Journal of Big Data

RESEARCH

Open Access

Explainable machine learning models for Medicare fraud detection

John T. Hancock[1*], Richard A. Bauder[1], Huanjing Wang[2] and Taghi M. Khoshgoftaar[1]

*Correspondence:
jhancoc4@fau.edu

[1] College of Engineering and Computer Science, Florida Atlantic University, Boca Raton, USA
[2] Ogden College of Science and Engineering, Western Kentucky University, Bowling Green, USA

Abstract

As a means of building explainable machine learning models for Big Data, we apply a novel ensemble supervised feature selection technique. The technique is applied to publicly available insurance claims data from the United States public health insurance program, Medicare. We approach Medicare insurance fraud detection as a supervised machine learning task of anomaly detection through the classification of highly imbalanced Big Data. Our objectives for feature selection are to increase efficiency in model training, and to develop more explainable machine learning models for fraud detection. Using two Big Data datasets derived from two different sources of insurance claims data, we demonstrate how our feature selection technique reduces the dimensionality of the datasets by approximately 87.5% without compromising performance. Moreover, the reduction in dimensionality results in machine learning models that are easier to explain, and less prone to overfitting. Therefore, our primary contribution of the exposition of our novel feature selection technique leads to a further contribution to the application domain of automated Medicare insurance fraud detection. We utilize our feature selection technique to provide an explanation of our fraud detection models in terms of the definitions of the selected features. The ensemble supervised feature selection technique we present is flexible in that any collection of machine learning algorithms that maintain a list of feature importance values may be used. Therefore, researchers may easily employ variations of the technique we present.

Keywords: Big Data, Class imbalance, Explainable machine learning models, Ensemble supervised feature selection, Medicare fraud detection

Introduction

Highly dimensional Big Data can be a challenge to work with, since brute-force approaches are not practical. For example, in the process of feature selection, with smaller datasets that have a few attributes, one may simply try all possible combinations of features, build a model with each combination, and select the model that performs the best. This approach quickly becomes impractical, especially as the number of features in the dataset grows. Put another way, high dimensionality may cause issues with machine learning model performance such as a diminished capacity to generalize and longer training times. Hence, feature selection techniques are a popular topic of research in machine learning for Big Data application domains. For more information on features

selection techniques, please see [1]. Another benefit of applying feature selection to the modeling process is that it yields more explainable models. Models with fewer features are easier to explain, since, at the very least, there are fewer factors to consider when theorizing about their effect on the dependent variable. In this study, we describe and apply an ensemble of supervised feature selection techniques. To the best of our knowledge, we discovered this feature selection technique, and we are certainly the first to apply it to the Medicare insurance fraud detection application domain. We show that our technique outperforms the baseline scenario of building models that use all features.

We apply our feature selection technique to build machine learning models for the automated detection of Medicare insurance fraud. Medicare is a public health insurance program in the United States. It is primarily tasked to insure individuals aged 65 and older. The Centers for Medicare and Medicaid Services (CMS) [2], is the institution responsible for overseeing the Medicare program. The CMS encourages research by maintaining publicly accessible repositories of Medicare insurance claims data. Records in the claims data have dozens of attributes. Therefore, feature selection is an appropriate subject of research that involves this data.

The principal sources of data for our investigation are two Medicare Plans, Medicare Part B, which covers treatments and procedures, and Medicare Part D, which covers prescription medications. The size and the rate at which these datasets grow reflects the flow of insurance claims submitted to CMS by healthcare providers. At present, fraudulent claims can go unnoticed, enabling dishonest providers to exploit the system. Even if only a small percentage of claims are fraudulent, the substantial volume of claims still translates to large amounts of money. In 2019, approximately three billion dollars were reclaimed from fraudulent activities by the Department of Justice [3]. However, the total amount lost to fraud remains uncertain as the CMS reports "improper payments," which encompasses both fraudulent and mistaken payments [4]. In 2019, the CMS reported about $100 billion in improper payments. For more details on the characteristics of criminal Medicare fraud activity, please see [5].

Not only is the data provided by CMS highly dimensional, in the sense that it has many attributes, but also it contains many records, since on the order of millions of records are added annually. An effective feature selection technique is therefore highly desirable, since that reduces the overall size of the data that must be processed.

Automated, reliable fraud detection could help the CMS estimate the proportion of improper payments due to fraud, providing a solid foundation for law enforcement to recuperate stolen funds. Our interest in the Medicare fraud detection application domain is focused on employing machine learning techniques to detect Medicare fraud, contributing towards the overall aim of automated Medicare fraud detection. Enhancing fraud detection capabilities could lead to the more efficient use of government funds, and potentially lower taxes as a result of reduced program costs. At the same time, we would like to be able to prove that the fraud detection process is fair, something that can only be accomplished with explainable machine learning models. Models with fewer features are more explainable. Therefore, the feature selection technique we present here is a method for building more explainable models.

In order to show our technique is viable and should be the subject of ongoing research, we compare the performance of models built after applying our feature selection

technique to the performance of models built with all features of the datasets. We show that the models we build on datasets where the feature selection technique is applied to outperform models built with all features. Therefore, this exposition of our feature selection technique constitutes a contribution to the field of machine learning applied to highly imbalanced Big Data, that future researchers can employ to reduce the size of the datasets they work with, and achieve more explainable models. In order to show the effectiveness and benefits of our feature selection technique, we have devised a study with the following sections: a survey of related work, a discussion of the datasets used, a discussion of the machine learning algorithms used, a description of our experimental methodology, including the feature selection technique, presentation of results, statistical analysis, and conclusions.

Related work

We explore the extensive research conducted in the field of machine learning on the subject of feature selection, with an emphasis on their applications in fraud detection. Although the body of work specifically addressing Medicare fraud detection is somewhat limited, we have incorporated studies examining other forms of fraud detection for a comprehensive understanding. The methodologies employed in these studies on fraud detection apply to the task of Medicare fraud detection. Although there are related studies, we find our study stands apart for its exposition of a novel feature selection technique, and its use of statistical analysis to prove the benefits of applying the technique.

Mayaki and Riveill [6] compile CMS data from the years 2017 to 2019, and label it with the List of Excluded Individuals and Entities (LEIE) [7] to form a dataset for supervised machine learning. They then build a model for detecting Medicare fraud in their dataset, which they named "Multiple Inputs Neural Network Auto-Encoder" (MINN-AE). The auto-encoder component of MINN-AE is a Long-Short Term Memory (LSTM). Auto-encoders have been successfully employed in other application domains involving the classification of highly imbalanced data [8]. Mayaki and Riveill measure the effectiveness of MINN-AE against Logistic Regression, Random Forest, Gradient Boosting, XGBoost, and five other artificial neural network models. The evaluation metrics reported in the study include precision, AUC, Area Under the Precision Recall Curve (AUPRC) [9], and geometric mean. In alignment with our views on AUC versus AUPRC, Mayaki and Riveill write about the usefulness of AUPRC when compared to AUC for datasets that are highly imbalanced. Their findings demonstrate that MINN-AE surpassed the performance of the other nine models. Despite this, the authors neglected to provide detailed information about the experimental dataset, including the count of instances and features. Moreover, unlike our study, Mayaki and Riveill do not discuss a feature selection technique.

Waspada et al. [10] apply a supervised feature selection technique, Random Forest, to rank the features in the Kaggle Credit Card Fraud Detection Dataset [11]. This dataset is similar to the Medicare Part B and Part D datasets that we use here because it is highly imbalanced. They carry out experiments with many factors. In the combination that yields the best results, they report that the top five features yield the best performance. Waspada et al. report results in terms of several metrics, including AUPRC. We agree with their conclusion that AUPRC is a more reliable metric for evaluating the

classification of highly imbalanced big data. However, in the multifactorial experiments that Waspada et al. conduct, we do not find a report of a statistical analysis performed to determine the significance of the effects of the factors. In our study we conduct a statistical analysis of the impact of our feature selection technique on experimental outcomes to give a clear idea of how varying the number of features used in experiments effects classification scores. Similar to Waspada et al., we use a supervised feature selection technique. However, ours is an ensemble feature selection technique, and we present a novel method for combining multiple supervised feature selection techniques. For an overview of ensemble feature selection methods, please refer to [12].

In their research, Sailaja et al. [13] perform experiments with CMS Medicare data from 2012–2015. They label the data with the LEIE. From the CMS and LEIE data, they compile a dataset with 37,147,213 instances and nine features. Their final dataset contains 3,331 instances marked as fraudulent. Therefore, their dataset exhibits a high class imbalance, with 0.008967% of instances being fraudulent. The machine learning algorithms Sailaja et al. use are Decision Tree, Support Vector machine (SVM), and Logistic Regression. They report that Decision Tree outperformed both SVM and Logistic Regression models in overall effectiveness. Moreover, it was found that Decision Tree worked best with an 80:20 class distribution when Random Undersampling was applied to address the class imbalance in their dataset. Sailaja et al. use Area Under Receiver Operating Characteristic Curve (AUC) [14] as the metric to evaluate the results of their experiments. Over the course of our research, we have found that AUC is a misleading metric for evaluating the results of classification experiments involving highly imbalanced big data [10].

Gupta et al. [15] conducted a comparative study aimed at detecting fraudulent cases in Indian health insurance claims data. They observed that certain features within the dataset showed strong correlations with each other. To tackle this, they removed one feature from each correlated pair and used the remaining subset for fraud detection. Gupta et al.'s feature selection technique is not an ensemble technique, such as the one we present in our study. While removing correlated features is a sensible step for preparing a dataset for classification, calculating feature correlation does not provide one with a ranking of the features in a dataset that enables one to intelligently remove features from a dataset. Our study provides an exposition of an ensemble feature selection technique, which is flexible and extensible in the sense that any technique that provides a ranking of features can be incorporated in it. Furthermore, since it provides an ordered list of features as a result, one may leverage it to control the size of the feature set for experiments. Feature correlation does not provide a sense of what features may be removed without impacting performance, whereas a ranking informs one of which features are less important, and hence may be discarded without negatively effecting classification scores.

