

This is an Author's Original Manuscript of an article whose final and definitive form, the Version of Record, has been published in the *Proceedings of the Third International Conference on Advances in Computing Research (ACR'25)*, available online at: https://link.springer.com/chapter/10.1007/978-3-031-87647-9_25

Performance of Machine Learning Classifiers for Anomaly Detection in Cyber Security Applications

Markus Haug^{[0009-0006-9581-6003]*} and Gissel Velarde^[0000-0001-5392-9540]

IU International University of Applied Sciences, Erfurt, 99084, Germany
markus.haug@iu-study.org, gissel.velarde@iu.org

Abstract. This work empirically evaluates machine learning models on two imbalanced public datasets (KDDCUP99 and Credit Card Fraud 2013). The method includes data preparation, model training, and evaluation, using an 80/20 (train/test) split. Models tested include eXtreme Gradient Boosting (XGB), Multi Layer Perceptron (MLP), Generative Adversarial Network (GAN), Variational Autoencoder (VAE), and Multiple-Objective Generative Adversarial Active Learning (MO-GAAL), with XGB and MLP further combined with Random-Over-Sampling (ROS) and Self-Paced-Ensemble (SPE). Evaluation involves 5-fold cross-validation and imputation techniques (mean, median, and IterativeImputer) with 10, 20, 30, and 50 % missing data. Findings show XGB and MLP outperform generative models. IterativeImputer results are comparable to mean and median, but not recommended for large datasets due to increased complexity and execution time. The code used is publicly available on GitHub (github.com/markushaug/acr-25).

Keywords: Imbalanced Classification, Machine Learning, Deep Learning, Cyber Security, Performance Evaluation, Fraud Detection, Anomaly Detection, Data Imputation

1 Introduction

In this work, we evaluate the performance of several machine learning algorithms for binary classification on tabular data. We compare the performance of supervised and unsupervised learning approaches, and study the case of fraud detection for cyber security. Still, our findings and lessons learned are relevant to any business application and domain where the solution can be framed as a binary classification problem including fraud detection, credit approval, medical diagnosis, or online advertising, to mention a few applications. Indeed, very few business use cases would have balanced data in terms of negative and positive ratios. Generally, the positive class samples appear less frequently than those of

the negative class. However, the positive samples are of high interest when diagnosing sicknesses, identifying fraudulent activities or predicting if a customer will click on an ad.

Binary classification of imbalance data has been a relevant topic for the machine learning community given that it is challenging to learn statistics from the under represented samples [8]. Approaches to imbalanced learning include hyperparameter tuning [19,18,9], sampling the data before training [7,5,18], and ensembling [22,8].

Cyber security applications are becoming increasingly important given that cyber crime is on the rise with cost estimations that reach the 400 billion USD globally [15]. For this experimental study we systematically evaluated several algorithms on fraud detection datasets, one of the applications of cyber security, however, the main take aways and findings of this study, apply to any application that can be framed as a binary classification problem. The motivation to compare supervised learning with unsupervised learning algorithms is justified since fraud patterns might change over time, fraudsters are very creative and try new possibilities to bypass any system, and therefore, some patterns may never appear during training or might be recognised in a larger time window when it is too late [18]. Therefore, an approach that can discriminate between legit and fraud activity without the need of collecting labels, is very appealing.

In our experiments, we focus first on the evaluation of different approaches and techniques to deal with imbalanced learning. Here we compare the performance of tree based and deep learning approaches in supervised and unsupervised fashion, and test the effect of sampling and ensembling. Then, we investigate the robustness of the best classifier when dealing with missing data, a problem that occurs either because certain features might not be available at the time of data collection, either because of the nature of the process or due to a system error. In the next section we describe the method.

