Journal of Big Data

RESEARCH Open Access

The effect of feature extraction and data sampling on credit card fraud detection

Zahra Salekshahrezaee, Joffrey L. Leevy* and Taghi M. Khoshgoftaar

*Correspondence:
jleevy2017@fau.edu

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

Abstract

Training a machine learning algorithm on a class-imbalanced dataset can be a difficult task, a process that could prove even more challenging under conditions of high dimensionality. Feature extraction and data sampling are among the most popular preprocessing techniques. Feature extraction is used to derive a richer set of reduced dataset features, while data sampling is used to mitigate class imbalance. In this paper, we investigate these two preprocessing techniques, using a credit card fraud dataset and four ensemble classifiers (Random Forest, CatBoost, LightGBM, and XGBoost). Within the context of feature extraction, the *Principal Component Analysis* (PCA) and *Convolutional Autoencoder* (CAE) methods are evaluated. With regard to data sampling, the *Random Undersampling* (RUS), *Synthetic Minority Oversampling Technique* (SMOTE), and SMOTE Tomek methods are evaluated. The F1 score and *Area Under the Receiver Operating Characteristic Curve* (AUC) metrics serve as measures of classification performance. Our results show that the implementation of the RUS method followed by the CAE method leads to the best performance for credit card fraud detection.

Keywords: Credit Card Fraud, Random Undersampling, SMOTE, SMOTE Tomek, PCA, Convolutonal Autoencoder, Feature Extraction

Introduction

An unequal distribution of classes in a dataset is known as class imbalance. Under this condition, the majority class can overburden machine learning algorithms, thus making recognition of the minority class more challenging. As a result, classification performance scores for the impacted algorithms can become biased in favor of the dominant class. *Random Undersampling* (RUS) [1], *Synthetic Minority Oversampling Technique* (SMOTE) [2], and SMOTE Tomek [3] are examples of data-level approaches for dealing with class imbalance. Algorithm-level approaches that address this imbalance typically involve various cost-sensitive strategies [4]. In this study, the focus is on data-level approaches that mitigate class imbalance.

The data sampling technique of RUS discards members of the majority class until the ratio of instances of majority and minority class members reaches a predetermined level. SMOTE is a data augmentation technique that is used to increase the representation of the minority class in datasets. This technique synthesizes components of the minority class based on those that already exist and are proximate to each other. SMOTE Tomek

combines the ability of SMOTE, which boosts instances of the minority class, and Tomek links [5], which removes instances of the majority class that have been identified as Tomek connections. Tomek links is further described in the "Background information" section, as this technique is not as well-known as SMOTE and RUS.

Feature extraction begins with a dataset that contains the original features and uses them to generate derived features which are designed to be informative and non-redundant. This process facilitates generalization and may improve interpretation and classification performance scores. Feature extraction generally leads to dimensionality reduction. As depicted in Figure 1, the original features of a dataset are transformed into a reduced set of features. Before training classifiers on largescale imbalanced datasets, feature extraction or dimensionality reduction is often performed. Feature extraction is carried out with various algorithms, such as *Principal Component Analysis* (PCA) [6] and *Convolutional Autoencoder*s (CAEs) [7]. PCA is based on linear transformations, while autoencoders use non-linear complex functions. The feature extraction techniques used in this study are further described in the "Background information" section.

Our motivation for this work comes from the fact that there are yearly increases in the number of credit card fraud incidents [9] and that machine learning techniques have been successfully used to detect fraudulent activity. Our paper examines the use of feature extraction and data sampling on a class-imbalanced dataset. To be more specific, our research involves the evaluation of PCA and CAE techniques for feature extraction, with RUS, SMOTE, and SMOTE Tomek used as the data sampling techniques. To the best of our knowledge, this is the first study that investigates the use of PCA, CAE, RUS, SMOTE, and SMOTE Tomek on classimbalanced data. Our research uses a credit card fraud detection dataset from the Kaggle [10] community, aptly named the Credit Card Fraud Detection Dataset. Given the solid performance of ensemble classifiers in many studies [11–13], we use four ensemble learners based on the Decision Tree [14] classifier: Random Forest [15], XGBoost [16], LightGBM [17], and CatBoost [18]. Classification performance is measured with the F1 score and *Area Under the Receiver Operating Characteristic Curve* (AUC) metric.

The contribution of our research is highlighted as follows:

- Examines effect of PCA and CAE on ensemble classifiers
- Examines effect of RUS, SMOTE, and SMOTE Tomek on ensemble classifiers
- Examines effect of the order of preprocessing tasks on ensemble classifiers

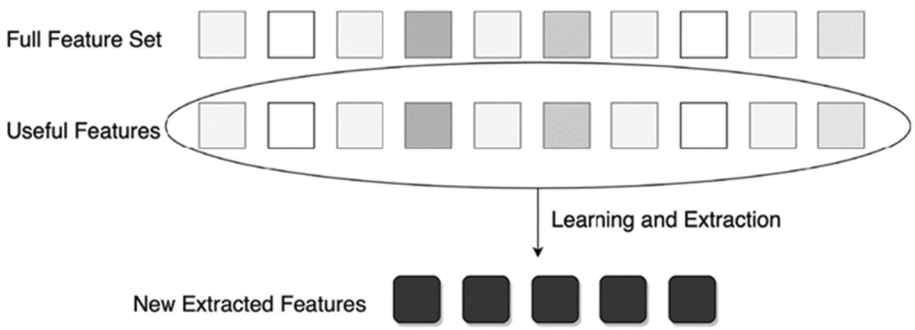

Fig. 1 Feature extraction process [8]

The remainder of this paper is organized as follows: the "Background information" section provides background information on the Tomek links, PCA, and CAE algorithms; the "Related work" section reviews relevant Bot-IoT literature; the "Methodology" section covers data preprocessing and classification tasks; the "Results and discussion" section provides and analyzes our findings; and the "Conclusion" section summarizes the key points of this paper, as well as providing suggestions for future work.

