Submitted to *INFORMS Journal on Computing*
manuscript JOC-2021-04-OA-076

Black-box Attack-Based Security Evaluation Framework for Credit Card Fraud Detection Models

Jin Xiao, Yuhang Tian
Business School, Sichuan University, Chengdu 610064, China, xiaojin@scu.edu.cn, tyh70537@outlook.com

Yanlin Jia
School of Sciences, Southwest Petroleum University, Chengdu 610500, China, yelinjyl@126.com

Xiaoyi Jiang
Faculty of Mathematics and Computer Science, University of Münster, Münster D-48149, Germany, xjiang@uni-muenster.de

Lean Yu*
Business School, Sichuan University, Chengdu 610064, China, yulean@amss.ac.cn

Shouyang Wang*
School of Entrepreneurship and Management, ShanghaiTech University, Shanghai 201210, China, sywang@amss.ac.cn

Key words: nonlinear optimization; credit card fraud detection models; security evaluation; black-box attack; adversarial examples; machine learning

History:

Appendix A: Semi-supervised Learning Techniques

A.1. Co-Forest

The Co-Forest algorithm first trains a random forest model that contains T decision tree base classifiers, $h_1,...,h_T$, on the initial class labeled dataset L, which is denoted as H^*. For each base classifier $h_i, i=1,...,T$, the $T-1$ base classifiers other than h_i in H^* form the concomitant ensemble of h_i, which is denoted by H_i. The Co-Forest iteratively labels the unlabeled samples in U and uses these samples to update each base classifier. For each base classifier h_i in every iteration, the concomitant ensemble H_i provides the labeling confidence for all samples in U and selects samples with confidences greater than the threshold θ for labeling. Let L_i denote the set of newly labeled samples, and the set $L \cup L_i$ is used to train a candidate model $h_i^{'}$. If the utility of $h_i^{'}$ is greater than that of h_i, replace h_i with $h_i^{'}$; otherwise, keep h_i unchanged in this iteration. This iteration process is repeated until none of the base classifiers changes.

A.2. FlexMatch

In order to make the neural network classifier more suitable for semi-supervised classification tasks, FlexMatch defines a dynamic loss function $\mathcal{L}_t$ for the neural network classifier based on the continuity assumption. The $\mathcal{L}_t$ in the t-th iteration consists of two cross-entropy loss terms: a fixed supervised loss $\mathcal{L}_s$ applied to labeled dataset L and an unsupervised loss $\mathcal{L}_{u,t}$ (varies with iteration) applied to unlabeled dataset U. Specifically, $\mathcal{L}_s$ is just the standard cross-entropy loss on weakly augmented labeled dataset:

$$\mathcal{L}_s = \mathbb{E}_{x,y\in L}[H(y, p(\alpha(x)))], \tag{A1}$$

where H is the standard cross entropy function, $\alpha(\cdot)$ is a weak augmentation function, and $p(\cdot)$ represents the predicted class distribution produced by the classifier for the input. The unsupervised loss $\mathcal{L}_{u,t}$ is defined as:

$$\mathcal{L}_{u,t} = \mathbb{E}_{x\in U}[I(max(p(\alpha(x))) > \mathcal{F}_t(q))H(q, p(\Omega(x)))], \tag{A2}$$

where $q = \arg\max(p(\alpha(x)))$ denotes the pseudo-label assigned to x by the classifier, $\mathcal{F}_t(q)$ is a threshold that changes dynamically with q, and $\Omega(\cdot)$ is a strong augmentation function. Based on the continuity assumption, $L_{u,t}$ promotes the classifier to be as consistent as possible in the classification results on weakly-augmented and strongly-augmented samples. Finally, the total loss is formulated as the weighted combination (by μ) of supervised and unsupervised loss:

$$\mathcal{L}_t = \mathcal{L}_s + \mu\mathcal{L}_{u,t}. \tag{A3}$$

According to loss function $\mathcal{L}_t$, the neural network classifier is trained until the preset maximum number of iterations is reached. See (Zhang et al. 2021) for more details on the two augmentation functions and threshold function $\mathcal{F}_t(q)$. In this paper, the classifier trained by FlexMatch is used to classify and label the samples in U. The samples with confidence greater than θ are merged with the initial labeled dataset L and then used to train the substitute models.

