MULTIPLE INPUTS NEURAL NETWORKS FOR MEDICARE FRAUD DETECTION

Mansour Zoubeirou A Mayaki
Université Côte d'Azur
CNRS, Inria, I3S
France

Michel Riveill
Université Côte d'Azur
CNRS, Inria, I3S
France

March 14, 2022

ABSTRACT

Medicare fraud results in considerable losses for governments and insurance companies and results in higher premiums from clients. Medicare fraud costs around 13 billion euros in Europe and between 21 billion and 71 billion US dollars per year in the United States. This study aims to use artificial neural network based classifiers to predict medicare fraud. The main difficulty using machine learning techniques in fraud detection or more generally anomaly detection is that the data sets are highly imbalanced. To detect medicare frauds, we propose a multiple inputs deep neural network based classifier with a Long-short Term Memory (LSTM) autoencoder component. This architecture makes it possible to take into account many sources of data without mixing them and makes the classification task easier for the final model. The latent features extracted from the LSTM autoencoder have a strong discriminating power and separate the providers into homogeneous clusters. We use the data sets from the Centers for Medicaid and Medicare Services (CMS) of the US federal government. The CMS provides publicly available data that brings together all of the cost price requests sent by American hospitals to medicare companies. Our results show that although baseline artificial neural network give good performances, they are outperformed by our multiple inputs neural networks. We have shown that using a LSTM autoencoder to embed the provider behavior gives better results and makes the classifiers more robust to class imbalance.

Keywords Medicare fraud detection · Anomaly detection · Imbalanced data · Machine learning · Deep neural networks

1 Introduction

The progress made in the field of big data and data management makes it possible to fight fraud more effectively in several business sectors such as finance, banking and insurance. Insurance fraud results in considerable losses for governments and insurance companies and results in higher premiums from clients. According to Insurance Europe, detected and undetected fraud would cost European customers and insurers around 13 billion euros [1] per year. In the field of medicare, in France the compulsory scheme detected 261.2 million euros of fraudulent services in 2018, mainly due to providers (opticians, pharmacists, medical auxiliaries, doctors, etc.) and institutions [2]. In the United States, according to the Federal Bureau of Investigation (FBI), fraud represents $5 - 10\%$ of medicare claims and costs insurance companies between 21 billion and 71 billion per year [3]. In a context where reducing management costs is a real issue for health insurers, the fight against fraud is a real expectation so that everyone receives a fair return for their contributions. Most insurance companies have business rule based fraud detection systems. These methods, although effective, are often very difficult to set up and maintain. Indeed, a rule-based fraud detection system constantly requires the presence of experts in the field and constant updates of the rules.

Models based on statistical methods and machine learning make it possible to automatically build patterns and thus detect fraudulent activities effectively. In the field of medicare, insurance companies receive requests for reimbursement (claims) sent by healthcare providers on behalf of their clients. For example, when an insured buys new corrective

lenses, the optician sends the invoice to his mutual medicare which covers a percentage of the costs (for example 90%). We find the same circuit in almost all other branches of insurance: auto insurance, business, home etc. The most common types of fraud include billing for appointments that the patient has missed, billing for services that are more complex than those performed, or billing for services not provided. Thus, in medicare, it happens that customers or healthcare providers send false claims to insurance companies in order to benefit from reimbursements. There are several other types of fraud that insurance companies face.

The main difficulty in applying machine learning techniques in fraud detection or more generally anomaly detection is that you don't have enough data labeled as anomalous or fraudulent. Thus, you end up in a situation of imbalanced class where one class is very poorly represented compared to the others. For example in medicare, fraudulent transactions often represent less than 5% of all transactions. The high imbalance rate makes it very difficult for machine learning algorithms to learn as they will tend to favor the majority class. Another challenge is that the labels are often not accurate. Some samples labeled fraudulent may not be real frauds and and vice versa.

In this study we use publicly available medicare data sets from the Centers for Medicare and Medicaid Services (CMS) for period 2017–2019 [4]. The data sets contain the hospitalization requests (Inpatient Data), the outpatient care requests (Outpatient Data) and the claims details. We also use the Office of Inspector General's list of excluded individuals and entities (LEIE) [5]. The LEIE table contains the list of healthcare providers excluded from the healthcare system for illegitimate or fraudulent activities. Since we are interested in fraud detection, we call the group of fraudsters the positive class and non-fraudulent group the negative class. The main challenge working with this data set is that it is highly imbalanced with a fraud rate between 0.038% and 0.074%. Another challenge is that it exhibits big data properties. Each year the CMS releases approximately 9 millions records from multiple sources, mixed-type and high-dimensional.

