Feature Extraction for Class Imbalance Using a Convolutional Autoencoder and Data Sampling

Zahra Salekshahrezaee*, Joffrey L. Leevy*, and Taghi M. Khoshgoftaar*
*Florida Atlantic University
Email: zsalekshahre2018@fau.edu, jleevy2017@fau.edu, khoshgof@fau.edu

***Abstract*—Training a machine learning algorithm from a class-imbalanced dataset is an inherently challenging task. The task becomes more challenging when compounded by high dimensionality (a high number of features). Feature extraction is a data reduction process that transforms features into linear or non-linear combinations of the original features, resulting in a smaller and richer set of attributes. Data sampling is a popular approach for addressing class imbalance. In this paper, our proposed method requires the implementation of feature extraction before data sampling, based on the idea that richer high-level features facilitate more efficient sampling and hence, produce better classification results. We use *principal component analysis* (PCA) and *convolutional autoencoder* (CAE) as the feature extraction techniques and *synthetic minority oversampling technique* (SMOTE) as the data sampling technique. In evaluating the performance of the random forest classifier on a credit card fraud dataset, our results show that CAE is a better feature extraction technique than PCA. The combination of CAE followed by SMOTE yields the best F1-score of 90.5%.**

***Index Terms*—feature extraction, SMOTE, class imbalance, PCA, convolutional autoencoder**

1. Introduction

Class imbalance in a dataset occurs when there is an unequal distribution of classes. The majority class(es) can overwhelm learning algorithms, making it difficult to detect minority class(es). As a result, classification performance scores for the affected algorithms become biased in favor of the dominant class(es). Class imbalance is addressed through data-level and/or algorithm-level approaches. Data-level approaches include sampling techniques, such as random undersampling [1], random oversampling [1], and *synthetic minority oversampling technique* (SMOTE) [2], and also a few specialized feature selection techniques [3]. Algorithm-level approaches typically involve various cost-sensitive techniques [4].

Feature extraction is concerned with the derivation of a reduced set of features from a larger set [5]. This transformed set of features should be informative, support generalization, and where possible, make for better human interpretation. It should be noted that feature extraction leads to dimensionality reduction. For large-scale imbalanced datasets, feature extraction or dimensionality reduction is often necessary before training predictive models. When class imbalance and high dimensionality co-occur, researchers and practitioners of machine learning should ask the following question: Can feature extraction maintain or improve the classification performance score of a predictive model?

Several algorithms are available for performing feature extraction. Among the most popular are autoencoders [6], [7] and *principal component analysis* (PCA) [8], [9]. An autoencoder is a type of neural network that learns a compressed representation of the original data. The saved data representation from the autoencoder can subsequently be used to train a different learner [10]. PCA achieves dimensionality reduction through the calculation of a few principal components that account for maximum variance. PCA is a linear transformation algorithm, while an autoencoder is an algorithm capable of modeling complex, non-linear functions.

This paper investigates the use of feature extraction and data sampling on a high-dimensional, class-imbalanced dataset. More specifically, our contribution revolves around the evaluation of *convolutional autoencoder* (CAE) and PCA methods for feature extraction, with SMOTE as the data sampling technique. To the best of our knowledge, this is the first work that investigates the use of CAE, PCA, and SMOTE on high-dimensional, class-imbalanced data. The research incorporates a credit card fraud detection dataset from Kaggle [11] and a random forest [12], [13] classifier. Our experiments assess the use of SMOTE only, feature extraction only, both SMOTE and feature extraction, and a baseline (no SMOTE and no feature extraction). Classification performance is reported with precision, recall, and F1-score metrics.

The remainder of this paper is organized as follows: Sections 1-A and 1-B provide more detail on class-imbalanced data and feature extraction. Section 2 provides an overview of related CAE literature; Section 3 covers our methodology; Section 4 presents and discusses our results; and Section 5 concludes with a brief summary and suggestions for future work.

A. Imbalanced Data

Machine learning methods are sensitive to the distribution of classes in a training set. In this paper, we focus on binary class imbalance, since multi-class imbalance problems can often be defined by a sequence of binary classification tasks [14], [15]. For binary class imbalance, the class of interest is usually the minority class, which is under-represented. Hence, it becomes a challenge for the learner to detect instances of the minority class. Under high class imbalance conditions, identifying minority class instances in a dataset is comparable to searching for a needle in a haystack. High class imbalance

is defined as a majority-to-minority ratio of class instances between 100:1 and 10,000:1 [16].