Appendix B: Detailed Derivation of the Method of Lagrange Multipliers

First, introduce a slack variable η and rewrite problem (6) in the main text into the following form:

$$\begin{gathered} x^* = \arg\min_x w^T x + b \\ s.t.\ (x^+ - x)^T (x^+ - x) - \rho^2 + \eta^2 = 0. \end{gathered} \tag{A4}$$

Second, the Lagrangian function of Equation (A4) is constructed as follows:

$$F(x, \lambda, \eta) = w^T x + b + \lambda((x^+ - x)^T (x^+ - x) - \rho^2 + \eta^2), \tag{A5}$$

where λ is the Lagrange multiplier. Third, the partial derivatives of Equation (A5) are constructed with respect to x, λ, and η, which are set to zero. Thus, we get:

$$\frac{\partial F}{\partial x} = w + 2\lambda(x - x^{+}) = 0, \tag{A6}$$

$$\frac{\partial F}{\partial \lambda} = (x^{+} - x)^{T}(x^{+} - x) - \rho^2 + \eta^2 = 0, \tag{A7}$$

$$\frac{\partial F}{\partial \eta} = 2\lambda\eta = 0. \tag{A8}$$

From Equation (A8), we know that $\lambda = 0$ or $\eta = 0$. However, when $\lambda = 0$, substituting into Equation (A6) results in $w = 0$; which contradicts the condition that w is not equal to 0. Therefore, the only solution is $\eta = 0$. Finally, the simultaneous Equations (A6)–(A8) can be solved as:

$$x^{*} = x^{+} - \frac{\rho}{\sqrt{w^T w}} w, \tag{A9}$$

$$\lambda = \frac{\sqrt{w^T w}}{2\rho}, \tag{A10}$$

$$\eta = 0. \tag{A11}$$

Appendix C: Experimental Setup

C.1. Handling Imbalanced Dataset

The class distribution of the first dataset Vesta credit is imbalanced. In this case, banks usually apply resampling techniques to balance the class distribution before training the machine learning-based CCFD models. Standard resampling techniques include over- and under-sampling (Van Vlasselaer et al. 2016). To prevent model overfitting and to reduce the demand for computing resources from over-sampling (Mead et al. 2018), we apply the random under-sampling method to balance the class distribution of the target model training set D_{tr}. The sampling ratio is set to 10 (normal : fraudulent transaction samples), which is the ideal ratio to train the CCFD models (Chen and Wasikowski 2008). Likewise, to ensure that the performance of the attack algorithms is not compromised by the imbalanced class distribution, we balance the distributions of the initial class labeled dataset L of semi-supervised learning and the training set L_{aug} for the substitute model C in the same manner. It is worth mentioning that the original class imbalance ratio of Lending club credit dataset is already lower than 10, and the random under-sampling cannot be performed, so we keep the original class distribution unchanged.

C.2. Time series cross-validation

As a variant of standard k-fold cross-validation, k-fold time series cross-validation first divides the entire dataset into $k+1$ folds according to time sequence. Then, in the i-th $(i = 1, 2, ..., k)$ validation, the first i folds are used to train the CCFD model O and optimize its hyperparameters (the last 1000 samples are the validation set D_{va}, and the remaining samples are the training set D_{tr}), the $(i+1)$-th fold is used as the test set D_{te} to evaluate the security of O. Finally, it calculates the mean value of k validation results. To obtain more reliable experimental results, we repeat the above cross-validation ten times and take the averages as the final computational results. All algorithms are coded in Python 3.7 and implemented on a PC with an Intel Core i7 CPU at 2.81 GHz.

