Journal of Big Data

RESEARCH Open Access

Threshold optimization and random undersampling for imbalanced credit card data

Joffrey L. Leevy[1*], Justin M. Johnson[1], John Hancock[1] and Taghi M. Khoshgoftaar[1]

*Correspondence: jleevy2017@fau.edu

[1] Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431, USA

Abstract

Output thresholding is well-suited for addressing class imbalance, since the technique does not increase dataset size, run the risk of discarding important instances, or modify an existing learner. Through the use of the Credit Card Fraud Detection Dataset, this study proposes a threshold optimization approach that factors in the constraint *True Positive Rate* (TPR) $\geq$ *True Negative Rate* (TNR). Our findings indicate that an increase of the *Area Under the Precision–Recall Curve* (AUPRC) score is associated with an improvement in threshold-based classification scores, while an increase of positive class prior probability causes optimal thresholds to increase. In addition, we discovered that best overall results for the selection of an optimal threshold are obtained without the use of *Random Undersampling* (RUS). Furthermore, with the exception of AUPRC, we established that the default threshold yields good performance scores at a balanced class ratio. Our evaluation of four threshold optimization techniques, eight threshold-dependent metrics, and two threshold-agnostic metrics defines the uniqueness of this research.

Keywords: Output thresholding, Credit Card Fraud Detection Dataset, Random undersampling, Machine learning

Introduction

Class imbalance within a dataset occurs when there is a higher number of instances in one or more classes than in the other class(es). From a binary class perspective, this imbalance means that there is one majority (typically negative) class and one minority (typically positive) class. If the difference between the number of majority and minority class instances is significant, as in the case of high class imbalance, the results of a machine learning study could be skewed. As stated by several researchers, a condition of high class imbalance exists when the minority-to-majority ratio ranges from 1:100 to 1:10,000 [1].

Various techniques are employed to reduce class imbalance or reduce the effect of this imbalance. These techniques can be implemented at the data or algorithm level or both. The most popular and established data-level approaches [1] involve *Random Undersampling* (RUS), *Random Oversampling* (ROS), and *Synthetic Minority Oversampling Technique* (SMOTE). However, these three techniques have well-known disadvantages. RUS, a method for randomly discarding instances from the majority class, may also remove important instances. ROS, a process for duplicating instances of the minority class, runs the risk of overfitting. Developed as an intelligent method for duplication of the minority class, SMOTE generates synthetic instances between existing instances of the minority class. The risk of overfitting is greatly reduced with SMOTE. One disadvantage, which is characteristic of any oversampling technique, is the increase in size of the dataset. Algorithm-level approaches include class-weighting and output thresholding techniques. Class-weighting, which is a direct algorithm-level method, modifies the learner [2]. It is a popular technique that has been integrated into many machine learning algorithms. Output thresholding is a process for tuning the decision threshold that is used to associate class labels with a model's probability estimates [3]. While both algorithm-level approaches reduce bias toward the majority class, output thresholding is a more beneficial technique because it does not modify existing learners and can be performed on any learner that provides probability scores. Therefore, this paper focuses primarily on threshold optimization. In addition, six different levels of RUS are applied to evaluate the interaction between threshold optimization and changing class distributions. We use RUS because studies show that it performs as good as or better than other methods of addressing class imbalance in most cases [4, 5].

In this paper, threshold optimization is used to assign class labels to a model's output probability scores. The optimal or best threshold is one that maximizes the score of a specified performance metric. A valuable tool in our study is the application of the constraint *True Positive Rate* (TPR) ≥ *True Negative Rate* (TNR). This constraint ensures that a threshold will not be selected where the positive class has been ignored by a classifier. For comparative purposes, the default threshold of 0.5 is also investigated to determine whether it is suitable for classifying imbalanced data. Since this is a comprehensive study, we investigate four threshold optimization techniques based on metrics: F-measure, Geometric Mean of TPR and TNR, *Matthews Correlation Coefficient* (MCC), and Precision. In addition, we evaluate eight threshold-dependent metrics: TPR, *False Positive Rate* (FPR), *False Negative Rate* (FNR), TNR, F-measure, Geometric Mean of TPR and TNR, MCC, and Precision. Also under evaluation are two threshold-agnostic metrics, *Area Under the Receiver Operating Characteristic Curve* (AUC) and AUPRC. The learners in this work are XGBoost [6], CatBoost [7], Random Forest [8], Extremely Randomized Trees [9], and Logistic Regression [10].

Our research is centered on the Credit Card Fraud Detection Dataset, which is a set of anonymized transactions available for download from Kaggle [11]. The dataset is based on credit card purchases by Europeans in 2013. There are 284,807 instances and 30 independent variables in the Credit Card Fraud Detection Dataset. Other publicly available datasets used for credit card fraud detection are orders of magnitude less in size. Fraudulent transactions comprise 0.172% of the total number of records, which means that the dataset is highly imbalanced. We use this dataset because it consists of real-world transactions and also because it is on track to become a gold standard for credit card fraud detection.

Our research findings are highlighted as follows:

- As the AUPRC score increases, the threshold-based performance scores also improve.

- As RUS is used to increase the positive class prior probability, the optimal thresholds also increase.
- Best overall results for the selection of an optimal threshold are obtained without the use of RUS.
- For most metrics, the default threshold yields its best results at a balanced (1:1) class ratio.
- However, the combination of the default threshold and balanced class ratio yields the lowest AUPRC scores for all classifiers, implying a significant tradeoff for balancing the classes.
- The default threshold does not yield good results when the dataset is imbalanced.

To the best of our knowledge, this is the first study to investigate threshold optimization using four different techniques based on metrics, while considering the TPR $\geq$ TNR constraint. Moreover, this is the first study to evaluate threshold optimization with eight threshold-dependent and two threshold-agnostic metrics. The remainder of this paper is organized as follows: "Related work" section reviews relevant literature on output thresholding; "Data description" describes the dataset; "Methodology" section covers the methodology, learners, and non-default hyperparameters used; "Results and discussion" section presents and analyzes our findings; and "Conclusion" section summarizes the key points of this paper, as well as providing suggestions for future work.

Related work

The objective of this section is to discuss similar studies that use optimal thresholds for dataset classification. We did not come across any studies on the use of output thresholding with the Credit Card Fraud Detection Dataset.

In relation to one dataset of scenic images and another of health records, Zhang et al. [12] proposed the use of threshold moving techniques to address class imbalance. This involved the adjustment of decision thresholds for binary classification, so that the class distribution of training data could match the predicted outcomes of new data. Using a multi-label version of Random Forest, the authors then performed multi-label classification, where instances may belong to more than one label. Performance-wise, their results indicate that the Random Forest model was just as good or better than more complex multi-label classifiers. Both our work and theirs incorporate the positive class prior probability threshold. However, we go further by comparing the positive class probability threshold with other thresholds.

Buda et al. [13] investigated the effect of output thresholding on *Convolutional Neural Networks* (CNNs) [14], with the aid of three benchmark datasets: MNIST [15], CIFAR-10 [16], and ImageNet [17]. Subsampling was used to render the datasets sufficiently imbalanced. The authors showed that making the threshold equivalent to the positive class prior probability noticeably improved accuracy. Not only do we use the positive class prior probability technique in our work, but we also evaluate four threshold optimization techniques. In addition, the inclusion of eight performance metrics in our study makes it a more comprehensive paper.

With a focus on the network security domain, Calvert and Khoshgoftaar [18] used threshold optimization for establishing alternatives to the AUC metric. They determined that optimal thresholds could be obtained with the Geometric Mean and F-Measure metrics. Using these optimal thresholds, the authors were able to evaluate the performance of various classifiers with different metrics. We note that the authors do not assess the effect of optimized thresholds for Geometric Mean and F-Measure on eight metrics. Another contribution of our work is the inclusion of the TPR $\geq$ TNR constraint.

Finally, Zhou et al. [19] developed a method for finding optimal classification thresholds during experimentation with a protein homology dataset. Their technique involved the comparison of optimal thresholds against "uniform" thresholds of 0.1, 0.2, 0.3 and the default threshold of 0.5. According to their results, the optimal thresholds yielded better scores than the "uniform" thresholds. Our approach is more general, in that we assess several optimal thresholds obtained by different techniques. Moreover, we demonstrate how constraints can be effectively imposed on the threshold optimization process.

We discovered that many studies use RUS to address class imbalance [1]. As stated earlier, there is an inherent risk of discarding important instances when using RUS. This risk is non-existent for output thresholding. In concluding this section, we reaffirm that this is the first paper to include four threshold optimization techniques based on metrics, while taking into account the TPR $\geq$ TNR constraint. It is also the first paper to do so using eight threshold-dependent and two threshold-agnostic metrics.