Herland et al. [16] focus on the detection of Medicare fraud with the use of datasets derived from the CMS's publicly available data. They use Medicare Physician & Other Practitioners—by Provider and Service (Part B) [17] data from the years 2012–2015, Medicare Part D Prescribers—by Provider and Drug (Part D) [18] data from the years 2013–2015, and data from a third part of Medicare, known as Medicare Durable Medical Equipment, Devices & Supplies—by Referring Provider and Service (DMEPOS) [19] from the years. 2013–2015. Furthermore, Herland et al. compile a fourth dataset, known

as the Combined dataset, by merging the Part B, Part D, and DMEPOS datasets. All four of the datasets are documented as highly imbalanced in their study. They provide details on the data processing methods for each of the four datasets, and they demonstrate how to label the datasets with the LEIE. Herland et al. build classification models with Logistic Regression, Random Forest, and Gradient Boosting classifiers for all four datasets. Their results show that the Combined dataset, when used with Logistic Regression, yields the best overall performance in detecting fraud. A key distinguishing factor between our studies is that Herland et al. do not employ feature selection techniques. Moreover, we only find results reported in terms of AUC, which, as mentioned previously, we find to be a misleading metric for classifying imbalanced Big Data.

Our review of related work leads us to the conclusion that the documentation of our work in the form of a study represents a contribution. We are the first to present the application of a new feature selection technique in the Medicare insurance fraud detection application domain. Moreover, we are the first to offer an explanation of our model's results in terms of the reduced feature set that results in applying our ensemble supervised feature selection technique.

Datasets

The datasets in this study are compiled from information provided by the CMS and the United States Office of Inspector General (OIG). First, we discuss characteristics of the CMS Medicare data. Next, we discuss how we aggregate it, as a preprocessing step. Later we describe the labeling process, which involves the data from the OIG. We utilize two primary sources for data from the CMS in our study. They are both Medicare Health insurance plans. One plan is known as Part D, and the other is known as Part B. In the context of health insurance, a plan is simply the agreement between the insurer and the insured as to what things are covered under the insurance policy. Part D covers prescription medications, and Part B covers treatments and procedures. CMS makes different raw data available for both programs, however, we use the same technique to compile both sources into datasets suitable for supervised machine learning. This technique also involves a third source of data, from the United States Office of Inspector General which we use for labeling. The third source is the List of Excluded Individuals and Entities (LEIE) [7]. The datasets used in this study are compiled in the manner described in [20].

We obtain the Part D and Part B data from on-line sources. The CMS website offers a user interface as well as an application programming interface for examining these datasets, and to carry out rudimentary exploration of data. We acquired the Medicare datasets from the CMS site in a comma-separated format files as the basis for our datasets. The datasets are available to the public, for download. We use data spanning the years 2013–2019. The CMS provides supplementary documentation that explains the Medicare data. We utilize publicly available, CMS-provided methodology documents detailing their data gathering and processing methods. We also use CMS-provided data dictionaries that explain all accessible attributes [21–24].

We use two sources for the Part D data. The first is Medicare Part D Prescribers—by Provider and Drug [18], and the second is Medicare Part D Prescribers—by Provider [25]. The key difference between the two sources is the level of specificity of the data. The first source, Medicare Part D Prescribers—by Provider and Drug has a record for

every combination of health care provider, medication that the health care provider prescribes, and year. We refer to this as the "provider-drug-level Part D data". The second source, Medicare Part D Prescribers—by Provider is less specific. It contains a record for each provider for each year. We refer to this as the "provider-level Part D data".

The provider-drug-level Part D data has 22 attributes. Not all of these attributes are relevant for machine learning. These are attributes related to the provider's name and address. We eschew these attributes since they could form a unique identifier that a machine learning model could memorize instead of properly generalizing the data. We retain one identifier, the provider's national provider identifier (NPI), which we use later for labeling purposes. The provider-drug-level Part D data has two categorical features, that identify the type of medication prescribed. During the aggregation phase of our dataset compilation, we discard these categorical features. Another feature which is ultimately discarded, but useful for processing is the year in which the claim was made. We use this feature for aggregating the Part D provider-drug-level data by year, but we do not use it as an attribute for supervised machine learning. Since it is at the provider level, when we aggregate records, we retain a categorical feature for the provider type. Numeric features in the provider-drug-level part D data are readily useful for supervised machine learning. These include data on the total volume and frequency of prescriptions a provider submits claims for, the number of patients, as well as the total cost of the claims. Furthermore, there are similar, additional features for patients aged 65 and over. There are approximately 174 million records in the collection of provider-drug-level Part D data files.

The provider-level Part D data contains 51 additional attributes pertaining to claim the provider submits to Medicare, across all the medications the provider prescribes for the year. They are listed in Table 1. The feature descriptions we provide here are from the provider-level Part D data dictionary [24]. The provider-level Part D data has ten features of summary statistics about the beneficiaries of the claims the provider submits. There is also an average beneficiary risk score. The score is calculated with a model that adjusts risk based on hierarchical condition categories (HCC). As per CMS's methodology, beneficiaries possessing risk scores higher than the average of HCC score of 1.08 are projected to have Medicare spending that exceeds the average. The provider-level Part D data also has features for the total number of claims, total number of 30-day prescription orders, total drug cost, total day's supply dispensed, and the total number of beneficiaries seen, in the form of subtotals within various categories of claims. The categories are Low-Income Subsidy (LIS) claim, Medicare Advantage Prescription Drug Plan (MAPD) coverage claims, and Medicare Prescription Drug Plan (PDP) claims. The statistics are also divided by several drug categories, including claims for opiate drugs, long-acting (LA) opiate drugs, antibiotic drugs, and anti-psychotic drugs.

We also have two sources for the Part B data, that differ in specificity just as the sources of the Part D data. The first source of the Part B data, which we refer to as the "provider-service-level Part B data" is Medicare Physician & Other Practitioners—by Provider and Service [17]. The second source, which we refer to as the "provider-level Part B data" is Medicare Physician & Other Practitioners—by Provider [22]. The

Table 1 Provider-level Part D features, descriptions copied from [24]

Feature	Description
GE65_Tot_Clms	The number of Medicare Part D claims for beneficiaries age 65 and older
GE65_Tot_30day_Fills	The number of Medicare Part D standardized 30-day fills for beneficiaries age 65 and older
GE65_Tot_Drug_Cst	The aggregate total drug cost paid for all associated claims for beneficiaries age 65 and older
GE65_Tot_Day_Suply	The aggregate number of day's supply for which this drug was dispensed, for beneficiaries age 65 and older
GE65_Tot_Benes	The total number of unique Medicare Part D beneficiaries age 65 and older with at least one claim for the drug
Brnd_Tot_Clms	Total claims of brand-name drugs, including refills
Brnd_Tot_Drug_Cst	Aggregate drug cost paid for brand-name drugs
Gnrc_Tot_Clms	Total claims of generic drugs, including refills
Gnrc_Tot_Drug_Cst	Aggregate cost paid for generic drugs
Othr_Tot_Clms	Total claims of other drugs, including refills. A drug is classified as "other" using any FDA approval categories not included in the brand or generic definitions
Othr_Tot_Drug_Cst	Aggregate cost paid for all other drugs not classified as brand or generic
MAPD_Tot_Clms	The number of claims for beneficiaries covered by (Medicare Advantage plan that includes Medicare Part (MDAPD)
MAPD_Tot_Drug_Cst	Aggregate cost paid for claims filled by beneficiaries in MAPD plans
PDP_Tot_Clms	The number of claims for beneficiaries covered by standalone Prescription Drug Plans (PDPs)
PDP_Tot_Drug_Cst	Aggregate drug cost paid for claims filled by beneficiaries in standalone PDPs
LIS_Tot_Clms	Total number of claims from this prescriber, including refills, for beneficiaries with a Part D low-income subsidy (LIS)
LIS_Drug_Cst	Aggregate drug cost paid for claims for beneficiaries with a Part D low-income subsidy
NonLIS_Tot_Clms	Total number of claims from this prescriber, including refills, for beneficiaries without a Part D low-income subsidy
NonLIS_Drug_Cst	Aggregate drug cost paid for claims for beneficiaries without a Part D low-income subsidy
Opioid_Tot_Clms	Total claims of opioid drugs, including refills
Opioid_Tot_Drug_Cst	Aggregate cost paid for opioid drugs
Opioid_Tot_Suply	The aggregate number of day's supply for opioid drugs
Opioid_Tot_Benes	The total number of unique Medicare Part D beneficiaries with at least one opioid claim
Opioid_Prscrbr_Rate	The percent of the Tot_Clms represented by the Opioid_Tot_Clms
Opioid_LA_Tot_Clms	The aggregate number of day's supply for long-acting (LA) opioid drugs
Opioid_LA_Tot_Drug_Cst	Aggregate cost paid for long-acting opioid drugs
Opioid_LA_Tot_Suply	The aggregate number of day's supply for long-acting opioid drugs
Opioid_LA_Tot_Benes	The total number of unique Medicare Part D beneficiaries with at least one long-acting opioid claim
Opioid_LA_Prscrbr_Rate	The percent of the Opioid_Tot_Clms represented by the Opioid_LA_Tot_Clms
Antbtc_Tot_Clms	Total claims of antibiotic drugs, including refills
Antbtc_Tot_Drug_Cst	Aggregate cost paid for antibiotic drugs
Antbtc_Tot_Benes	The total number of unique Medicare Part D beneficiaries with at least one antibiotic claim
Antpsyct_GE65_Tot_Clms	Total claims of antipsychotic drugs, including refills, for beneficiaries age 65 and older
Antpsyct_GE65_Tot_Drug_Cst	Aggregate cost paid for antipsychotic drugs for beneficiaries age 65 and older
Antpsyct_GE65_Bene_Suprsn_Flag	A flag indicating the reason the Antpsyct_GE65_Tot_Benes variable is suppressed

Table 1 (continued)

Feature	Description
Antpsyct_GE65_Tot_Benes	The total number of unique Medicare Part D beneficiaries age 65 and older with at least one antipsychotic claim
Bene_Avg_Age	Average age of beneficiaries
Bene_Age_LT_65_Cnt	Number of beneficiaries under the age of 65
Bene_Age_65_74_Cnt	Number of beneficiaries between the ages of 65 and 74
Bene_Age_75_84_Cnt	Number of beneficiaries between the ages of 75 and 84
Bene_Age_GT_84_Cnt	Number of beneficiaries over the age of 84
Bene_Feml_Cnt	Number of female beneficiaries
Bene_Male_Cnt	Number of male beneficiaries
Bene_Dual_Cnt	Number of Medicare beneficiaries qualified to receive Medicare and Medicaid benefits
Bene_Ndual_Cnt	Number of Medicare beneficiaries qualified to receive Medicare only benefits
Bene_Avg_Risk_Scre	Average Hierarchical Condition Category (HCC) risk score of beneficiaries

provider-service-level part B data contains one record, for each year, for each specific treatment or procedure that the provider submits claims to Medicare for. The provider-level part B data has records that contain information about the provider's activity over all claims for all treatments and procedures for a year.

