2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA) | 978-1-6654-6283-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICMLA55696.2022.00224

Informative Evaluation Metrics for Highly Imbalanced Big Data Classification

John Hancock, Taghi M. Khoshgoftaar, and Justin M. Johnson
College of Engineering and Computer Science
Florida Atlantic University
Boca Raton, Florida 33431
jhancoc4@fau.edu, khoshgof@fau.edu, jjohn273@fau.edu

***Abstract*—We conduct experiments that show the Area Under the Precision Recall Curve (AUPRC) metric provides a more meaningful insight into the impact of Random Undersampling than Area Under the Receiver Operating Characteristic Curve (AUC). Evaluating experiments with multiple metrics is a robust method for overcoming challenges in Machine Learning, such as class imbalance. Random Undersampling is a technique to deal with class imbalance. We find Random Undersampling may provide an improvement to AUC scores. However, at the same time, Random Undersampling may be detrimental to AUPRC scores. AUPRC is a metric that involves precision, whereas AUC does not. In the classification of imbalanced Big Data, an increase in false positive counts has a more noticeable drop in precision scores. Therefore, in application domains where false positives are undesirable, optimizing models for AUPRC is a wise choice. Our contribution is to compare the performance of models in terms of AUPRC and AUC to show the impact of Random Undersampling on the classification of imbalanced Big Data. We compare the performance via experiments in the classification of highly imbalanced Big Data. Models are built with data in its original class ratio, and with data undersampled into 5 distinct class ratios. We report the results of 600 experiments where we apply Random Undersampling to a dataset with about 175 million instances. To the best of our knowledge we are the first to utilize Medicare Part D data which became available in 2021.**

***Keywords*—Extremely Randomized Trees, XGBoost, Class Imbalance, Big Data, Undersampling, AUC, AUPRC**

Introduction

Random Undersampling (RUS) is a technique for addressing class imbalance. For a survey of other techniques to address class imbalance, please see [1]. It is attractive for working with imbalanced Big Data because it reduces the size of the training data. Less training data means lower resource consumption, both in terms of time, and hardware resources. Applying RUS to imbalanced Big Data with a minority to majority class ratio of less than one in a thousand to induce a class ratio of 1:1 will reduce the size of the training data by a factor of more than one thousand. Such a reduction in size can make experiments possible that would otherwise not be, due to resource constraints. With RUS, one should note that there is a trade-off between size reduction and the discarding of important information about the majority class. On one hand, researchers may be delighted to find out that applying RUS to large datasets to reduce the size of the training data has no impact, or even a positive impact, on one performance metric. On the other hand, they would be equally crestfallen to see RUS can be quite detrimental to performance in terms of another metric. Our contribution is to show that, for classifying highly imbalanced Medicare Big Data to detect insurance fraud, RUS yields the best performance in terms of Area Under the Receiver Operating Characteristic Curve (AUC) [2], while simultaneously having a negative impact on Area Under the Precision Recall Curve (AUPRC) [3]. More specifically, we show applying RUS to induce a class ratio of 1:9 with imbalanced Big Data yields the best performance, in terms of AUC, for two popular ensemble learners. However, for the same experiments, applying RUS to induce a 1:9 class ratio causes the performance of both classifiers, in terms of AUPRC, to be significantly worse. For a comparative study of classification performance metrics, please see [4]. Our results and analysis show that the AUC performance metric is overly optimistic because it is relatively insensitive to false positives.

Our research is also a contribution to the field of automated Medicare insurance fraud detection. Primarily for Americans aged 65 or older, Medicare is the United States' public health insurance program. To foment research, the Centers for Medicare and Medicaid Services, (CMS), publishes the "Medicare Provider Utilization and Payment Data: Part D Prescriber" [5]. This data is made publicly available on the Internet. For the remainder of our study, we refer to this data as the Part D data. The CMS adds to the Part D data periodically. It now spans the seven years 2013 through 2019. The total number of records in the data has grown to approximately 175 million records.

The rate at which data grows at is one characteristic of Big Data. Each record in the Part D data contains information about one medication that a particular healthcare provider prescribes for one year. Hence, it is entirely possible that one record in the Part D data could represent hundreds of prescriptions a provider writes over the course of a year. We point this out to give a sense for the rate at which providers submit claims to CMS. A tiny percentage of this number of claims may therefore amount to a large sum of money. Moreover, a small fraction of fraudulent claims could be difficult to detect, as the following information bears out. In 2019, the United States Department of Justice reported that it recovered about three billion dollars' worth of funds that were paid out due to fraudulent claims [6]. Separately, the CMS estimates it made $100 billion in improper payments during the same year [7]. The CMS is uncertain about the fraction of the improper payments due to fraud. They admit some improper payments are due to mistakes on their part. This is also documented in [7].