Data description

The Credit Card Fraud dataset [11] was published by Worldline and the *Université Libre de Bruxelles* (ULB). There are 284,807 instances and 30 independent variables or input features in the raw dataset, which shows credit card purchases by Europeans in September 2013. Using *Principal Component Analysis* (PCA) [20], the dataset publishers transformed 28 of the 30 input features. The remaining two features, "Time" and "Amount" were not transformed. "Time" contains the seconds elapsed between each transaction and the first transaction in the dataset. "Amount", which we normalized, is the transaction amount. "Time" was dropped as this is a uninformative feature for the purpose of the study.

The label (dependent variable) of this binary dataset is 1 for a fraudulent transaction and 0 for a non-fraudulent transaction. Fraudulent transactions constitute 492 instances, or 0.172%, thus making the dataset highly imbalanced with regard to the minority and majority classes.

Methodology

Experiments were run on 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. Our programs for training and testing machine learning models were implemented in the Python programming language. CatBoost and XGBoost are standalone Python libraries. Random Forest, Extremely Randomized Trees, and Logistic Regression are part of the Sci-kit Learn library [21]. These five learners represent different families of machine learning algorithms, thus benefiting the generalization of results.

CatBoost is designed around Ordered Boosting, an algorithm that orders instances used by Decision Trees. XGBoost is based on a weighted quantile sketch and a sparsity-aware function. A weighted quantile sketch uses approximate tree learning [22] for merging and pruning operations of Decision Trees, while the sparsity-aware function is an optimization that efficiently locates the best value to split a dataset on for a Decision Tree node when data is sparse. Random Forest is an ensemble of Decision Trees, and it uses the bagging [23] technique. The Extremely Randomized Trees learner, which also relies on the bagging technique, is an extension of Random Forest. However, for Random Forest, the optimal values for splits in the Decision Tree are usually calculated systematically, whereas these optimal values are selected randomly for Extremely Randomized Trees. Logistic Regression produces a value corresponding to the probability of belonging to a particular class. It is a linear model that relies on a sigmoid function to output a number between 0 and 1.

For every experiment, we select a classifier and a class ratio that we wish to apply to the training data. We use the Imblearn [24] 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. Hence, we perform experiments with the six class ratios of 1:1 (balanced class ratio), 1:3, 1:9, 1:27, 1:81, and 1:578 (original class ratio of the Credit Card Fraud Detection dataset). Selection of the first five ratios is based on preliminary experimentation through RUS.

For each experiment, training is subsequently performed on 80% of the data using *k*-fold cross-validation, where the model is trained on *k*-1 folds each time and tested on the remaining fold. This ensures that as much data as possible is used during the classification phase. Our cross-validation process is stratified, which seeks to ensure that each class is proportionally represented across the folds. In this experiment, a value of five was assigned to *k*, where four folds were used in training and one fold was used in testing. We perform 10 iterations of cross-validation as a precaution against data loss due to random sampling of instances from the majority class.

The threshold optimization technique, as outlined in Algorithm 1, is implemented on the training data. For the remaining 20% of the data, classifier output probabilities are calculated. Instances of the test data are then assigned to classes based on the computed thresholds. After the model is fit to the training partitions, Algorithm 1 is used to identify the optimal decision threshold that maximizes some user-defined performance metric. Given the training data's probability estimates and ground truth labels, we enumerate all possible thresholds and then select the threshold that maximizes the desired performance metric. This design is flexible, as it allows for the user to optimize against any performance metric that is suitable for their problem. It also allows for additional constraints, e.g., we desire that the TPR be greater than the TNR in many classification problems. Finally, we apply the optimal decision threshold that has been learned from the training partitions to the test partition and record the test performance.

```
Input
    $y$    ground truth labels
    $\hat{y}$    classifier output probabilities
    $f$    optimization function, e.g. geometric mean
    $c$    flag controlling whether optimization is constrained
Output
    $\lambda$    Optimized threshold
tprs, fprs, thresholds ← roc_curve($y$, $\hat{y}$)
tnrs ← 1 - fprs
scores ← $f(y, \hat{y})$
if $c$ then
    mask ← tprs ≥ tnrs
    scores, thresholds ← scores[mask], thresholds[mask]
max_score_idx = $\arg\max$ (scores)
return thresholds[max_score_idx]
```

Algorithm 1: Pseudocode for threshold optimization; roc_curve is assumed to be a library function capable of returning all true positive rates, and false positive rates for all thresholds in the interval from zero to one

We point out that our threshold optimization algorithm takes a function as an argument. This function parameter may be one of the four classification metrics: Geometric Mean of TPR and TNR, MCC, Precision, or F-measure. Furthermore, our threshold optimization algorithm has a flag that controls whether optimization is constrained. If the flag is set to true, the threshold, where TPR ≥ TNR, is chosen for the best value of a specific metric. Optimized thresholds are computed for all combinations of the constraint flag and classification metrics.

For each of the optimized thresholds, as well as the default threshold of 0.5 and the positive class prior probability threshold, scores are calculated for the following metrics: TPR, FPR, FNR, TNR, F-measure, Geometric Mean of TPR and TNR, MCC, and Precision. For Logistic Regression, the hyperparameter values were not changed. To prevent overfitting of the Decision Tree-based classifiers, the maximum tree depths shown in Table 1 are used. These depths were obtained from preliminary experimentation. Overall classification performance for each model at each ratio was evaluated with the threshold-agnostic AUC and AUPRC metrics.

Results and discussion

To start off, we define the words and abbreviations used in the tables. "Classifier" is the type of learning algorithm used for classification. AUC is the Area under the Receiver Operating Characteristic Curve. AUPRC is the Area under the Precision–Recall Curve. "Technique" is the thresholding technique used. An example is the selection of the output probability threshold that optimizes F-measure. "Threshold" is the value of the threshold found for a particular thresholding technique. TPR stands for True Positive Rate. FPR stands for False Positive Rate. FNR stands for False Negative Rate. F-meas

Table 1 Maximum tree depths used in experiments

Classifier	Maximum tree depth
XGBoost	`max_depth=1` for all class ratios
CatBoost	`max_depth=1` for 1:1, 1:3, 1:9, `max_depth=5` for 1:27, 1:81
Random forest	`max_depth=4` for all class ratios
Extremely randomized trees	`max_depth=8` for all class ratios

Table 2 Mean AUC and AUPRC scores for 10 iterations of fivefold cross validation

Classifier	AUC	AUPRC
CatBoost depth 5	0.9834	0.8592
Extremely randomized trees depth 8	0.9721	0.8092
Logistic regression	0.9737	0.7586
Random forest depth 4	0.9601	0.8067
XGBoost depth 1	0.9775	0.8261

Table 3 Results for the CatBoost depth 5 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0018	0.9039	0.0150	0.0961	0.9850	0.1764	0.9434	0.2935	0.0982
F-meas NC	0.3541	0.8053	0.0001	0.1947	0.9999	0.8687	0.8970	0.8717	0.9449
G-mean	0.0015	0.9063	0.0180	0.0937	0.9820	0.1514	0.9432	0.2698	0.0829
G-mean NC	0.0023	0.8970	0.0108	0.1030	0.9892	0.2395	0.9418	0.3476	0.1405
MCC	0.0015	0.9063	0.0180	0.0937	0.9820	0.1510	0.9432	0.2695	0.0826
MCC NC	0.0023	0.8970	0.0108	0.1030	0.9892	0.2395	0.9418	0.3476	0.1405
Precision	0.0018	0.9039	0.0150	0.0961	0.9850	0.1764	0.9434	0.2935	0.0982
Precision NC	0.4869	0.7953	0.0001	0.2047	0.9999	0 .8676	0.8915	0.8714	0.9560
C	0.0017	0.9037	0.0147	0.0963	0.9853	0.1755	0.9435	0.2934	0.0973
D	0.5000	0.7937	0.0001	0.2063	0.9999	0.8665	0.8905	0.8704	0.9559

stands for the F-measure score. G-mean stands for the score of the Geometric Mean of TPR and TNR. MCC stands for Matthews Correlation Coefficient.

Under the "Technique" column, there are several abbreviations for the thresholding techniques. F-meas, G-mean and MCC have already been defined. NC stands for no constraint. In other words, the constraint has not been applied. The constraint is that the True Positive Rate is greater than the True Negative Rate. Therefore, "NC" after any thresholding technique means that the constraint is not used. C stands for class prior; i.e., the threshold value chosen is the fraction of positive instances in the dataset. Since the class prior threshold is determined, and not calculated via optimization code, the constraint that True Positive Rate be greater than True Negative Rate cannot be applied. D stands for the default threshold of 0.5. The constraint cannot be applied to the default threshold as well.