If class imbalance is not sufficiently addressed during the training stage of a predictive model, it can lead to the misclassification of minority instances in unseen (test) data. This concern is especially important in real-world applications, where misclassifying instances from the minority class may prove costly [17]–[19]. For example, a classifier that labels a defective jet engine for a commercial aircraft as "non-defective" becomes an agent of disaster. As mentioned previously, there are several techniques available for tackling class imbalance, and in the remaining paragraphs of this subsection, we explain further.

During random undersampling, instances from the majority class are randomly discarded. This reduces the size of the dataset, but there is always the risk of losing important data. Conversely, random oversampling is the indiscriminate duplication of minority class instances. This oversampling technique increases the size of the dataset but may lead to the overfitting of the classifier [1], [20]. Another popular oversampling technique is SMOTE, which synthetically creates new minority class instances by interpolating between existing minority class instances that are near each other. Unlike random oversampling, SMOTE safeguards against overfitting because the duplication of new instances under SMOTE is a restricted process [21], [22].

The use of specialized feature selection techniques for mitigating class imbalance is a less popular data-level approach. This strategy is based on the concept that the most influential features can provide additional information for class discrimination [1]. Lastly, algorithm-level approaches use various cost-sensitive techniques to minimize class imbalance. In the event of a misclassification of an instance, more weight is assigned to the specific instance.

For addressing class imbalance, data-sampling approaches are more established and popular than algorithm-level approaches [23]–[25]. Taking this into consideration, we adopt a data-sampling approach. We choose SMOTE because it has been repeatedly shown to be a reliable sampling technique for many applications in different domains [26]–[28].

B. Feature Extraction

Feature extraction is a method of feature reduction. As shown in Figure 1 [29], the original features are transformed into a reduced set of linear or non-linear combinations of features. In this particular figure, all the original features are depicted as useful. However, this is not always the case.

The new extracted features are described by fewer, more meaningful attributes. Applications of feature extraction are widely used in the data compression, pattern recognition, and data decomposition domains. Feature extraction is also used to accelerate the training of supervised learners [30].

According to Eklund [31], feature extraction has several other advantages. It reduces the computational burden on hardware resources, allows for easier visualization of data, and in some cases, may boost the classification performance of models. For the remainder of this section, we describe PCA and the autoencoder, along with the autoencoder variant that we use, the CAE.

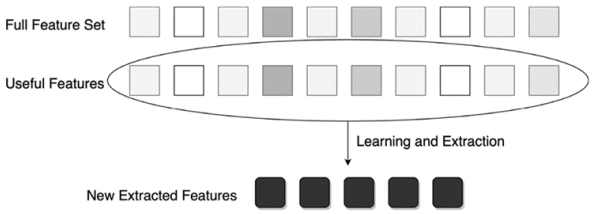

Fig. 1: Feature Extraction Process [29]

1) Principal Component Analysis: PCA is an unsupervised algorithm that creates a linear combination of original features, with the new features being uncorrelated [32]. The new features are ranked in order of their explained variance. For example, principal component 1 explains the component associated with the most variance, principal component 2 explains the component associated with the second highest variance, and so on. By restricting the number of principal components, the dimensionality of data can be reduced. The number of principal components is determined by the threshold of explained variance that is set. PCA is a simple and fast process. However, data must be standardized before it can be processed by PCA.

2) Autoencoder: An autoencoder is a type of neural network model for learning a compressed format of input data. Autoencoders are usually trained as part of a larger model that aims to recreate the input data. Although autoencoders are an unsupervised learning method, they can be considered a form of semi-supervised learning, since they are trained by using automatically generated labels equal to their inputs.

Figure 2 [33] 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 is the input to the hidden layer, and the decoder is the output to the hidden layer. The vectors in the input layer are mapped to the same number of vectors in the output layer by the autoencoder [34]. An autoencoder model is expected to reconstruct the same inputs that pass through the input layer. As a result, the decoder functions as a mirror image of the encoder, with a matching number of neurons.