An automated method for detecting fraud is therefore called for, as a way for the CMS to become more certain about where improper payments are going. This would empower the CMS to stop improper payments and spend more funds on appropriate healthcare for Medicare beneficiaries.

Here, we investigate the effect of RUS on the AUC and AUPRC scores of Extremely Randomized Trees (ET) [8] and XGBoost [9] as they classify highly imbalanced Big Data. AUPRC is the area under the curve obtained by plotting precision versus recall as the classifier output probability threshold is varied from zero to one.

We give an informal argument as to why AUPRC is a preferable performance metric in the classification of imbalanced Big Data. We assume a classification of a highly imbalanced Big Data set has occurred, and we have counted the correctly and incorrectly classified instances of the test set. Furthermore, let us assume the positive class is the minority class. The definition of precision is

$$\frac{\text{true positives}}{\text{true positives} + \text{false positives},}$$

and recall is defined as

$$\frac{\text{true positives}}{\text{true positives} + \text{false negatives}.}$$

Models that have high precision and recall scores for any choice of output probability threshold will yield AUPRC scores close to the maximum possible value of one, and models that fit data poorly will yield scores closer to the minimum possible value of zero.

AUC is calculated in a manner similar to AUPRC, however it is the area under the curve obtained from plotting the true positive rate versus the false positive rate, as the classifier output probability threshold varies from zero to one. The true positive rate is defined as

$$\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

and false positive rate is

$$\frac{\text{false positives}}{\text{true negatives} + \text{false positives}}.$$

Notice that recall and true positive rate have the same definitions. Therefore, recall and true positive rate do not play a role in the difference between AUC and AUPRC scores, since they are common to both. Hence, the difference in AUC and AUPRC scores stems from the difference in false positive rate and precision. The denominator of the false positive rate has a term for true negatives. In highly imbalanced Big Data, this number is large, so a classifier must perform poorly before the number of false positives begins to impact the false positive rate. Shifting our focus to precision, under our assumption of highly imbalanced Big Data, the number of true positives is small, thus allowing the false positives term to quickly dominate the calculation of precision. Therefore, precision provides a stronger indication of false positives.

To make our argument concrete, let us use some numbers in a calculation. Assuming we have a test dataset of highly imbalanced Big Data where 1,000,000 instances are in the negative class, 1000 are in the positive class, and we have done a classification where 900 positive instances are classified correctly as true positives, and 1,000 instances of the negative class are classified as false positives. Then, precision is

$$\frac{900}{900 + 1,000} \approx 0.47.$$

However, false positive rate is

$$\frac{1,000}{1,000,000} = 0.001.$$

Now assume the number of false positives has grown to 2,000. Then our calculation for precision becomes

$$\frac{900}{900 + 2,000} \approx 0.31,$$

and the false positive rate becomes

$$\frac{2,000}{1,000,000} = 0.002.$$

We see precision drops by 0.16 (lower is worse), whereas false positive rate increases by 0.001 (higher is worse). Both metrics show worse performance, but the size of the negative class overwhelms the change in false positives for the false positive rate calculation. However, it is not a factor in the calculation of precision. Hence, the magnitude of the drop in precision is a more noticeable indicator of the increased number of false positives.

Since our application domain is fraud detection, a false positive is tantamount to convicting an innocent provider of fraudulent activity. Thus, diminishing the number of law-abiding providers eligible to provide healthcare to Medicare beneficiaries, and limiting the availability of their care. Therefore, it would be self-defeating for us to judge the effect of RUS on the performance of classifiers with a metric that does not give us a sense of false positives. For these reasons, we suggest that researchers in other application domains use AUPRC when there are consequences for false positives. We rely on reports of prior work on Medicare Fraud detection that show supervised techniques outperform other methods [10]. To convey our message on the impact of RUS on AUC and AUPRC scores, the remainder of our study is organized into the following sections: Related Works, Algorithms, Data Description and Preparation, Methodology, Results and Statistical Analysis, and Conclusions.

Related Works

We conducted a thorough review of studies related to the classification of imbalanced Big Data to determine whether we have a contribution to make. The search was conducted by carefully reviewing the collection of studies returned by searching academic research databases for peer-reviewed studies on applications of Machine Learning to classify imbalanced Big Data, and techniques to address class imbalance. For a general review of techniques for addressing class imbalance in Big Data, please see [11]. After our review of the studies, we found none of the documented experiments involve a dataset as large as the one we use. Moreover, none of the studies cover application of the ET classifier; therefore we are the first to report on the impact of RUS when ET is used to classify Big Data. Most importantly, none of the studies investigate the effect of RUS on AUC versus AUPRC. Therefore, our study makes a contribution because it provides novel findings.