C.3. Parameter Setting of the Attack Algorithms

For the STBA algorithms proposed in this paper, the key parameters include the number of samples in D'_{tr} (τ), the number of queried samples in D'_{tr} (δ), the labeling threshold (θ), and the number of modifiable features from the original features (β). First, as the number of unlabeled samples that the fraudsters can collect may be very limited, we let $\tau = 1000$ in this paper. Next, to illustrate that the Linear-STBA and RBF-STBA can achieve satisfactory attack performance when δ is relatively small, we let $\delta = 40$, and it is not difficult for the fraudsters to conduct 40 queries to the target model. Then, we find through repeated experiments that the performance of the attack algorithms is the strongest at $\theta = 0.75$. Finally, we let $\beta = 10$, which means that the fraudsters can only modify the fraudulent samples on 10 specified features. The 10 features are selected from the original feature set according to the advice of domain experts and are easy to be manipulated by the fraudsters. For the Vesta credit dataset, the 10 features are related to credit card transaction behavior and include “transaction amount”, “transaction location”, “transaction frequency”, etc. For the Lending club credit dataset, the 10 features include “loan amount”, “the number of payments on the loan”, “self-reported annual income”, etc. For the semi-supervised learning algorithm Co-Forest, we set the number of base classifiers T to 6 (Li and Zhou 2007). For the semi-supervised learning algorithm FlexMatch, we determine the structure of the neural network classifier according to suggestion of Khosrojerdi et al. (2016). The weight factor μ is set to the default of 1, and the maximum number of iterations is set to 50000 to ensure that the algorithm can converge. It is noted that the compared algorithms Linear-classic and RBF-classic also include the parameters τ, δ, and β. To ensure a fair comparison, we use the same parameter setting as the STBA algorithms. In addition, as the fraudsters cannot use the validation set D_{va} to optimize the parameters of the substitute models and semi-supervised learning algorithms. For the substitute models Linear-SVM and RBF-SVM used by our proposed and traditional attack algorithms, we adopt the default parameters in Scikit-Learn[1].

C.4. Parameter Setting of the Target Models

When the maximum number of iterations exceeds 500 for LR, the classification performance no longer improves; thus, we set it to 500. For DT, we employ the classical classification and regression tree (CART) method (Breiman et al. 1984). The classification performance of the model is optimal when the minimum

[1] https://scikit-learn.org/.

number of samples required to split the internal nodes is 2 and the minimum number of samples required for the leaf nodes is 1. As CART is the base classifier of XGBoost, choosing it as one of the target models is beneficial to compare the security differences between single and ensemble learning models. We employ LIBSVM[2] to implement the classic Linear-SVM and RBF-SVM. The kernel parameter γ of RBF-SVM is set to $1/m$ by default, where m is the number of features in the dataset. In addition, we select the optimal penalty coefficient ϵ for the two SVMs from the set $\{1,\ 10,\ 100,\ 1000\}$ (Xiao et al. 2020). It is worth mentioning that the fitting time complexity of RBF-SVM is more than quadratic with the number of samples. Since the number of training samples for both experimental datasets in the 5-th cross-validation exceeds 180,000, we cannot directly train RBF-SVM due to the limitation of computing resources. Therefore, on the premise of not severely affecting the classification performance of the model, only 10,000 samples randomly selected from the training set are used in each training of RBF-SVM. For XGBoost, when the number of base classifiers exceeds 100, its classification performance no longer improves; thus, we set it to 100. For DNN, we apply the classical feedforward neural network (Smith et al. 2009) with the ReLU activation function, and three hidden layers are used for its structure. The range for the number of neurons in each hidden layer is [1, 20] and the step size is 1 (Khosrojerdi et al. 2016). For IF, we find that satisfactory classification performance can be achieved when the number of isolation trees is 100 and the training samples of each isolation tree are 500. DAE consists of a compression network and an estimation network. We find that the performance of DAE is the best when the structures of the two neural networks are as follows: The compression network contains three hidden layers, the number of neurons in each layer is 16, 8 and 16, respectively, and the activation function is ReLU; the estimation network contains a hidden layer of 8 neurons, and the activation function is ReLU.