To deal with class imbalance issue, there are two main approaches with varying performance depending on the field of application and the complexity of the problem: the resampling approach (or data level) which consists in balancing the classes by adding or removing data from classes and the approach which consists in modifying the learning algorithms so that they take into account the class imbalance (algorithm level).

To detect medicare frauds, we propose a multiple inputs deep neural networks based classifier with a Long-short Term Memory (LSTM) autoencoder component. We call this architecture **MINN-AE**. This architecture makes it possible to take into account many sources of data without mixing them and makes the classification task easier for the final model. The LSTM autoencoder part of MINN-AE plays a dimension reduction role for the provider data and its latent vector describes the provider behavior over time. The rest of the paper is outline as follows. The **Related works** section discusses the other studies and articles related to imbalance data handling, deep learning for anomaly and medicare fraud detection. In section **Data sets**, we describe the data sets used in this study and the data pre-processing steps. The **Classifiers** section describes in details our models architecture, the loss functions we tested and the hyperparameters optimization steps. The results are presented and discussed in section **Results and discussions**.

2 Related works

The work presented here does not only concern research in the fields of medicare or fraud detection. We also present some techniques proposed to remedy the problem of class imbalance. These two concepts are inseparable because in fraud detection we always face the problem of class imbalance.

2.1 Medicare Fraud Detection and Resampling Methods

The Centers for Medicare and Medicaid Services (CMS) data has been used in numerous studies to detect medicare fraud. Most of these studies use resampling techniques to overcome the imbalance class issue (Bauder et al. [6]; Liu et al. [7]; Herland et al. [8]; Johnson et al.[3]; Van et al. [9]). In there study, Herland et al.[8] show that the combination of the three parts of CMS data makes it possible to detect more precisely fraudulent activities. They compared the performances of logistic regression, random forest and gradient boosting classifiers on each part of the data taken separately with those obtained grouping all the parts and results show that the performance of all classifiers improves dramatically using all parts of the data, and that logistic regression outperforms all other models. Using the CMS data from 2010, Liu et al. [7] added some geo-location information to detect fraud. They went from the hypothesis that medicare beneficiaries are senior, disabled or poor and prefer to choose the health service providers locating in a relatively short distance and if the distance between the providers location and the client living place is too long, it may imply a fraud. Bauder et al. [6] used three different classifiers to detect fraudulent medicare provider claims: C4.5 decision tree (C4.5), Support Vector Machine (SVM), and Logistic Regression (LR). They used the CMS data over the period 2012-2015 combined with the Office of Inspector General's list of excluded individuals and entities (LEIE) [5].

The authors also used random undersampling technique to handle the class imbalance problem. Their results show that the C4.5 decision tree and logistic regression classifiers have the best performance on detecting fraud. In their study, Johnson et al.[3] compared six resampling techniques for imbalanced classes using the CMS data over the period 2012–2016 [4]. These authors combined artificial neural network models with class imbalance techniques to predict fraud. They tested random undersampling (RUS), random oversampling (ROS), mean square error (MSE) and Focal Loss techniques among others. According to their results, RUS improves the performance of the classification algorithm if the majority class share is above 99%. The authors then conclude that maintaining sufficient representation of the majority class is more important than reducing the level of class imbalance, and that down-sampling until classes are balanced can deteriorate classification performance. Van et al. [9], in another study also compared different resampling techniques using 11 types of classifiers. In their experiments, they used 35 different data sets with degrees of imbalance (ratio between the number of samples in the minority class and that of the majority class) varying between 1.33% and 35%. The resampling techniques used in this article are: random undersampling (RUS), random oversampling (ROS), one-sided selection (OSS), cluster-based oversampling (CBOS), Wilson's editing (WE), synthetic minority oversampling technic (SMOTE) and borderline-SMOTE (BSM). To facilitate comparison of the results, they grouped the data sets into 3 categories according to the percentage of data in the minority class: $< 5\%$, between 5 % and 10 %, $> 10\%$. Their results show that whatever the category considered, the RUS technique tends to give better performances (32% of the time).