Background information

The Tomek links algorithm acts as a label noise filter and is denoted by pairs of instances. A Tomek link is a pair of data points x and y from different classes, such that, if d stands for the distance metric, there exists no example z such that $d(x,z)$ is lower than $d(x,y)$, or $d(y,z)$ is lower than $d(x,y)$. Hence, where the two examples x and y form a Tomek link, either one is noise or both are borderline. These two examples are thus eliminated from the training data. To elaborate further, in a binary classification environment with classes 0 and 1, a Tomek link pair would have an instance of each class and would be nearest neighbors across the dataset [19]. These cross-class pairs are valuable in defining the class boundary [20]. Figure 2 shows an alignment of Tomek link pairs at the class boundary. It is important to note that the use of Tomek links for label noise detection does not involve the calculation of reconstruction error.

The PCA algorithm is a feature extraction technique with a variety of applications in exploratory data analysis, visualization, and dimensionality reduction [21]. It is an unsupervised algorithm that generates a linear mixture of original features and new features which are not correlated with the original features. Furthermore, generated features are ranked according to the amount of variance that can be explained by them. As a result, Principal Component 1 represents the first principal that explains the greatest amount of variance in the dataset, Principal Component 2 represents the second principal that explains the second greatest amount of variance in the dataset, and so on. It is therefore possible to minimize the dimensionality of data with principal components.

An autoencoder is a neural network architecture that tries to learn a compact or latent representation of an input. This latent representation contains the extracted features.

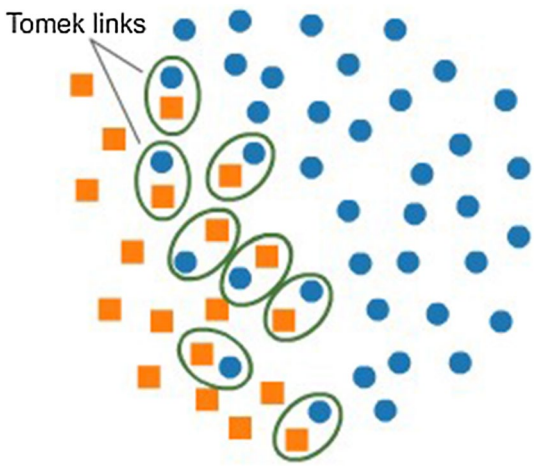

Fig. 2 Tomek link pairs [8]

The autoencoder is frequently part of a larger model that tries to recreate the input. Despite the fact that an autoencoder is an unsupervised learning method, it is technically trained via supervised learning and hence can be considered a type of semi-supervised learning.

Figure 3 illustrates the structure of a typical autoencoder, which is a feed-forward neural network containing an encoder, one or more hidden layers, and a decoder. The encoder feeds information from the input into the hidden layer, and the decoder feeds information from the hidden layer into the output layer. It is assumed that an autoencoder model will reconstruct the identical inputs that flowed through the input layer during the training process. Consequently, the decoder acts as a mirror image of the encoder, with a matching number of neurons to the encoder in both directions. For feature extraction and dimensionality reduction, the smallest hidden layer in the architecture (also referred to as the bottleneck) is used to compress the input to the lowest level of space (also known as latent space) in order to achieve the desired dimensionality reduction [22]. During the training phase, the decoder is used to calculate the error rate of the model, but it is not utilized to recover the original input dimension of the data. Several distinct types of autoencoders are available, and their uses range widely.

The CAE has a similar architecture to the *Convolutional Neural Network* (CNN) [7]. Both algorithms use some of the same fundamental components, including convolutional filters and pooling layers [24]. The encoder performs feature extraction and dimensionality reduction by using the convolution filters and pooling layers of the CNN. The decoder performs the reverse operation. Figure 4 shows the structure of a typical CAE.

Related work

The reduction of high-dimensional data, such as genomic information, images, videos, and text, is seen as an important and necessary data preprocessing step that generates high-level representations. One reason for reducing dimensionality is to provide deeper insight into the inherent structure of data. Various feature extraction techniques have

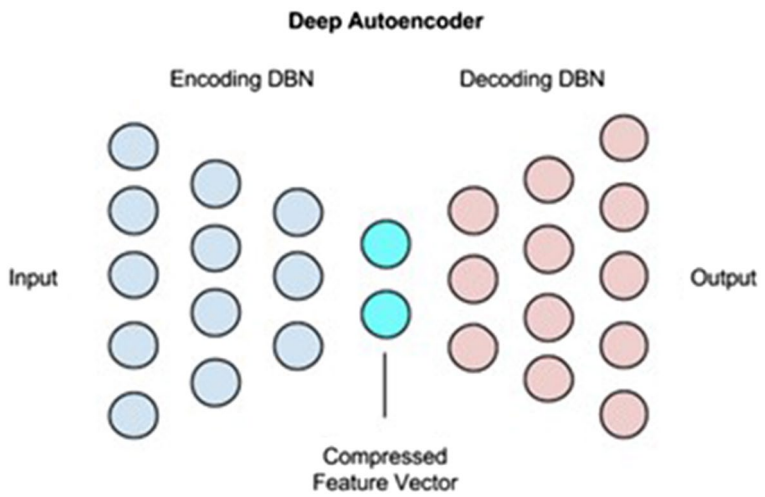

Fig. 3 Autoencoder architecture [23]