Appendix D: Sensitivity Analyses of Parameters

In the security evaluation experiments of the CCFD models based on machine learning, the use of an attack algorithm with a weak attack performance leads to an overestimation of the model security. Therefore, it is necessary to conduct sensitivity analyses of the parameters to reasonably set the parameters of the attack algorithms. In this section, the Linear-STBA (Co-Forest) is used as an example to study the effects of the model parameter selection on the performance of attack algorithms.

The Linear-STBA (Co-Forest) algorithm contains six primary parameters: the number of samples in $D_{tr}^{'}$ (τ), the number of queried samples in $D_{tr}^{'}$ (δ), the labeling threshold (θ), the number of base classifiers (T), the number of modifiable features in the original features (β), and the attack strength (ρ). The attack performance of the proposed algorithms is analyzed in detail for different attack strengths, and the parameter (β) can only be a small number; thus, we only perform sensitivity analyses on the remaining parameters. As there are multiple parameters, before analyzing a particular one, the values of all other parameters are fixed based on the parameter setting in Appendix C.2. Consider the parameter τ as an example. We fix the other parameters and allow τ to vary. Then, the SEIs of the eight considered target models are computed individually. Finally, the average value of SEIs is calculated, which is denoted by u. A lower u indicates a stronger overall attack performance of the Linear-STBA (Co-Forest).

[2] https://www.csie.ntu.edu.tw/ cjlin/libsvm/.

Table A1 SEIs of Eight Target Models when the Number of Unlabeled Samples Collected by the Fraudsters τ Takes Different Values

Target model	τ=200	τ=400	τ=600	τ=1000	τ=5000
Vesta credit dataset					
LR	0.0844	0.0844	0.0839	0.0834	0.0834
DT	0.3044	0.3043	0.3042	0.3032	0.3030
Linear-SVM	0.0687	0.0685	0.0683	0.0678	0.0677
RBF-SVM	0.0688	0.0687	0.0683	0.0673	0.0670
XGBoost	0.2348	0.2348	0.2347	0.2347	0.2346
DNN	0.1377	0.1375	0.1372	0.1367	0.1367
IF	0.6200	0.6200	0.6199	0.6204	0.6201
DAE	0.1845	0.1842	0.1841	0.1841	0.1841
u	0.2129	0.2128	0.2126	0.2122	0.2121
Lending club credit dataset					
LR	0.0656	0.0656	0.0652	0.0647	0.0644
DT	0.1057	0.1057	0.1055	0.1050	0.1047
Linear-SVM	0.0732	0.0728	0.0727	0.0717	0.0718
RBF-SVM	0.0851	0.0847	0.0843	0.0833	0.0834
XGBoost	0.0739	0.0737	0.0737	0.0737	0.0734
DNN	0.0872	0.0869	0.0867	0.0862	0.0861
IF	0.5658	0.5656	0.5653	0.5643	0.5643
DAE	0.7014	0.7014	0.7010	0.7000	0.7001
u	0.2198	0.2195	0.2193	0.2186	0.2185

D.1. Effects of Parameter τ on the Attack Algorithm Performance

In STBA, τ represents the number of unlabeled samples collected by the fraudsters. Considering that the fraudsters may only collect a few unlabeled samples, we mainly investigate changes in the attack performance for the Linear-STBA (Co-Forest) algorithm at $\tau = 200, 400, 600, 1000$, and 5000. Table A1 and Figure A1 show the changes in the SEIs of the eight target models and their mean value u as the parameter τ increases, respectively. It can be seen from Table A1 and Figure A1 that with the increase of τ, the fraudsters have more unlabeled samples for semi-supervised learning, so the attack performance of Linear-STBA (Co-Forest) has also become stronger. Considering both the attack performance and the cost of obtaining unlabeled samples, it is reasonable to set $\tau = 1000$.