In 2018, Bauder *et al.* released a study on sampling techniques for classifying imbalanced Big Data [12]. Their application domain is the same as ours, Medicare Fraud detection. However, the authors perform experiments on a smaller dataset, based on a combination of data from Medicare Parts A, B and Durable Medical Equipment Prosthetics Orthotics and Supplies (DMEPOS). Their combined dataset contains fewer than one million instances. In their study, Bauder *et al.* use six data sampling techniques: Random Oversampling (ROS), Adaptive Synthetic (ADASYN) [13], Synthetic Minority Oversampling Technique (SMOTE) [14], two variations of Borderline SMOTE (also covered in [14]), and RUS. They use each sampling technique to induce class ratios of 50:50, 65:35, 25:75, 10:90, and 1:99. In addition, they also conduct experiments with models trained on the dataset in its initial state, which has a class ratio of 473:759,267. All their experiments are conducted as programs running on the Apache Spark [15] platform. Therefore, they use the Apache Spark implementations of Gradient Boosted Trees [16], Logistic Regression [17], and Random Forest. Bauder *et al.* evaluate experimental outcomes in terms of AUC, and AUC only. Their conclusion is that classifiers fitted to data that is pre-processed with RUS yield the best performance. Hence, RUS is the only sampling technique we use in our study. However, in addition to AUC, we also evaluate experimental outcomes in terms of the AURPC metric. Like Bauder *et al.*, we find that classifiers trained on data that is treated with RUS yield higher AUC scores, but we go on to show that RUS can be a detriment to AUPRC scores. This is an important aspect of our study that sets our it apart from Bauder *et al.*'s.

Calvert and Khoshgoftaar study the performance of multiple classifiers in terms of multiple metrics [18]. In their paper, they document experimental results pertaining to the classification of a highly imbalanced information systems network traffic dataset. The classification objective is to distinguish malicious and benign network traffic. The overwhelming majority of traffic in their dataset is benign. The dataset has approximately 1.7 million instances, and has a minority to majority class ratio of approximately 0.0014. On one hand, the class imbalance in their data is similar to ours; on the other, they work with fewer instances than we do. Calvert and Khoshgoftaar write that data for their experiments is collected in a real-world network environment. However, they also write that they generate the attack traffic themselves, which means their results pertain to a simulated phenomenon. Our results pertain to instances of Medicare fraud perpetrated by convicted individuals and entities; therefore, our results pertain to a natural phenomenon. Calvert and Khoshgoftaar report that their rankings of classifiers vary widely depending on the performance metric. Their contribution is to show that while one classifier yields the best performance in terms of AUC, it yields far worse performance in terms of the remaining metrics. Their conclusion is that the best classifier is the one that yields the best performance for the largest number of metrics. Calvert and Khoshgoftaar's study has nothing to do with the impact of RUS on different performance metrics.

Del Río *et al.* published a study in 2014 that documents experiments to determine the impact of RUS in classifying Big Data. In their study, they evaluate the performance of Random Forest classifiers in terms of Geometric Mean [19], and β-f-measure [20]. Their results show mixed results in performance when RUS is applied. There are two important differences between our study and Del Río *et al.*'s. The first is that the largest data set used in their experiments has less than six million instances. The second major difference in our studies is that Del Río *et al.* apply RUS to induce a class ratio of 1:1. We apply RUS to induce many class ratios. Therefore, our study gives a better insight into the impact of RUS on performance metrics.

Johnson and Khoshgoftaar investigate the impact of RUS on the performance of Neural Network classifiers [21]. They employ AUC, Geometric Mean, true positive rate, and true negative rate to evaluate the performance of their classifiers in the task of Medicare Fraud detection. They do not report results in terms of AUPRC, so there is no investigation into the impact of RUS on AUPRC scores like we do here. They find that applying RUS to induce a class ratio where the minority class occupies more than 20% of the dataset causes performance in terms of AUC to suffer. Furthermore, their results show that all metrics indicate worse performance as RUS is applied to induce class ratios where the minority class forms a larger fraction of the training data. The data Johnson and Khoshgoftaar work with comes from the same source as the data we work with, the CMS. However, they apply an aggregation technique that reduces the size of the data to under five million instances. As part of the aggregation process they discard the highest cardinality categorical features, and one-hot encode the remaining categorical features. There are many options for encoding categorical features; we urge interested readers to consult [22] for a review of those options. Since we employ CatBoost encoding to handle categorical features, we are able to use all categorical features. Another advantage to CatBoost encoding is that it does not require augmenting the dataset with additional features. This makes it more practical for working with large datasets.