The encoder maps each input x_i to a reduced representation y_i using a function $g(x)$, as shown in Equation 1, where $g(x)$ is a sigmoid function, and W is a weight matrix $dy \times dx$:

$$y_i = g(Wx_i) \tag{1}$$

Using the same reduced representation y_i, the decoder reconstructs x_i, as shown in Equation 2:

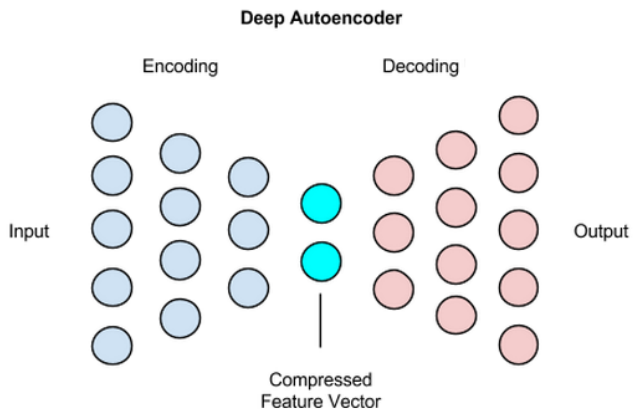

Fig. 2: Autoencoder Architecture [33]

$$\acute{x_i} = f(\acute{W} y_i) \tag{2}$$

An autoencoder is typically trained with both an encoder and decoder. For dimensionality reduction, the smallest hidden layer in the architecture (also known as the bottleneck) is used to compress the input to the lowest level of space (latent space) [35]. The decoder is used during training to calculate the error rate of the model but not to restore the original input dimension. To process more complex data structures, the hidden layers can be expanded from a single layer to multiple layers. There are many different types of autoencoders, and their applications vary.

3) Convolutional Autoencoder: The CAE is an unsupervised feature extraction algorithm, similar in structure to the *convolutional neural network* (CNN) [36]. Both algorithms share some basic components, such as convolutional filters [37]. The CNN is typically used for image recognition and classification, video recognition, natural language processing, and time series classification.

According to Lee et al. [38], the latent representation of the $k-th$ feature map in the encoder for a mono-channel input x is given by Equation 3, where * denotes the two-dimensional convolution and σ defines the activation function:

$$h^k = \sigma(x * W^k + b^k) \tag{3}$$

In Equation 4, c represents a bias per input channel and H represents a group of latent feature maps:

$$y = \sigma(\sum_{k \in H} h^k * W^k + c) \tag{4}$$

The *mean squared error* (MSE) is the objective function to minimize, as shown in Equation 5:

$$E(\theta) = \frac{1}{2n} \sum_{1}^{n} (y_i - x_i)^2 \tag{5}$$

The structure of the CAE is shown in Figure 3. It is divided into two parts: encoder and decoder. The encoder extracts features and reduces the size of the input by using the convolutional filters and pooling layers of the CNN. The decoder does the reverse process.

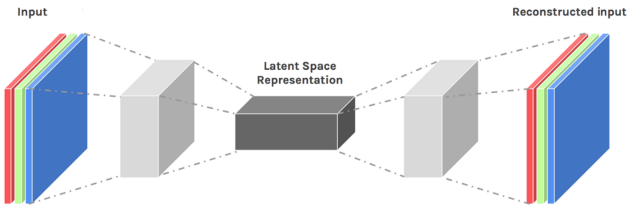

Fig. 3: Convolutional Autoencoder Architecture [39]

2. 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 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 [40]. PCA and *linear discriminant analysis* (LDA) [41] 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 non-linear methods [42], [43]. 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 [35]. 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.

One advantage of using the CAE is its ability to preserve and manipulate spatial information. Polic et al. [44] 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 [45] 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. [35] note that the plain autoencoder fails to take into account relationships between data samples. 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 [46], denoising autoencoder [47], and variational autoencoder [48]. 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. [38] 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. [49] use a CAE to extract features in data-driven applications for virtual metrology. Values for optical emission spectrometry serve as the input data.