The provider-service-level Part B data has 29 features, but as is the case with the Part D data, many of the attributes are provider demographic data, which we do not use for modeling purposes. It also has categorical data that describes the treatment or procedure rendered to the patient, which we do not include in the final datasets, since we aggregate at the provider level. On the other hand, we utilize categorical features at the provider level for place of service, provider gender, and provider type. The provider-service-level Part B data also includes numeric data on claims submitted for treatments and procedures. There are features for the total number of times the service is provided, the total number of patients the service is provided to, the daily number of beneficiaries treated, the average amount the provider charges for the service, and data on the average amounts Medicare pays for the service. The provider-service-level Part B data has approximately 68 million records.

The provider-level Part B data has 47 features. They are listed in Table 2. The simplest group of attributes in the provider-level Part B data has data on the total number of patients treated, by age. Therefore, there are four features, one for patients under 65, one for patients aged 65–75, and one for patients aged 75–84, and one for patients aged 85 and over. There is a final age-related feature that has the average number of patients in each age group. There are seven features which pertain to claims the provider sends to Medicare for all treatments and procedures that the provider renders to their patients for the year. These features contain data on the total number of distinct procedures, total number of patients treated, and the sum of money the provider has billed Medicare for, for the year. The seven features also include the amount Medicare is allowed to pay a provider, and the amount Medicare actually paid the provider. Finally, there is a feature for a standardized total payment amount. The standardized total Medicare payment amount is calculated by adjusting for geographic differences in costs, making it easier

Table 2 Provider level Part B features, descriptions copied from [21]

Tot_HCPCS_Cds	Total number of unique HCPCS codes
Tot_Benes	Total Medicare beneficiaries receiving services from the provider
Tot_Srvcs	Total provider services
Tot_Sbmtd_Chrg	The total charges that the provider submitted for all services
Tot_Mdcr_Alowd_Amt	The Medicare allowed amount for all provider services
Tot_Mdcr_Pymt_Amt	Total amount that Medicare paid after deductible and coinsurance amounts have been deducted for all the provider's line item service
Tot_Mdcr_Stdzd_Amt	Total amount that Medicare paid after deductible and coinsurance amounts have been deducted for the line item service and after standardization of the Medicare payment has been applied
Drug_Tot_HCPCS_Cds	Total number of HCPCS codes for drug services
Drug_Tot_Benes	Total Medicare beneficiaries receiving drug services
Drug_Tot_Srvcs	Total drug services
Drug_Sbmtd_Chrg	The total charges that the provider submitted for drug services
Drug_Mdcr_Alowd_Amt	The Medicare allowed amount for drug services
Drug_Mdcr_Pymt_Amt	Total amount that Medicare paid after deductible and coinsurance amounts have been deducted for all the provider's line item drug services
Drug_Mdcr_Stdzd_Amt	Total amount that Medicare paid after deductible and coinsurance amounts have been deducted for the line item drug service
Med_Tot_HCPCS_Cds	Total number of HCPCS codes associated with medical services
Med_Tot_Benes	Total Medicare beneficiaries receiving medical services
Med_Tot_Srvcs	Total medical services
Med_Sbmtd_Chrg	The total charges that the provider submitted for medical services
Med_Mdcr_Alowd_Amt	The Medicare allowed amount for medical services
Med_Mdcr_Pymt_Amt	Total amount that Medicare paid after deductible and coinsurance amounts have been deducted for all of the provider's line item medical services
Med_Mdcr_Stdzd_Amt	Total amount that Medicare paid after deductible and coinsurance amounts have been deducted for the line item medical service
Bene_Avg_Age	Average age of beneficiaries. Beneficiary age is calculated at the end of the calendar year or at the time of death
Bene_Age_LT_65_Cnt	Number of beneficiaries under the age of 65. Beneficiary age is calculated at the end of the calendar year or at the time of death
Bene_Age_65_74_Cnt	Number of beneficiaries between the ages of 65 and 74. Beneficiary age is calculated at the end of the calendar year or at the time of death
Bene_Age_75_84_Cnt	Number of beneficiaries between the ages of 75 and 84
Bene_Age_GT_84_Cnt	Number of beneficiaries over the age of 84. Beneficiary age is calculated at the end of the calendar year or at the time of death
Bene_Feml_Cnt	Number of female beneficiaries
Bene_Male_Cnt	Number of male beneficiaries
Bene_Dual_Cnt	Number of Medicare beneficiaries qualified to receive Medicare and Medicaid benefits
Bene_Ndual_Cnt	Number of Medicare beneficiaries qualified to receive Medicare only benefits
Bene_CC_AF_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for atrial fibrillation
Bene_CC_Alzhmr_Pct	Percent of beneficiaries meeting the Chronic Conditions Data Warehouse (CCW) chronic condition algorithm for Alzheimer's, related disorders, or dementia
Bene_CC_Asthma_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for Asthma
Bene_CC_Cncr_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithms for cancer
Bene_CC_CHF_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for heart failure
Bene_CC_CKD_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for chronic kidney disease
Bene_CC_COPD_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for chronic obstructive pulmonary disease
Bene_CC_Dprssn_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for depression

Table 2 (continued)

Bene_CC_Dbts_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for diabetes
Bene_CC_Hyplpdma_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for hyperlipidemia
Bene_CC_Hyprtnsn_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for hypertension
Bene_CC_IHD_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for ischemic heart disease
Bene_CC_Opo_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for osteoporosis
Bene_CC_RAOA_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for rheumatoid arthritis/osteoarthritis
Bene_CC_Sz_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for schizophrenia and other psychotic disorders
Bene_cc_strok_pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for stroke
Bene_Avg_Risk_Scre	Average Hierarchical Condition Category (HCC) risk score of beneficiaries

Table 3 Prescription-level Part D base features, descriptions copied from [23]

Feature	Description
Prscrbr_Type	The Medicare specialty code, describes the type of practice
Tot_Clms	The number of Medicare Part D claims this includes original prescriptions and refills
Tot_30day_Fills	The aggregate number of Medicare Part D standardized 30-day fills
Tot_Day_Suply	The aggregate number of day's supply for which this drug was dispensed
Tot_Drug_Cst	The aggregate drug cost paid for all associated claims
Tot_Benes	The total number of unique Medicare Part D beneficiaries with at least one claim for the drug

to compare prices across regions. The provider-level Part B data has another 14 features that are the result of splitting these seven features into two groups: one group for medication-related services, and the second for all other services. There are features for the total number of male, and female patients. The provider-level Part B data has another 18 features for the percentages of patients with certain chronic conditions. Alzheimer's, kidney disease, and asthma are all examples of chronic conditions. There is also a feature for the HCC risk score that has the same definition as in the provider-level Part D data.

For both the Part B and Part D data, our approach is to aggregate data from the first source at the drug/service level, and then enrich the aggregated data with the data from the second source that is at the provider level. For the provider-service-level Part B data, we retain the features for the provider type, place of service, and provider gender, since they are provider-level features. We discard features related to specific services, such as the Healthcare Common Procedure Coding System (HCPCS) code. We then take the remaining numeric features and for each of them, we calculate six summary statistics, sum, mean, median, minimum, maximum, and standard deviation, using all records for the provider, for the year. The final base features for the Part D data are listed in Table 3. For every one of the features listed in Table 3, except Prscbr_Type, there are six features calculated from summary statistics. Table 4 is similar to Table 3, but for the Part B data. For the features listed in Table 4, there are six features of descriptive statistics in the

Table 4 Service-level Part B base features, descriptions copied from [26]

Rndrng_Prvdr_Type	Derived from the provider specialty code reported on the claim
Place_Of_Srvc	Identifies whether the place of service submitted on the claims is a facility
Rndrng_Prvdr_Gndr	The provider's gender
Tot_Srvcs	Number of services provided; note that the metrics used to count the number provided can vary from service to service
Tot_Benes	Number of distinct Medicare beneficiaries receiving the service
Tot_Bene_Day_Srvcs	Number of distinct Medicare beneficiary/per day services
Avg_Sbmtd_Chrg	Average of the charges that the provider submitted for the service
Avg_Mdcr_Pymt_Amt	Average amount that Medicare paid after deductible and coinsurance amounts have been deducted for the line item service

actual dataset, except for the Rndrng_Prvdr_Type, Place_Of_Srvc, and Rndrng_Prvdr_Gndr features.

We aggregate the provider-drug-level Part D data almost exactly as we do for the provider-service-level part B data. The provider-drug-level Part D data does not have attributes for the provider gender, or place of service. However, it has a feature for provider type, so we retain that since it is at the provider level. We then utilize the numeric features in the same manner as the provider-service-level Part B data, calculating the six descriptive statistics for each numeric feature, at the level of every provider for the year. Aggregation reduces the size of the Part B and Part D datasets. The aggregated Part D data has approximately 6.3 million instances, and the aggregated Part B data has approximately 8.7 million instances.

Next we enrich both the Part D and the Part B data by joining them to the provider level datasets. This means we join the aggregated Part D data to the provider-level Part D data by NPI, and we join the aggregated Part B data to the provider-level Part B data, also by NPI. The joining process yields an unlabeled dataset. "Joining data by NPI" means that we add a record to our unlabeled Part D dataset if, and only if, a record exists in the aggregated Part D data and the provider-level Part D data with the same NPI and year values. The same rule applies for our unlabeled Part B dataset as well. The unlabeled Part B dataset is exactly the same size as the aggregated Part B dataset. However, the unlabeled Part D dataset has about one million fewer records than the aggregated Part D dataset. This is due to a lack of provider level Part D records for some providers and some years. The unlabeled Part B dataset has 82 attributes, and the unlabeled part D dataset has 80 attributes.

As a last step in compiling the Part B and Part D datasets, we label them with records from the LEIE. We adopt a uniform approach to labeling the two datasets. The LEIE is released on a monthly schedule by the OIG. When a healthcare provider is listed in the LEIE, it may denote a conviction for actions that prohibits them from filing insurance claims with Medicare. The LEIE, Part B, and Part D data sources all share a common element, the NPI. There are different types of exclusions that apply to providers that are on the LEIE. We use the same exclusion types as fraud indicators that Bauder and Khoshgoftaar use in [27]. When a provider is listed in the LEIE for any of these types of exclusions, we mark all associated records dated before the end of the exclusion period from as fraudulent.