Most of these studies come to the conclusion that undersampling (down-sampling) is more efficient than over-sampling. These results go against what one might have expected as undersampling often leads to a loss of information. One possible explanation is that in some situations, adding new artificial data will add more noise than useful information to the model. Depending on the complexity of the problem (or data), it is necessary to test the two approaches (down-sampling and over-sampling) to see which one fits best.

2.2 Algorithm Level Methods for Imbalanced Classes

To overcome the problem of class imbalance, some authors propose to alter the learning algorithm in the way that it takes into account the problem (Wang et al. [10]; Haishuai et al.[11]; Lin et al.[12]) . The main idea of algorithm level method is to modify the learning algorithms so that they give more importance to the samples from the minority class which is often the class of interest.

Lin et al.[12] proposed an algorithm level method which consists in rewriting the classical entropy loss function by integrating two new parameters: α takes into account the imbalanced issue and γ (gamma) the complexity of classifying the samples. This new loss function called **Focal Loss** is obtained by multiplying the classical cross entropy (CE) by a modulation factor $\alpha(1-p)^\gamma$. hyperparameter $\gamma \geq 0$ adjusts the rate at which easy examples are down weighted and α is a class-wise weight used to give more importance to the minority class [13]. Lin et al. [12] applied their new cost function (Focal Loss) to object detection in images and their results show that this loss function gives better performance than most benchmark models. Wang et al. [10] proposed another algorithm level method called **mean false error** (MFE) which consist in decomposing the classical mean squared error (MSE) in two components in other to give more weights to the minority class samples. They rewrite the classical MSE as a kind of weighted average of the errors of the two classes. In this way, all the classes participate equally in the final loss function. Haishuai et al.[11] in their paper used an artificial neural network based model with a **cost matrix** to predict readmission of patients from a hospital. They defined a **cost matrix** such that the cost of misclassified readmission (False negative) is greater than that of misclassified non-readmission (false positive). This technique can be seen as an algorithm level method because during optimization, the model will tend to penalize more or give more weight to the minority class (readmission) samples in the loss function.

Algorithm level methods often give better results than data level methods as they don't alter the training data and don't lead to a loss of information. However in some situations, when you don't have enough data, oversampling can be a good way to extend your data set. Moreover, when the distribution of the samples in the majority class is stationary (the samples are very close to each other) undersampling may work very well as we don't loose lot of information by deleting some samples.

3 Data sets

The Centers for Medicare and Medicaid Services (CMS) [4] is an important source of data for research on fraud detection in medicare. The CMS publishes a series of publicly available data each year containing information on the use and payments of medical procedures, services and prescription drugs provided to beneficiaries as well as data on physicians and other actors in the healthcare system. These CMS data combined with the Office of Inspector General's list of excluded individuals and entities (LEIE) [5] containing the list of healthcare providers excluded from

the healthcare system for illegal activity allows researchers in the field of statistics and artificial intelligence to propose new methods to fight against fraud in medicare. One of the main challenges when working with these data sets is that they exhibit big data properties. Each year the CMS releases approximately 9 millions records from multiple sources, mixed-type features and high-dimensional. In this study we use publicly available medicare data sets from the Centers for Medicare and Medicaid Services (CMS) for the period 2017–2019. The dataset can be downloaded from the CMS website [4].

The CMS data sets contains mainly three types of information: hospitalization requests (Inpatient Data), outpatient care requests (Outpatient Data) and beneficiary information (Beneficiary Details Data). The Inpatient Data table contains information on patients admitted to hospitals. Table Outpatient Data gathers information on patients who have visited the hospital without being hospitalized there. These tables have been combined into a single table containing patient information, provider information and claims details (see Fig. 1). We thus have information such as the identifier of the beneficiary (BeneID), the identifier of the claims (ClaimID), the unique identifier of the healthcare provider (NPI), the refunded amount (AmtReimbursed) etc. Records within the data set also contain various provider-level attributes, e.g. National Provider Identifier (NPI), first and last name, gender, credentials, address etc. Fig. 1 shows the aggregating process used.
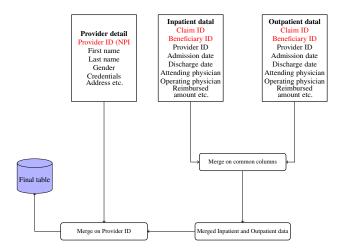

Figure 1: Data merging flow. The tables are aggregated using unique identifiers highlighted in red.