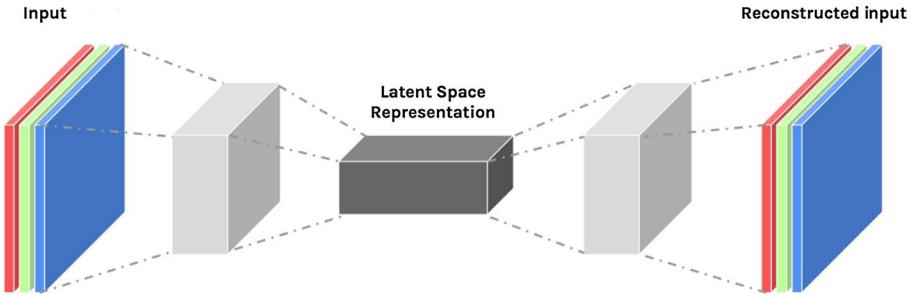

Fig. 4 Convolutional autoencoder architecture [25]

been explored. The early approaches are based on projection and involve mapping input features in the original high-dimensional space to a new low-dimensional space while minimizing information loss [26]. PCA and *Linear Discriminant Analysis* (LDA) [27] are two of the most well-known projection techniques. The former is an unsupervised method that maximizes variance to project original data into its principal directions. The latter is a supervised approach for locating a linear subspace by optimizing distinguishing data between classes. The main disadvantage of these approaches is that they conduct linear projection. Subsequent research overcame this problem by utilizing nonlinear methods. Another drawback of the early approaches is that the majority of these works tend to map data from high-dimensional to low-dimensional space by extracting features once, rather than stacking them to build deeper levels of representation progressively [22]. Autoencoders compress dimensionality by minimizing reconstruction loss using artificial neural networks. As a result, it is simple to stack autoencoders by adding hidden layers. This gives the autoencoder and its variants, such as the CAE, the ability to extract meaningful features.

Compared to the plain autoencoder, the CAE has the ability to extract smooth features by use of its pooling layers, which is advantageous for classification. Polic et al. [28] employ a CAE to reduce the optical-based output for a tactile sensor image. The authors validate their method with a set of benchmarking cases. Shallow neural networks and other machine learning models are used to estimate contact object shape, edge position, orientation, and indentation depth. A contact force estimator [29] is also trained, resulting in the confirmation that the extracted features contain sufficient information on both the spatial and mechanical properties of the object.

Meng et al. [22] note that the plain autoencoder fails to take into account relationships between data features. These relationships may impact results if original and/or novel features are used. For feature extraction, Meng et al. propose a relational autoencoder model that factors in both data features and their relationships. The authors also make their model compatible with other autoencoder variants, such as a sparse autoencoder [30], denoising autoencoder [31], and variational autoencoder [32]. Upon testing the proposed model on a set of benchmark datasets, results show that the incorporation of data relationships generates more robust features with lower reconstruction error loss, when compared to the other autoencoder variants.

In another related work, Lee et al. [33] use a CAE to perform feature extraction and dimensionality reduction for radar data analysis. The aim of their study is to obtain a

fast, accurate, and human-like image-processing algorithm. Finally, Maggipinto et al. [7] use a CAE to extract features in data-driven applications for virtual metrology. Values for optical emission spectrometry serve as the input data.

Finally, we investigated autoencoder studies performed on the Credit Card Fraud Detection Dataset published by Kaggle. The relevant works are described in the following two paragraphs.

Using both a plain autoencoder algorithm and a Logistic Regression algorithm, Al-Shabi [34] evaluated balanced and imbalanced data to detect credit card fraud in the dataset. Results show that the autoencoder outperformed Logistic Regression. However, we note that the F1 score for the autoencoder is only 0.04, due to the low value for the Precision metric.

Within the framework of one-class classification, Chen et al. [35] combined a sparse autoencoder with a Generative Adversarial Network to detect credit card fraud in the dataset. Other one-class classification algorithms were evaluated, namely OneClass Gaussian Process [36] and Support Vector Data Description [37]. Based on the results, the authors' proposed model, with a top F1 score of 0.8736, performed the best. The reproducibility of their work is questionable, since the hyperparameters used for the One-Class Gaussian Process and Support Vector Data Description algorithms have not been provided.

With regard to the final two related works, we point out that their best values for F1 score are noticeably lower than our best value obtained in this study. Further, we note that none of the related works discussed in this section use data sampling in conjunction with feature extraction.

Methodology

The Credit Card Fraud Detection Dataset [10] was published by Worldline and the *Universit´e Libre de Bruxelles* (ULB). There are 284,807 instances and 30 independent variables in the raw dataset, which shows credit card purchases by Europeans in September 2013. 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 rendering the dataset highly imbalanced with regard to the majority and minority classes.

In this study, we evaluate three data sampling techniques (RUS, SMOTE, and SMOTE Tomek) and two feature extraction techniques (PCA and CAE). The impact on classifier performance is evaluated with various scenarios, as depicted in Table 1.

The RUS technique is used to obtain five different ratios (1:1, 1:5, 1:10, 1:20 and 1:50), as shown in Tables 2, 3, 4, 5, 6. These ratios represent the minority to majority instances for each dataset obtained by down-sampling the original dataset. The use of this range of ratios strengthens the validity of our study. The SMOTE and SMOTE Tomek data sampling techniques are associated with Tables 7 and 8, respectively. Table 9 shows the results where no preprocessing was performed, i.e., no data sampling and no feature extraction.