D.2. Effects of Parameter δ on the Attack Algorithm Performance

In the proposed STBA, δ, which is the number of queried samples in $D_{tr}^{'}$, is associated with the cost of fraud. A larger δ indicates more query time is required by the fraudsters and the risk of being caught by the bank increases. To study the effects of δ on the attack performance of the Linear-STBA (Co-Forest) algorithm, we conduct experiments in the following steps. First, set δ to six different values (10, 20, 40, 60, 80, and 100) to construct the training set. Second, train the substitute model on the constructed training set. Third, generate the adversarial example set based on the substitute model and calculate the average u of the SEIs for the eight target models.

Table A2 shows the SEIs of eight target models when δ takes different values on the two datasets, where the final row on each dataset shows the average u of the SEIs. Figure A2 displays the trend of u as the parameter δ increases. We note from Table A2 and Figure A2 that as δ increases, the overall performance of the Linear-STBA (Co-Forest) attack algorithm gradually increases. In particular, when δ increases from 10

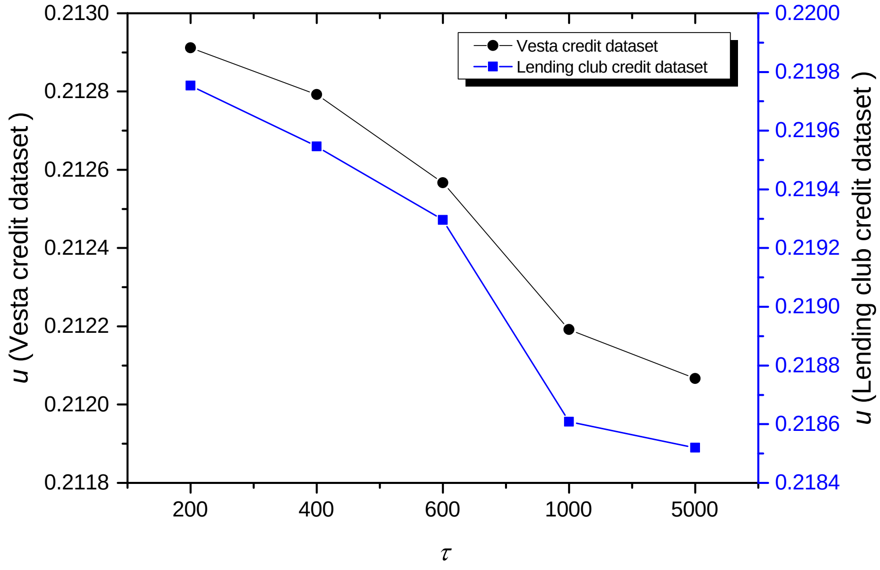


Figure A1 Trend of u as the Number of Unlabeled Samples Collected by the Fraudsters τ Increases on Two Experimental Datasets

Table A2 SEIs of Eight Target Models when the Number of Queried Samples δ in $D_{tr}^{'}$ Takes Different Values

Target model	δ=10	δ=20	δ=40	δ=60	δ=80	δ=100
			Vesta credit dataset			
LR	0.0864	0.0854	0.0834	0.0834	0.0831	0.0826
DT	0.3032	0.3032	0.3032	0.3032	0.3032	0.3022
Linear-SVM	0.0708	0.0698	0.0678	0.0678	0.0681	0.0681
RBF-SVM	0.0663	0.0653	0.0673	0.0665	0.0668	0.0658
XGBoost	0.2327	0.2327	0.2347	0.2347	0.2344	0.2334
DNN	0.1417	0.1407	0.1367	0.1359	0.1362	0.1367
IF	0.6214	0.6204	0.6204	0.6200	0.6200	0.6200
DAE	0.1851	0.1861	0.1841	0.1845	0.1845	0.1850
u	0.2134	0.2129	0.2122	0.2120	0.2120	0.2117
			Lending club credit dataset			
LR	0.0677	0.0687	0.0647	0.0647	0.0642	0.0632
DT	0.1050	0.1030	0.1050	0.1050	0.1040	0.1035
Linear-SVM	0.0737	0.0717	0.0717	0.0712	0.0707	0.0702
RBF-SVM	0.0873	0.0853	0.0833	0.0823	0.0823	0.0818
XGBoost	0.0777	0.0757	0.0737	0.0737	0.0742	0.0742
DNN	0.0862	0.0842	0.0862	0.0867	0.0862	0.0852
IF	0.5683	0.5663	0.5643	0.5648	0.5643	0.5643
DAE	0.6990	0.7000	0.7000	0.6990	0.6985	0.6985
u	0.2206	0.2194	0.2186	0.2184	0.2180	0.2176