Several tables of results were generated for the various class ratios for each classifier. For ease of understanding, in this section we show the results of CatBoost, which is the top performing classifier overall. Results obtained with the remaining classifiers are shown in "Appendices".

Classification results for the original class ratio (no RUS applied)

In Table 2, CatBoost is the best performer with regard to AUC and AUPRC, obtaining scores of 0.9834 and 0.8592, respectively. For Table 3, use of the default threshold yields comparatively low TPR scores and comparatively high FNR scores. Furthermore, the threshold values obtained using constrained optimal thresholds are lower than their non-constrained counterparts.

Classification results for the 1:1 class ratio

In Table 4, Extremely Randomized Trees is the best performer in reference to AUC and AUPRC, obtaining scores of 0.9803 and 0.7379, respectively. For Table 5, use of the default threshold yields comparatively low TPR scores and comparatively high FNR scores. In addition, the threshold values obtained using constrained optimal thresholds are lower than their non-constrained counterparts.

Classification results for the 1:3 class ratio

In Table 6, CatBoost generated the highest score for AUC (0.9790), while the top score for AUPRC (0.7481) was obtained with XGBoost. For Table 7, use of the default threshold yields comparatively low TPR scores and comparatively high FNR scores. Also, the threshold values obtained using constrained optimal thresholds are lower than their non-constrained counterparts.

Table 4 Mean AUC and AUPRC scores for 10 iterations of fivefold cross validation

Classifier	AUC	AUPRC
Random forest depth 4	0.9771	0.7347
Extremely randomized trees depth 8	0.9803	0.7379
Logistic regression	0.9770	0.6030
XGBoost depth 1	0.9790	0.7121
CatBoost depth 1	0.9785	0.5928

Table 5 Results for the CatBoost depth 1 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.3100	0.9281	0.0690	0.0719	0.9310	0.0458	0.9293	0.1412	0.0235
F-meas NC	0.3760	0.9207	0.0552	0.0793	0.9448	0.0609	0.9325	0.1623	0.0317
G-mean	0.2992	0.9301	0.0717	0.0699	0.9283	0.0440	0.9291	0.1383	0.0225
G-mean NC	0.3783	0.9201	0.0544	0.0799	0.9456	0.0612	0.9326	0.1629	0.0319
MCC	0.2921	0.9313	0.0741	0.0687	0.9259	0.0427	0.9285	0.1361	0.0219
MCC NC	0.3916	0.9193	0.0524	0.0807	0.9476	0.0641	0.9331	0.1669	0.0334
Precision	0.3311	0.9250	0.0627	0.0750	0.9373	0.0494	0.9310	0.1473	0.0254
Precision NC	0.8357	0.8594	0.0095	0.1406	0.9905	0.3031	0.9219	0.3922	0.2045
C	0.5000	0.9099	0.0339	0.0901	0.9661	0.0873	0.9375	0.1993	0.0459
D	0.5000	0.9099	0.0339	0.0901	0.9661	0.0873	0.9375	0.1993	0.0459

Table 6 Mean AUC and AUPRC scores for 10 iterations of fivefold cross validation

Classifier	AUC	AUPRC
Random forest depth 4	0.9742	0.7313
Extremely randomized trees depth 8	0.9786	0.7387
Logistic regression	0.9790	0.7090
XGBoost depth 1	0.9786	0.7481
CatBoost depth 1	0.9790	0.7374

Table 7 Results for the CatBoost depth 1 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.1507	0.9217	0.0518	0.0783	0.9482	0.0584	0.9347	0.1614	0.0302
F-meas NC	0.3901	0.8907	0.0166	0.1093	0.9834	0.1730	0.9356	0.2861	0.0975
G-mean	0.1324	0.9260	0.0606	0.0740	0.9394	0.0508	0.9325	0.1497	0.0261
G-mean NC	0.2208	0.9094	0.0347	0.0906	0.9653	0.0890	0.9367	0.2003	0.0470
MCC	0.1317	0.9260	0.0610	0.0740	0.9390	0.0505	0.9323	0.1492	0.0260
MCC NC	0.2350	0.9073	0.0326	0.0927	0.9674	0.0960	0.9367	0.2081	0.0510
Precision	0.1509	0.9217	0.0517	0.0783	0.9483	0.0585	0.9348	0.1616	0.0302
Precision NC	0.8467	0.7722	0.0025	0.2278	0.9975	0.5380	0.8716	0.5810	0.4962
C	0.2500	0.9057	0.0276	0.0943	0.9724	0.1029	0.9383	0.2181	0.0546
D	0.5000	0.8836	0.0103	0.1164	0.9897	0.2305	0.9349	0.3393	0.1332

Classification results for the 1:9 class ratio

In Table 8, XGBoost produced the top scores for both AUC (0.9801) and AUPRC (0.7804). For Table 9, use of the default threshold yields comparatively low TPR scores and comparatively high FNR scores. In addition, the threshold values obtained using constrained optimal thresholds are lower than their non-constrained counterparts.

Classification results for the 1:27 class ratio

In Table 10, CatBoost is the top performer for both AUC and AUPRC, registering scores of 0.9817 and 0.7963, respectively. For Table 11, use of the default threshold yields comparatively low TPR scores and comparatively high FNR scores. Also, the

Table 8 Mean AUC and AUPRC scores for 10 iterations of fivefold cross validation

Classifier	AUC	AUPRC
Random forest depth 4	0.9719	0.7418
Extremely randomized trees depth 8	0.9779	0.7523
Logistic regression	0.9786	0.7298
XGBoost depth 1	0.9801	0.7804
CatBoost depth 1	0.9789	0.7680

Table 9 Results for the CatBoost depth 1 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0535	0.9183	0.0529	0.0817	0.9471	0.0571	0.9324	0.1591	0.0295
F-meas NC	0.4056	0.8667	0.0042	0.1333	0.9958	0.4201	0.9288	0.4886	0.2820
G-mean	0.0477	0.9217	0.0600	0.0783	0.9400	0.0510	0.9307	0.1497	0.0262
G-mean NC	0.1089	0.8982	0.0252	0.1018	0.9748	0.1206	0.9355	0.2349	0.0651
MCC	0.0475	0.9217	0.0603	0.0783	0.9397	0.0507	0.9306	0.1493	0.0261
MCC NC	0.1250	0.8947	0.0215	0.1053	0.9785	0.1408	0.9355	0.2552	0.0772
Precision	0.0535	0.9183	0.0529	0.0817	0.9471	0.0571	0.9324	0.1591	0.0295
Precision NC	0.8815	0.6851	0.0005	0.3149	0.9995	0.6875	0.8208	0.7039	0.7591
C	0.1000	0.8992	0.0249	0.1008	0.9751	0.1115	0.9362	0.2273	0.0595
D	0.5000	0.8599	0.0027	0.1401	0.9973	0.5053	0.9259	0.5539	0.3597

Table 10 Mean AUC and AUPRC scores for 10 iterations of fivefold cross validation

Classifier	AUC	AUPRC
CatBoost depth 5	0.9817	0.7963
Random forest depth 4	0.9699	0.7483
Extremely randomized trees depth 8	0.9754	0.7693
Logistic regression	0.9785	0.7504
XGBoost depth 1	0.9802	0.7885

Table 11 Results for the CatBoost depth 5 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0681	0.8819	0.0053	0.1181	0.9947	0.3856	0.9365	0.4641	0.2533
F-meas NC	0.1711	0.8685	0.0018	0.1315	0.9982	0.5979	0.9310	0.6289	0.4584
G-mean	0.0333	0.8977	0.0132	0.1023	0.9868	0.2238	0.9410	0.3316	0.1323
G-mean NC	0.0734	0.8801	0.0048	0.1199	0.9952	0.4054	0.9358	0.4795	0.2699
MCC	0.0333	0.8977	0.0132	0.1023	0.9868	0.2238	0.9410	0.3316	0.1323
MCC NC	0.0734	0.8801	0.0048	0.1199	0.9952	0.4054	0.9358	0.4795	0.2699
Precision	0.0681	0.8819	0.0053	0.1181	0.9947	0.3856	0.9365	0.4641	0.2533
Precision NC	0.4621	0.8480	0.0009	0.1520	0.9991	0.7224	0.9202	0.7336	0.6446
C	0.0357	0.8923	0.0091	0.1077	0.9909	0.2517	0.9402	0.3591	0.1469
D	0.5000	0.8526	0.0007	0.1474	0.9993	0.7560	0.9229	0.7610	0.6812

Table 12 Mean AUC and AUPRC scores for 10 iterations of fivefold cross validation

Classifier	AUC	AUPRC
CatBoost depth 5	0.9832	0.8490
Random forest depth 4	0.9657	0.7852
Extremely randomized trees depth 8	0.9747	0.7915
Logistic regression	0.9782	0.7580
XGBoost depth 1	0.9801	0.8057

threshold values obtained using constrained optimal thresholds are lower than their non-constrained counterparts

Classification results for the 1:81 class ratio

In Table 12, CatBoost is the top performer for both AUC (0.9832) and AUPRC (0.8490). For Table 13, use of the default threshold yields comparatively low TPR scores and comparatively high FNR scores. Also, the threshold values obtained using constrained optimal thresholds are lower than their non-constrained counterparts.