Sleeman and Krawczyk published a study in 2021 on their work pertaining to use of the Apache Spark platform for classification of imbalanced Big Data [23]. The datasets they perform their experiments with contain at most three million instances. Sleeman and Krawczyk show improved performance when RUS is applied, however, they do not include results for the AUPRC metric. One important point that differentiates our work from Sleeman and Krawczyk's is the number of levels of RUS in our experiments. Sleeman and Krawczyk apply RUS to induce a 1:1 minority to majority class ratio, whereas we apply RUS to induce five levels of class ratios. Therefore, we provide a more complete picture of the impact of RUS when employed to induce different class levels. The aspects of our study that set it apart from Sleeman and Krawczyk also enable us to provide novel conclusions about the impact of RUS and which metrics provide better insight into RUS's impact.

In their 2017 publication Fernandez *et al*. document experiments performed on the Apache Spark [15] and Hadoop [24] platforms. Their experiments cover the application of RUS to imbalanced Big Data. The larger of the two datasets they work with contains approximately 12 million instances. They find that applying RUS to induce a class ratio of 1:1 in their training data improves performance in terms of Geometric Mean. We provide a more extensive review of the impact of RUS since we perform experiments where we induce many class ratios in the training data. Another difference we notice between our study and Fernandez *et al*. is the initial imbalance ratio of our datasets. Fernandez *et al*. report that the datasets they perform experiments with have an initial minority to majority class ratio of 1:50. The dataset we use has an initial imbalance ratio of approximately 1:256. Therefore, our results pertain not only to a larger dataset, but also to a more imbalanced dataset. Finally, Fernandez *et al*. do not cover the subject of our research here, which is to thoroughly compare the impact of varying levels of RUS on AUC versus AUPRC scores.

In surveying the literature, we see that there are no studies that offer the careful examination of the impact of RUS on both AUC and AUPRC scores in the classification of imbalanced Big Data. Furthermore, since none of the related works cover experiments with the ET classifier, we make a novel contribution to show how the ET classifier performs in a classification task on imbalanced Big Data. Clearly, findings we impart in our study are unique, and represent a contribution to Machine Learning research.

Algorithms

We use two popular, open source classifiers for the experiments documented in this study, XGBoost and ET. Both classifiers employ ensemble techniques; however they come from different families of Machine Learning algorithms. XGBoost comes from the Gradient Boosted Decision Tree (GBDT) family of algorithms. ET is a bagging technique. Since XGBoost and ET are different approaches to Decision Tree-based modeling, we believe that when they yield similar patterns in performance metrics there is a better potential that the patterns will hold more generally.

Conceptually, ET is the simpler of the two classifiers. Geurts *et al*. introduce ET in their seminal paper published in 2006 [8]. ET is a direct descendant of Random Forest [25]. Therefore, ET exploits the same probabilistic argument that buttresses Random Forest. The argument is that the probability of a majority in an ensemble of weak learners to make a correct prediction increases with the size of the ensemble. To further explain ET, we must first define the term "split" in the context of decision trees. A split in a decision tree pertains to the internal nodes in the tree. The split is the value that a feature is compared to in order to determine which edge to follow out of the node. Since splits are crucial to the evaluation of decision trees, methods for determining the optimal values for splits are a key area of research for decision trees. The method ET employs for determining splits is to randomly choose their values. Interestingly, random selection of splits may yield performance similar to, or better than, more sophisticated methods. In fact, our experimental results show ET sometimes outperforms XGBoost.

XGBoost is an implementation of the GBDT ensemble learning technique which includes several enhancements to the original GBDT implementation. Friedman published the seminal work on GBDTs in 2001 [26]. In that paper, Friedman proposes an iterative process for building an ensemble of learners. In this process, the next learner to be added to the ensemble is the one whose output values, when added to the output of the ensemble, best reduce the error of the ensemble. Chen and Guestrin introduce XGBoost in their 2016 study [9]. For a study comparing the performance of XGBoost with other GBDT learners in the classification of Big Data, please see [27]. XGBoost builds on the success of the GBDT technique in several ways. One way XGBoost improves on the GBDT technique is with the introduction of a regularized loss function that helps prevent overfitting. Another enhancement XGBoost makes involves splits in its constituent trees that is an optimization for finding splits in data that has sparsity. Sparsity in data is data where most attributes are of one value with occasional aberrations. As its name implies, XGBoost's "sparsity aware split finding" technique is an enhancement for calculating splits that works more efficiently when input data is sparse. XGBoost offers further enhancements to calculating splits efficiently, particularly in the form of a technique for estimating optimal split values. Chen and Guestrin simply name this technique the "approximate algorithm". The approximate algorithm for calculating optimal splits enables application of the GBDT technique in distributed computing environments as well as application domains where it is not feasible to load all the data one wishes to use into Random Access Memory.