The SMOTE, SMOTE Tomek, and RUS algorithms are included in the imbalancedlearn [38] Python library. For SMOTE and SMOTE Tomek, we set the k neighbors parameter to 5. After the implementation of the PCA algorithm with ScikitLearn [39],

Table 1 Preprocessing scenarios

Step 1	Step 2
Sampling (RUS, SMOTE, or SMOTE Tomek)	None
Sampling (RUS, SMOTE, or SMOTE Tomek)	PCA
Sampling (RUS, SMOTE, or SMOTE Tomek)	CAE
PCA	None
PCA	Sampling (RUS, SMOTE, or SMOTE Tomek)
CAE	None
CAE	Sampling (RUS, SMOTE, or SMOTE Tomek)
None	None

Table 2 Performance scores for baseline

Learner	Step 1	Step 2	F1	AUC
Random Forest	None	None	0.846	0.886
XGBoost	None	None	0.840	0.883
LightGBM	None	None	0.249	0.687
CatBoost	None	None	0.853	0.891

Table 3 Performance scores for RUS (1:1 ratio)

Learner	Step 1	Step 2	F1	AUC
Random Forest	PCA	RUS	0.138	0.950
XGBoost	PCA	RUS	0.128	0.949
LightGBM	PCA	RUS	0.155	0.948
CatBoost	PCA	RUS	0.158	0.948
Random Forest	RUS	PCA	0.238	0.976
XGBoost	RUS	PCA	0.145	0.976
LightGBM	RUS	PCA	0.189	0.976
CatBoost	RUS	PCA	0.280	0.981
Random Forest	RUS	CAE	0.215	0.963
XGBoost	RUS	CAE	0.127	0.953
LightGBM	RUS	CAE	0.151	0.957
CatBoost	RUS	CAE	0.229	0.958
Random Forest	CAE	RUS	0.122	0.934
XGBoost	CAE	RUS	0.098	0.934
LightGBM	CAE	RUS	0.113	0.940
CatBoost	CAE	RUS	0.118	0.944
Random Forest	RUS	None	0.273	0.967
XGBoost	RUS	None	0.151	0.965
LightGBM	RUS	None	0.192	0.967
CatBoost	RUS	None	0.283	0.970

Table 4 Performance scores for RUS (1:5 ratio)

Learner	Step 1	Step 2	F1	AUC
Random Forest	PCA	RUS	0.331	0.942
XGBoost	PCA	RUS	0.260	0.940
LightGBM	PCA	RUS	0.341	0.942
CatBoost	PCA	RUS	0.334	0.945
Random Forest	RUS	PCA	0.486	0.979
XGBoost	RUS	PCA	0.403	0.979
LightGBM	RUS	PCA	0.516	0.980
CatBoost	RUS	PCA	0.473	0.968
Random Forest	RUS	CAE	0.602	0.965
XGBoost	RUS	CAE	0.312	0.974
LightGBM	RUS	CAE	0.412	0.968
CatBoost	RUS	CAE	0.466	0.961
Random Forest	CAE	RUS	0.573	0.928
XGBoost	CAE	RUS	0.245	0.925
LightGBM	CAE	RUS	0.232	0.932
CatBoost	CAE	RUS	0.366	0.934
Random Forest	RUS	None	0.537	0.983
XGBoost	RUS	None	0.371	0.986
LightGBM	RUS	None	0.444	0.983
CatBoost	RUS	None	0.505	0.983

Table 5 Performance scores for RUS (1:10 ratio)

Learner	Step 1	Step 2	F1	AUC
Random Forest	PCA	RUS	0.624	0.929
XGBoost	PCA	RUS	0.397	0.928
LightGBM	PCA	RUS	0.503	0.943
CatBoost	PCA	RUS	0.590	0.944
Random Forest	RUS	PCA	0.644	0.965
XGBoost	RUS	PCA	0.600	0.965
LightGBM	RUS	PCA	0.565	0.965
CatBoost	RUS	PCA	0.619	0.954
Random Forest	RUS	CAE	0.638	0.954
XGBoost	RUS	CAE	0.487	0.961
LightGBM	RUS	CAE	0.498	0.965
CatBoost	RUS	CAE	0.556	0.957
Random Forest	CAE	RUS	0.509	0.920
XGBoost	CAE	RUS	0.437	0.923
LightGBM	CAE	RUS	0.562	0.920
CatBoost	CAE	RUS	0.563	0.928
Random Forest	RUS	None	0.783	0.966
XGBoost	RUS	None	0.639	0.973
LightGBM	RUS	None	0.733	0.965
CatBoost	RUS	None	0.756	0.962

the number of principal components obtained was 15, which is half the the number of original dataset features. To recreate the original transactions from the PCA components, the inverse transform function from Scikit-Learn is used. The CAE is

Table 6 Performance scores for RUS (1:20 ratio)

Learner	Step 1	Step 2	F1	AUC
Random Forest	PCA	RUS	0.709	0.929
XGBoost	PCA	RUS	0.642	0.929
LightGBM	PCA	RUS	0.624	0.933
CatBoost	PCA	RUS	0.743	0.926
Random Forest	RUS	PCA	0.879	0.977
XGBoost	RUS	PCA	0.766	0.977
LightGBM	RUS	PCA	0.790	0.977
CatBoost	RUS	PCA	0.748	0.966
Random Forest	RUS	CAE	0.827	0.973
XGBoost	RUS	CAE	0.778	0.977
LightGBM	RUS	CAE	0.853	0.977
CatBoost	RUS	CAE	0.788	0.966
Random Forest	CAE	RUS	0.697	0.917
XGBoost	CAE	RUS	0.668	0.921
LightGBM	CAE	RUS	0.688	0.928
CatBoost	CAE	RUS	0.712	0.921
Random Forest	RUS	None	0.848	0.981
XGBoost	RUS	None	0.816	0.981
LightGBM	RUS	None	0.840	0.981
CatBoost	RUS	None	0.859	0.981