Overall analysis of results

A thorough investigation of results should also involve a collective analysis of the various experiments performed. As such, interesting observations about the overall study have been discussed in the paragraphs below.

Table 13 Results for the CatBoost depth 5 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0158	0.8923	0.0081	0.1077	0.9919	0.2831	0.9406	0.3844	0.1700
F-meas NC	0.1760	0.8553	0.0008	0.1447	0.9992	0.7489	0.9243	0.7555	0.6711
G-mean	0.0120	0.8988	0.0122	0.1012	0.9878	0.2133	0.9421	0.3256	0.1223
G-mean NC	0.0201	0.8872	0.0061	0.1128	0.9939	0.3458	0.9389	0.4338	0.2182
MCC	0.0119	0.8990	0.0123	0.1010	0.9877	0.2123	0.9422	0.3248	0.1216
MCC NC	0.0201	0.8872	0.0061	0.1128	0.9939	0.3458	0.9389	0.4338	0.2182
Precision	0.0158	0.8923	0.0081	0.1077	0.9919	0.2831	0.9406	0.3844	0.1700
Precision NC	0.4323	0.8347	0.0004	0.1653	0.9996	0.8107	0.9133	0.8115	0.7912
C	0.0122	0.8977	0.0110	0.1023	0.9890	0.2200	0.9421	0.3328	0.1256
D	0.5000	0.8333	0.0003	0.1667	0.9997	0.8268	0.9126	0.8269	0.8218

As RUS is used to increase the positive class prior probability, the AUC metric remains practically unchanged. For the original class ratio, the highest score in Table 2 is 0.9834 (CatBoost). At the balanced class ratio, the highest score in Table 4 is 0.9803 (Extremely Randomized Trees). In terms of AUPRC, the increase of positive class prior probability negatively affects the scores. At the original class ratio, the highest score in Table 2 is 0.8592 (CatBoost), which is also the highest value of AUPRC for the entire study. At the balanced class ratio, the highest score in Table 4 is 0.7379 (Extremely Randomized Trees). AUPRC is impacted due to the sensitivity of this metric to the percentage of positive class instances. When RUS is used to balance training data, the models become less and less biased toward the majority class. Instead, they begin to favor the minority class more, causing them to over-predict the minority class and obtain many false positives. These large numbers of false positives are detrimental to Precision and AUPRC.

Increasing model bias toward the minority class may result in a higher TPR score on the test set. This action can significantly lower TNR scores, which in turn negatively impacts Precision and F-Measure. In general, the increase in bias results in lower G-Mean, F-Measure, MCC, and Precision scores (all metrics that take both classes into consideration). This is because overall, the models trained with RUS are worse at discriminating between the classes, as observed with the AUPRC results. Some may argue that RUS should be used to obtain higher TPR scores. Based on our results, however, we counterargue that instead of applying RUS to raise TPR scores, the threshold optimization process should be used. This optimization would yield better results, because a model trained without RUS is better at discriminating between classes.

Since the AUPRC indicates how good a model is at separating classes, an increase in the AUPRC score means that, in theory, there will be an improvement in threshold-based performance. Hence, the use of AUPRC and optimal thresholding are two techniques that complement each other. We point out that the AUPRC does not identify a specific threshold for selection. Rather, the AUPRC is helpful because it allows for the selection of the best model, model hyperparameters, or sampling rates. Within this framework, optimal thresholding should be implemented after the AUPRC is plotted in order to select the correct operating point on the curve. It is intuitively apparent that the best AUPRC values occur when RUS is not applied (at the original class ratio). Therefore, in the absence of RUS, the practice of optimal thresholding can be used to identify the ideal operating point on an AUPRC curve. This does not mean that the use of RUS should be avoided at all costs, as there is always a run-time tradeoff to consider. For example, undersampling to a 1:81 class ratio may yield statistically similar results, which translates to the saving of training resources in the case of big data.

An increase in positive class prior probability with RUS generally increases the optimal thresholds. This phenomenon can be observed using results obtained with CatBoost (and also results shown in "Appendices" for the remaining classifiers). For the original class ratio, Table 3 shows the constrained optimal threshold for Geometric Mean as 0.0015, while for the balanced class ratio in Table 5, the related value is 0.2992. This phenomenon is due to the shift of the optimal thresholds toward the positive class ratio.

Finally, it should be noted that the best results for the default threshold are obtained at the balanced class ratio. This observation can be shown using results obtained with CatBoost (and also results shown in "Appendices" for the remaining classifiers). For the original class ratio, Table 3 associates the default threshold with a Geometric Mean score of 0.8905. For the balanced class ratio, the related value in Table 5 is 0.9375. This observation can be attributed to the shifting of probability scores closer to the value of 0.5 when RUS is used to obtain the 1:1 class ratio. We also point out that there is a comparatively significant difference between the Geometric Mean score of 0.9375 at the balanced class ratio and 0.8905 at the original class ratio. For the 1:81, 1:27, 1:9, and 1:3 ratios, the related Geometric Mean scores for Logistic Regression at the default threshold are 0.9126, 0.9229, 0.9259, and 0.9349, respectively. Hence, it should also be noted that the default threshold does not yield strong results when the dataset is imbalanced.

Conclusion

In our research, we use the Credit Card Fraud Detection Dataset to investigate output thresholding. A useful aid in this work is the application of the constraint $TPR \geq TNR$, which ensures that the positive class is never ignored during the selection of an optimal threshold. We evaluate four threshold optimization techniques, eight threshold-dependent metrics, and two threshold-agnostic metrics.

Our primary observations made in this research suggest that: an increase of the AUPRC score is associated with an improvement of threshold-based performance scores; increasing the positive class prior probability will increase optimal thresholds; best overall results for an optimal threshold are obtained without the need for RUS; determining whether to use the default threshold with a 1:1 class ratio obtained via RUS requires a proper consideration of the tradeoff involved; the default threshold yields poor results when the dataset is imbalanced. Future work will use our threshold optimization approach with datasets from other application domains. Also, the use of additional constraints will be evaluated.

Appendices

Appendix 1: Results for original class ratio

See Tables 14, 15, 16 and 17.

Table 14 Results for the extremely randomized trees depth 8 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0012	0.9087	0.0678	0.0913	0.9322	0.0446	0.9202	0.1380	0.0229
F-meas NC	0.1975	0.7735	0.0002	0.2265	0.9998	0.8143	0.8790	0.8159	0.8628
G-mean	0.0012	0.9114	0.0734	0.0886	0.9266	0.0414	0.9188	0.1326	0.0212
G-mean NC	0.0026	0.8807	0.0262	0.1193	0.9738	0.1164	0.9258	0.2274	0.0631
MCC	0.0011	0.9114	0.0737	0.0886	0.9263	0.0413	0.9186	0.1323	0.0211
MCC NC	0.0037	0.8725	0.0198	0.1275	0.9802	0.1639	0.9245	0.2715	0.0942
Precision	0.0012	0.9087	0.0678	0.0913	0.9322	0.0446	0.9202	0.1380	0.0229
Precision NC	0.6220	0.4836	0.0000	0.5164	1.0000	0.6437	0.6939	0.6844	0.9740
C	0.0017	0.8937	0.0390	0.1063	0.9610	0.0739	0.9266	0.1809	0.0386
D	0.5000	0.5652	0.0001	0.4348	0.9999	0.7047	0.7513	0.7275	0.9386