Note that for a fraudster provider, we do not know which of its claims are fraudulent and which are legitimate. To overcome this label issue, we considered that if a provider is fraudster, all its claims are also fraudulent [3]. This assumption makes sense because if a provider has been declared as a fraudster, the decision certainly comes from a deep analysis of his recent activity and his claims reflect illegal activities. We created additional features for the providers by aggregating the variables at the invoice level. For each provider we created new variables by taking the mean, the variance, the sum, the skewness coefficient of the numerical variables per trimester. In order to capture the provider behavior over time, we use a slicing technique called bucketing [11]. The behavior of each provider with respect to each variable can be considered as a time series. Indeed, over the years the provider makes several claims for different patients. For example, we can calculate the total amount paid by the insurance company to the provider each month. The bucketing technique consists of separating the claims from each provider into groups according to time and aggregation indicators are calculated in each group. There are two possible levels of aggregation: first order (first order features) and second order (second order features). The first order indicators are mean, standard deviation, variance, sum, maximum, minimum, skewness and kurtosis. The second order indicators are: Energy (E1), Entropy (E2), Correlation ($\rho_{x,y}$), Inertia and Local Homogeneity (LH). In this study, we only calculate the first order indicators for each numeric variable per trimester. The data of each provider over each year is separated into 4 blocks. In each block, the aggregation variables are calculated: mean, standard deviation, skewness, maximum, minimum, sum etc. After cleaning and preprocessing, the final data set has a fraud rate of 0.5%. Table 1 shows a subset of the provider level data and Table 2 a subset of the LEIE data.

Ultimately, we have three sources of data to predict fraud: the first table which contains all information on the claims (total amount, equipment, amount of each equipment etc.), the second table contains aggregated data per year for each provider (total amount on the year, number of each type of equipment, number of patients, etc.) and the third table contains aggregated data per trimester (total amount per trimester, number of clients per trimester etc.) for each provider.

Table 1: Subset of the agregated data set on provider level

NPI	BeneID count	DeductibleAmtPaid mean	InscClaimAmtReimbursed sum	Fraud
1000051001	24	213.60	104640	No
1000051101	117	502.16	605670	Yes
1000051005	138	2.08	52170	No
1000051007	58	45.33	33710	No
1000051009	36	53.86	35630	No

Table 2: Subsset of LEIE data set

NPI	CITY	STATE	EXCLTYPE	EXCLDATE
1306111111	GARDEN CITY	NY	1128a1	20180220
1306111111	WARREN	OH	1128b5	20190220
1306111111	PHILADELPHIA	PA	1128b7	20191231
1306111111	FLUSHING	NY	1128a1	20190220
1306111111	SPRINGFIELD	MO	1128b4	20200220

4 Methodology

In this section, we present our **MINN-AE** model and the other classifiers we tested. We compared MINN-AE to baseline artificial neural networks and non neural networks models.

4.1 State-of-the-art Classifiers

We compared the artificial neural network models to three state-of-the-art classifiers : logistic regression, random forest and gradient boosting. These three classifiers are good baseline models for classification tasks. The **logistic regression** makes it possible to predict the probability of an event happening (fraud value of 1) or not (non fraud value of 0) from the optimization of the regression coefficients. This result always varies between 0 and 1. When the predicted value is greater than a threshold, the event is likely to occur, while when this value is below the same threshold, it is not. **Random forest** makes an aggregation of several decision trees classifiers trained on slightly different data subsets. It thus makes it possible to have results that are robust to variations and generally gives good performance in inference. **Gradient boosting** is another method of aggregating decision trees. Decision trees are built sequentially by giving more and more weight to misclassified data [14]. The algorithm thus uses the errors of the present model to fit the future model. We also tested another implementation of Gradient boosting algorithm called **XGBoost** . The XGBoost implementation is computational efficiency and often better model performance. In this study, the hyper parameters were chosen using a grid search.