Table 7 Performance scores for RUS (1:50 ratio)

Learner	Step 1	Step 2	F1	AUC
Random Forest	PCA	RUS	0.775	0.918
XGBoost	PCA	RUS	0.767	0.918
LightGBM	PCA	RUS	0.771	0.915
CatBoost	PCA	RUS	0.799	0.911
Random Forest	RUS	PCA	0.890	0.984
XGBoost	RUS	PCA	0.853	0.988
LightGBM	RUS	PCA	0.882	0.988
CatBoost	RUS	PCA	0.847	0.966
Random Forest	RUS	CAE	0.902	0.984
XGBoost	RUS	CAE	0.849	0.984
LightGBM	RUS	CAE	0.840	0.981
CatBoost	RUS	CAE	0.852	0.973
Random Forest	CAE	RUS	0.802	0.902
XGBoost	CAE	RUS	0.733	0.910
LightGBM	CAE	RUS	0.745	0.910
CatBoost	CAE	RUS	0.814	0.917
Random Forest	RUS	None	0.860	0.984
XGBoost	RUS	None	0.856	0.988
LightGBM	RUS	None	0.909	0.988
CatBoost	RUS	None	0.882	0.988

Table 8 Performance scores for SMOTE

Learner	Step 1	Step 2	F1	AUC
Random Forest	PCA	SMOTE	0.815	0.915
XGBoost	PCA	SMOTE	0.664	0.925
LightGBM	PCA	SMOTE	0.358	0.927
CatBoost	PCA	SMOTE	0.620	0.929
Random Forest	SMOTE	PCA	0.787	0.907
XGBoost	SMOTE	PCA	0.731	0.929
LightGBM	SMOTE	PCA	0.509	0.936
CatBoost	SMOTE	PCA	0.725	0.937
Random Forest	SMOTE	CAE	0.900	0.930
XGBoost	SMOTE	CAE	0.872	0.928
LightGBM	SMOTE	CAE	0.522	0.926
CatBoost	SMOTE	CAE	0.650	0.922
Random Forest	CAE	SMOTE	0.850	0.926
XGBoost	CAE	SMOTE	0.757	0.930
LightGBM	CAE	SMOTE	0.502	0.925
CatBoost	CAE	SMOTE	0.639	0.940
Random Forest	SMOTE	None	0.856	0.937
XGBoost	SMOTE	None	0.815	0.937
LightGBM	SMOTE	None	0.714	0.940
CatBoost	SMOTE	None	0.733	0.940

Table 9 Performance scores for SMOTE Tomek

Learner	Step 1	Step 2	F1	AUC
Random Forest	PCA	SmoteTomek	0.818	0.913
XGBoost	PCA	SmoteTomek	0.729	0.933
LightGBM	PCA	SmoteTomek	0.445	0.937
CatBoost	PCA	SmoteTomek	0.700	0.928
Random Forest	SmoteTomek	PCA	0.807	0.918
XGBoost	SmoteTomek	PCA	0.609	0.938
LightGBM	SmoteTomek	PCA	0.368	0.931
CatBoost	SmoteTomek	PCA	0.593	0.927
Random Forest	SmoteTomek	CAE	0.899	0.970
XGBoost	SmoteTomek	CAE	0.852	0.970
LightGBM	SmoteTomek	CAE	0.478	0.967
CatBoost	SmoteTomek	CAE	0.741	0.940
Random Forest	CAE	SmoteTomek	0.855	0.922
XGBoost	CAE	SmoteTomek	0.702	0.929
LightGBM	CAE	SmoteTomek	0.509	0.925
CatBoost	CAE	SmoteTomek	0.606	0.947
Random Forest	SmoteTomek	None	0.872	0.918
XGBoost	SmoteTomek	None	0.813	0.923
LightGBM	SmoteTomek	None	0.693	0.938
CatBoost	SmoteTomek	None	0.746	0.928

implemented with Keras and TensorFlow, with optimum parameters selected during preliminary experimentation [40].

The learners used in this study are Random Forest, XGBoost, LightGBM, and CatBoost. Random Forest, which is an ensemble of Decision Trees, uses the bagging [41] technique. XGBoost, LightGBM, and CatBoost are *Gradient-Boosted Decision Tree*s (GBDTs) [42], which are ensembles of Decision Trees that are trained sequentially with the boosting [43] technique. XGBoost is based on a weighted quantile sketch and a sparsity-aware function. A weighted quantile sketch uses approximate tree learning [44] for merging and pruning operations, while sparsity is concerned with zero or missing values. LightGBM is defined by Exclusive Feature Bundling and Gradient-based One-Side Sampling. Exclusive Feature Bundling reduces the count of variables through the categorization of mutually exclusive features, while One-Side Sampling excludes a chunk of instances associated with small gradients. CatBoost is designed around Ordered Boosting, an algorithm that orders instances used by Decision Trees.

Training and testing are performed with 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 crossvalidation 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: four folds used in training and one fold used in testing. The process was repeated five times.

The AUC metric is used to measure classifier performance. AUC refers to the area under the *Receiver Operating Characteristic* (ROC) curve, which plots *True Positive Rate* (TPR) against *False Positive Rate* (FPR). AUC summarizes overall model performance and is reflective of all classification thresholds along the curve [45]. The F1 score metric is the harmonic mean of precision and recall. Like AUC, the F1 score is well-suited for datasets with a high class imbalance [46]. For the F1 score, the default threshold of 0.5 was used.