2 The Method

The method can be seen in Figure 1. Each dataset is initially preprocessed in a standardized way so that they can be reused for the various models and sampling methods. The respective properties of the datasets can be seen in Table 1. Categorical features are first converted into numerical features and then all numerical features are normalized using a *Standard Scaler* [12]. Model training and evaluation follows a 80/20 (train/test) split. The partitioning is not purely stochastic, but is carried out as part of stratified sampling, so that the respective imbalance ratio (IR) is retained in all created partitions. Models tested include XGB [2], MLP [14], GAN [4], VAE [1], and MO-GAAL [20], with XGB and MLP further combined with ROS and SPE. The experiment was executed in a 5-fold cross-validation on the training set with scikit-learn pipelines [12]. After evaluation, XGB was selected to study the effect of imputation techniques (mean, median, and IterativeImputer) with 10, 20, 30, and 50 % missing data. This last step is not represented in Figure 1. The evaluation is based on the number of True

positive (TP), True negative (TN), False positive (FP), and False negative (FN), considering $Precision = TP/(TP + FP)$, $Recall = TP/(TP + FN)$ and $F_1-Score = 2 \times Recall \times Precision / (Recall + Precision)$. Finally, significance tests are performed between imputation techniques. All experiments were conducted on a 2021 MacBook Pro with the Apple M1 Max and 32 GB of RAM.

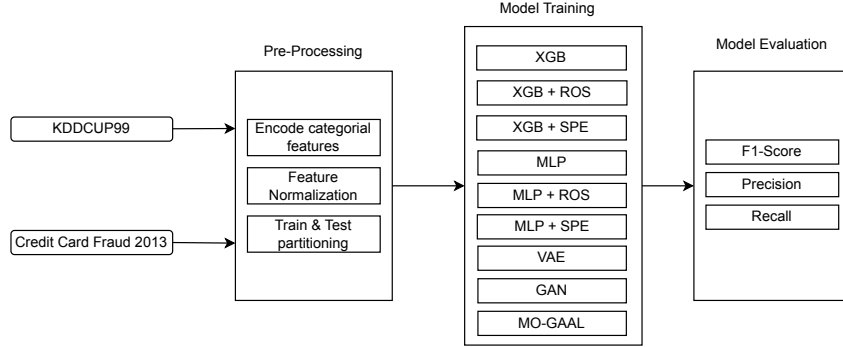


Fig. 1. Graphical representation of the Method. Each dataset goes through a Pre-Processing stage before Model Training and Evaluation. Models are trained and tested independently. XGB and MLP are selected and combined with ROS and SPE.

2.1 Datasets

We used two highly imbalanced datasets *KDDCUP99* and *Credit Card Fraud 2013*, see Table 1. A scikit-learn pipeline was used to initially preprocess the datasets in a standardized way to be reused for the various models and sampling methods. The categorical features were first converted into numerical features and then all numerical features were scaled. Finally, a training and test data partition was created. The partitioning was not purely stochastic, but was carried out as part of stratified sampling, so that the respective IR was retained in all created partitions.

Table 1. Characteristics of the datasets used. Data from [13,17].

Dataset	Negative Samples	Positive Samples	IR	NaN Values	Numerical Features	Categorical Features
Credit Card	284 315	492	577.88	0	29	0
KDDCUP99	60 593	250 436	4.13	0	34	7

2.2 Hyperparameters

Table 2. Hyperparameters found for the Credit Card dataset.