We chose these three classifiers because they are commonly used and provide reasonably good performance. We compare their performance to those of of artificial neural networks based classifiers. The selected hyper parameters are listed in the following table 3:

Table 3: State-of-the-art classifiers hyper parameters

Models	n_estimators	min_samples_split	max_features	max_depth	min_samples_leaf
Random forests	50	20	-	15	-
Gradient Boosting	80	20	auto	10	20

4.2 Baseline Artificial Neural Network

We first tested some baseline Multi-Layer Perceptrons (MLP) models consisting of a single input layer, multiple hidden layers, and an output layer. These models take an invoice as input and predicts if it's fraud or not. The number of layers and the number of neurons in each layer are variables (hyperparameters) that must be chosen carefully for neural network models to give good results. These variables remain constant throughout the training process and have a direct impact on the performance of the models. We refer to the baseline neural network as **BNN**. The BNN is a simple multilayers perceptron model where all the features are concatenated and used as input for the model. We tested some

version of BNN using the loss functions as describe in subsection 4.6. **BNN weighted** stands for BNN with weighted loss, **BNN focal** with focal loss, **BNN mfe** with the mean false error loss and **BNN rus** the best BNN obtained by random under sampling. The choice of hyperparameters is described in the next subsection 4.5.

4.3 Our MINN-AE Model's Architecture

MINN-AE is made up of two different inputs layers. The first input layer receives the data relating to the claims and the second input layer receives the data relating to the healthcare provider. The model is thus composed of two blocks which meet at the end. Each block consists of an input layer, hidden layers and an output layer. The outputs of the two blocks are then concatenated to form a single vector. Such an architecture makes it possible to simultaneously take into account data on claims and those on healthcare provider separately. In our version of the multi-input model, the second block is a Long-short Term Memory (LSTM) autoencoder. We first trained the LSTM autoencoder on the provider level data. This autoencoder learns to reconstitute a healthcare provider behavior over time. Then we used the latent vector from the LSTM autoencoder as an input vector for our final model. The final model is thus composed of an input layer which takes as input the claims details and another input layer which makes it possible to inject the latent vectors coming from the autoencoder. In this architecture, the autoencoder plays a dimension reduction role for the provider data and its latent vector describes the healthcare provider behavior. Note that the autoencoder parameters remain constant when learning the final model. The model's architecture is presented in Fig. 2 below.
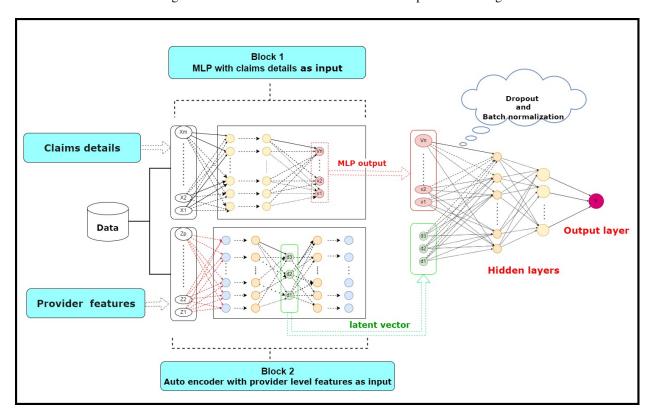

Figure 2: Visualization of the proposed neural network architecture. Block 1 receives features related to the invoice. Block 2 receives features related to the provider behavior and trains a LSTM based autoencoder. The latent vector of the autoencoder and the output of block 1 are concatenated and used as input for the next hidden layers of the model.

4.4 Performance Metrics

The classifiers are evaluated using the precision-recall curve (PRC). This plot shows precision values for corresponding recall values. It provides a model-wide evaluation like the receiver operating characteristic (ROC) plot or the cost curve (CC). The area under the curve (AUC) score of precision-recall curve, denoted as AUC (PRC), is likewise effective in multiple-classifier comparisons [15]. The AUC (PRC) measures the entire two-dimensional area under the entire precision-recall curve (by integral calculations) from (0,0) to (1, 1). We don't use AUC (ROC) as most studies do

because as Saito et al. [15] show in their study, the AUC (ROC) is not well suited in case of imbalance class. In their article [15], these authors showed that AUC (ROC) could be misleading when applied in imbalanced classification scenarios instead AUC (PRC) should be used. Their study showed via multiple simulations that AUC (ROC) fails to capture the variation in class distribution contrary to the AUC (PRC). As the AUC (ROC) is used as performance metric in most studies in the literature, we will give its value for each of our classifiers just as an indication. In order to have more detail on classifiers performance, we also compute the precision and the G-means score. The precision gives the performance of the classifier on the positive class and the G-mean metric makes a compromise between the true positive rate TPR or recall and the true negative rate (TNR).

$$TNR = \frac{TN}{TN + FP} \qquad Gmean = \sqrt{Recall \cdot TNR}$$

$$Recall = \frac{TP}{TP + FN} \qquad Precision = \frac{TP}{TP + FP}$$

4.5 Hyperparameters Optimization

We used the version of the mini-batch stochastic gradient descent (SGD) called SGD Adam with a batch size of 256. This optimizer allows to adapt the step of the gradient during the training of the model and is known to give better performances compared to other versions of SGD [3]. We kept the default values of the other hyperparameters: lr=0.001, $\beta_1 = 0.9$ and $\beta_1 = 0.999$. The rectified linear unit (ReLU) activation function is used in neurons of the hidden layers, and the sigmoid activation function is used for output layer to estimate posterior probabilities.