3. Methodology

The credit card fraud detection dataset [11] was published by Worldline and the *Université Libre de Bruxelles* (ULB). There are 248,807 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 the effect of using the SMOTE data sampling technique and feature extraction techniques (PCA and CAE) on classifier performance. Our experiments involve the use of SMOTE only, feature extraction only, both SMOTE and feature extraction, and a baseline (no SMOTE and no feature extraction). The feature extraction preprocessing step, when used with the SMOTE preprocessing step, is performed first in one set of experiments. In another set of experiments, the feature extraction preprocessing step, when used with the SMOTE preprocessing step, is performed second.

The majority-to-minority class distribution is balanced with the implementation of SMOTE, an algorithm included in the imbalanced-learn [50] Python library. The PCA algorithm is implemented with Scikit-Learn [51]. The number of principal components is the same as the number of original features from our dataset. To recreate the original transactions from the PCA components, the inverse transform function from Scikit-Learn is used. The CAE is implemented with Keras and TensorFlow [52]. Based on preliminary experimentation that produced good results, we use the Adam optimization algorithm and a learning rate of 0.001. With respect to the encoder, the first layer generates 8 features, the second generates 16 features, and the third generates 32 features. A one-dimensional convolutional window of size 3 is used for these layers. Two fully connected layers come after the set of convolutional layers. The architecture of the decoder is the reverse of the encoder's, meaning that the decoder starts with two fully connected layers, followed by three convolutional layers.

TABLE 1: Scores for Random Forest with Feature Extraction and SMOTE

First Step	Second Step	Precision	Recall	F1-score
none	none	**97.0%**	74.0%	84.0%
SMOTE	no feature extraction	89.0%	84.0%	86.4%
PCA	no SMOTE	91.0%	73.0%	81.0%
SMOTE	PCA	82.0%	79.0%	80.5%
PCA	SMOTE	86.0%	82.0%	84.0%
CAE	no SMOTE	89.0%	86.0%	87.5%
CAE	SMOTE	92.0%	**89.0%**	**90.5%**
SMOTE	CAE	91.0%	86.0%	88.4%

TABLE 2: Two-factor ANOVA for F1-score

	Df	Sum Sq	Mean Sq	F Value	Pr(>F)
First Step	3	874.7	291.57	36.393	<2.00E-16
Second Step	3	794.6	264.87	33.061	<2.00E-16
Interaction	1	9.6	9.55	1.192	0.277
Residuals	152	1217.8	8.01		

Random forest is an ensemble classifier of decision trees that uses a bagging technique [53], [54]. Bagging involves the aggregation of predictions of each classifier, upon which the predicted class becomes the one with the most votes. For random forest, we note the use of the following non-default hyperparameter values: $criterion$= 'entropy' and max_depth=10.

The training and testing approach is based on k-fold cross-validation, where the model is trained on k-1 folds each time and tested on the remaining fold. This is to ensure that as much data as possible is used in the classification. Our cross-validation process is stratified, which attempts to ensure that each class is equally represented across the folds. In this study, we assigned a value of 5 to k: four folds for training and one fold for testing. The process is repeated 4 times.

Since we randomly shuffle instances during cross-validation, different results may be obtained when the order of instances is changed [55], [56]. One way of addressing the undesirable effect of this randomness is by performing several iterations, as we have done by repeating the process 4 times. Note that each value shown in Table 1 is an average of (4 repetitions x 5 folds) 20 performance scores.

Our primary metric, the F1-score, is the harmonic mean of precision and recall. This metric is well-suited for datasets with a high class imbalance [57]–[59]. Precision and recall scores have been included in this study to help interpret the F1-score.

4. Results and Discussion

Table 1 shows values for precision, recall, and F1-score for the random forest model. Each row provides results for a specific and ordered combination of the first and second preprocessing steps. The highest column score for each performance metric is boldfaced.

TABLE 3: Tukey's HSD Results

First Step - Second Step	Group
CAE - SMOTE	a
SMOTE - CAE	ab
CAE - no SMOTE	b
SMOTE - no feature extraction	b
none - none	c
PCA - SMOTE	c
PCA - no SMOTE	cd
SMOTE - PCA	d

The combination of CAE followed by SMOTE appears to be the top performer. It has the highest F1-score (90.5%), the highest recall (89.0%), and the second-highest precision (92.0%). However, there is no clear-cut worst performer. For example, the baseline (no feature extraction with no SMOTE) yields the second-lowest recall (74.0%) and third-lowest F1-score (84.0%), but this combination has the highest precision (97.0%). As another example, SMOTE used with no feature extraction results in the third-lowest precision (89.0%), the fourth-lowest F1-score (86.4%), and the third-highest recall (84.0%).