Results and discussion

Tables 8, 9 show performance scores obtained for the F1 Score and AUC metrics. In each table, each row provides results for a particular combination of data sampling and feature extraction technique.

Table 9 reflects the baseline scores, where no preprocessing activity (no data sampling and no feature extraction) was implemented. The highest values of 0.853 and 0.891 are for the F1 score and AUC, respectively. These two scores were obtained with CatBoost.

The highest values among the tabulated results obtained through the RUS sampling technique are in Table 6, which is associated with a minority-to-majority class ratio of 1:50. For the F1 score and AUC, the highest values in this table are 0.909 and 0.988, respectively. The score of 0.909 was obtained by LightGBM, while the score of 0.988 was obtained by XGBoost, LightGBM, and CatBoost, all GBDTs. Interestingly, the highest F1 score for the baseline (Table 9) is greater than any of the F1 scores for the RUS ratios of 1:1, 1:5, and 1:10 (Tables 2, 3, and 4, respectively).

Table 7 was obtained with the SMOTE sampling technique. The highest values in this table for the F1 score and AUC are 0.872 and 0.940, respectively. The score of 0.872 was obtained by XGBoost, while the score of 0.940 was obtained by LightGBM and CatBoost.

Table 10 Three-factor ANOVA for F1 score

	Df	Sum Sq	Mean Sq	F Value	Pr(> F)
Scenario	7	4.72	0.675	740.4	< 2.00E-16
Classifier	3	44.8	14.934	16,383.78	< 2.00E-16
Sampling Technique	2	3.45	1.727	1894.47	< 2.00E-16
Scenario:Algorithm	21	17.97	0.856	938.76	< 2.00E-16
Scenario:Sampling	14	2.92	0.208	228.49	< 2.00E-16
Algorithm:Sampling	6	3.46	0.576	632	< 2.00E-16
Residuals	2304	2.1	0.001		

Table 11 Three-factor ANOVA for AUC

	Df	Sum Sq	Mean Sq	F Value	Pr(> F)
Scenario	7	4.749	0.6785	1215.93	< 2.00E-16
Classifier	3	1.016	0.3387	606.99	< 2.00E-16
Sampling Technique	2	0.078	0.0388	69.59	< 2.00E-16
Scenario:Algorithm	21	2.396	0.1141	204.44	< 2.00E-16
Scenario:Sampling	14	0.473	0.0338	60.6	< 2.00E-16
Algorithm:Sampling	6	0.053	0.0088	15.79	< 2.00E-16
Residuals	2304	1.286	0.0006		

In Table 8, which was obtained with the SMOTE Tomek sampling technique, the highest values of 0.899 and 0.970 are associated with the F1 score and AUC, respectively. The score of 0.899 was obtained by Random Forest, while the score of 0.970 was obtained by Random Forest and XGBoost.

To determine the statistical significance of the performance scores, we perform three-way *ANalysis Of VAriance* (ANOVA) tests. ANOVA reveals whether there is a significant difference between the group means [47]. A 95% ($\alpha = 0.05$) confidence level is used for our ANOVA tests. The results are shown in Tables 10 and 11 for the F1 score and AUC, respectively.

In these tables, *Df* is the degrees of freedom, *Sum Sq* is the sum of squares, *Mean Sq* is the mean sum of squares, *F value* is the F-statistic, and *Pr(>F)* is the *p*-value. Note that for the Sampling Technique factor, only the 1:50 minority-to-majority class ratio is considered for the RUS technique, since this ratio yields the highest performance scores among all the RUS ratios obtained. As shown in Tables 10 and 11, the *p*-value for each factor is practically 0, well below the level of α. Hence, we infer that all factors have a significant impact on performance in terms of AUC. Since this is the case, Tukey's *Honestly Significant Difference* (HSD) tests [48] are carried out to find out which groups are significantly different from each other. For a particular experiment, letter groups assigned through the Tukey method indicate similarity or significant differences in performance results within a factor.

The Tukey method is first applied within the scope of the F1 score metric. With regard to the Scenario factor (Table 12), data sampling alone is ranked in group 'a', the best-performing group. Data sampling followed by CAE is ranked in group 'b', the second-best performing group. The bottom group 'f' consists of PCA followed by data sampling. In

Table 12 Tukey's HSD Results for F1 Score: Scenario Factor

First Step	Second Step	F1-Score	Order
Sampling	None	0.807	a
Sampling	CAE	0.788	b
Sampling	PCA	0.730	c
CAE	None	0.717	d
CAE	Sampling	0.713	d
PCA	None	0.691	e
None	None	0.688	e
PCA	Sampling	0.676	f

Table 13 Tukey's HSD Results for F1 Score: Classifier Factor

Classifier	F1-Score	Order
Random Forest	0.837	a
XGBoost	0.805	b
CatBoost	0.771	c
LightGBM	0.493	d

Table 14 Tukey's HSD results for F1 score: sampling factor

Sampling Technique	F1-Score	Order
RUS	0.800	a
SMOTE Tomek	0.703	b
SMOTE	0.696	c

Table 15 Tukey's HSD results for AUC: scenario factor

First step	Second Step	F1-Score	Order
Sampling	CAE	0.954	a
Sampling	None	0.946	b
Sampling	PCA	0.939	c
CAE	Sampling	0.929	d
PCA	Sampling	0.922	e
CAE	None	0.862	f
PCA	None	0.852	g
None	None	0.833	h

terms of the Classifier factor (Table 13), Random Forest, the top performer, is in group 'a', XGBoost is in group 'b', and at the bottom is LightGBM in group 'd'. For the Sampling factor (Table 14), RUS, the best performer, is in group 'a', SMOTE Tomek is in group 'b', and SMOTE is in group 'c'.