ET relies more on random chance to build an ensemble of weak learners. XGBoost takes a more systematic approach. Leveraging the difference in approaches makes for sound experimental design, and lends confidence that phenomena we uncover pertain to more than one Machine Learning approach.

Data Description and Preparation

In the interest of providing reproducible research, we discuss our techniques for preparing the Part D data used in our experiments. As mentioned previously, the CMS offers the Part D data for public use for download from its website. The most recently available data we use became available on the CMS's site in 2021. To the best of our knowledge we are the first to utilize the latest Part D data in a study. A valuable tool for understanding that data is the CMSs data dictionary [28]. This data dictionary supplies the definitions for features in Table I. We would like to emphasize that the nine features listed in Table I are a subset of elements defined in the CMS's data dictionary. We would like to point out that some elements of the CMS data are not appropriate for Machine Learning because they serve as identifiers for specific providers. Many algorithms will overfit to these features and fail to generalize. Therefore, we discard six such features. They are the National Provider ID (NPI), and features for the provider name, and provider address. The names of the features are Prscrbr_NPI, Prscrbr_Last_Org_Name, Prscrbr_First_Name, Prscrbr_City, Prscrbr_State_Abrvtn, and Prscrbr_State_FIPS. The nine features we decided to use are summarized in Table I. We augment the feature definitions from CMS in Table I with information on categorical features of the data we work with.

TABLE I
FEATURES OF THE PART D DATASET

Feature Name	Description
Prscrbr_Type	Derived from the Medicare provider/supplier specialty code; categorical, 249 distinct values
Prscrbr_Type_Src	Source of the Medicare provider/supplier specialty code; categorical, 2 distinct values
Brnd_Name	Brand name (trademarked name) of the drug filled; categorical, 3,907 distinct values
Gnrc_Name	A term referring to the chemical ingredient of a drug rather than the trademarked brand name under which the drug is sold; categorical, 2,272 distinct values
Tot_Clms	The number of Medicare Part D claims
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

After preparing the data from CMS we must label it in order to do supervised classification. Although we do not use the NPI as a feature for Machine Learning, it is vital as part of the labeling process. We join records of the Part D data by the NPI to records of the List of Excluded Individuals and Entities (LEIE), [29]. The OIG publishes the LEIE on a monthly basis. Individuals and Entities listed in the LEIE are prohibited from filing claims to Medicare due to fraudulent activity. If a provider appears in the LEIE we label all records pertaining to the provider dated prior to, and up until the end of the exclusion period as fraudulent. We use the same logic documented in [30] for selecting the NPIs of providers that we label as fraudulent in the Part D data. Exclusions have definite beginning and ending times. When the exclusion period for a particular individual or entity ends, that individual or entity is removed from the LEIE. Therefore, it is necessary to use the Internet Archive Tool[1] to obtain older versions of the LEIE, to label the Part D data for providers that were once on the LEIE but have been removed. When we have completed assigning labels to the Part D data, we have a dataset with 173,677,665 instances.

METHODOLOGY

Our method of conducting experiments is to run programs to train and evaluate Machine Learning models. Our programs are implemented in the Python programming language [31]. The Machine Learning algorithms used in this study are libraries that our programs rely on. All the software is available at no cost to the end user. The XGBoost library is available for download [2]. ET is part of the Scikit Learn library [32]. Scikit Learn is a large library containing many Machine Learning algorithms and other supporting software for Machine Learning research and applications.

An additional library that is vital for reproducing our methods is the category_encoders Python library [33]. This library provides a CatBoost encoding module one may use to encode categorical features according to the technique Prokhorenkova *et al.* introduce in [34]. For abroad survey of CatBoost's applications, please see [35]. One important practice to observe when using the CatBoost encoder has to do with a preparation step one must perform prior to using the encoder to encode categorical features. The preparation step involves fitting the encoder to the data to be encoded. If one fits the encoder to the entire dataset, information about the test data will leak into the encoded values in the training data, potentially causing models that use the encoded features to overfit. Therefore, the CatBoost encoder must be fit to the training data only.