Model	Parameters
XGB (Normal)	Vanilla XGB with default parameters from [21]
XGB + ROS	objective = Vanilla XGB with default parameters from [21]
XGB + SPE	Vanilla XGB with default parameters from [21]
MLP (Normal)	batch_size = 2048, epochs = 50, learning_rate = 0.01, optimizer = Adam, layers = [256, 256, Dropout(0.3), 256, Dropout(0.3), 1], activation = [ReLU, ReLU, ReLU, sigmoid], metrics = [f1, fn, fp, tn, tp, precision, recall], loss = binary_crossentropy
MLP + ROS	MLP (Normal), sampling_strategy = 1
MLP + SPE	estimator = MLP (Normal), n_estimators = 50
VAE	epochs = 10, batch_size = 32, lr=0.001, dropout_rate = 0.2
GAN	epochs = 10, latent_dim = 29, batch_size = 32, d_optimizer = SGD with learning rate = 0.0002, g_optimizer = Adam with learning rate = 0.00001 & beta_1 = 0.5
MO-GAAL	contamination = Proportion of fraud cases in training partition, n_sub_generators = 3, learning rate discriminator = 0.01, learning rate generator = 0.0001, epochs = 2

Table 3. Hyperparameters found for the KDDCUP99 dataset.

Model	Parameters
XGB (Normal)	objective = binary:logistic, booster = gbtrees, colsample_bytree = 0.7, device = cpu, eval_metric = aucpr, gamma = 0.3, learning_rate = 0.3, max_depth = 6, max_leaves = 64, n_estimators = 1000
XGB + ROS	Vanilla XGB with default parameters from [21], sampling_strategy = 1
XGB + SPE	Vanilla XGB with default parameters from [21], n_estimators (SPE) = 50
MLP (Normal)	epochs = 50, batch_size = 2048, learning_rate = 0.02, optimizer = Adam, layers = [128, 64, 1], activation = [relu, tanh, sigmoid], metrics = [f1, recall, precision], loss = binary_crossentropy
MLP + ROS	estimator = MLP (Normal), sampling_strategy = 1
MLP + SPE	estimator = MLP (Normal), n_estimators = 50
VAE	latent_dim = 2, batch_size = 32, learning_rate = 0.001, KL_beta = 1.0, encoder_neuron_list=[128, 64, 32], decoder_neuron_list=[32, 64, 128], activation = ReLU, dropout_rate = 0.2
GAN	epochs = 10, latent_dim = 121, batch_size = 32, d_optimizer = Adam with learning rate = 0.0001 & beta_1 = 0.5, g_optimizer = Adam with learning rate = 0.0001 & beta_1 = 0.5 & clipvalue = 1.0
MO-GAAL	contamination = Proportion of fraud cases in training partition, n_sub_generators = 3, learning rate discriminator = 0.01, learning rate generator = 0.0001, epochs = 2

Tables 2 and 3 present the nine setups per dataset, which include XGB, MLP, VAE, GAN, MO-GAAL, XGB and MLP with ROS and SPE. For XGB, we tested different setups. First, a vanilla XGB model with standard parameters [21]. Second, a XGB model was fine-tuned using *RandomSearchCV* across parameters such as *n_estimators*, *max_leaves*, *learning_rate*, *gamma*, *max_depth*, *subsample*, *reg_alpha*, *reg_lambda*, *scale_pos_weight*, with RandomSearch yielding the most promising results on the KDD dataset. These tuned hyperparameters were then compared to the performance of the vanilla XGB model, XGB + ROS and XGB + SPE, and the best performing model was selected for further evaluation.

The setup for the MLPs was taken from [3]. Furthermore, hyperparameter tuning based on scikit-learn’s *RandomSearchCV* was performed for the KDD-CUP99 dataset. VAE was trained in an unsupervised fashion on the negative classes to learn the underlying probability distribution, following [1]. GAN was implemented following [16]. GAN and VAE were not combined to ROS or SPE, as these were trained using the information of the negative class only. MO-GAAL was trained using the parameters from [6], whereby *n_sub_generators* was reduced to 3 and the number of epochs to 2, as already [6] reported large execution time. The python library *PyOD* was used to implement VAE and MO-GAAL [20].

3 Results

The models’ performance can be seen in Figures 2 and 3 and their training times in Figure 4. Figures 2 and 3 show that XGB outperforms all models and its performance is similar on both datasets. MLP is the second best. MO-GAAL is able to learn and detect anomalies on KDDCUP but not on Credit Card Dataset. VAE and GAN show poorer generalization performance for both datasets. As seen in Figure 4, it is clear that training on KDDCUP takes longer time than that taken on Credit Card for all models, being XGB the most efficient and MO-GAAL the least.