to 40, the performance of the attack algorithm improves the most. Considering the high query cost of the fraudsters, it is reasonable to set $\delta = 40$.

D.3. Effects of Parameter θ on the Attack Algorithm Performance

In the STBA, the unlabeled samples with a labeling confidence greater than θ are labeled by the concomitant ensemble and used for subsequent iterations in the model training. For binary classification problems, $\theta \geq 0.5$

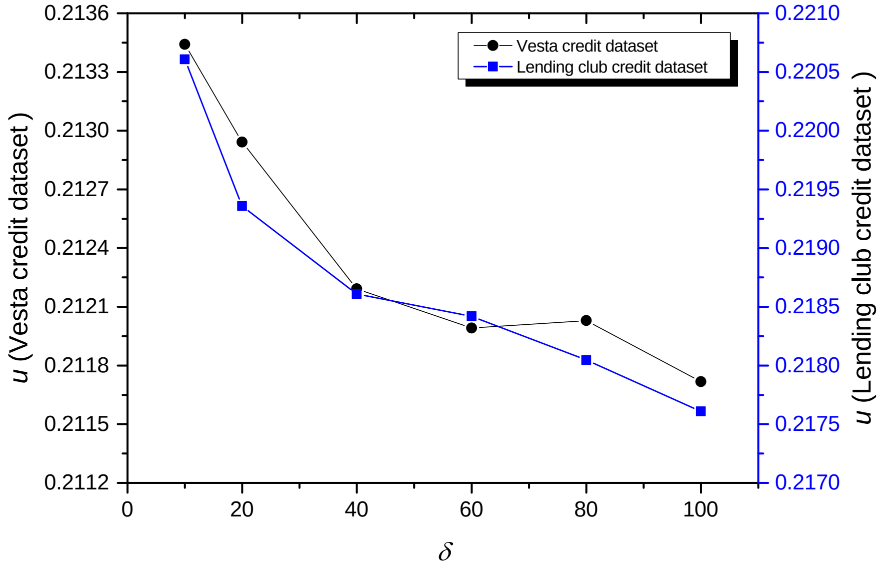


Figure A2 Trend of u as the Number of Queried Samples δ in D'_{tr} Increases on Two Experimental Datasets

(Li and Zhou 2007). Therefore, we mainly investigate changes in the attack performance for the Linear-STBA (Co-Forest) algorithm at $\theta = 0.55, 0.65, ...,$ and 0.95. Table A3 and Figure A3 show the changes in the SEIs of the eight target models and their mean value u as the parameter θ increases, respectively. From Table A3 and Figure A3, we find that when $\theta = 0.75$, the value of u is the smallest. Therefore, it is reasonable to set $\theta = 0.75$.