For every experiment, we select a classifier (XGBoost or ET) and a class ratio that we wish to induce on the training data. We use the Imblearn [36] library's RandomUnderSampler module to control class ratios for all experiments, except for the case when we leave the class ratio at its initial value. We apply RUS to obtain 6 different positive:negative class ratios, including 1:1, 1:3, 1:9, 1:27, 1:81, and 1:256, where 1:256 is the original distribution. For every combination of class ratio and classifier, we perform ten iterations of five-fold cross validation. Therefore, the total number of experimental outcomes we record is $2 \times 5 \times 10 \times 6 = 600$.

When the first two steps of encoding and undersampling are completed, we have prepared training data that we can then pass to either XGBoost or ET for fitting. After fitting to the training data, we evaluate the models' classification performance by recording the AUC and AUPRC scores. Since

[1]http://archive.org/web
[2]https://pypi.org/project/xgboost/

we do ten iterations of five-fold cross validation, we obtain 50 records of both metrics. We do repeated iterations of cross validation as a precaution against data loss due to random sampling of instances from the majority class.

Another important aspect of our methodology is the hyperparameter settings we use for XGBoost and ET. We set the maximum tree depth of XGBoost to 24. Preliminary experimental results revealed much stronger performance of XGBoost with maximum tree depth set to that level. We also set the "tree_method" parameter of XGBoost to "gpu_hist" which causes XGBoost to use Graphics Processing Unit (GPU) hardware. We do not modify any hyperparameter settings for ET, since we obtained best results in preliminary experiments with default settings for hyperparameters. We run our experiments in a distributed computing platform where available nodes have Intel Xeon Central Processing Units (CPUs) with 16 cores, 256 GB RAM per CPU and Nvidia V100 GPUs.

Results and Statistical Analysis

Please see Table II for a listing of AUC and AUPRC scores. One should note that AUC scores for both ET and XGBoost fluctuate according to the level of RUS applied. Overall, ET has slightly better AUC scores than XGBoost. However, the more striking trend in the results is the effect of RUS on the AUPRC scores of ET and XGBoost. We see the AUPRC scores of XGBoost are particularly impacted, when we apply RUS to induce the 1:1, 1:3 and 1:9 class ratios.

TABLE II
Part-D Mean and Standard Deviation of AUC and AUPRC with varying levels of RUS (10 iterations of 5-fold cross-validation)

Class Ratio	ET		XGB-24	
	AUC	AUPRC	AUC	AUPRC
1:1	0.97625 (0.00033)	0.83614 (0.00307)	0.95292 (0.00059)	0.11168 (0.00158)
1:3	0.97586 (0.00038)	0.88826 (0.00181)	0.96809 (0.00045)	0.32721 (0.00572)
1:9	0.97464 (0.00038)	0.91341 (0.00103)	0.97256 (0.00037)	0.58986 (0.00571)
1:27	0.97244 (0.00037)	0.92451 (0.00080)	0.97363 (0.00037)	0.70815 (0.00375)
1:81	0.96966 (0.00035)	0.92925 (0.00073)	0.97348 (0.00039)	0.74950 (0.00335)
Unchanged	0.96746 (0.00038)	0.93288 (0.00073)	0.97273 (0.00042)	0.76105 (0.00406)

Standard deviations are below AUC and AUPRC scores in parenthesis. ET is the Extremely Randomized Trees classifier. XGB-24 is the XGBoost classifier with maximum tree depth set to 24.

Our next step in the analysis of our results is to use the same experimental outcome data summarized in Table II in two analysis of variance (ANOVA) tests [37]. We select a significance level of $\alpha = 0.01$ for all tests. The first ANOVA test we report on is an analysis of the variance in AUC scores. Specifically, we compute a two-factor analysis. The first factor is RUS. As stated previously the RUS factor has six levels. The second factor is the type of classifier (CLF). Since we use XGBoost and ET, the classifier factor has two levels. Table III has the results of the ANOVA test with RUS and classifier as factors. The Pr(>F), or p-values, for the RUS and CLF factors in the ANOVA test are quite small. Therefore, both factors have a statistically significant effect on AUC scores.

TABLE III
ANOVA for RUS and CLF as factors of performance in terms of AUC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
RUS	5	0.00539	0.00108	44.90	*
CLF	1	0.00218	0.00218	90.99	*
Residuals	593	0.01423	0.00002		

* indicates the value is less than 1×10^{-15}.

Because the ANOVA test results show all the factors have a significant impact on AUC scores, we can conduct Tukey's Honestly Significant Difference (HSD) Tests [38], to rank the levels of the factors according to their impact on AUC scores. Factors in the HSD test results are grouped. The labels on the groups are in alphabetical order according to their association with AUC scores. Group 'a' is the group with the highest AUC scores, group 'b' is the group with the second-highest AUC scores, and so on. When there are several levels of a factor in an HSD group, the differences in outcomes for those levels is not significant. The HSD test results are in Tables IV and V.