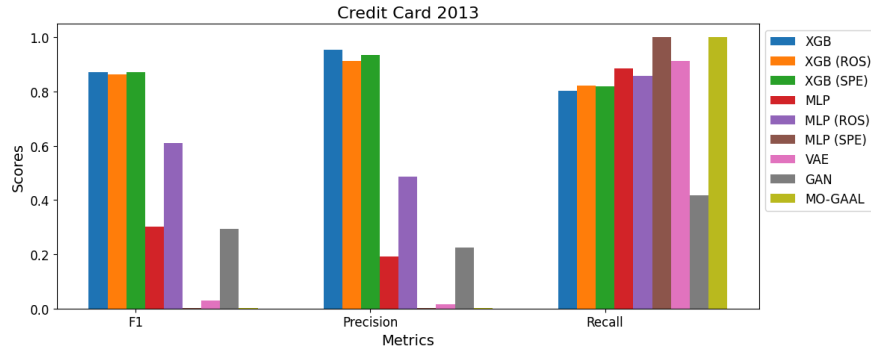


Fig. 2. Models’ performance on the credit card dataset, based on 5-fold cross-validation on the training data. For XGB and MLP, only the best sampling combinations are shown.

Furthermore, we observe that the combination of MLP + ROS led to an improvement in anomaly detection for both datasets. Particularly in the case of the credit card dataset, large improvements were achieved by oversampling the positive class.

Finally, since XGB performed best for both datasets, it was used to study the effect of missing data with random deletion of 10, 20, 30, and 50 % of

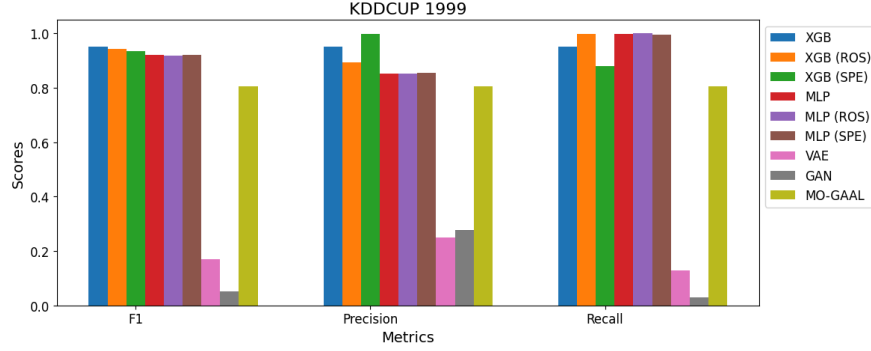


Fig. 3. Models' performance on the KDDCUP99 dataset, based on 5-fold cross-validation on the training data. For XGB and MLP, only the best sampling combinations are shown.