Table 5 Summary of Part B and Part D datasets

Dataset	Instance Count	Fraudulent	Ratio fraudulent (%)
Part D	5,344,106	3700	0.0693
Part B	8,669,497	3954	0.0456

The data from Part B and Part D covers entire calendar years, while exclusion periods end at specific months. For this reason, we approximate the end of an exclusion period to the closest year. When an exclusion period concludes, the healthcare provider in question is expunged from the LEIE, allowing them to once again file claims with Medicare. Consequently, any data related to claims made by the provider after the expiration of the exclusion period is considered non-fraudulent. Compiling a dataset encompassing all available years will not include records of providers previously listed in the LEIE, as the current LEIE will not include such records. As such, tools like the Internet Archive Tool[1] should be utilized to access earlier versions of the LEIE and label the older records in the CMS data accordingly. Table 5 summarizes the outcome of the labeling process for the Part B and Part D datasets.

Algorithms

For reproducible results, we utilize five well-known, open-source ensemble learning methods for our classification tasks, and one linear algorithm. As stated above, the ensemble algorithms are XGBoost [28], LightGBM [29], Extremely Randomized Trees (ET) [30], Random Forest [31], and CatBoost [32]. The linear algorithm is Logistic Regression [33]. For feature selection, we use the five ensemble methods, as well as Decision Tree [34]. Here we review the key attributes of all machine learning algorithms used in this study.

The first algorithm we discuss is Decision Tree, since all the ensemble techniques we use rely on it, and it is used in our ensemble feature selection technique as well. More specifically, we use Decision Trees for binary classification. A Decision Tree is built in an iterative fashion. To start, the Decision Tree consists of a single rule that separates a dataset in two classes based on the value of a single attribute in relation to a threshold value, known as a "split". This is because we can visualize the rule as a node, with two edges emanating from it. One edge ends at a node that represents a classification of the instance as member of one class, and the other edge ends at a node that represents the alternative classification. Nodes that represent classifications are called leaf nodes. As a Decision Tree is built, one repeatedly splits the paths to the leaf nodes by adding more rules. This can be visualized as adding more steps along additional paths to the leaf nodes, and leaf nodes being duplicated to accommodate the increasing number of paths. Decision rules are added based on their capacity to segregate the data into subsets that exhibit greater homogeneity in relation to the target variable. Put another way, those features that lead to splits significantly enhancing the reduction of impurity (for instance,

[1] http://archive.org/web.

via Gini impurity or entropy measures) are deemed more critical for models based on Decision Trees.

The ensemble algorithms in our study belong to two families: Bagging and Gradient Boosted Decision Trees (GBDTs). ET and Random Forest are part of the Bagging family, while CatBoost, XGBoost, and LightGBM are representatives of the Gradient Boosted Decision Trees family. The rationale behind using methods that incorporate diverse underlying techniques is to demonstrate that our findings are not confined to a single algorithmic type. Both Bagging and Gradient Boosted Decision Trees embody distinct strategies for leveraging a set of learners for classification tasks. We employ Logistic Regression as a baseline to determine whether the computing overhead of the ensemble techniques yields better performance. Here we provide an overview of these algorithms.

The GBDT classifiers in our study are derived from the Gradient Boosted Machine algorithm, first introduced by Friedman [35]. Previous research shows GBDTs yield successful results in the Medicare fraud detection application domain [36]. Friedman's technique is an ensemble technique. It proceeds in an iterative fashion. This method starts with one learner that provides an initial set of predictions $\hat{\mathbf{y}}$ for the dependent variable $\mathbf{y}$. The discrepancies, or residuals, between these predictions $\hat{\mathbf{y}}$ and the actual values $\mathbf{y}$ form a vector $\mathbf{y} - \hat{\mathbf{y}}$. This vector can be regarded as a new dependent variable which we can estimate using the original independent variables with a second learner. Subsequently, the combined output of the two models will offer a more precise prediction of the dependent variable compared to the first model's outputs. This process can be extended with the addition of more learners to the ensemble, with each new learner trained to predict the residuals of the current ensemble. Consequently, every additional learner enhances the estimation of the dependent variable. XGBoost, LightGBM, and CatBoost are all advanced adaptations of Friedman's initial idea. Since they all utilize Decision Trees, we refer to them as Gradient Boosted Decision Trees (GBDTs).

XGBoost was the earliest of the three GBDTs we use. It was released by Chen and Guestrin in 2016. Earlier research proves XGBoost is a good choice of classifier for classifying imbalanced Big Data [37]. XGBoost adds several features to the standard GBDT technique. XGBoost incorporates an improved loss function during the training phase, which includes an additional regularization term to mitigate overfitting. XGBoost enhances the calculation of splits in the ensemble of Decision Trees that it uses. Chen and Guestrin introduced an "approximate algorithm" that estimates optimal split values. The approximate algorithm is useful in situations where the complete dataset cannot fit into main memory. The approximate algorithm is also advantageous in distributed environments. Finally, XGBoost brings another advancement in the form of an algorithm for determining splits that works effectively with sparse data. Sparse data typically exhibits near-constant values with sporadic deviations. XGBoost's "sparsity aware split finding" feature allows it to effectively exploit sparse data to efficiently construct its constituent Decision Trees.

LightGBM is the second GBDT used in our study. Ke et al. published the foundational paper on LightGBM in 2017 [29]. Their aim was to develop a GBDT implementation that offered performance on par with XGBoost, but with lower resource consumption. To realize this goal, Ke et al. introduced two key improvements to the GBDT approach. The first is Exclusive Feature Bundling (EFB). EFB is a method for reducing the dimensions

of a dataset by merging two attributes into one. It proves particularly effective for sparse data. If two attributes of a dataset display sparsity and the infrequent values of both attributes occur in mutually exclusive rows, they can be safely merged into a single feature with minimal loss of information. EFB lowers the number of the dataset's dimensions, subsequently reducing training time. The second improvement LightGBM brings to the GBDT method is known as Gradient-based One-Side Sampling (GOSS). GOSS is a strategy for sensibly reducing the number of training instances when fitting Decision Trees. It chooses instances for training based on how much they add to the loss function. The loss function must be an aggregate function, which is computed as part of fitting the GBDT ensemble to the training data. If an instance contributes greater than an adjustable threshold value to the model's loss, it is retained for further iterations of the fitting process. Conversely, instances that contribute less than the threshold value are discarded. Through GOSS and EFB, Ke et al. have crafted a GBDT implementation that requires fewer computational resources.

One more GBDT implementation used in our study is CatBoost, introduced by Prokhorenkova et al. in 2018 [32]. A recent study on CatBoost's applications across various domains can be found in [38]. One objective expressed in the seminal paper on CatBoost is the mitigation of overfitting, with two strategies employed to meet the objective. The first strategy, called Ordered Boosting, pertains to the selection of training instances used to fit Decision Trees in CatBoost's ensemble. In order to add a Decision Tree to the GBDT ensemble, there are two steps: first, fitting the candidate Decision Trees to the dependent variable in the training data, and then, evaluating these trees to select the one that most effectively enhances the ensemble's overall performance. In Ordered Boosting, it is ensured that the training instances used to fit the Decision Tree are not used to evaluate it for inclusion into the ensemble. This prevents the ensemble from overfitting to the training data.

The second overfitting protection in CatBoost involves its Ordered Target Statistics method for encoding categorical features. Ordered Target Statistics stems from the idea of target encoding. To use target encoding for a categorical feature, the encoded value of a feature is set to the mean value of the dependent variable that it co-occurs with. However, this encoding strategy could lead to what Prokhorenkova et al. refer to as "target leakage". To explain target leakage, we consider the scenario where one value of the encoded feature co-occurs with different dependent variable values. If the feature is encoded with target encoding, it may not be a useful predictor of the dependent variable since it carries information about the specific value of the target that it co-occurs with. To mitigate target leakage, the Ordered Target Statistics technique guarantees that the encoded value for a categorical feature of a given instance is derived from other instances. In other words, the encoded value of a categorical feature cannot be calculated from the specific label it appears with. This precludes the possibility of the encoded feature value of the instance being directly related to the value of the dependent variable. Hence, Ordered Target Statistics is CatBoost's second protection against overfitting.

Breiman introduced the concept of Bagging in the domain of machine learning in a 1996 paper [39]. Breiman describes the versatility of Bagging in both classification and regression contexts. As our research revolves around binary classification, our focus is on Breiman's ideas about Bagging applied to binary classification. Bagging involves

fitting multiple instances of a machine learning algorithm to multiple bootstrap samples of the training data, thereby generating an ensemble of learners. A bootstrap sample is derived from the original training data by sampling with replacement [40]. Moreover, the machine learning algorithms are allowed to be so-called, "weak" learners, in the sense that the Bagging technique may still be effective even when one of the learners on their own would yield unacceptably poor performance. Once the learners are fit to these bootstrap samples, each of them classifies instances of the test data. The final classification output is the class identified by the majority of the ensemble learners. Bagging's potential to enhance classification performance centers on probabilistic reasoning. Consider a scenario where a weak learner correctly classifies more often than it makes errors. In such a case, the probability of the majority of weak learners in an ensemble making the right classification grows as the number of these weak learners is increased. Consequently, if we consider the classification result to be the category selected by the majority of the weak learners, Bagging yields better results as the size of the ensemble grows.

The first member of the bagging family we use is Random Forest, which was introduced by Breiman [31]. Random Forest builds upon the Bagging principle with an added improvement. It applies Bagging to Decision Trees, but includes a modification. To understand this modification, we first need to define what a "split" signifies within the scope of Decision Trees. The non-leaf nodes of a Decision Tree contain rules that dictate which node should be visited next. This rule is formed based on a comparison between a numerical value, termed as the split, and the current value of one of the dataset's independent attributes. Therefore, the process of training a Decision Tree-based model largely revolves around determining the best values for these splits. Random Forest enhances the Bagging technique by randomly selecting a subset of the features when deciding the most suitable value for a split.

Another classifier that we utilize from the Bagging family of machine learning algorithms is the Extremely Randomized Trees (ET) classifier [30]. ET extends the concept of Random Forest by selecting values for Decision Tree splits at random. In contrast, Random Forest and other Decision Tree-based learners typically calculate splits according to some deterministic logic. For instance, the optimal value for a split could be determined based on a specific metric evaluating how effectively the split rule segregates the training data into subsets with identical labels. However, ET departs from this deterministic approach and opts for a random selection of split values. Interestingly, our findings indicate that this seemingly random selection strategy in ET often yields effective performance when it comes to classifying highly imbalanced Big Data, particularly in the context of Medicare fraud detection.

