representation for imbalanced classification. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). 2016. pp. 5375–84. https://doi.org/10.1109/CVPR.2016.580.
52. Buda M, Maki A, Mazurowski MA. A systematic study of the class imbalance problem in convolutional neural networks. Neural Netw. 2018;106:249–59. https://doi.org/10.1016/j.neunet.2018.07.011.
53. Anand R, Mehrotra KG, Mohan CK, Ranka S. An improved algorithm for neural network classification of imbalanced training sets. IEEE Trans Neural Netw. 1993;4(6):962–9. https://doi.org/10.1109/72.286891.
54. Masko D, Hensman P. The impact of imbalanced training data for convolutional neural networks. Stockholm: KTH, School of Computer Science and Communication (CSC); 2015.
55. Krizhevsky A, Nair V, Hinton G. Cifar-10 (Canadian Institute for Advanced Research).
56. Chawla NV, Japkowicz N, Kotcz A. Editorial: Special issue on learning from imbalanced data sets. SIGKDD Explor Newsl. 2004;6(1):1–6. https://doi.org/10.1145/1007730.1007733.
57. Dong Q, Gong S, Zhu X. Imbalanced deep learning by minority class incremental rectification. IEEE Trans Pattern Anal Mach Intell. 2018;41:1367–81. https://doi.org/10.1109/TPAMI.2018.2832629.
58. Liu Z, Luo P, Wang X, Tang X. Deep learning face attributes in the wild. In: Proceedings of international conference on computer vision (ICCV). 2015.
59. Wang H, Cui Z, Chen Y, Avidan M, Abdallah AB, Kronzer A. Predicting hospital readmission via cost-sensitive deep learning. IEEE/ACM Trans Comput Biol Bioinform. 2018;15:1968–78. https://doi.org/10.1109/TCBB.2018.2827029.
60. Krizhevsky A, Nair V, Hinton G. Cifar-100 (Canadian Institute for Advanced Research).
61. 20 Newsgroups Dataset. http://people.csail.mit.edu/jrennie/20Newsgroups/. Accessed 15 Oct 2018.
62. Lin T-Y, Maire M, Belongie SJ, Bourdev LD, Girshick RB, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL. Microsoft coco: common objects in context. In: ECCV. 2014.
63. Fu C, Liu W, Ranga A, Tyagi A, Berg AC. DSSD : deconvolutional single shot detector. CoRR. 2017. arxiv:abs/1701.06659.
64. Shrivastava A, Sukthankar R, Malik J, Gupta A. Beyond skip connections: top-down modulation for object detection. CoRR. 2016. arxiv:abs/1612.06851.
65. Shrivastava A, Gupta A, Girshick RB. Training region-based object detectors with online hard example mining. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). 2016. pp. 761–9.
66. Nemoto K, Hamaguchi R, Imaizumi T, Hikosaka S. Classification of rare building change using cnn with multi-class focal loss. In: IGARSS 2018—2018 IEEE international geoscience and remote sensing symposium. 2018. pp. 4663–6. https://doi.org/10.1109/IGARSS.2018.8517563.
67. Centers for Medicare & Medicaid Services. National provider identifier standard (NPI). https://www.cms.gov/Regulations-and-Guidance/Administrative-Simplification/NationalProvIdentStand/. Accessed 01 Mar 2019.
68. Centers For Medicare & Medicaid Services. HCPCS general information. https://www.cms.gov/Medicare/Coding/MedHCPCSGenInfo/index.html. Accessed 20 Jan 2019.
69. Office of Inspector General. Exclusion authorities. https://oig.hhs.gov/exclusions/authorities.asp. Accessed 06 Feb 2019.
70. Guo C, Berkhahn F. Entity embeddings of categorical variables. CoRR. 2016. arxiv:abs/1604.06737.
71. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th international conference on neural information processing systems, vol. 2. NIPS'13. Curran Associates Inc., USA. 2013. pp. 3111–9.
72. Jayalakshmi T, Santhakumaran A. Statistical normalization and back propagation for classification. Int J Comput Theor Eng. 2011;3:89–93.
73. Linux S. About. https://www.scientificlinux.org/about/. Accessed 02 Jan 2019.
74. Wilson D, Martinez T. The general inefficiency of batch training for gradient descent learning. Neural Netw. 2004;16:1429–51. https://doi.org/10.1016/S0893-6080(03)00138-2.

75. Kingma DP, Ba J. Adam: a method for stochastic optimization. CoRR. 2015. arxiv:abs/1412.6980.
76. Lippmann RP. Neural networks, bayesian a posteriori probabilities, and pattern classification. In: Cherkassky V, Friedman JH, Wechsler H, editors. From statistics to neural networks. Berlin: Springer; 1994. p. 83–104.
77. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–58.
78. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd international conference on international conference on machine learning, vol. 37. ICML'15. 2015. pp. 448–56.
79. Seliya N, Khoshgoftaar TM, Van Hulse J. A study on the relationships of classifier performance metrics. In: 2009 21st IEEE international conference on tools with artificial intelligence. 2009. pp. 59–66. https://doi.org/10.1109/ICTAI.2009.25.
80. Zdaniuk B. In: Michalos AC, editor. Ordinary least-squares (OLS) model. Dordrecht: Springer; 2014. pp. 4515–17.
81. Bengio Y, Courville A, Vincent P. Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell. 2013;35(8):1798–828. https://doi.org/10.1109/TPAMI.2013.50.
82. Ando S, Huang CY. Deep over-sampling framework for classifying imbalanced data. In: Ceci M, Hollmén J, Todorovski L, Vens C, Džeroski S, editors. Machine learning and knowledge discovery in databases. Cham: Springer; 2017. p. 770–85.

Publisher's Note