Table 15 Results for the logistic regression classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0012	0.9193	0.0699	0.0807	0.9301	0.0438	0.9245	0.1375	0.0225
F-meas NC	0.0921	0.7746	0.0004	0.2254	0.9996	0.7778	0.8797	0.7778	0.7827
G-mean	0.0011	0.9208	0.0759	0.0792	0.9241	0.0404	0.9223	0.1315	0.0206
G-mean NC	0.0025	0.8915	0.0239	0.1085	0.9761	0.1199	0.9327	0.2343	0.0646
MCC	0.0011	0.9208	0.0760	0.0792	0.9240	0.0403	0.9223	0.1314	0.0206
MCC NC	0.0029	0.8846	0.0196	0.1154	0.9804	0.1433	0.9311	0.2575	0.0787
Precision	0.0012	0.9193	0.0699	0.0807	0.9301	0.0438	0.9245	0.1375	0.0225
Precision NC	1.0000	0.1833	0.0000	0.8167	1.0000	0.3087	0.4267	0.4261	0.9988
C	0.0017	0.9043	0.0398	0.0957	0.9602	0.0729	0.9317	0.1807	0.0380
D	0.5000	0.6178	0.0002	0.3822	0.9998	0.7219	0.7853	0.7329	0.8721

Table 16 Results for the random forest depth 4 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0005	0.8929	0.0752	0.1071	0.9248	0.0400	0.9085	0.1285	0.0205
F-meas NC	0.3299	0.7547	0.0002	0.2453	0.9998	0.7997	0.8682	0.8019	0.8550
G-mean	0.0005	0.8996	0.0836	0.1004	0.9164	0.0362	0.9078	0.1220	0.0185
G-mean NC	0.0014	0.8612	0.0247	0.1388	0.9753	0.1632	0.9161	0.2644	0.0951
MCC	0.0005	0.8996	0.0836	0.1004	0.9164	0.0362	0.9078	0.1220	0.0185
MCC NC	0.0033	0.8524	0.0123	0.1476	0.9877	0.2766	0.9173	0.3654	0.1751
Precision	0.0005	0.8929	0.0752	0.1071	0.9248	0.0400	0.9085	0.1285	0.0205
Precision NC	0.8217	0.4545	0.0000	0.5455	1.0000	0.6190	0.6729	0.6659	0.9805
C	0.0017	0.8498	0.0089	0.1502	0.9911	0.2459	0.9176	0.3471	0.1440
D	0.5000	0.6898	0.0001	0.3102	0.9999	0.7832	0.8302	0.7906	0.9074

Table 17 Results for the XGBoost depth 1 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0010	0.9161	0.0476	0.0839	0.9524	0.0628	0.9339	0.1675	0.0325
F-meas NC	0.4605	0.7739	0.0002	0.2261	0.9998	0.8286	0.8792	0.8315	0.8961
G-mean	0.0009	0.9189	0.0527	0.0811	0.9473	0.0573	0.9329	0.1595	0.0296
G-mean NC	0.0015	0.9045	0.0315	0.0955	0.9685	0.0996	0.9357	0.2117	0.0531
MCC	0.0009	0.9189	0.0530	0.0811	0.9470	0.0571	0.9327	0.1592	0.0295
MCC NC	0.0016	0.9031	0.0300	0.0969	0.9700	0.1071	0.9357	0.2194	0.0576
Precision	0.0010	0.9161	0.0476	0.0839	0.9524	0.0628	0.9339	0.1675	0.0325
Precision NC	0.9418	0.3137	0.0000	0.6863	1.0000	0.4698	0.5552	0.5500	0.9830
C	0.0017	0.8990	0.0229	0.1010	0.9771	0.1196	0.9371	0.2364	0.0641
D	0.5000	0.7700	0.0002	0.2300	0.9998	0.8238	0.8771	0.8260	0.8877

Appendix 2: Results for 1:1 class ratio

See Tables 18, 19, 20 and 21.

Table 18 Results for the extremely randomized trees depth 8 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.2232	0.9423	0.1047	0.0577	0.8953	0.0308	0.9183	0.1139	0.0157
F-meas NC	0.2306	0.9384	0.0969	0.0616	0.9031	0.0339	0.9204	0.1196	0.0173
G-mean	0.2165	0.9459	0.1137	0.0541	0.8863	0.0286	0.9155	0.1092	0.0145
G-mean NC	0.2319	0.9380	0.0951	0.0620	0.9049	0.0344	0.9211	0.1205	0.0175
MCC	0.2148	0.9465	0.1165	0.0535	0.8835	0.0280	0.9143	0.1079	0.0142
MCC NC	0.2372	0.9364	0.0905	0.0636	0.9095	0.0368	0.9226	0.1247	0.0188
Precision	0.2275	0.9411	0.0987	0.0589	0.9013	0.0324	0.9208	0.1172	0.0165
Precision NC	0.3903	0.8921	0.0218	0.1079	0.9782	0.1546	0.9339	0.2653	0.0876
C	0.5000	0.8665	0.0073	0.1335	0.9927	0.2946	0.9273	0.3894	0.1794
D	0.5000	0.8665	0.0073	0.1335	0.9927	0.2946	0.9273	0.3894	0.1794

Table 19 Results for the logistic regression classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.3001	0.9321	0.0700	0.0679	0.9300	0.0452	0.9310	0.1405	0.0232
F-meas NC	0.4586	0.9189	0.0426	0.0811	0.9574	0.0774	0.9378	0.1857	0.0406
G-mean	0.2742	0.9356	0.0773	0.0644	0.9227	0.0410	0.9290	0.1334	0.0209
G-mean NC	0.4675	0.9185	0.0411	0.0815	0.9589	0.0792	0.9384	0.1883	0.0416
MCC	0.2685	0.9366	0.0792	0.0634	0.9208	0.0400	0.9286	0.1318	0.0205
MCC NC	0.5330	0.9116	0.0342	0.0884	0.9658	0.0944	0.9381	0.2062	0.0501
Precision	0.3101	0.9303	0.0667	0.0697	0.9333	0.0467	0.9317	0.1432	0.0240
Precision NC	0.8922	0.8272	0.0091	0.1728	0.9909	0.2769	0.8990	0.3673	0.1878
C	0.5000	0.9155	0.0350	0.0845	0.9650	0.0844	0.9398	0.1966	0.0443
D	0.5000	0.9155	0.0350	0.0845	0.9650	0.0844	0.9398	0.1966	0.0443

Table 20 Results for the random forest depth 4 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.2546	0.9352	0.0845	0.0648	0.9155	0.0377	0.9252	0.1273	0.0193
F-meas NC	0.2900	0.9256	0.0629	0.0744	0.9371	0.0519	0.9312	0.1502	0.0268
G-mean	0.2368	0.9384	0.0994	0.0616	0.9006	0.0322	0.9192	0.1167	0.0164
G-mean NC	0.2908	0.9252	0.0622	0.0748	0.9378	0.0522	0.9313	0.1508	0.0269
MCC	0.2352	0.9390	0.1008	0.0610	0.8992	0.0318	0.9188	0.1158	0.0162
MCC NC	0.3023	0.9224	0.0557	0.0776	0.9443	0.0578	0.9331	0.1592	0.0299
Precision	0.2570	0.9344	0.0824	0.0656	0.9176	0.0386	0.9258	0.1288	0.0197
Precision NC	0.5203	0.8756	0.0121	0.1244	0.9879	0.2453	0.9299	0.3463	0.1545
C	0.5000	0.8801	0.0122	0.1199	0.9878	0.2044	0.9323	0.3155	0.1164
D	0.5000	0.8801	0.0122	0.1199	0.9878	0.2044	0.9323	0.3155	0.1164

Table 21 Results for the XGBoost depth 1 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.3385	0.9270	0.0623	0.0730	0.9377	0.0500	0.9322	0.1483	0.0257
F-meas NC	0.3647	0.9248	0.0570	0.0752	0.9430	0.0556	0.9337	0.1565	0.0287
G-mean	0.3201	0.9282	0.0666	0.0718	0.9334	0.0470	0.9307	0.1434	0.0241
G-mean NC	0.3670	0.9240	0.0565	0.0760	0.9435	0.0560	0.9335	0.1570	0.0289
MCC	0.3127	0.9291	0.0685	0.0709	0.9315	0.0458	0.9302	0.1414	0.0235
MCC NC	0.3810	0.9229	0.0540	0.0771	0.9460	0.0588	0.9343	0.1611	0.0304
Precision	0.3526	0.9248	0.0589	0.0752	0.9411	0.0525	0.9328	0.1522	0.0270
Precision NC	0.6989	0.8848	0.0176	0.1152	0.9824	0.1880	0.9318	0.2967	0.1215
C	0.5000	0.9120	0.0344	0.0880	0.9656	0.0854	0.9383	0.1975	0.0448
D	0.5000	0.9120	0.0344	0.0880	0.9656	0.0854	0.9383	0.1975	0.0448

Appendix 3: Results for 1:3 class ratio

See Tables 22, 23, 24 and 25.