The last classifier left to discuss is Logistic Regression [33]. Logistic Regression is a method that fits a sigmoid function to a dataset by setting the parameters of the sigmoid function to the maximum likelihood estimate of their value, given the training data. A Logistic Regression model has on the order of one parameter for each independent variable, which in many cases makes it simpler than the ensemble methods we discussed above. Since the value of a split is a parameter of a Decision Tree, ensemble techniques that result in large collections of tall and wide Decision Trees would have far more parameters than a Logistic Regression model. We include Logistic Regression in

Table 6 Hyperparameter settings used in experiments

Classifier	Parameter name	Parameter setting
CatBoost	`task_type`	`'GPU'`[a]
	`max_ctr_complexity`	1
	`max_depth`	5
ET	`max_depth`	8
XGBoost	`max_depth`	3
	`tree_method`	`'gpu_hist'`[a]
LightGBM	`max_depth`	4
Random Forest	`max_depth`	4

[a] Setting selects Graphics Processing Unit GPU) implementation of the classifier

our study as a check to ensure that a simple model would suffice to carry out automated Medicare fraud detection.

Hyperparameter settings

In our experiments, we have adopted the identical hyperparameter configurations that were used in the tests detailed in [41] for the classifiers. Preliminary trials for the aforementioned study indicated that these hyperparameter configurations exhibit robust performance in classifying highly imbalanced data, without showing tendencies of overfitting when determining the best classification thresholds based on multiple metrics. These metrics include f-Measure [42], geometric mean of true positive rate and true negative rate [43], Matthews Correlation Coefficient (MCC) [44], and precision [45].

Table 6 includes hyperparameter settings that were adjusted from their default values. Besides the settings outlined in Table 6, we have also fixed random number generator seeds for all classifiers to ensure the reproducibility of the results. All remaining settings are kept at their default settings. Furthermore, we have not made any changes to the hyperparameter settings for Logistic Regression and Decision Tree.

This concludes the discussion of the seven types of machine learning algorithms used in this study. Most of the models discussed are used in both feature selection, and classification. These are Random Forest, ET, XGBoost, CatBoost, and LightGBM. We use Logistic Regression for classification only, and we use Decision Tree for feature selection, only.

Methodology

Our study has two main methodologies. We employ one methodology for ensemble feature selection, and a second one for Medicare Fraud detection. To the best of our knowledge, the ensemble feature selection technique is novel. First, we provide a description of the ensemble feature selection technique. Then, we explain how we approach Medicare Fraud detection as a supervised machine learning classification task.

Ensemble feature selection

Here we describe our novel ensemble feature selection technique. A synopsis of our technique is that it is an efficient means to intelligently leverage the built-in feature importance functionality of multiple machine learning algorithms.

Our technique begins with the selection of a collection of supervised machine learning algorithms. In the context of our technique for ensemble feature selection, we call a supervised machine learning algorithm a *ranker*. For our study, we employ six rankers: CatBoost, XGBoost, Random Forest, ET, LightGBM, and Decision Tree [34]. These algorithms are a convenient choice since they all maintain a built-in list of feature importance values during the model fitting phase of supervised machine learning. Each algorithm's implementation may have a unique method for calculating feature importance. For example, a Decision Tree implementation may assign a feature its importance value based on the change in Gini Impurity [45], or Shannon Entropy [46], that is obtained by splitting the training data into subsets based on some criteria related to the feature. This is because a Decision Tree implementation could employ either Gini Impurity change or Shannon Entropy change as a splitting criteria during model training. Therefore, these statistics are readily available to return to the user as feature importance values. Once fitting is completed, one may access the ordered list of features, ranked according to their importance to the algorithm. Since we use six rankers, we have six ordered lists of features. We refer to an ordered list of features as a *ranking*.

The novelty of our feature selection technique is the method by which we combine the rankings. Our approach is to assign each feature the median value of its rank in each ranking. We use the median rank for several reasons. First, the median robust to outliers. Therefore, if one ranker assigns a rank r to a feature that is much different from the rank the other rankers assign, the median value will be the same if r were above or below ranks assigned by other rankers, regardless of how extreme it is. Therefore, using the median rank of a feature helps prevent a single ranker from exerting too much influence on the final rank assigned by the ensemble. Another reason we use the median value of a feature's rank is that it preserves information. If all rankers agree that a feature has a particular ranking, then the median rank will also be that value. This is an advantage over a majority rules approach to combining rankings, where we select features based on the number of rankers that agree that the feature is among the highest n ranked features. Such an approach loses the relative position of features in the final ranking. Since selecting the median rank of a feature can preserve information about the ranks assigned to it, the outcome leads to more explainable models.

Assigning each feature the median value of its positions in all of the rankings results in an ordered list of features. In order to complete the feature selection process, one decides on a number of features to retain by selecting the first n features in the ordered list, where n is an integer between one and the number of features in the dataset. Algorithm 1 below is a formal description of our ensemble feature selection technique.

Algorithm 1: Ensemble Supervised Feature Selection Technique

input :

D ▷ labeled dataset

$F^j, j = 1, \ldots, m,$ ▷ F features of D

$y^j, j = 1, \ldots, m,$ ▷ y labels of D

$\omega_1 \ldots \omega_k$ ▷ a set of k supervised machine learning algorithms that maintian feature importance lists (rankers)

output: $\mathbb{R}$ ▷ Ensemble output, a ranking of features F^j of dataset D

$\mathbf{R} \leftarrow \emptyset$ **for** $i = 1, \ldots, k$ **do**

  Fit ω_i to F^j, y^j;

  $\mathbf{R}_i \leftarrow$ feature ranking list from ranker ω_i;

  $\mathbf{R} \leftarrow \mathbf{R} \cup \mathbf{R}_i$;

for $F^j \in F$ **do**

  $l \leftarrow \emptyset$ ▷ list of feature rankings;

  for $\mathbf{R}_i \in \mathbf{R}$ **do**

    $l \leftarrow \mathbf{R}_i(F^j)$ ▷ Get a list of all of the ranks of feature F^j

  $\mathbb{R}(F^j) \leftarrow \text{median}(l)$ ▷ The rank of feature F^j is the median of all of its rank values in $\mathbf{R}$.

return $\mathbb{R}$

We choose to make the return value of Algorithm 1 the ordered list of features $\mathbb{R}$ to emphasize that one may then use first n elements of $\mathbb{R}$ to select n features. We do this to make it clear that our ensemble feature selection technique has an application for building explainable models. We recommend using statistical analysis such as Analysis of Variance (ANOVA) [47] tests, and Tukey's Honestly Significant Difference (HSD) [48] tests, to find the minimum value of the number of features n that we can build a model with, that yields performance similar to, or better than using all features. Then this minimal subset is the one we use to explain a model's behavior.

We refer to the output of the feature selection technique by the number of features selected. We define a "feature set" as a group of features identified by a feature selection technique. Therefore, for the Part D data, we have feature sets 7a, 7b, 8, 9, 10, 15, 20, 25, 30 and 80. Feature sets 7a and 7b arose from the fact that, in the Part D ranking, we have two features that share a median rank of 7. Furthermore, we decided to assess the performance of models built with fewer than 10 features, since we found that models built with ten features of the Part D data outperformed models built with all features. We did not see a similar trend in AUPRC scores from experiments with the Part B data. Hence, it was not necessary to continue with smaller feature sets for experiments with the Part B data. Therefore, for the Part B data we have feature sets 10, 15, 20, 25, 30, and 82. The trend in rising AUPRC scores for experiments with 10 features from the Part D was possibly

because classification results for experiments on the Part B data with 10 features were not as strong as the results for experiments on the Part D data with 10 features. In the next section, we document how the application of our feature selection technique is used to build explainable models for Medicare fraud detection.

Classification

Our classification methodology is where all the various components of this study come together. Earlier we describe our technique for compiling the Part B and Part D datasets. Part of the description includes the labeling process. Since we have a labeled dataset, it is possible to apply popular machine learning algorithms to classify it. In our discussion of supervised feature selection, we explain how we obtain many datasets by selecting different numbers of features based on their median rank, derived from multiple feature importance lists. This results in ten Part D feature sets, and six Part B feature sets. In our discussion on machine learning algorithms above, we mention that we use six algorithms for classification. We use all combinations of feature sets and algorithms to classify the Part B and Part D data.

For every pair of classifier and feature set, we conduct fivefold cross validation [49]. This yields five Area Under the Receiver Operating Characteristic Curve (AUC) [14] and five Area Under the Precision Recall Curve (AUPRC) [9] values, which we record as experimental outcomes. Since we are dealing with imbalanced data, we consider AUPRC to be a more robust metric when it comes to evaluating the performance of classifiers. However, we include AUC as well, since it is a secondary, threshold-agnostic classification metric. In order to mitigate the effects of random chance on experimental outcomes, and to obtain a sufficient number of results for statistical analysis, we repeat fivefold cross validation ten times. Different random number generator seeds are used in order to ensure our machine learning algorithms run under different initial conditions every time. This results in fifty AUC and AUPRC values for each combination of classifier and feature set. Since we have six feature sets of Part B data, and six classifiers, we have a total of 1800 AUC and AUPRC values in the set of experimental outcomes. For experiments with the Part D data, we have ten feature sets, and therefore, 3000 AUC and AUPRC values in the set of experimental outcomes.

All the software used in our experiments is open-source, and therefore publicly available for free. Every classification experiment is run as a Python program [50]. We use the scikit-learn [51] library to handle fivefold cross validation, and its implementations of the Decision Tree, Random Forest, ET and Logistic Regression classifiers. CatBoost, XGBoost, and LightGBM are available as individual libraries. A program written in the R language is used to perform the analysis of variance (ANOVA) [47] and Tukey's Honestly Significant Difference (HSD) [48] tests.

All of our experiments are carried out as batch operations on a distributed computing platform. Each node on this platform is equipped with an Intel Xeon Central Processing Unit (CPU), which comes with 16 cores, along with 256 GB of RAM per CPU and Nvidia V100 Graphics Processing Units (GPUs). The resources available on a single node are sufficient to conduct any of the experiments discussed in this study. Now that we have elucidated our methods, we move on to discuss the results they produce.