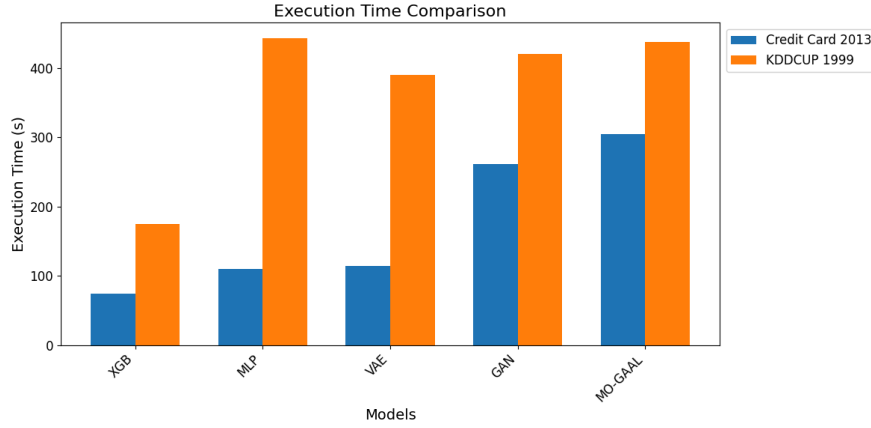


Fig. 4. Training times in seconds (Execution Time). XGB proves to be the most efficient model, while MO-GAAL stands out due to its very long training time. For XGB and MLP, only the best sampling combinations are shown.

the data, see results in Figure 5. For the Credit Card dataset, XGB + SPE and for the KDDCUP99 dataset, XGB without any sampling methods were the baseline models (see Tables 2 and 3). Three imputation methods (Mean, Median, IterativeImputer) were evaluated in ten independent runs to avoid coincidences. No significant differences between the imputation methods on both datasets ($p > 0.05, n = 5$) were found.

We also found enormously high execution times for the *IterativeImputer* for the KDDCUP99 dataset.

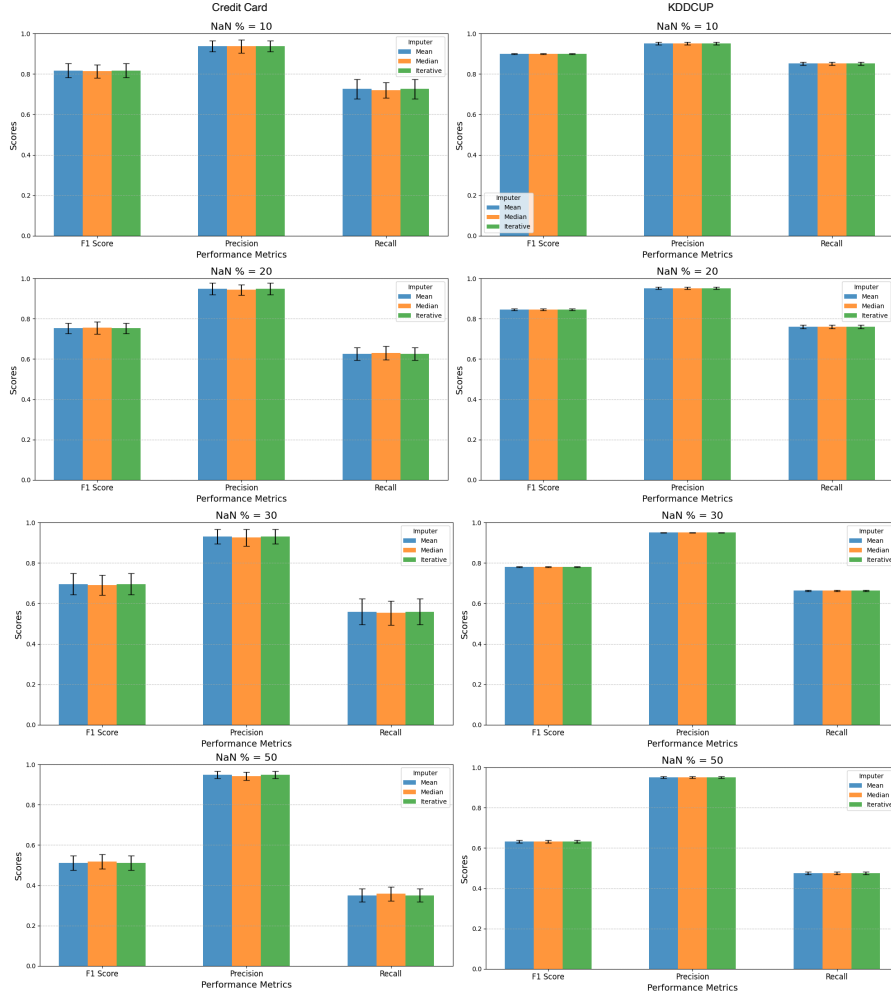


Fig. 5. XGB performance with missing data using imputation techniques on credit card and KDDCUP dataset. The error bars represent the standard deviations of the measurements, each of which was performed ten times.