Table 22 Results for the extremely randomized trees depth 8 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.1126	0.9317	0.0788	0.0683	0.9212	0.0399	0.9263	0.1312	0.0204
F-meas NC	0.1663	0.9035	0.0339	0.0965	0.9661	0.0926	0.9341	0.2033	0.0492
G-mean	0.1066	0.9352	0.0898	0.0648	0.9102	0.0352	0.9224	0.1225	0.0179
G-mean NC	0.1289	0.9214	0.0589	0.0786	0.9411	0.0534	0.9310	0.1530	0.0276
MCC	0.1062	0.9356	0.0907	0.0644	0.9093	0.0349	0.9222	0.1220	0.0178
MCC NC	0.1317	0.9201	0.0563	0.0799	0.9437	0.0563	0.9317	0.1570	0.0291
Precision	0.1129	0.9313	0.0784	0.0687	0.9216	0.0400	0.9263	0.1315	0.0205
Precision NC	0.3501	0.8591	0.0068	0.1409	0.9932	0.3814	0.9235	0.4580	0.2838
C	0.2500	0.8789	0.0129	0.1211	0.9871	0.1913	0.9313	0.3042	0.1076
D	0.5000	0.8415	0.0009	0.1585	0.9991	0.7138	0.9167	0.7222	0.6223

Table 23 Results for the logistic regression classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.1336	0.9307	0.0588	0.0693	0.9412	0.0524	0.9358	0.1528	0.0270
F-meas NC	0.4471	0.8918	0.0134	0.1082	0.9866	0.2030	0.9379	0.3143	0.1175
G-mean	0.1215	0.9337	0.0659	0.0663	0.9341	0.0475	0.9338	0.1447	0.0244
G-mean NC	0.2451	0.9130	0.0299	0.0870	0.9701	0.1027	0.9410	0.2172	0.0547
MCC	0.1210	0.9340	0.0663	0.0660	0.9337	0.0473	0.9337	0.1444	0.0243
MCC NC	0.2696	0.9087	0.0271	0.0913	0.9729	0.1141	0.9401	0.2292	0.0613
Precision	0.1337	0.9307	0.0588	0.0693	0.9412	0.0525	0.9358	0.1528	0.0270
Precision NC	0.9082	0.7439	0.0023	0.2561	0.9977	0.5093	0.8474	0.5555	0.4803
C	0.2500	0.9120	0.0272	0.0880	0.9728	0.1050	0.9418	0.2214	0.0558
D	0.5000	0.8880	0.0110	0.1120	0.9890	0.2218	0.9370	0.3324	0.1274

Table 24 Results for the random forest depth 4 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.1161	0.9254	0.0728	0.0746	0.9272	0.0427	0.9261	0.1358	0.0219
F-meas NC	0.2347	0.8850	0.0183	0.1150	0.9817	0.1550	0.9319	0.2688	0.0857
G-mean	0.1073	0.9305	0.0853	0.0695	0.9147	0.0371	0.9224	0.1258	0.0189
G-mean NC	0.1551	0.9069	0.0421	0.0931	0.9579	0.0755	0.9318	0.1822	0.0396
MCC	0.1072	0.9305	0.0854	0.0695	0.9146	0.0370	0.9223	0.1256	0.0189
MCC NC	0.1607	0.9045	0.0393	0.0955	0.9607	0.0809	0.9319	0.1889	0.0426
Precision	0.1162	0.9254	0.0727	0.0746	0.9273	0.0427	0.9262	0.1359	0.0219
Precision NC	0.4851	0.8484	0.0038	0.1516	0.9962	0.5213	0.9190	0.5716	0.4317
C	0.2500	0.8811	0.0149	0.1189	0.9851	0.1724	0.9315	0.2867	0.0958
D	0.5000	0.8490	0.0015	0.1510	0.9985	0.6337	0.9206	0.6554	0.5103

Table 25 Results for the XGBoost depth 1 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.1735	0.9173	0.0446	0.0827	0.9554	0.0672	0.9360	0.1738	0.0349
F-meas NC	0.3210	0.8994	0.0213	0.1006	0.9787	0.1358	0.9381	0.2519	0.0739
G-mean	0.1590	0.9193	0.0496	0.0807	0.9504	0.0614	0.9346	0.1654	0.0318
G-mean NC	0.1992	0.9122	0.0387	0.0878	0.9613	0.0794	0.9363	0.1889	0.0416
MCC	0.1572	0.9195	0.0503	0.0805	0.9497	0.0606	0.9344	0.1643	0.0314
MCC NC	0.2054	0.9106	0.0375	0.0894	0.9625	0.0824	0.9360	0.1923	0.0433
Precision	0.1752	0.9169	0.0441	0.0831	0.9559	0.0679	0.9361	0.1747	0.0353
Precision NC	0.7930	0.8209	0.0035	0.1791	0.9965	0.4734	0.9024	0.5310	0.3857
C	0.2500	0.9061	0.0280	0.0939	0.9720	0.1012	0.9383	0.2163	0.0536
D	0.5000	0.8846	0.0105	0.1154	0.9895	0.2279	0.9355	0.3373	0.1313

Appendix 4: Results for 1:9 class ratio

see Tables 26, 27, 28 and 29.

Table 26 Results for the Extremely randomized trees depth 8 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0507	0.9213	0.0651	0.0787	0.9349	0.0470	0.9280	0.1431	0.0241
F-meas NC	0.1665	0.8628	0.0066	0.1372	0.9934	0.3168	0.9257	0.4066	0.1964
G-mean	0.0478	0.9254	0.0736	0.0746	0.9264	0.0420	0.9258	0.1347	0.0215
G-mean NC	0.0658	0.9063	0.0407	0.0937	0.9593	0.0764	0.9323	0.1840	0.0400
MCC	0.0478	0.9254	0.0737	0.0746	0.9263	0.0419	0.9257	0.1346	0.0215
MCC NC	0.0674	0.9055	0.0390	0.0945	0.9610	0.0799	0.9327	0.1883	0.0419
Precision	0.0507	0.9213	0.0651	0.0787	0.9349	0.0470	0.9280	0.1431	0.0241
Precision NC	0.3681	0.8323	0.0013	0.1677	0.9987	0.6793	0.9114	0.6963	0.6006
C	0.1000	0.8876	0.0178	0.1124	0.9822	0.1466	0.9335	0.2631	0.0800
D	0.5000	0.8167	0.0004	0.1833	0.9996	0.7911	0.9033	0.7915	0.7685

Table 27 Results for the logistic regression classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0496	0.9260	0.0560	0.0740	0.9440	0.0545	0.9349	0.1557	0.0281
F-meas NC	0.4944	0.8598	0.0036	0.1402	0.9964	0.4652	0.9254	0.5227	0.3275
G-mean	0.0441	0.9297	0.0642	0.0703	0.9358	0.0481	0.9326	0.1456	0.0247
G-mean NC	0.1006	0.9081	0.0277	0.0919	0.9723	0.1140	0.9395	0.2285	0.0613
MCC	0.0440	0.9297	0.0643	0.0703	0.9357	0.0480	0.9326	0.1455	0.0247
MCC NC	0.1212	0.9024	0.0218	0.0976	0.9782	0.1382	0.9393	0.2539	0.0756
Precision	0.0497	0.9258	0.0559	0.0742	0.9441	0.0545	0.9348	0.1558	0.0281
Precision NC	0.9541	0.5959	0.0005	0.4041	0.9995	0.6020	0.7487	0.6345	0.7548
C	0.1000	0.9081	0.0243	0.0919	0.9757	0.1153	0.9412	0.2326	0.0616
D	0.5000	0.8596	0.0033	0.1404	0.9967	0.4687	0.9254	0.5254	0.3262

Table 28 Results for the random forest depth 4 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0381	0.9152	0.0694	0.0848	0.9306	0.0441	0.9228	0.1376	0.0226
F-meas NC	0.1973	0.8520	0.0051	0.1480	0.9949	0.3822	0.9205	0.4561	0.2545
G-mean	0.0362	0.9189	0.0771	0.0811	0.9229	0.0398	0.9208	0.1302	0.0203
G-mean NC	0.0575	0.8955	0.0336	0.1045	0.9664	0.0885	0.9301	0.1988	0.0467
MCC	0.0362	0.9189	0.0771	0.0811	0.9229	0.0398	0.9208	0.1302	0.0203
MCC NC	0.0619	0.8931	0.0303	0.1069	0.9697	0.0975	0.9305	0.2093	0.0517
Precision	0.0382	0.9152	0.0694	0.0848	0.9306	0.0441	0.9228	0.1376	0.0226
Precision NC	0.6686	0.7846	0.0006	0.2154	0.9994	0.7638	0.8845	0.7702	0.7717
C	0.1000	0.8693	0.0142	0.1307	0.9858	0.1739	0.9255	0.2868	0.0968
D	0.5000	0.8398	0.0006	0.1602	0.9994	0.7746	0.9160	0.7771	0.7212