Results

Our primary metric for gauging performance is AUPRC. In their study [52], Calvert and Khoshgoftaar show that AUC on its own may be a misleading metric to gauge experimental outcomes, due to the fact that it does not align with results in terms of several

Table 7 Mean AUPRC values by classifier and number of features for ten iterations of fivefold cross validation, for classifying the Medicare Part D data (Part 1)

Features classifier	7a	7b	8	9	10
CatBoost	0.7575	0.7582	0.7570	0.7558	0.7585
ET	0.5941	0.5171	0.5585	0.6006	0.5878
LightGBM	0.4548	0.4533	0.4689	0.5116	0.5529
Logistic Regression	0.3468	0.3368	0.3497	0.3482	0.3537
Random Forest	0.5873	0.4937	0.5927	0.5393	0.5903
XGBoost	0.7533	0.7539	0.7533	0.7514	0.7571

Table 8 Mean AUPRC values by classifier and number of features for ten iterations of fivefold cross validation, for classifying the Medicare Part D data (Part 2)

Features classifier	15	20	25	30	82
CatBoost	0.8016	0.7953	0.7962	0.7949	0.7797
ET	0.4954	0.4647	0.4605	0.4391	0.3275
LightGBM	0.4447	0.4661	0.4603	0.4841	0.4982
Logistic Regression	0.3669	0.3536	0.3519	0.2939	0.3047
Random Forest	0.6097	0.5398	0.5519	0.5249	0.2429
XGBoost	0.7889	0.7448	0.7589	0.7548	0.7376

Table 9 Mean AUC values by classifier and number of features for ten iterations of fivefold cross validation, for classifying the Medicare Part D data (Part 1)

Features classifier	7a	7b	8	9	10
CatBoost	0.9223	0.9231	0.9228	0.9293	0.9383
ET	0.8541	0.8415	0.8448	0.8614	0.8574
LightGBM	0.7541	0.7449	0.7730	0.7904	0.8298
Logistic Regression	0.8850	0.8698	0.8816	0.8832	0.8869
Random Forest	0.8340	0.8222	0.8384	0.8244	0.8332
XGBoost	0.9276	0.9282	0.9278	0.9346	0.9408

other metrics in their study. Here we present experimental outcomes in tabular form. Tables 7 and 8 contain the mean AUPRC scores for each combination of classifier and feature set. For each combination of classifier and feature set, we perform ten iterations of fivefold cross validation. Therefore, each AUPRC score is the mean value of 50 recorded AUPRC scores.

Tables 9 and 10 are similar to Tables 7 and 8, but they contain AUC scores instead of AUPRC scores. In our opinion, it is more difficult to see a trend in the data in Tables 9 and 10 than Tables 7 and 8. Furthermore, we conjecture the cause for this is the fact that the AUC metric is calculated from the receiver operating characteristic (ROC) curve. The ROC curve is a plot of true positive and false positive rates. In a large, imbalanced dataset a model can have a low false positive rate, but still incorrectly classify a large fraction of the positive class. This is because the false positive rate is calculated as $\frac{FP}{FP+TN}$, where FP is the number of false positives and TN is the number

Table 10 Mean AUC values by classifier and number of features for ten iterations of fivefold cross validation, for classifying the Medicare Part D data (Part 2)

Features classifier	15	20	25	30	82
CatBoost	0.9436	0.9560	0.9567	0.9588	0.9587
ET	0.8294	0.8323	0.8352	0.8429	0.8116
LightGBM	0.7505	0.7929	0.7913	0.8311	0.8455
Logistic Regression	0.9006	0.9143	0.9133	0.8620	0.8536
Random Forest	0.8315	0.8288	0.8316	0.8496	0.7909
XGBoost	0.9472	0.9390	0.9447	0.9449	0.9426

Table 11 Mean AUPRC values by classifier and number of features for ten iterations of fivefold cross validation, for classifying the Medicare Part B data

Features classifier	10	15	20	25	30	80
CatBoost	0.6581	0.6792	0.7069	0.7009	0.7016	0.6817
ET	0.0400	0.0462	0.0443	0.0524	0.0424	0.0433
LightGBM	0.3939	0.3830	0.4261	0.4589	0.4293	0.4146
Logistic Regression	0.0093	0.0326	0.0338	0.0065	0.0064	0.0103
Random Forest	0.4356	0.3990	0.3736	0.3800	0.3395	0.2462
XGBoost	0.6611	0.6715	0.6995	0.6956	0.6955	0.6886

Table 12 Mean AUC values by classifier and number of features for ten iterations of fivefold cross validation, for classifying the Medicare Part B data

Features classifier	10	15	20	25	30	80
CatBoost	0.9346	0.9409	0.9493	0.9537	0.9539	0.9569
ET	0.7907	0.8080	0.8122	0.8255	0.8214	0.8409
LightGBM	0.8300	0.8176	0.8412	0.8679	0.8643	0.8477
Logistic Regression	0.8349	0.8207	0.8240	0.7874	0.7896	0.8166
Random Forest	0.8096	0.8245	0.8374	0.8525	0.8476	0.8643
XGBoost	0.9375	0.9399	0.9493	0.9521	0.9529	0.9561

of true negatives. When *TN* is extremely large relative to the number of instances of the positive class, the false positive rate calculation may be overwhelmed by the *TN* factor. The AUPRC metric does not suffer this drawback since it involves precision in place of the false positive rate, and precision does not involve instances of the negative class. Therefore, the AUPRC metric may reveal the effect of a factor in a set of experiments on the classification of imbalanced Big Data. The visibility of the trend in AUPRC scores in Tables 7 and 8 is a testament to the advantage of using AUPRC to evaluate results in the classification of imbalanced Big Data.

Table 11 is similar to Tables 7 and 8, but for the Part B data. We detect a trend in Table 11, similar to the trend we see in Tables 7 and 8. AUPRC scores are generally higher for models built with fewer features than for models built with the maximum number of features.

Table 13 ANOVA for Features and Classifier as factors of performance in terms of AUPRC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	9	2.97	0.33	47.15	[a]
Classifier	5	71.64	14.33	2050.15	[a]
Residuals	2985	20.86	0.01		

[a] Indicates the value is less than 1×10^{-4}

Table 14 HSD test groupings after ANOVA of AUPRC for the Features factor

Group a consists of: 10
Group ab consists of: 15, 9, 7a, 8
Group bc consists of: 25, 20
Group c consists of: 7b, 30
Group d consists of: 82

Also, as the case with the Part D Data, the AUC scores in Table 12 do not show as clear of a trend in the effect of features used. We find it difficult to distinguish which feature set out of 25, 30 and 80 yields the best performance. In our opinion, the results in Tables 9, 10, and 12 show that AUC is a less suitable metric for evaluating the classification of imbalanced Big Data, since trends are more difficult to detect. Put another way, we find that AUC yields ambiguous results.

Statistical analysis

Here we move on to use statistical methods to process the data presented in the results in a way that allows us to make definitive conclusions as to the effect of the feature selection, and classifier factors in our experiments.

Two factor ANOVA for feature selection experiments with Part D data analysis of results in terms of AUPRC

The first statistical method we use is an ANOVA test to determine whether the classifier and number of features factors have a significant impact on experiments involving the Part D data. The result of the ANOVA test is in Table 13. The Pr(> F), or p-values, in Table 13 are practically 0, so we conclude that both factors have a significant effect. In the ANOVA tables that follow, the term "Features" indicates the treatment of the number of features in the dataset as a factor in the experimental design. For example, in Table 13, there are ten possible values for the number of features used in the datasets, therefore, nine degrees of freedom (Df) for the "Features" factor.

Since we conclude that both the classifier and number of features have a have significant effect on experimental outcomes, we conduct an HSD test to rank the levels of the factors in terms of their impact. The result of the HSD test is the assignment of levels of the factor to groups. The alphabetical order of the group label corresponds to the level of performance, or rank, of the group.

Table 14 contains the result of the HSD test that we use to rank the impact of feature selection. Averaged across the performance of all the classifiers, 10 features yields the

Table 15 HSD test groupings after ANOVA of AUPRC for the Classifier factor

Group a consists of: CatBoost
Group b consists of: XGBoost
Group c consists of: Random Forest
Group d consists of: ET
Group e consists of: LightGBM
Group f consists of: Logistic Regression

Table 16 ANOVA for Features and Classifier as factors of performance in terms of AUC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	9	0.21	0.02	10.43	[a]
Classifier	5	9.38	1.88	833.75	[a]
Residuals	2985	6.71	0.00		

[a] Indicates the value is less than 1×10^{-4}

Table 17 HSD test groupings after ANOVA of AUC for the Features factor

Group a consists of: '30'
Group ab consists of: '10'
Group abc consists of: '25', '20'
Group abcd consists of: '9'
Group bcde consists of: '82', '15'
Group cde consists of: '8'
Group de consists of: '7a'
Group e consists of: '7b'

best performance. In Table 14, group 'a' is associated with the highest AUPRC scores, group 'ab' with scores in a range overlapping groups 'a' and 'ab', group 'ab' is associated with scores overlapping both groups 'a' and 'bc', and so on. Put another way, groups in the HSD test result correspond with their rank in alphabetical order.

Table 15 contains the HSD result for the effect of the choice classifier on AUPRC scores in our experiments with the Part D data. The pattern apparent in the test result is that the Gradient Boosted Decision Tree algorithms CatBoost and XGBoost do the best, followed by the Bagging algorithms, Random Forest and ET, followed by the linear method of Logistic Regression. The rank of LightGBM is an exception to the pattern.

Two factor ANOVA for feature selection experiments with Part D data analysis of results in terms of AUC

Next, we include ANOVA and HSD test results similar to those in Tables 13, 14, and 15. The difference is that, for the results in Tables 16, 17, and 18, we use the AUC scores recorded in experimental outcomes as opposed to the AUPRC scores. The ANOVA test result recorded in Table 16 shows that the Pr(>F) values are practically zero, which means that both the classifier and the number of features used have a significant impact on the AUC scores we record as experimental outcomes.

Table 18 HSD test groupings after ANOVA of AUC for the Classifier factor

Group a consists of: CatBoost, XGBoost
Group b consists of: Logistic Regression
Group c consists of: ET
Group d consists of: Random Forest
Group e consists of: LightGBM

Table 19 ANOVA for Features and Classifier as factors of performance in terms of AUPRC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	5	0.24	0.05	13.16	[a]
Classifier	5	130.10	26.02	7242.04	[a]
Residuals	1789	6.43	0.00		

[a] Indicates the value is less than 1×10^{-4}

Table 20 HSD test groupings after ANOVA of AUPRC for the Features factor

Group a consists of: 25, 20, 30, 15, 10
Group b consists of: 80

Although the ANOVA test result in Table 16 implies that both the classifier and number of features have a significant impact on experimental outcomes, the HSD test result in Table 17 accentuates the difficulty one may encounter when using AUC to gauge the performance of models when working with imbalanced Big Data. It is more difficult to identify the best choice since all factors are members of some interval that overlaps with other intervals.