As previously stated, the F1-score takes into account both precision and recall. For this reason, more weight should be given to the F1-score than to its individual components. Therefore, based on the F1-score values in Table 1, the combinations of CAE followed by SMOTE and SMOTE followed by CAE produce the best results of 90.5% and 88.4%, respectively.

To understand the statistical significance of the F1-score values, we perform two-factor *ANalysis Of VAriance* (ANOVA) tests. ANOVA establishes whether there is a significant difference between group means [60]. A 95% (α = 0.05) confidence level is used for our ANOVA tests. The results are shown in Table 2, where *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.

The two factors of concern for our ANOVA tests are the first and second preprocessing steps. As these two factors show a p-value of less than 0.05 in Table 2, this indicates that both are significant in terms of the F1-score. Hence, we perform Tukey's *Honestly Significant Difference* (HSD) tests [61] to determine which groups are significantly different from each other for both factors. Letter groups assigned via the Tukey method indicate similarity or significant differences in performance results within a factor. In our case, the best-performing group is assigned the letter 'a' and the worst-performing group, the letter 'd'.

Our HSD test results are shown in Table 3. The best group, 'a', comprises only the combination of CAE followed by SMOTE. Unsurprisingly, group 'ab' contains the combination of SMOTE followed by CAE. Group 'b' contains combinations of CAE with no SMOTE and SMOTE with no feature extraction. We expected the latter combination in group 'b' to be ranked lower, so this demonstrates that high levels of class imbalance in datasets should always be addressed. The worst group,'d', consists only of the combination of SMOTE followed by PCA.

It is evident from the HSD tests and F1-score values that the combination of CAE followed by SMOTE produces the best results. With regard to our two feature extraction techniques, the ability of the CAE to model non-linear functions gives it an edge over PCA.

5. Conclusion

Using a credit card fraud dataset, we evaluate the impact of two feature extraction techniques and one data sampling technique on the random forest classifier. For feature extraction, the PCA and CAE methods are compared. PCA is basically a linear transformation, while CAE is capable of modeling non-linear functions. To balance the highly imbalanced dataset, SMOTE is applied as a means of oversampling. The feature extraction step, when used with the SMOTE step, is performed first in one set of experiments. In another set of experiments, the feature extraction step, when used with the SMOTE step, is performed second. We evaluate the use of SMOTE only, feature extraction only, both SMOTE and feature extraction, and a baseline (no SMOTE and no feature extraction). Best results are obtained with the use of CAE, followed by the use of SMOTE.

Future work will involve the use of additional feature extraction techniques and classifiers. In addition, datasets from different application domains will be incorporated into the research.

6. Acknowledgments

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

References

[1] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *Journal of Big Data*, vol. 5, no. 1, p. 42, 2018.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[3] S. M. Abd Elrahman and A. Abraham, "A review of class imbalance problem," *Journal of Network and Innovative Computing*, vol. 1, no. 2013, pp. 332–340, 2013.

[4] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme, "Cost-sensitive learning methods for imbalanced data," in *The 2010 International joint conference on neural networks (IJCNN)*. IEEE, 2010, pp. 1–8.

[5] N. Tajziyehchi, M. Moshirpour, G. Jergeas, and F. Sadeghpour, "A predictive model of cost growth in construction projects using feature selection," in *2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE, 2020, pp. 142–147.

[6] Z. Salekshahrezaee, J. L. Leevy, and T. M. Khoshgoftaar, "A reconstruction error-based framework for label noise detection," *Journal of Big Data*, vol. 8, no. 1, pp. 1–16, 2021.

[7] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[8] C. Peng, Y. Chen, Z. Kang, C. Chen, and Q. Cheng, "Robust principal component analysis: A factorization-based approach with linear complexity," *Information Sciences*, vol. 513, pp. 581–599, 2020.

[9] T. M. Khoshgoftaar and R. M. Szabo, "Improving neural network predictions of software quality using principal components analysis," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 5. IEEE, 1994, pp. 3295–3300.