We choose the best hyperparameters using the **KerasTuner** library and on a small holdout set (10%) of the validation dataset. KerasTuner is an easy-to-use, scalable hyperparameter optimization framework that solves the pain points of hyperparameter search [16]. The AUC (PCR) metric is used to compare the set of hyperparameters. We also added between the hidden layers a Batch Normalization layer followed by a dropout layer. Batch Normalization [17] makes artificial neural network faster and more stable by normalizing and rescaling layers inputs. Dropout consists in "deactivating" randomly some neurons during training [18]. Each neuron being possibly inactive during a learning iteration, this forces each unit to "learn well" independently of the others and thus avoid overfitting. The best architecture for each model obtained using Keras tunner are listed in Table 4.

Table 4: Neural network architecture for each data set.

Data set	architecture	Hyper-parameters	Epochs
BNN	MLP	(45,25,1)	50
BNN weighted	MLP	(30,15,1)	50
BNN focal	MLP	(30,12,1)	50
BNN MFE	MLP	(25,10,1)	50
BNN RUS	MLP	(15,5,1)	50
MINN-AE (ours)	MLP	(20,10,1)+(15,1)	50
	LSTM autoencoder	(64,32,10,32,64)	30

4.5.1 Optimal Decision Threshold

The output of an artificial neural network is the posterior probability of a sample being fraudulent. To classify each sample, we need a decision threshold above witch the samples are labeled as fraudulent. The choice of the optimal decision threshold is made on the validation data during the training phase by cross validation. Thus during each iteration, we choose the optimal threshold by varying it between 0 and 1. We test several values between 0 and 1 and we choose the threshold which maximizes the G-Mean score on the validation data. At the end of the training, we end up with 10 threshold values corresponding to the optimal values of the 10 models trained during $k = 10$ folds (iterations). The 10 values are then averaged to have the final optimal threshold that will be used during inference [3].

4.6 Loss Functions

In this subsection, we describe the loss function tested with our classifiers.

4.6.1 Weighted Cross Entropy

This cost function integrates class-wise weights. The loss of each data is multiplied by the weight of the class it belongs to. The total cost function is written as follows:

$$Weighted\ CE\ loss = -\sum_{i=1}^{C} w_i P_i \log(P_i) \tag{1}$$

With w_i the weight associated to class i, P_i the probability of class i and C the total number of classes. The class weights are calculated as follows [19] :

$$class\_0 = \frac{1}{neg} \cdot \frac{total}{2} \quad class\_1 = \frac{1}{pos} \cdot \frac{total}{2}$$

$total$ refers to the total number of samples, pos numbre of samples in the positive class et neg number of samples in the negative class. Multiplying by the $\frac{total}{2}$ makes sure that the loss function keeps the same amplitude for each sample.

4.6.2 Focal Loss

To address class imbalance, Lin et al.[12] propose to rewrite the entropy function by integrating two new parameters: α takes into account the imbalanced issue and γ (gamma) the complexity complexity of classifying samples. This new loss function called Focal Loss is obtained by multiplying the cross entropy (CE) by a modulation factor $\alpha(1-p)^\gamma$. hyperparameter $\gamma \geq 0$ adjusts the rate at which easy examples are down weighted and α is a class-wise weight used to give more importance to the minority class [13]. The proposed loss function is written as follows:

$$FL(p) = \alpha(1-p)^\gamma \log p$$

For easy classified samples ($p- > 1$) the modulation factor tends towards 0 which reduces their importance in the final loss function. Moreover, if a sample is badly classified ($p- > 0$), the modulation factor is close to 1 and the cost function is little affected. The parameter γ therefore makes it possible to control the contribution of a sample in the final loss function according to its classification complexity. In this study, we don't aim to study this cost function, we will not dwell on the choice of the hyperparameters α and γ. We use the optimal parameters obtained by Johnson et al. in [3]: $\alpha = 0.25$ and $\gamma = 3$.