We see the 1:9 and 1:27 RUS levels associated with the best performance in Table IV. In the next, row we see the HSD group 'ab' contains the 1:3 and 1:81 levels of the RUS factor. The 'ab' label stems from the fact that the range of AUC scores in this group overlap with in the ranges of scores in groups 'a' and 'b'. Furthermore, we see applying RUS to induce the 1:1 class ratio is associated with the worst performance. Finally, the initial class ratio, where no RUS is applied, is the group with the third-best performance. We averaged AUC scores across both ET and XGB-24 to provide the HSD test results in Table IV.

TABLE IV
HSD test groupings of AUC for the RUS factor

Group a consists of: 1:9, 1:27
Group ab consists of: 1:3, 1:81
Group b consists of: Unchanged
Group c consists of: 1:1

The HSD test results for the classifier factor are in Table V. ET yields the best performance across all levels of RUS. However, the experiments involving XGBoost and the larger datasets ran much faster than the experiments involving ET because of XGBoost's GPU support. Hence, there is a trade-off between performance and running time in selecting between the two classifiers. Table V contains results that are averaged across all levels of the RUS factor.

TABLE V
HSD test groupings after ANOVA of AUC for the CLF factor

Group a consists of: ET
Group b consists of: XGB-24

Similar to the analysis we do for AUC scores, we use the AUPRC scores summarized in Table II for a second ANOVA table, Table VI. The ANOVA test results show both RUS and choice of classifier have a significant impact on experimental outcomes.

Table VIII contains the result of the HSD test for the classifier factor. The HSD test results confirm that ET is associated with the highest AUPRC scores. Furthermore, we see that not applying RUS is associated with the best performance. The HSD test results for the RUS factor reinforce the key message of our study: a factor in experiments may have a noticeable, negative impact on AUPRC, and simultaneously have no effect on AUC scores.

TABLE VI
ANOVA FOR RUS AND CLF AS FACTORS OF PERFORMANCE IN TERMS OF AUPRC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
RUS	5	11.38	2.28	204.99	*
CLF	1	19.75	19.75	1778.97	*
Residuals	593	6.58	0.01		

* indicates the value is less than 1×10^{-15}. The factors in the ANOVA are Random Undersampling (RUS), and classifier (CLF).

TABLE VII
HSD TEST GROUPINGS AFTER ANOVA OF AUPRC FOR THE RUS FACTOR

Group a consists of: Unchanged, 1:81, 1:27
Group b consists of: 1:9
Group c consists of: 1:3
Group d consists of: 1:1

TABLE VIII
HSD TEST GROUPINGS AFTER ANOVA OF AUPRC FOR THE CLF FACTOR

Group a consists of: ET
Group b consists of: XGB-24

CONCLUSIONS

The outcomes of 600 experiments show that RUS does not significantly improve AUPRC performance. On the other hand, we get a conflicting message from the results concerning RUS and AUC scores, where we see RUS associated with higher AUC scores. As explained previously, precision is the key differentiator between AUC and AUPRC, and precision is more sensitive to false positives. Hence, AUPRC gives a stronger indication of false positive classification errors. Therefore, in application domains where false positives are important, AUPRC may be more informative. Our results imply that one should do analysis of the impact of RUS on AUPRC scores in situations where false positives should be avoided. For Medicare fraud detection, where a false positive is equivalent to convicting the innocent, AUPRC is an important metric.

ET's performance is a noteworthy facet of our results. In terms of either metric, AUC or AUPRC, we find the HSD test results show ET is the classifier associated with higher scores. Moreover, we note that the performance of ET does not drop as steeply as that of XGBoost as we apply RUS to induce smaller majority class sizes. Therefore, we conclude ET is more robust to the removal of instances of the majority class, as compared to XGBoost. Further investigation into whether this robustness holds for other datasets should be a subject for future work.

As we mention above, to the best of our knowledge our experiments constitute a novel combination of ET and CatBoost encoding. Also, to the best of our knowledge, this is the first report of results regarding the impact of RUS on AUC and AUPRC scores in the classification of a dataset on the order of 175 million instances for Medicare insurance fraud detection. Our primary conclusion is that in the classification of imbalanced Big Data AUPRC is a more reliable metric for gauging the impact of RUS on the performance of classifiers.

REFERENCES

[1] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, pp. 1–54, 2019.

[2] M. Bekkar, H. K. Djemaa, and T. A. Alitouche, "Evaluation measures for models assessment over imbalanced data sets," *J Inf Eng Appl*, vol. 3, no. 10, 2013.