Table 29 Results for the XGBoost depth 1 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0678	0.9199	0.0399	0.0801	0.9601	0.0744	0.9396	0.1841	0.0388
F-meas NC	0.3362	0.8791	0.0054	0.1209	0.9946	0.3703	0.9349	0.4518	0.2390
G-mean	0.0607	0.9233	0.0459	0.0767	0.9541	0.0661	0.9384	0.1727	0.0343
G-mean NC	0.0883	0.9114	0.0308	0.0886	0.9692	0.0999	0.9396	0.2135	0.0531
MCC	0.0603	0.9239	0.0464	0.0761	0.9536	0.0657	0.9385	0.1721	0.0341
MCC NC	0.0894	0.9114	0.0305	0.0886	0.9695	0.1011	0.9398	0.2150	0.0539
Precision	0.0678	0.9199	0.0399	0.0801	0.9601	0.0744	0.9396	0.1841	0.0388
Precision NC	0.8462	0.7161	0.0007	0.2839	0.9993	0.6796	0.8386	0.6991	0.7259
C	0.1000	0.9083	0.0245	0.0917	0.9755	0.1139	0.9412	0.2312	0.0608
D	0.5000	0.8646	0.0026	0.1354	0.9974	0.5170	0.9284	0.5639	0.3708

Appendix 5: Results for 1:27 class ratio

See Tables 30, 31, 32 and 33.

Table 30 Results for the extremely randomized trees depth 8 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0211	0.9134	0.0612	0.0866	0.9388	0.0494	0.9259	0.1465	0.0254
F-meas NC	0.1246	0.8518	0.0025	0.1482	0.9975	0.5367	0.9217	0.5785	0.4020
G-mean	0.0196	0.9189	0.0703	0.0811	0.9297	0.0435	0.9242	0.1370	0.0223
G-mean NC	0.0315	0.8980	0.0315	0.1020	0.9685	0.0952	0.9324	0.2069	0.0504
MCC	0.0196	0.9191	0.0704	0.0809	0.9296	0.0435	0.9243	0.1370	0.0223
MCC NC	0.0349	0.8949	0.0273	0.1051	0.9727	0.1110	0.9329	0.2240	0.0596
Precision	0.0211	0.9134	0.0612	0.0866	0.9388	0.0494	0.9259	0.1465	0.0254
Precision NC	0.5029	0.7598	0.0003	0.2402	0.9997	0.7823	0.8707	0.7845	0.8149
C	0.0357	0.8910	0.0238	0.1090	0.9762	0.1146	0.9326	0.2299	0.0612
D	0.5000	0.7872	0.0003	0.2128	0.9997	0.7981	0.8869	0.7981	0.8105

Table 31 Results for the logistic regression classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0170	0.9270	0.0587	0.0730	0.9413	0.0520	0.9340	0.1519	0.0268
F-meas NC	0.3684	0.8465	0.0013	0.1535	0.9987	0.6562	0.9193	0.6736	0.5395
G-mean	0.0156	0.9305	0.0650	0.0695	0.9350	0.0474	0.9326	0.1446	0.0243
G-mean NC	0.0355	0.9041	0.0252	0.0959	0.9748	0.1181	0.9386	0.2338	0.0635
MCC	0.0155	0.9307	0.0654	0.0693	0.9346	0.0472	0.9325	0.1443	0.0242
MCC NC	0.0422	0.8994	0.0201	0.1006	0.9799	0.1411	0.9386	0.2578	0.0769
Precision	0.0170	0.9270	0.0587	0.0730	0.9413	0.0520	0.9340	0.1519	0.0268
Precision NC	0.9873	0.3665	0.0001	0.6335	0.9999	0.4655	0.5745	0.5312	0.8709
C	0.0357	0.9037	0.0231	0.0963	0.9769	0.1193	0.9394	0.2367	0.0639
D	0.5000	0.8327	0.0009	0.1673	0.9991	0.7092	0.9119	0.7171	0.6197

Table 32 Results for the random forest depth 4 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0114	0.9096	0.0657	0.0904	0.9343	0.0461	0.9217	0.1406	0.0237
F-meas NC	0.3419	0.8392	0.0008	0.1608	0.9992	0.7330	0.9156	0.7413	0.6627
G-mean	0.0111	0.9116	0.0714	0.0884	0.9286	0.0424	0.9199	0.1345	0.0217
G-mean NC	0.0158	0.8886	0.0378	0.1114	0.9622	0.0802	0.9245	0.1870	0.0422
MCC	0.0111	0.9116	0.0714	0.0884	0.9286	0.0424	0.9199	0.1345	0.0217
MCC NC	0.0193	0.8823	0.0310	0.1177	0.9690	0.1007	0.9245	0.2097	0.0541
Precision	0.0114	0.9096	0.0657	0.0904	0.9343	0.0461	0.9217	0.1406	0.0237
Precision NC	0.7582	0.7147	0.0002	0.2853	0.9998	0.7689	0.8435	0.7739	0.8448
C	0.0357	0.8632	0.0125	0.1368	0.9875	0.1911	0.9231	0.3018	0.1075
D	0.5000	0.8218	0.0004	0.1782	0.9996	0.8049	0.9062	0.8049	0.7895

Table 33 Results for the XGBoost depth 1 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0241	0.9170	0.0384	0.0830	0.9616	0.0767	0.9390	0.1870	0.0400
F-meas NC	0.3270	0.8551	0.0015	0.1449	0.9985	0.6273	0.9238	0.6510	0.4996
G-mean	0.0212	0.9219	0.0451	0.0781	0.9549	0.0666	0.9382	0.1736	0.0346
G-mean NC	0.0335	0.9057	0.0280	0.0943	0.9720	0.1095	0.9381	0.2237	0.0587
MCC	0.0211	0.9219	0.0452	0.0781	0.9548	0.0665	0.9381	0.1733	0.0345
MCC NC	0.0365	0.9042	0.0267	0.0958	0.9733	0.1199	0.9379	0.2334	0.0653
Precision	0.0241	0.9170	0.0384	0.0830	0.9616	0.0767	0.9390	0.1870	0.0400
Precision NC	0.9284	0.6142	0.0002	0.3858	0.9998	0.7054	0.7760	0.7261	0.8842
C	0.0357	0.9046	0.0234	0.0954	0.9766	0.1177	0.9398	0.2350	0.0630
D	0.5000	0.8429	0.0008	0.1571	0.9992	0.7294	0.9175	0.7362	0.6452

Appendix 6: Results for 1:81 class ratio

See Tables 34, 35, 36 and 37.

Table 34 Results for the extremely randomized trees depth 8 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0083	0.9096	0.0607	0.0904	0.9393	0.0500	0.9241	0.1469	0.0257
F-meas NC	0.1187	0.8405	0.0008	0.1595	0.9992	0.7396	0.9163	0.7453	0.6637
G-mean	0.0076	0.9136	0.0692	0.0864	0.9308	0.0438	0.9220	0.1370	0.0224
G-mean NC	0.0155	0.8870	0.0245	0.1130	0.9755	0.1209	0.9300	0.2340	0.0654
MCC	0.0076	0.9136	0.0693	0.0864	0.9307	0.0437	0.9220	0.1370	0.0224
MCC NC	0.0169	0.8844	0.0214	0.1156	0.9786	0.1352	0.9301	0.2487	0.0739
Precision	0.0083	0.9096	0.0607	0.0904	0.9393	0.0500	0.9241	0.1469	0.0257
Precision NC	0.6452	0.6098	0.0001	0.3902	0.9999	0.7219	0.7791	0.7368	0.8960
C	0.0122	0.8947	0.0321	0.1053	0.9679	0.0881	0.9305	0.1994	0.0464
D	0.5000	0.7206	0.0002	0.2794	0.9998	0.7903	0.8482	0.7945	0.8780

Table 35 Results for the logistic regression classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0061	0.9256	0.0597	0.0744	0.9403	0.0511	0.9328	0.1503	0.0263
F-meas NC	0.2694	0.8297	0.0007	0.1703	0.9993	0.7427	0.9103	0.7474	0.6767
G-mean	0.0057	0.9286	0.0666	0.0714	0.9334	0.0463	0.9309	0.1425	0.0237
G-mean NC	0.0130	0.9012	0.0232	0.0988	0.9768	0.1257	0.9381	0.2417	0.0680
MCC	0.0056	0.9290	0.0668	0.0710	0.9332	0.0462	0.9310	0.1424	0.0237
MCC NC	0.0151	0.8969	0.0192	0.1031	0.9808	0.1488	0.9378	0.2646	0.0818
Precision	0.0061	0.9256	0.0597	0.0744	0.9403	0.0511	0.9328	0.1503	0.0263
Precision NC	1.0000	0.1928	0.0000	0.8072	1.0000	0.3080	0.4288	0.4105	0.9213
C	0.0122	0.9030	0.0235	0.0970	0.9765	0.1174	0.9389	0.2345	0.0628
D	0.5000	0.8069	0.0004	0.1931	0.9996	0.7859	0.8978	0.7862	0.7676