4.6.3 MFE Loss

The mean false error (MFE) cost function decomposes the mean squared error (MSE) into two components during optimization in order to give more weight to the minority class samples. The MSE is rewritten as a kind of weighted average of the errors of the two classes. In this way, all the classes participate equally in the final loss function. The final loss function is a sum of to means : **mean false positive error** (FPE) and **mean false negative error** (FNE).
$MFE = FPE + FNE$ and $MSFE = FPE^2 + FNE^2$

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

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

N is the number of samples in the negative class, **P** number of samples in the positive class, $d^{(i)}$ the true label of sample i, $y^{(i)}$ the predicted label for sample i.

4.7 Experimental Design

The dataset has been separated into a training dataset (80 %) and test (20 %) dataset. To avoid any risk of data leakage, we split the dataset according to the healthcare provider Id. Thus, if a provider is in the training subset, his claims are only used in the training steps. The models are trained using a k-folds cross-validation ($k = 10$ in our case). Each model is therefore trained and validated on 10 different sub-samples. The final performance is computed on the test set. Note that the test sets do not intervene at any time in the training steps. The final value of each performance metric is computed by taking the average of the ten measurements obtained during the ten iterations of the cross-validation. The final models are compared using the AUC (PRC) metric. We set the number of epochs to 50 for each model and used **early stopping** to avoid overfitting.

5 Results and discussions

In this section we present the results of our classifiers on data sets from 2017 to 2019 collected on the CMS website [4]. The classifiers performances metrics are listed in Table 5. Recall that we refer to the baseline neural network as **BNN** and our multiple inputs neural network as **MINN-AE**. **BNN weighted** stands for BNN with weighted loss, **BNN focal** with focal loss, **BNN mfe** with the mean false error loss and **BNN rus** the best BNN obtained by random under sampling. Despite the class imbalance in the training data, MINN-AE outperform all other classifiers in terms of AUC (PRC). Our model's AUC (PRC) is 0.745 and that of the second best classifier is 0.616. Note that the **no skill** (random) classifier has an AUC (PCR) of 0.03. Using the ROC (AUC) as performance metric, the baseline neural network with mean false error function (BNN mfe) has the best performance (0.864) but it has a very low precision (0.445). This is due to the fact this model fails to capture the provider behavior and the context of the data. The advantage of MINN-AE is that the autoencoder separates the providers into homogeneous groups and creates contextual features. In fraud detection the context matters. For example two providers can provide very similar claims but due to their previous behaviors (context) one will be considered fraudulent and the other one genius. State-of-the-art classifiers (logistic regression, random forest, Gradient boosting and XGBoost) perform worst than artificial neural network classifiers because they fails to capture complex structures in sequence datasets and large scale data [20]. Deep learning models have excellent capabilities in learning expressive representations of complex data such as high-dimensional data, temporal data and spatial data [21]. Our results suggest that using an LSTM autoencoder to embed the provider level features makes it easier for the neural network to separate fraudulent transaction from legitimate ones. The autoencoder acts like a dimensional reduction layer and also learns the provider behavior. The model is also robust toward the imbalance class due to the fact that the latent features extracted from the LSTM autoencoder have strong clustering power. The latent features allows the model to group the providers into clusters (see Fig. 3) and makes it easier to identify fraudulent behaviors. We can see on Fig.3 that the providers are clustered into homogeneous groups. This clustering was done using the **MeanShift** algorithm from **Scikit learn**. These results show that although it is important to take in account the provider level data in medicare fraud detection, it is also important to separate them from the features related to the claims when feeding an artificial neural network.

Table 5: Experimental results of the proposed method and some state-of-the-art methods. Mean time refers to the execution time expressed in minutes.