4 Discussion

This section discusses the main findings.

4.1 On the general performance

In preliminary experiments we observed other metrics such as Matthews Correlation Coefficient (MCC) and Geometric Mean, and noticed that F1, Precision,

and Recall are the most relevant to drive conclusions. The results show that XGB and MLP combined to sampling or ensembling achieved the best performance for both datasets. The results confirm the great popularity of XGB in the machine learning community for tabular data. In recent years, the use of XGB has led to many wins in machine learning competitions, with MLPs just behind XGB in terms of popularity [2, 1].

ROS and MLP on the credit card dataset produced a jump in *precision*, but this was at the expense of deteriorating *recall*. Exactly the opposite could be observed with Random-Under-Sampling (RUS) in [6] in combination with XGB and other models. SPE applied to MLP had a negative effect in performance on the credit card dataset. [10] evaluated the performance of MLPs in combination with several sampling methods, including RUS.

We therefore recommended to evaluate several techniques for coping with imbalanced data for each dataset. However, it must also be mentioned that due to the limited time frame of this research, no in-depth experiments with different hyperparameters for SPE were conducted. Besides, future work could investigate classifier performance on different imbalance ratios as it was done in [18].

4.2 On generative models

The generative models (VAE, GAN, MO-GAAL) achieved significantly poorer results compared to XGB and MLP. This could be due to their complexity. In particular, the optimization of the hyperparameters for generative models proved to be more difficult, as several MLPs with different objectives were often trained here and the implementation and execution of the optimization for XGB and a simple MLP were significantly simpler. Our results are similar to those obtained by [6] on another dataset called *PaySim* dataset [11].

4.3 On imputation methods

The investigation of the three imputation methods showed that for the *Credit-Card* dataset, the *IterativeImputer* outperforms *Mean* and *Median* imputation methods on 10 % and 20 % of missing data. On the KDDCUP99 dataset, there is no difference on whichever method is used for imputation, except for the execution time. *IterativeImputer* poses an enormous processing time, which is particularly evident in the KDDCUP99 dataset.

The *IterativeImputer* complexity increases with an increasing number of features (columns) and size of the dataset (rows) and impairs the practicability of the method. The complexity \mathcal{O} is defined as $\mathcal{O}(knp^3 \min(n, p))$, where k is the number of maximum iterations, n is the number of entries in the dataset and p is the number of features. It is important to note at this point that the *IterativeImputer* can be adjusted by parameters to counteract a high runtime [12]. In this case, however, these parameters were not optimized as this was not possible due to time constraints for this project.

5 Conclusion

This work contributes to a better understanding of the problem of imbalance classification, considering a systematic evaluation of supervised and unsupervised learning models. We demonstrate on two representative datasets for cybersecurity that supervised learning algorithms outperform unsupervised learning approaches. Moreover, depending on the model and the dataset, some sampling and ensembling techniques can help improve recognition. But these should be used carefully depending on the performance. We observed that Random Oversampling and Self-Paced Ensembling should be tested on each dataset and model before its application. Interestingly, there was no difference on the imputation methods tested (IterativeImputer, mean and median), except that IterativeImputer is not recommended for large datasets due to increased complexity and execution time.

6 Author Contributions

M. H. software implementation, experimental design, data analysis, initial paper draft including figures and tables, paper writing and revisions. G.V. supervision, experimental design, paper writing and revisions.