[3] K. Boyd, K. H. Eng, and C. D. Page, "Area under the precision-recall curve: Point estimates and confidence intervals," in *Joint European conference on machine learning and knowledge discovery in databases*, Springer, 2013, pp. 451–466.

[4] N. Seliya, T. M. Khoshgoftaar, and J. Van Hulse, "A study on the relationships of classifier performance metrics," in *2009 21st IEEE international conference on tools with artificial intelligence*, IEEE, 2009, pp. 59–66.

[5] The Centers for Medicare and Medicaid Services, *Medicare part d prescribers - by provider and drug*, 2021.

[6] Civil Division, U.S. Department of Justice, *Fraud statistics, overview*, 2020.

[7] Centers for Medicare and Medicaid Services, *2019 estimated improper payment rates for centers for medicare & medicaid services (cms) programs*, 2019.

[8] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[9] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016. DOI: 10.1145/2939672.2939785.

[10] R. A. Bauder and T. M. Khoshgoftaar, "Medicare fraud detection using machine learning methods," in *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, IEEE, 2017, pp. 858–865.

[11] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *Journal of Big Data*, vol. 5, no. 1, pp. 1–30, 2018.

[12] R. A. Bauder, T. M. Khoshgoftaar, and T. Hasanin, "Data sampling approaches with severely imbalanced big data for medicare fraud detection," in *2018 IEEE 30th international conference on tools with artificial intelligence (ICTAI)*, IEEE, 2018, pp. 137–142.

[13] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, IEEE, 2008, pp. 1322–1328.

[14] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*, Springer, 2005, pp. 878–887.

[15] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, *et al.*, "Apache spark: A unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[16] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, *et al.*, "Mllib: Machine learning in apache spark," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–1241, 2016.

[17] S. Le Cessie and J. C. Van Houwelingen, "Ridge estimators in logistic regression," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 41, no. 1, pp. 191–201, 1992.

[18] C. L. Calvert and T. M. Khoshgoftaar, "Threshold based optimization of performance metrics with severely imbalanced big security data," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2019, pp. 1328–1334.

[19] L. I. Kuncheva, Á. Arnaiz-Gonzalez, J.-F. Díez-Pastor, and I. A. Gunn, "Instance selection improves geometric mean accuracy: A study on imbalanced data classification," *Progress in Artificial Intelligence*, vol. 8, no. 2, pp. 215–228, 2019.

[20] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[21] J. M. Johnson and T. M. Khoshgoftaar, "Medicare fraud detection using neural networks," *Journal of Big Data*, vol. 6, no. 1, pp. 1–35, 2019.

[22] J. T. Hancock and T. M. Khoshgoftaar, "Survey on categorical data for neural networks," *Journal of Big Data*, vol. 7, no. 1, pp. 1–41, 2020.

[23] W. C. Sleeman IV and B. Krawczyk, "Multi-class imbalanced big data classification on spark," *Knowledge-Based Systems*, vol. 212, p. 106 598, 2021.

[24] Apache Software Foundation, *Hadoop*, version 0.20.2, Feb. 19, 2010.

[25] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[26] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[27] J. T. Hancock and T. M. Khoshgoftaar, "Gradient boosted decision tree algorithms for medicare fraud detection," *SN Computer Science*, vol. 2, no. 4, pp. 1–12, 2021.

[28] The Centers for Medicare and Medicaid Services, *Medicare part d prescribers - by provider and drug data dictionary*, 2021.

[29] LEIE. (2022). Office of inspector general leie downloadable databases. [Online]. Available: https://oig.hhs.gov/exclusions/index.asp.

[30] M. Herland, T. M. Khoshgoftaar, and R. A. Bauder, "Big data fraud detection using multiple medicare data sources," *Journal of Big Data*, vol. 5, no. 1, pp. 1–21, 2018.

[31] G. Van Rossum and F. L. Drake, *Python/c api manual-python 3*, 2009.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[33] W. McGinnis. (2022). Category encoders, [Online]. Available: https://contrib.scikit-learn.org/category_encoders/.

[34] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.

[35] J. T. Hancock and T. M. Khoshgoftaar, "Catboost for big data: An interdisciplinary review," *Journal of big data*, vol. 7, no. 1, pp. 1–45, 2020.

[36] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 559–563, 2017.

[37] G. R. Iversen and H. Norpoth, *Analysis of variance*, 1. Newbury Park: Sage, 1987.

[38] J. W. Tukey, "Comparing individual means in the analysis of variance," *Biometrics*, pp. 99–114, 1949.