The HSD results for the effect of the choice of classifier on AUC scores are more interpretable. However, they serve to further illustrate difficulties one may encounter when using AUC to gauge the performance of a model. Here we see that the more biased Logistic Regression model is ranked above Random Forest and ET. The HSD test result for performance of the classifier in terms of AUPRC in Table 15 disagrees with this ranking. We speculate this is due to class imbalance. In particular, we conjecture that the higher rank of Logistic Regression in Table 18 versus Table 15 is due to the overwhelming number of instances in the negative class in the Part D data, and how the false positive rate is used to calculate the AUC score versus how Precision is used to calculate the AUPRC score.

Two factor ANOVA for Supervised feature selection experiments with Part B data analysis of results in terms of AUPRC

Now we move on to present a statistical analysis of results of experiments conducted with the Part B data. The result of the ANOVA test in Table 19 shows that both the classifier and the number of features have a significant impact on AUPRC scores. Therefore, we conduct further HSD tests to rank levels of the factors according to their impact on AUPRC scores.

Table 21 HSD test groupings after ANOVA of AUPRC for the Classifier factor

Group a consists of: CatBoost, XGBoost
Group b consists of: LightGBM
Group c consists of: Random Forest
Group d consists of: ET
Group e consists of: Logistic Regression

Table 22 ANOVA for Features and Classifier as factors of performance in terms of AUC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	5	0.13	0.03	23.51	[a]
Classifier	5	5.99	1.20	1111.70	[a]
Residuals	1789	1.93	0.00		

[a] Indicates the value is less than 1×10^{-4}

Table 23 HSD test groupings after ANOVA of AUC for the Features factor

Group a consists of: 80
Group ab consists of: 25, 30
Group b consists of: 20
Group c consists of: 15, 10

The HSD test result in Table 20 indicates a different effect of supervised feature selection than what we find previously in the Part D data. In Table 20, it is clear that applying supervised feature selection yields better performance in terms of AUPRC, than not using supervised feature selection. However, the number of features used is not significant. Therefore, we are free to select the number of features that is most convenient. Since feature set 10 is the smallest, in terms of the size of the data, we would select that. This is because in our experience machine learning algorithms run to completion more quickly with smaller datasets.

The trend of classifier performance we observed for the Part B data in Table 21 is slightly more clear in the outcome of the HSD test recorded in Table 17, since LightGBM is ranked directly under CatBoost and XGBoost. Thus, the ranking in Table 21 shows that the Gradient Boosted Decision Tree algorithms CatBoost, LightGBM, and XGBoost yield the best performance, followed by the Bagging technique algorithms Random Forest and ET, and finally the linear technique, Logistic Regression.

Two factor ANOVA for Supervised feature selection experiments with Part B data analysis of results in terms of AUC

Just as we do for the Part D data, here we conduct an analysis of the performance of the AUC score results of experiments involving the Part B data. As in the other cases, the outcome of the ANOVA test documented in Table 22, for the significance of the effect of the classifier and number of features factors on experimental outcomes, shows that both

Table 24 HSD test groupings after ANOVA of AUC for the Classifier factor

Group a consists of: CatBoost, XGBoost
Group b consists of: LightGBM, Random Forest
Group c consists of: ET, Logistic Regression

Table 25 Medicare Part B 10 most important features

Tot_Srvcs
Avg_Mdcr_Pymt_Amt_sum
Avg_Mdcr_Pymt_Amt_max
Tot_Bene_Day_Srvcs_sum
Avg_Sbmtd_Chrg_sum
Tot_Benes_sum
Tot_Mdcr_Pymt_Amt
Rndrng_Prvdr_Type
Tot_Srvcs_sum
Avg_Mdcr_Pymt_Amt_min

factors have a significant impact. Therefore, we do additional HSD tests to rank the factors in terms of their impact.

The HSD test result in Table 23 highlights the misleading nature of the AUC score. The outcome of the HSD test implies that feature sets 25, 30, and 80 all yield overlapping AUC scores. This is true because the AUC score involves the false positive rate, as opposed to the AUPRC score, which involves precision, and therefore the fraction of the positive class correctly identified, instead. We conjecture that for many threshold values, models built with feature set 80 have lower false positive rates, and lower precision scores, and the differences in rankings of feature sets is evidence of that.

The HSD results in Table 24 are a case where we find AUC not to be so misleading. The results align with the trend we see in other HSD results. The trend is that the GBDT algorithms are ranked highest, followed by the Bagging methods, and finally the linear technique.

Model interpretability from feature selection and statistical analysis

Results of the statistical tests above enable an explanation of how the models used in this study accomplish Medicare fraud detection. Specifically, the HSD test results for the effect of the feature set (number of features) on the mean AUPRC scores for classifying both the Part B and Part D data tell us which subset of features can be used to build models that yield performance similar to, or better than, using all features. Of particular interest is the smallest such number of features. We conclude that the smallest set of features that yields performance similar to using all features explains the information a model uses to do a machine learning task. In our case the machine learning task is Medicare Fraud detection.

The results of our experiments with Part B data show that the model's generalization was improved with the use of 10 features, compared to using all 80 features, based on AUPRC scores. All 10 features are listed in Table 25.

Table 26 Medicare Part D 10 most important features

Tot_Clms
Tot_30day_Fills_sum
Tot_Benes_sum
Tot_Clms_sum
Tot_Day_Suply_sum
Tot_Day_Suply
Tot_30day_Fills
Gnrc_Tot_Clms
GE65_Tot_Clms
Tot_Drug_Cst_std

Looking at these 10 features for possible causal inferences can provide information on why fraud may have occurred and what features to focus on for particular fraudulent activities. Note that there are several engineered features with suffixes such as "_sum" or"_std" which are lumped in with their base feature. For this analysis we discard those features, and focus on the remaining six features. From these, there are some potential causal interpretations in relation to fraudulent activities.

- Tot_Srvcs represents the number of services provided, which can vary by provider. A higher value of Tot_Srvcs could indicate fraudulent activities like unnecessary procedures or services.
- Avg_Mdcr_Pymt_Amt depicts the average amount that Medicare paid after deductible and coinsurance amounts have been deducted for each service item. A higher value could indicate that a provider is charging more than what is typical for a service—possible fraud.
- Tot_Bene_Day_Srvcs denotes the number of distinct Medicare beneficiary per day services. This metric removes double counting from the line service variable for an occurrence of a unique service. A higher value could imply that a provider is delivering unnecessary services or performing services more frequently than required.
- Avg_Sbmtd_Chrg represents the average of the charges that the provider submitted for a service. A higher value could suggest that a provider is charging more than what is typical for a service—pointing toward possible fraud.
- Tot_Benes represents the number of distinct Medicare beneficiaries receiving the service for each Rndrng_NPI, HCPCS_Cd, and Place_Of_Srvc. A higher value could indicate that a provider is performing unnecessary procedures or services or that they are seeing an unusually high number of patients.
- Rndrng_Prvdr_Type is derived from the provider specialty code reported on the claim. For providers that reported more than one specialty code on their claims, this is the code associated with the largest number of services. This, coupled with other features, could provide insight into what type of procedures and services the provider is offering.

The results of our experiments with Part D data show that the model's generalization was improved with the use of 10 features, compared to using all 82 features, based on

AUPRC scores. Moreover, 7 features improved the model over using all features. Even so, all 10 features are listed in Table 26 for reference.

Looking at these 10 features for possible causal inferences can provide information on why fraud may have occurred and what features to focus on for particular fraudulent activities. Note that there are several engineered features with suffixes such as "_sum" or "_std" which are lumped in with their base feature. From these, there are some potential causal interpretations in relation to fraudulent activities.

- Tot_Clms, Tot_30day_Fills, and Tot_Day_Suply, are variables related to the number of drugs prescribed and dispensed. High values in these features could indicate possible fraudulent activities such as over-prescribing, over-dispensing, or even drug diversion.[2]
- Tot_Drug_Cst, which reflects the total cost of the associated claims, can also be a useful variable to detect fraud, especially if there are significant deviations from the expected cost in total or in relation to other features like total claims or beneficiaries.
- Tot_Benes is the total number of unique beneficiaries with one or more claims for a drug. As such, providers may claim many treatments for beneficiaries that could increase the possibility of a fraudulent claim.
- Additionally, other features like Gnrc_Tot_Clms and GE65_Tot_Clms could also be useful in detecting fraud. The former is related to the number of claims for generic drugs, which could be an indicator of lower quality care or prescription errors. The latter reflects the number of claims for beneficiaries aged 65 and older, a population that is at higher risk of healthcare fraud.

Conclusions

We have shown how a novel feature selection technique can be employed to generate more explainable models, as well as significantly decrease the size of a highly imbalanced Big Data dataset without necessarily compromising classification performance. Statistical analysis is essential in order to make the determination as to what the minimum number of features are that one can eliminate from a dataset and build models that yield acceptable performance. Therefore, we would like to point out that one should employ statistical tests similar to the tests we conduct as a part of the feature selection technique.

In this study, we employ an ensemble supervised feature selection technique to rank features of two Big Data datasets. The datasets are derived from Medicare Part D and Medicare Part B insurance claims data, and labeled with data from the LEIE. They qualify as Big Data, and they are highly imbalanced. Therefore, our results demonstrate that our feature selection technique is a viable technique that can work for reducing the dimensionality of other highly imbalanced Big Data for supervised machine learning tasks. Taking our results for the Part D dataset as an example, our feature selection technique reduces a dataset from 82 to 10 features. Yet, models built on the reduced set of features yield classification results that are significantly better than using all 82 features.

[2] https://www.cdc.gov/injectionsafety/drugdiversion/index.htm.

Another important advantage of our feature selection technique is its running time. Our technique runs in the time that it takes to fit a collection of six classifiers to the dataset. If the fitting process for each of the classifiers is run in parallel, then the running time of the feature selection technique is bounded from above by the time it takes to fit one classifier. This is a far better running time than the brute-force approach of trying all possible combinations of features. The brute-force approach requires exponential resources. Since our technique reduces the size of the dataset, this also improves the running time of many learners that one can fit to it, to train models for Medicare Fraud detection. Moreover, the reduction in size of the dataset also helps control overfitting. We conjecture this is why we observe better performance in terms of AUPRC for models built with fewer features

Explainable models are an additional result of our feature selection technique. When one can show that models built with a reduced number of features yield performance equivalent to using all features, one can make the case that the reduced number of features contains all the information that the algorithm requires in order to successfully carry out the machine learning task. When fewer features are required, it is easier to understand and explain what sort of information a model uses. Future work should consider applying the techniques discussed here to other Big Data in other application domains.