[10] W. Wang, Y. Huang, Y. Wang, and L. Wang, "Generalized autoencoder: A neural network framework for dimensionality reduction," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 490–497.

[11] Kaggle, "Credit card fraud detection," https://www.kaggle.com/mlg-ulb/creditcardfraud.

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

[13] T. M. Khoshgoftaar, M. Golawala, and J. Van Hulse, "An empirical study of learning from imbalanced data using random forest," in *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, vol. 2. IEEE, 2007, pp. 310–317.

[14] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of artificial intelligence research*, vol. 2, pp. 263–286, 1994.

[15] X.-Y. Liu, Q.-Q. Li, and Z.-H. Zhou, "Learning imbalanced multi-class data with optimal dichotomy weights," in *2013 IEEE 13th International Conference on Data Mining*. IEEE, 2013, pp. 478–487.

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

[17] A. A. Shanab, T. M. Khoshgoftaar, R. Wald, and J. Van Hulse, "Comparison of approaches to alleviate problems with high-dimensional and class-imbalanced data," in *2011 IEEE International Conference on Information Reuse & Integration*. IEEE, 2011, pp. 234–239.

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

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

[20] R. A. Bauder and T. M. Khoshgoftaar, "A study on rare fraud predictions with big medicare claims fraud data," *Intelligent Data Analysis*, vol. 24, no. 1, pp. 141–161, 2020.

[21] A. Fernández, S. del Río, N. V. Chawla, and F. Herrera, "An insight into imbalanced big data classification: outcomes and challenges," *Complex & Intelligent Systems*, vol. 3, no. 2, pp. 105–120, 2017.

[22] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 935–942.

[23] H. Ali, M. N. M. Salleh, K. Hussain, A. Ahmad, A. Ullah, A. Muhammad, R. Naseem, and M. Khan, "A review on data preprocessing methods for class imbalance problem," *International Journal of Engineering & Technology*, vol. 8, pp. 390–397, 2019.

[24] A. M. Mahmood, "Class imbalance learning in data mining–a survey," *International Journal of Communication Technology for Social Networking Services*, vol. 3, no. 2, pp. 17–38, 2015.

[25] A. Singh and A. Purohit, "A survey on methods for solving data imbalance problem for classification," *International Journal of Computer Applications*, vol. 127, no. 15, pp. 37–41, 2015.

[26] A. Patil, A. Framewala, and F. Kazi, "Explainability of smote based oversampling for imbalanced dataset problems," in *2020 3rd International Conference on Information and Computer Technologies (ICICT)*. IEEE, 2020, pp. 41–45.

[27] A. Fernández, S. Garcia, F. Herrera, and N. V. Chawla, "Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary," *Journal of artificial intelligence research*, vol. 61, pp. 863–905, 2018.

[28] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Building useful models from imbalanced data with sampling and boosting." in *FLAIRS conference*, 2008, pp. 306–311.

[29] S. A. Alsenan, I. M. Al-Turaiki, and A. M. Hafez, "Feature extraction methods in quantitative structure–activity relationship modeling: A comparative study," *IEEE Access*, vol. 8, pp. 78 737–78 752, 2020.

[30] H. Sandya, P. Hemanth Kumar, and S. K. R. Himanshi Bhudiraja, "Fuzzy rule based feature extraction and classification of time series signal," *International Journal of Soft Computing and Engineering*, vol. 3, no. 2, pp. 2231–2307, 2013.

[31] M. Eklund, U. Norinder, S. Boyer, and L. Carlsson, "Choosing feature selection and learning algorithms in qsar," *Journal of Chemical Information and Modeling*, vol. 54, no. 3, pp. 837–843, 2014.

[32] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.

[33] Nicholson, Chris, "A beginner's guide to important topics in ai, machine learning, and deep learning: Deep autoencoders," https://wiki.pathmind.com/deep-autoencoder.

[34] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke, "The rise of deep learning in drug discovery," *Drug discovery today*, vol. 23, no. 6, pp. 1241–1250, 2018.

[35] Q. Meng, D. Catchpoole, D. Skillicom, and P. J. Kennedy, "Relational autoencoder for feature extraction," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 364–371.