Models	Precision	AUC(ROC)	Gmean	AUC (PRC)	Mean Time
Random classifier	-	0	0.5	0.03	-
Logistic regression	0.438	0.827	0.826	0.629	0.51
Random forest	0.599	0.807	0.796	0.658	2.35
Gradient boosting	0.713	0.715	0.666	0.617	15.43
XGBoost	0.707	0.713	0.663	0.613	0.53
BNN	0.436	0.863	0.862	0.739	2.54
BNN weighted [3]	0.429	0.860	0.859	0.733	2.96
BNN focal [12]	0.432	0.861	0.861	0.737	2.50
BNN mfe [10]	0.445	0.864	0.863	0.741	6.03
BNN RUS	0.534	0.720	0.668	0.512	1.20
MINN-AE (Ours)	**0.770**	**0.794**	**0.762**	**0.765**	10

6 Conclusion

Medicare fraud results in considerable losses for governments, insurance companies and taxpayers. Frauds or anomalies are very rare events and difficult to detect. In most of the cases, the collected data are streaming time series data and due to their intrinsic characteristics it is a challenging problem to detect frauds precisely in them. Performance of traditional algorithms in detecting frauds is sub-optimal on large scale data since they fails to capture complex structures in the data. Deep learning based models have shown excellent capabilities in learning expressive representations of complex data such as high-dimensional data, temporal data, spatial data and graph data.

In this study, we proposed a deep neural networks with multiple inputs called **MINN-AE** to detect medicare frauds. Our model has a LSTM based autoencoder component that learns contextual features from the input data. Our results showed that this kind of architecture outperforms a classical multi-layer perceptron models using a single input layer. The model is also robust toward the imbalance class due to the fact that the latent features extracted from the autoencoder
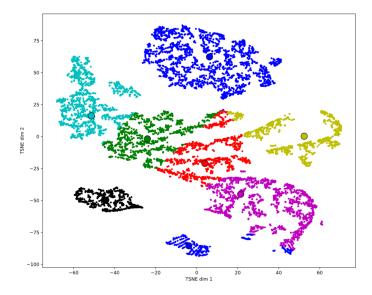
Figure 3: Mean-shift clustering on the autoencoder latent vector. This output of the autoencoder separates the providers into homogeneous groups.

have strong discriminating power. Future work will include employing the multiple inputs models with data sampling techniques or algorithm level techniques to combat the imbalanced nature of the data.

References

[1] Fraud prevention.

[2] Bilan 2018 des actions de lutte contre la fraude et actions de contrôles.

[3] Justin M Johnson and Taghi M Khoshgoftaar. Medicare fraud detection using neural networks. *Journal of Big Data*, 6(1):1–35, 2019.

[4] US and Government. Centers for medicare, medicaid services. medicare fee-for-service provider utilization , payment data physician and other supplier public use file: a methodological overview., 2018.

[5] US and Government. Office of inspector general. list of excluded individuals and entities, 2020.

[6] Richard A Bauder and Taghi M Khoshgoftaar. The detection of medicare fraud using machine learning methods with excluded provider labels. In *The Thirty-First International Flairs Conference*, 2018.

[7] Qi Liu and Miklos Vasarhelyi. Healthcare fraud detection: A survey and a clustering model incorporating geo-location information. In *29th world continuous auditing and reporting symposium (29WCARS), Brisbane, Australia*, 2013.

[8] Matthew Herland, Taghi M Khoshgoftaar, and Richard A Bauder. Big data fraud detection using multiple medicare data sources. *Journal of Big Data*, 5(1):29, 2018.

[9] Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th international conference on Machine learning*, pages 935–942, 2007.

[10] Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul J Kennedy. Training deep neural networks on imbalanced data sets. In *2016 international joint conference on neural networks (IJCNN)*, pages 4368–4374. IEEE, 2016.

[11] Haishuai Wang, Zhicheng Cui, Yixin Chen, Michael Avidan, Arbi Ben Abdallah, and Alexander Kronzer. Predicting hospital readmission via cost-sensitive deep learning. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(6):1968–1978, 2018.

[12] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[13] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27, 2019.

[14] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[15] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.

[16] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Keras tuner. `https://github.com/keras-team/keras-tuner`, 2019.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[18] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[19] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[20] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[21] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

[22] Justin M. Johnson and Taghi M. Khoshgoftaar. Robust thresholding strategies for highly imbalanced and noisy data. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1182–1188, 2021.

[23] Jiahe Yao, Siyu Yu, Changwu Wang, Tiejun Ke, and Hongjun Zheng. Medicare fraud detection using wtbagging algorithm. In *2021 7th International Conference on Computer and Communications (ICCC)*, pages 1515–1519, 2021.

[24] Robert K. L. Kennedy, Justin M. Johnson, and Taghi M. Khoshgoftaar. The effects of class label noise on highly-imbalanced big data. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1427–1433, 2021.