The Tukey method is next applied within the scope of the AUC metric. With regard to the Scenario factor (Table 15), data sampling followed by CAE is ranked in group 'a', the best-performing group. Data sampling alone is ranked in group 'b', the second-best

Table 16 Tukey's HSD results for AUC: classifier factor

Classifier	AUC	Order
CatBoost	0.921	a
XGBoost	0.917	b
Random Forest	0.911	c
LightGBM	0.870	d

Table 17 Tukey's HSD results for AUC: sampling factor

Sampling technique	F1-Score	Order
RUS	0.911	a
SMOTE Tomek	0.905	b
SMOTE	0.897	c

performing group. The bottom group 'h' is associated with no preprocessing activity. In terms of the Classifier factor (Table 16), CatBoost, the top performer, is in group 'a', XGBoost is in group 'b', and at the bottom is LightGBM in group 'd'. For the Sampling factor (Table 17), RUS, the best performer, is in group 'a', SMOTE Tomek is in group 'b', and SMOTE is in group 'c'.

Based on the Tukey's HSD results for the F1 score and AUC metrics, the RUS technique is the clear-cut top choice for the Sampling factor. However, the choice of best classifier could not be established from the rankings. This is because the HSD results for the F1 score metric (Table 13) show Random Forest as the best classifier, while the HSD results for the AUC metric (Table 16) show CatBoost as the best classifier. The Scenario factor shows data sampling followed by CAE as the second-best choice for the F1 score metric (Table 12) and the best choice for the AUC metric (Table 15). Conversely, data sampling alone is the top choice for the F1 score metric and the second-best choice for the AUC metric. We recommend the use of data sampling followed by CAE for the Scenario factor. This is because the implementation of CAE, which is a feature extraction technique, tends to reduce computational burden and decrease the training time of machine learning algorithms. As stated earlier, feature extraction may also improve generalization and the interpretation of results. With regard to the Scenario factor for the F1 score and AUC, the following was observed: Sampling + CAE is better than Sampling + PCA; CAE + Sampling is better than PCA + Sampling; and CAE + None is better than PCA + None. We believe that CAE has an advantage over PCA because the autoencoder can model non-linear functions.

Conclusion

In this research, we use a credit fraud dataset to investigate the effect of data sampling and feature extraction on four ensemble classifiers. Three data sampling techniques, RUS, SMOTE, and SMOTE Tomek are evaluated, and two feature extraction techniques, PCA and CAE, are evaluated. The results indicate that the use of the RUS data sampling technique followed by the use of the CAE feature extraction technique yields the best results.

Future work will involve additional classifiers and datasets, with a focus on incorporating data that contains audio and images. In addition, further work will consider other data sampling and feature extraction algorithms.

Abbreviations

AUC	Area Under the Receiver Operating Characteristic Curve
ANOVA	ANalysis Of VAriance
CAE	Convolutional Autoencoder
CNN	Convolutional Neural Network
FPR	False Positive Rate
GBDT	Gradient-Boosted Decision Tree
HSD	Honestly Significant Difference
LDA	Linear Discriminant Analysis
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristic
RUS	Random Undersampling
SMOTE	Synthetic Minority Oversampling Technique
TPR	True Positive Rate
ULB	Universit´e 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
ZS carried out the conception and design of the research, performed the implementation and experimentation, and drafted the manuscript. All authors provided feedback to ZS and helped shape the research. ZS and JLL prepared the manuscript. TMK introduced this topic to ZS and helped to complete and finalize this 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: 7 July 2022 Accepted: 5 January 2023
Published online: 17 January 2023

References

1. Liu B, Tsoumakas G. Dealing with class imbalance in classifier chains via random undersampling. Knowl-Based Syst. 2020;192: 105292.
2. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. Smote: synthetic minority over-sampling technique. J Artif Intell Res. 2002;16:321–57.
3. Jonathan B, Putra PH, Ruldeviyani Y. Observation imbalanced data text to predict users selling products on female daily with smote, tomek, and smote-tomek. In: 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), IEEE. pp. 81–85; 2020.
4. Thai-Nghe N, Gantner Z, Schmidt-Thieme L. Cost-sensitive learning methods for imbalanced data. In: The 2010 International Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1–8; 2010.
5. Tomek I, et al. Two modifications of cnn. IEEE Trans Syst Man Cybern. 1976;11:769–72.
6. Peng C, Chen Y, Kang Z, Chen C, Cheng Q. Robust principal component analysis: a factorization-based approach with linear complexity. Inf Sci. 2020;513:581–99.
7. Maggipinto M, Masiero C, Beghi A, Susto GA. A convolutional autoencoder approach for feature extraction in virtual metrology. Procedia Manufacturing. 2018;17:126–33.
8. Alsenan SA, Al-Turaiki IM, Hafez AM. Feature extraction methods in quantitative structure–activity relationship modeling: a comparative study. IEEE Access. 2020;8:78737–52.