[36] G. Castaneda, P. Morris, and T. M. Khoshgoftaar, "Evaluation of maxout activations in deep learning across several big data domains," *Journal of Big Data*, vol. 6, no. 1, pp. 1–35, 2019.

[37] P. Safayenikoo and I. Akturk, "Weight update skipping: Reducing training time for artificial neural networks," *arXiv preprint arXiv:2012.02792*, 2020.

[38] H. Lee, J. Kim, B. Kim, and S. Kim, "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, 2018, pp. 81–84.

[39] Chablani, Manish, "Autoencoders: Introduction and implementation in tf," https://towardsdatascience.com/autoencoders-introduction-and-implementation-3f40483b0a85.

[40] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in *2014 science and information conference*. IEEE, 2014, pp. 372–378.

[41] A. Sharma and K. K. Paliwal, "Linear discriminant analysis for the small sample size problem: an overview," *International Journal of Machine Learning and Cybernetics*, vol. 6, no. 3, pp. 443–454, 2015.

[42] S. J. Wetzel, "Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders," *Physical Review E*, vol. 96, no. 2, p. 022140, 2017.

[43] S. Jo, W. Sohng, H. Lee, and H. Chung, "Evaluation of an autoencoder as a feature extraction tool for near-infrared spectroscopic discriminant analysis," *Food Chemistry*, vol. 331, p. 127332, 2020.

[44] M. Polic, I. Krajacic, N. Lepora, and M. Orsag, "Convolutional autoencoder for feature extraction in tactile sensing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3671–3678, 2019.

[45] J. G. García, A. Robertsson, J. G. Ortega, and R. Johansson, "Generalized contact force estimator for a robot manipulator," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 4019–4024.

[46] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with svm for network intrusion detection," *IEEE Access*, vol. 6, pp. 52 843–52 856, 2018.

[47] Z. Meng, X. Zhan, J. Li, and Z. Pan, "An enhancement denoising autoencoder for rolling bearing fault diagnosis," *Measurement*, vol. 130, pp. 448–454, 2018.

[48] S. Zavrak and M. Iskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder," *IEEE Access*, vol. 8, pp. 108 346–108 358, 2020.

[49] M. Maggipinto, C. Masiero, A. Beghi, and G. A. Susto, "A convolutional autoencoder approach for feature extraction in virtual metrology," *Procedia Manufacturing*, vol. 17, pp. 126–133, 2018.

[50] Imbalanced-learn, "Imbalanced-learn documentation," https://imbalanced-learn.org/stable/.

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

[52] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd (2017).

[53] S. González, S. García, J. Del Ser, L. Rokach, and F. Herrera, "A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities," *Information Fusion*, vol. 64, pp. 205–237, 2020.

[54] T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Comparing boosting and bagging techniques with noisy and imbalanced data," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 3, pp. 552–568, 2010.

[55] S. Suzuki, T. Yamashita, T. Sakama, T. Arita, N. Yagi, T. Otsuka, H. Semba, H. Kano, S. Matsuno, Y. Kato *et al.*, "Comparison of risk models for mortality and cardiovascular events between machine learning and conventional logistic regression analysis," *PLoS One*, vol. 14, no. 9, p. e0221911, 2019.

[56] T. Hasanin, T. M. Khoshgoftaar, J. L. Leevy, and N. Seliya, "Examining characteristics of predictive models with imbalanced big data," *Journal of Big Data*, vol. 6, no. 1, pp. 1–21, 2019.

[57] Q. Gu, L. Zhu, and Z. Cai, "Evaluation measures of the classification performance of imbalanced data sets," in *International symposium on intelligence computation and applications*. Springer, 2009, pp. 461–471.

[58] M. F. Sohan, M. A. Kabir, M. I. Jabiullah, and S. S. M. M. Rahman, "Revisiting the class imbalance issue in software defect prediction," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*. IEEE, 2019, pp. 1–6.

[59] J. Van Hulse, T. M. Khoshgoftaar, A. Napolitano, and R. Wald, "Feature selection with high-dimensional imbalanced data," in *2009 IEEE International Conference on Data Mining Workshops*. IEEE, 2009, pp. 507–514.

[60] G. R. Iversen, H. Norpoth, and H. P. Norpoth, *Analysis of variance*. Sage, 1987, no. 1.

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