References

1. Bank, D., Koenigstein, N., Giryas, R.: Autoencoders (2021). DOI 10.48550/arXiv.2003.05991
2. 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, pp. 785–794 (2016). DOI 10.1145/2939672.2939785
3. Chollet, F.: Imbalanced classification: Credit card fraud detection. https://keras.io/examples/structured_data/imbalanced_classification/ (2019)
4. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative Adversarial Nets. In: Neural Information Processing Systems (2014)
5. Hajek, P., Abedin, M.Z., Sivarajah, U.: Fraud detection in mobile payment systems using an xgboost-based framework. *Information Systems Frontiers* pp. 1–19 (2022)
6. Hajek, P., Abedin, M.Z., Sivarajah, U.: Fraud Detection in Mobile Payment Systems using an XGBoost-based Framework. *Information Systems Frontiers* **25**(5), 1985–2003 (2023). DOI 10.1007/s10796-022-10346-6
7. Kim, M., Hwang, K.B.: An empirical evaluation of sampling methods for the classification of imbalanced data. *PLoS ONE* **17**(7) (2022). <https://doi.org/10.1371/journal.pone.0271260>
8. Lemaitre, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research* **18**(17), 1–5 (2017). URL <http://jmlr.org/papers/v18/16-365.html>

9. Li, Y., Jin, J., Ma, J., Zhu, F., Jin, B., Liang, J., Chen, C.P.: Imbalanced least squares regression with adaptive weight learning. *Information Sciences* **648**, 119,541 (2023)
10. Liu, Z., Cao, W., Gao, Z., Bian, J., Chen, H., Chang, Y., Liu, T.Y.: Self-paced Ensemble for Highly Imbalanced Massive Data Classification. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp. 841–852 (2020). DOI 10.1109/ICDE48307.2020.00078
11. Lopez-Rojas, E.A., Elmir, A., Axelsson, S.: PaySim: A financial mobile money simulator for fraud detection. In: The 28th European Modeling and Simulation Symposium-EMSS. Larnaca, Cyprus (2016)
12. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011)
13. Pozzolo, A.D., Caelen, O., Johnson, R.A., Bontempi, G.: Calibrating Probability with Undersampling for Unbalanced Classification. In: 2015 IEEE Symposium Series on Computational Intelligence, pp. 159–166 (2015). DOI 10.1109/SSCI.2015.33
14. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. *Nature* **323**(6088), 533–536 (1986). DOI 10.1038/323533a0
15. Sarker, I.H., Kayes, A., Badsha, S., Alqahtani, H., Watters, P., Ng, A.: Cybersecurity data science: an overview from machine learning perspective. *Journal of Big data* **7**, 1–29 (2020)
16. Sayak, P.: Conditional GAN. https://keras.io/examples/generative/conditional_gan/ (2021)
17. The UCI KDD Archive: KDD Cup 1999 Data. <https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html> (1999)
18. Velarde, G., Weichert, M., Deshmunkh, A., Deshmane, S., Sudhir, A., Sharma, K., Joshi, V.: Tree boosting methods for balanced and imbalanced classification and their robustness over time in risk assessment. *Intelligent Systems with Applications* **22**, 200,354 (2024). DOI <https://doi.org/10.1016/j.iswa.2024.200354>. URL <https://www.sciencedirect.com/science/article/pii/S2667305324000309>
19. Wang, C., Deng, C., Wang, S.: Imbalance-xgboost: leveraging weighted and focal losses for binary label-imbalanced classification with xgboost. *Pattern Recognition Letters* **136**, 190–197 (2020)
20. Winston Li: Source code for pyod.models.mo_gaal. https://pyod.readthedocs.io/en/latest/_modules/pyod/models/mo_gaal.html (2022)
21. xgboost developers: XGBoost Parameters (2022). URL <https://xgboost.readthedocs.io/en/stable/parameter.html>
22. Yang, K., Yu, Z., Chen, C.P., Cao, W., Wong, H.S., You, J., Han, G.: Progressive hybrid classifier ensemble for imbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **52**(4), 2464–2478 (2021)