9. Popat RR, Chaudhary J. A survey on credit card fraud detection using machine learning. In: 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), IEEE. 2018. p. 1120–1125.
10. Kaggle: Credit Card Fraud Detection. https://www.kaggle.com/mlg-ulb/creditcardfraud
11. Hancock JT, Khoshgoftaar TM. Catboost for big data: an interdisciplinary review. J Big data. 2020;7(1):1–45.
12. Zuech R, Hancock J, Khoshgoftaar TM. Detecting web attacks using random undersampling and ensemble learners. J Big Data. 2021;8(1):1–20.
13. Leevy JL, Hancock J, Zuech R, Khoshgoftaar TM. Detecting cybersecurity attacks across different network features and learners. J Big Data. 2021;8(1):1–29.
14. Patel HH, Prajapati P. Study and analysis of decision tree based classification algorithms. Int J Computer Sci Eng. 2018;6(10):74–8.
15. Breiman L. Random forests. Mach Learning. 2001;45(1):5–32.
16. Shi X, Wong YD, Li MZ-F, Palanisamy C, Chai C. A feature learning approach based on xgboost for driving assessment and risk prediction. Accid Anal Prev. 2019;129:170–9.
17. Tang C, Luktarhan N, Zhao Y. An efficient intrusion detection method based on lightgbm and autoencoder. Symmetry. 2020;12(9):1458.
18. Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, Gulin A. Catboost: unbiased boosting with categorical features. In: Advances in Neural Information Processing Systems, p. 6638–6648. 2018.
19. He H, Ma Y. Imbalanced Learning: Foundations, Algorithms, and Applications. New York: Wiley; 2013.
20. Brownlee J. Undersampling algorithms for imbalanced classification. https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/
21. Jolliffe IT, Cadima J. Principal component analysis: a review and recent developments. Philos Trans R Soci. 2016;374(2065):20150202.
22. Meng Q, Catchpoole D, Skillicom D, Kennedy PJ. Relational autoencoder for feature extraction. In: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE; p. 364–371. 2017.
23. Nicholson C. A Beginner's Guide to Important Topics in AI, Machine Learning, and Deep Learning: Deep utoencoders. https://wiki.pathmind.com/deep-autoencoder
24. Safayenikoo P, Akturk I. Weight update skipping: Reducing training time for artificial neural networks. arXiv preprint arXiv:2012.02792. 2020.
25. Chablani M. Autoencoders: Introduction and Implementation in TF. https://towardsdatascience.com/autoencoders-introduction-and-implementation-3f40483b0a85
26. Khalid S, Khalil T, Nasreen S. A survey of feature selection and feature extraction techniques in machine learning. In: 2014 Science and Information Conference, IEEE. p. 372–378; 2014.
27. Sharma A, Paliwal KK. Linear discriminant analysis for the small sample size problem: an overview. Int J Mach Learn Cybern. 2015;6(3):443–54.
28. Polic M, Krajacic I, Lepora N, Orsag M. Convolutional autoencoder for feature extraction in tactile sensing. IEEE Robot Autom Lett. 2019;4(4):3671–8.
29. Garcıa JG, Robertsson A, Ortega JG, Johansson R. Generalized contact force estimator for a robot manipulator. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006; p. 4019–4024 (2006). IEEE.
30. Al-Qatf M, Lasheng Y, Al-Habib M, Al-Sabahi K. Deep learning approach combining sparse autoencoder with svm for network intrusion detection. IEEE Access. 2018;6:52843–56.
31. Meng Z, Zhan X, Li J, Pan Z. An enhancement denoising autoencoder for rolling bearing fault diagnosis. Measurement. 2018;130:448–54.
32. Zavrak S, Iskefiyeli M. Anomaly-based intrusion detection from network flow features using variational autoencoder. IEEE Access. 2020;8:108346–58.
33. Lee H, Kim J, Kim B, Kim S. Convolutional autoencoder based feature extraction in radar data analysis. In: 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), IEEE. p. 81–84; 2018.
34. Al-Shabi M. Credit card fraud detection using autoencoder model in unbalanced datasets. J Adv Math Computer Sci. 2019;33(5):1–16.
35. Chen, J., Shen, Y., Ali, R.: Credit card fraud detection using sparse autoencoder and generative adversarial network. In: 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 1054–1059 (2018). IEEE
36. Kemmler M, Rodner E, Wacker E-S, Denzler J. One-class classification with gaussian processes. Pattern Recogn. 2013;46(12):3507–18.
37. Kim S, Choi Y, Lee M. Deep learning with support vector data description. Neurocomputing. 2015;165:111–7.
38. imbalanced-learn developers T. Imbalanced-learn documentation. https://imbalanced-learn.org/stable/
39. 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.
40. Gulli A, Pal S. Deep Learning with Keras. New York: Packt Publishing Ltd; 2017.
41. Gonzalez S, Garıa 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. Inform Fusion. 2020;64:205–37.
42. Wen Z, He B, Kotagiri R, Lu S, Shi J. Efficient gradient boosted decision tree training on gpus. In: 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 234–243 (2018). IEEE.
43. Basha SM, Rajput DS, Vandhan V. Impact of gradient ascent and boosting algorithm in classification. Int J Intell Eng Syst (IJIES). 2018;11(1):41–9.
44. Gupta A, Nagarajan V, Ravi R. Approximation algorithms for optimal decision trees and adaptive tsp problems. Math Oper Res. 2017;42(3):876–96.
45. Seliya N, Khoshgoftaar TM, Van Hulse J. A study on the relationships of classifier performance metrics. In: ICTAI'09. 21st International Conference On Tools with Artificial Intelligence, 2009, IEEE. 2009. p. 59–66.

46. Gu Q, Zhu L, Cai Z. Evaluation measures of the classification performance of imbalanced data sets. In: International Symposium on Intelligence Computation and Applications. 2009; Springer. p. 461–71.
47. Iversen GR, Norpoth H, Norpoth HP. Analysis of Variance. New York: Sage; 1987.
48. Tukey JW. Comparing individual means in the analysis of variance. Biometrics. 1949;8:99–114.

Publisher's Note