Abbreviations

ANOVA	Analysis of variance
AUC	Area under the receiver operating characteristic curve
AUPRC	Area Under the Precision Recall Curve
CMS	Centers for Medicare and Medicaid Services
CPU	Central Processing Unit
DMEPOS	Durable Medical Equipment Prosthetics Orthotics and Supplies
EFB	Exclusive Feature Bundling
ET	Extremely Randomized Trees
GBDT	Gradient Boosted Decision Trees
GOSS	Gradient Based One-Side Sampling
GPU	Graphics processing unit
HCC	Hierarchical Condition Categories
HCPCS	Healthcare Common Procedure Coding System
HSD	Honestly Significant Difference
LA	Long-acting
LEIE	List of Excluded Individuals and Entities
LIS	Low-Income Subsidy
LSTM	Long-Short Term Memory
MAPD	Medicare Advantage Prescription Drug Plan
MCC	Matthews' Correlation Coefficient
MINN-AE	Multiple Inputs Neural Network Auto-Encoder
NPI	National Provider Identifier
OIG	Office of the Inspector General
PDP	Prescription Drug Plan
SVM	Support Vector Machine

Acknowledgements
The authors would like to thank the various members of the Data Mining and Machine Learning Laboratory, Florida Atlantic University, for their assistance with the reviews.

Author contributions
JTH conducted experiments, and contributed to the manuscript. RAB and HJW contributed to the manuscript. TMK provided oversight of experiments, coordinated research, and contributed to the manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

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

Received: 17 June 2023 Accepted: 30 August 2023
Published online: 09 October 2023

References

1. Zuech R, Khoshgoftaar TM. A survey on feature selection for intrusion detection. In: Proceedings of the 21st issat international conference on reliability and quality in design; 2015. p. 150–5.
2. Centers for medicare and medicaid services: about CMS; 2023. https://www.cms.gov/About-CMS/About-CMS.
3. Civil Division, U.S. Department of Justice: fraud statistics, overview; 2020. https://www.justice.gov/opa/press-release/file/1354316/download.
4. Centers for Medicare and Medicaid Services: 2019 estimated improper payment rates for centers for medicare & medicaid services (CMS) programs; 2019. https://www.cms.gov/newsroom/fact-sheets/2019-estimated-improper-payment-rates-centers-medicare-medicaid-services-cms-programs.
5. Bauder R, Khoshgoftaar TM, Seliya N. A survey on the state of healthcare upcoding fraud analysis and detection. Health Serv Outcomes Res Methodol. 2017;17:31–55.
6. Mayaki MZA, Riveill M. Multiple inputs neural networks for fraud detection. In: 2022 international conference on machine learning, control, and robotics (MLCR). New York: IEEE; 2022. p. 8–13.
7. LEIE: office of inspector general Leie downloadable databases. https://oig.hhs.gov/exclusions/index.asp.
8. Salekshahrezaee Z, Leevy JL, Khoshgoftaar TM. A class-imbalanced study with feature extraction via pca and convolutional autoencoder. In: 2022 IEEE 23rd international conference on information reuse and integration for data science (IRI). New York: IEEE; 2022. p. 63–8.
9. Boyd K, Eng KH, Page CD. Area under the precision-recall curve: point estimates and confidence intervals. In: Joint European conference on machine learning and knowledge discovery in databases. Berlin: Springer; 2013. p. 451–66.
10. Waspada I, Bahtiar N, Wirawan PW, Awan BDA. Performance analysis of isolation forest algorithm in fraud detection of credit card transactions. Khazanah Informatika: Jurnal Ilmu Komputer dan Informatika 2020;6(2):165–75.
11. Kaggle: credit card fraud detection dataset; 2016. https://www.kaggle.com/mlg-ulb/creditcardfraud.
12. Wang H, Khoshgoftaar TM, Napolitano A. A comparative study of ensemble feature selection techniques for software defect prediction. In: 2010 ninth international conference on machine learning and applications. New York: IEEE; 2010. p. 135–40.
13. Sailaja C, Teja GSSK, Mahesh G, Reddy PRS. Detection of fraudulent medicare providers using decision tree and logistic regression models. J Cardiovasc Dis Res. 2021;12(3):3343–52.
14. Bekkar M, Djemaa HK, Alitouche TA. Evaluation measures for models assessment over imbalanced data sets. J Inf Eng Appl. 2013;3(10):27–38.
15. Gupta RY, Mudigonda SS, Baruah PK. A comparative study of using various machine learning and deep learning-based fraud detection models for universal health coverage schemes. Int J Eng Trends Technol. 2021;69(3):96–102.
16. Herland M, Khoshgoftaar TM, Bauder RA. Big data fraud detection using multiple medicare data sources. J Big Data. 2018;5(1):1–21.
17. The centers for medicare and medicaid services: medicare physician & other practitioners—by provider and service; 2021. https://data.cms.gov/provider-summary-by-type-of-service/medicare-physician-other-practitioners/medicare-physician-other-practitioners-by-provider-and-service.
18. The Centers for Medicare and Medicaid Services: medicare part D prescribers—by provider and drug; 2021. https://data.cms.gov/provider-summary-by-type-of-service/medicare-part-d-prescribers/medicare-part-d-prescribers-by-provider-and-drug.
19. The Centers for Medicare and Medicaid Services: medicare durable medical equipment, devices & supplies—by referring provider and service; 2021. https://data.cms.gov/provider-summary-by-type-of-service/medicare-durable-medical-equipment-devices-supplies/medicare-durable-medical-equipment-devices-supplies-by-referring-provider-and-service.
20. Johnson JM, Khoshgoftaar TM. Data-centric ai for healthcare fraud detection. SN Comput Sci. 2023;4(4):389.
21. The Centers for Medicare and Medicaid Services: medicare physician & other practitioners—by provider data dictionary; 2021. https://data.cms.gov/resources/medicare-physician-other-practitioners-by-provider-data-dictionary.
22. The Centers for Medicare and Medicaid Services: medicare physician & other practitioners—by provider; 2021. https://data.cms.gov/provider-summary-by-type-of-service/medicare-physician-other-practitioners/medicare-physician-other-practitioners-by-provider.
23. The Centers for Medicare and Medicaid Services: medicare part D prescribers—by provider and drug data dictionary. https://data.cms.gov/resources/medicare-part-d-prescribers-by-provider-and-drug-data-dictionary 2021.
24. The Centers for Medicare and Medicaid Services: medicare part D prescribers—by provider data dictionary; 2020. https://data.cms.gov/resources/medicare-part-d-prescribers-by-provider-data-dictionary.
25. The Centers for Medicare and Medicaid Services: medicare part D prescribers—by provider; 2021. https://data.cms.gov/provider-summary-by-type-of-service/medicare-part-d-prescribers/medicare-part-d-prescribers-by-provider.

26. The Centers for Medicare and Medicaid Services: medicare physician & other practitioners—by provider and service data dictionary; 2021. https://data.cms.gov/resources/medicare-physician-other-practitioners-by-provider-and-service-data-dictionary.
27. 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). New York: IEEE; 2016. p. 11–9.
28. Chen T, Guestrin C. Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining—KDD '16; 2016.
29. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu T-Y. Lightgbm: a highly efficient gradient boosting decision tree. Adv Neural Inf Process Syst. 2017;30:3146–54.
30. Geurts P, Ernst D, Wehenkel L. Extremely randomized trees. Mach Learn. 2006;63(1):3–42.
31. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.
32. Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, Gulin A. Catboost: unbiased boosting with categorical features. In: Advances in neural information processing systems. 2018. Vol. 31, p. 2–11.
33. Le Cessie S, Van Houwelingen JC. Ridge estimators in logistic regression. J R Stat Soc Ser C (Appl Stat). 1992;41(1):191–201.
34. Breiman L, Friedman J, Stone CJ, Olshen RA. Classification and regression trees. Taylor & Francis; 1984.
35. Friedman JH. Greedy function approximation: a gradient boosting machine. Ann Stat. 2001;29:1189–232.
36. Hancock JT, Khoshgoftaar TM. Gradient boosted decision tree algorithms for Medicare fraud detection. SN Comput Sci. 2021;2(4):1–12.
37. Leevy JL, Hancock JT, Zuech R, Khoshgoftaar TM. Detecting cybersecurity attacks using different network features with lightgbm and xgboost learners. In: 2020 IEEE second international conference on cognitive machine intelligence (CogMI). New York: IEEE; 2020. p. 190–7.
38. Hancock JT, Khoshgoftaar TM. Catboost for big data: an interdisciplinary review. J big data. 2020;7(1):1–45.
39. Breiman L. Bagging predictors. Mach Learn. 1996;24(2):123–40.
40. Efron B, Tibshirani RJ. An introduction to the bootstrap. Boca Raton: CRC Press; 1994. p. 5–6.
41. Hancock JT, Khoshgoftaar TM, Johnson JM. A comparative approach to threshold optimization for classifying imbalanced data. In: The international conference on collaboration and internet computing (CIC). New York: IEEE; 2022.
42. Gu Q, Cai Z, Zhu L, Huang B. Data mining on imbalanced data sets. In: 2008 international conference on advanced computer theory and engineering. New York: IEEE; 2008. p. 1020–1024.
43. Kuncheva LI, Arnaiz-Gonzalez A, Díez-Pastor J-F, Gunn IA. Instance selection improves geometric mean accuracy: a study on imbalanced data classification. Progr Artif Intell. 2019;8(2):215–28.
44. Chicco D, Jurman G. The advantages of the Matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. BMC Genom. 2020;21(1):1–13.
45. Hastie T, Tibshirani R, Friedman JH, Friedman JH. The elements of statistical learning: data mining, inference, and prediction, vol. 2. Heidelberg: Springer; 2009.
46. Shannon CE. A mathematical theory of communication. Bell Syst Tech J. 1948;27(3):379–423.
47. Iversen GR, Norpoth H. Analysis of variance, vol. 1. Newbury Park: Sage; 1987.
48. Tukey JW. Comparing individual means in the analysis of variance. Biometrics. 1949;5:99–114.
49. Witten IH, Frank E, Hall MA. Data mining: practical machine learning tools and techniques. The Morgan Kaufmann series in data management systems. Pittsburgh: Elsevier Science; 2011.
50. Van Rossum G, Drake F. Python 3 reference manual createspace. Scotts Valley; 2009.
51. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12:2825–30.
52. Calvert CL, Khoshgoftaar TM. Threshold based optimization of performance metrics with severely imbalanced big security data. In: 2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI). New York: IEEE; 2019. p. 1328–34.

Publisher's Note