Table 36 Results for the random forest depth 4 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0036	0.9037	0.0668	0.0963	0.9332	0.0452	0.9182	0.1386	0.0232
F-meas NC	0.3479	0.8276	0.0004	0.1724	0.9996	0.8001	0.9094	0.8007	0.7763
G-mean	0.0035	0.9055	0.0714	0.0945	0.9286	0.0423	0.9169	0.1337	0.0217
G-mean NC	0.0053	0.8787	0.0336	0.1213	0.9664	0.0942	0.9212	0.2014	0.0504
MCC	0.0035	0.9055	0.0714	0.0945	0.9286	0.0423	0.9169	0.1337	0.0217
MCC NC	0.0067	0.8746	0.0274	0.1254	0.9726	0.1190	0.9220	0.2274	0.0652
Precision	0.0036	0.9037	0.0668	0.0963	0.9332	0.0452	0.9182	0.1386	0.0232
Precision NC	0.8941	0.5714	0.0001	0.4286	0.9999	0.6930	0.7541	0.7115	0.8917
C	0.0122	0.8551	0.0109	0.1449	0.9891	0.2109	0.9195	0.3182	0.1204
D	0.5000	0.7892	0.0003	0.2108	0.9997	0.8149	0.8881	0.8152	0.8431

Table 37 Results for the XGBoost depth 1 classifier

Technique	Threshold	TPR	FPR	FNR	TNR	F-meas	G-mean	MCC	Precision
F-meas	0.0077	0.9150	0.0414	0.0850	0.9586	0.0712	0.9364	0.1793	0.0370
F-meas NC	0.4845	0.8307	0.0004	0.1693	0.9996	0.7988	0.9110	0.7996	0.7716
G-mean	0.0072	0.9158	0.0455	0.0842	0.9545	0.0656	0.9348	0.1715	0.0340
G-mean NC	0.0110	0.9087	0.0293	0.0913	0.9707	0.1058	0.9390	0.2197	0.0566
MCC	0.0071	0.9166	0.0461	0.0834	0.9539	0.0648	0.9349	0.1704	0.0336
MCC NC	0.0117	0.9075	0.0279	0.0925	0.9721	0.1128	0.9391	0.2270	0.0607
Precision	0.0077	0.9150	0.0414	0.0850	0.9586	0.0712	0.9364	0.1793	0.0370
Precision NC	0.9753	0.4784	0.0001	0.5216	0.9999	0.6186	0.6827	0.6589	0.9326
C	0.0122	0.9040	0.0233	0.0960	0.9767	0.1184	0.9395	0.2357	0.0634
D	0.5000	0.8303	0.0004	0.1697	0.9996	0.8029	0.9108	0.8033	0.7785

Abbreviations

ANN	Artificial neural network
ANOVA	Analysis of variance
AUC	Area Under the Receiver Operating Characteristic Curve
AUPRC	Area Under the Precision–Recall Curve
CAE	Convolutional autoencoder
CNN	Convolutional neural network
ET	Extremely Randomized Trees
FAU	Florida Atlantic University
FN	False negative
FNR	False negative rate
FP	False positive
FPR	False positive rate
GBDT	Gradient-boosted decision tree
HSD	Honestly significant difference
k-NN	k-Nearest neighbor
MCC	Matthews Correlation Coefficient
PCA	Principal component analysis
ROS	Random oversampling
RUS	Random undersampling
SMOTE	Synthetic minority oversampling technique
SVM	Support vector machine
TN	True negative
TNR	True negative rate
TP	True positive
TPR	True positive rate
ULB	Université Libre de Bruxelles

Acknowledgements
We would like to thank the reviewers in the Data Mining and Machine Learning Laboratory at Florida Atlantic University.

Author contributions
JLL searched for relevant papers and drafted the manuscript. All authors provided feedback to JLL and helped shape the work. JLL, JMJ, and JH prepared the manuscript. TMK introduced this topic to JLL and helped to complete and finalize the work. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

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

Received: 24 November 2022 Accepted: 19 April 2023
Published online: 06 May 2023

References

1. Leevy JL, Khoshgoftaar TM, Bauder RA, Seliya N. A survey on addressing high-class imbalance in big data. J Big Data. 2018;5(1):42.
2. Kesici M, Saner CB, Yaslan Y, Genc VI. Cost sensitive class-weighting approach for transient instability prediction using convolutional neural networks. In: 2019 11th international conference on electrical and electronics engineering (ELECO). IEEE; 2019. p. 141–5.
3. Johnson JM, Khoshgoftaar TM. Output thresholding for ensemble learners and imbalanced big data. In: 2021 IEEE 33rd international conference on tools with artificial intelligence (ICTAI). IEEE; 2021. p. 1449–54.
4. Hasanin T, Khoshgoftaar TM, Leevy JL, Bauder RA. Severely imbalanced big data challenges: investigating data sampling approaches. J Big Data. 2019;6(1):1–25.
5. Seiffert C, Khoshgoftaar TM, Van Hulse J, Napolitano A. A comparative study of data sampling and cost sensitive learning. In: 2008 IEEE international conference on data mining workshops. IEEE; 2008. p. 46–52.

6. 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; 2016. p. 785–94.
7. Hancock JT, Khoshgoftaar TM. CatBoost for big data: an interdisciplinary review. J Big Data. 2020;7(1):1–45.
8. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.
9. Acosta MRC, Ahmed S, Garcia CE, Koo I. Extremely randomized trees-based scheme for stealthy cyber-attack detection in smart grid networks. IEEE Access. 2020;8:19921–33.
10. Wang Q, Yu S, Qi X, Hu Y, Zheng W, Shi J, Yao H. Overview of logistic regression model analysis and application. Zhonghua yu fang yi xue za zhi [Chin J Prev Med]. 2019;53(9):955–60.
11. Kaggle: credit card fraud detection. https://www.kaggle.com/mlg-ulb/creditcardfraud.
12. Zhang X, Gweon H, Provost S. Threshold moving approaches for addressing the class imbalance problem and their application to multi-label classification. In: 2020 4th international conference on advances in image processing; 2020. p. 72–7.
13. Buda M, Maki A, Mazurowski MA. A systematic study of the class imbalance problem in convolutional neural networks. Neural Netw. 2018;106:249–59.
14. Zhang H, Huang L, Wu CQ, Li Z. An effective convolutional neural network based on smote and gaussian mixture model for intrusion detection in imbalanced dataset. Comput Netw. 2020;177: 107315.
15. Cohen G, Afshar S, Tapson J, Van Schaik A. EMNIST: extending MNIST to handwritten letters. In: 2017 international joint conference on neural networks (IJCNN). IEEE; 2017. p. 2921–6.
16. Yang L, Bankman D, Moons B, Verhelst M, Murmann B. Bit error tolerance of a CIFAR-10 binarized convolutional neural network processor. In: 2018 IEEE international symposium on circuits and systems (ISCAS). IEEE; 2018. p. 1–5.
17. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Commun ACM. 2017;60(6):84–90.
18. 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). IEEE; 2019. p. 1328–34.
19. Zou Q, Xie S, Lin Z, Wu M, Ju Y. Finding the best classification threshold in imbalanced classification. Big Data Res. 2016;5:2–8.
20. Salekshahrezaee Z, Leevy JL, Khoshgoftaar TM. Feature extraction for class imbalance using a convolutional autoencoder and data sampling. In: 2021 IEEE 33rd international conference on tools with artificial intelligence (ICTAI). IEEE; 2021. p. 217–23.
21. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12:2825–30.
22. Gupta A, Nagarajan V, Ravi R. Approximation algorithms for optimal decision trees and adaptive TSP problems. Math Oper Res. 2017;42(3):876–96.
23. González S, García S, Del Ser J, Rokach L, Herrera F. A practical tutorial on bagging and boosting based ensembles for machine learning: algorithms, software tools, performance study, practical perspectives and opportunities. Inf Fusion. 2020;64:205–37.
24. Lemaître G, Nogueira F, Aridas CK. Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. J Mach Learn Res. 2017;18(1):559–63.

Publisher's Note