Table A3 SEIs of Eight Target Models when the Labeling Threshold θ Takes Different Values

Target model	θ=0.55	θ=0.65	θ=0.75	θ=0.85	θ=0.95
Vesta credit dataset					
LR	0.0933	0.0884	0.0834	0.0910	0.0868
DT	0.3078	0.2982	0.3032	0.3065	0.3062
Linear-SVM	0.0768	0.0628	0.0678	0.0703	0.0714
RBF-SVM	0.0687	0.0723	0.0673	0.0727	0.0737
XGBoost	0.2421	0.2397	0.2347	0.2378	0.2429
DNN	0.1385	0.1317	0.1367	0.1394	0.1433
IF	0.6209	0.6204	0.6204	0.6292	0.6260
DAE	0.1894	0.1941	0.1841	0.1878	0.1912
u	0.2172	0.2134	0.2122	0.2168	0.2177
Lending club credit dataset					
LR	0.0684	0.0597	0.0647	0.0691	0.0730
DT	0.1144	0.1150	0.1050	0.1140	0.1067
Linear-SVM	0.0763	0.0767	0.0717	0.0794	0.0722
RBF-SVM	0.0879	0.0833	0.0833	0.0886	0.0894
XGBoost	0.0773	0.0687	0.0737	0.0780	0.0788
DNN	0.0949	0.0962	0.0862	0.0941	0.0908
IF	0.5680	0.5743	0.5643	0.5698	0.5696
DAE	0.7078	0.7100	0.7000	0.7100	0.7008
u	0.2244	0.2230	0.2186	0.2254	0.2227

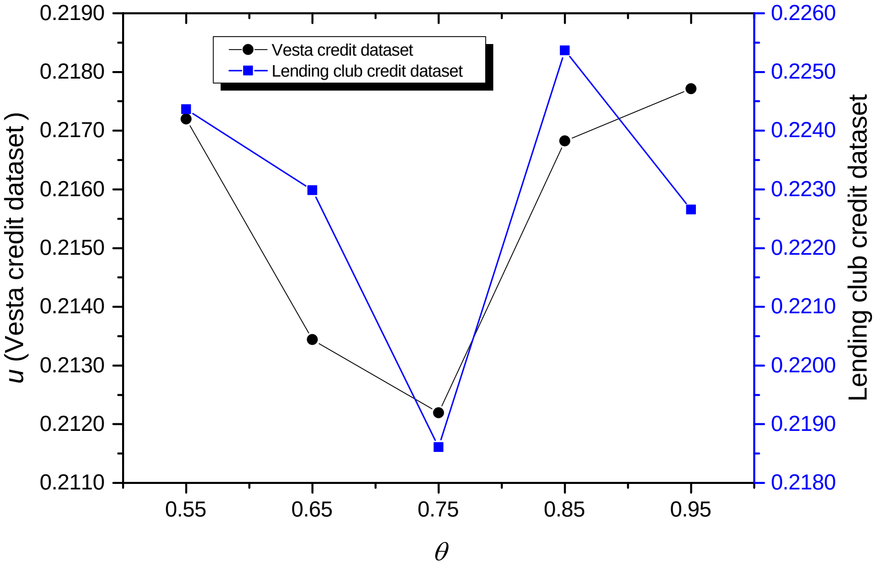


Figure A3 **Trend of u as the Labeling Threshold θ Increases on Two Experimental Datasets**

Table A4 **SEIs of Eight Target Models when the Number of Base Classifiers T Takes Different Values**

Target model	T=4	T=6	T=8	T=10	T=12
			Vesta credit dataset		
LR	0.0966	0.0834	0.0934	0.0865	0.0865
DT	0.3198	0.3032	0.3032	0.3082	0.3118
Linear-SVM	0.0789	0.0678	0.0678	0.0728	0.0758
RBF-SVM	0.0843	0.0673	0.0623	0.0753	0.0692
XGBoost	0.2504	0.2347	0.2397	0.2355	0.2371
DNN	0.1431	0.1367	0.1317	0.1409	0.1433
IF	0.6271	0.6204	0.6204	0.6303	0.6241
DAE	0.1952	0.1841	0.1941	0.1843	0.1894
u	0.2244	0.2122	0.2141	0.2167	0.2171
			Lending club credit dataset		
LR	0.0795	0.0647	0.0747	0.0650	0.0715
DT	0.1181	0.1050	0.1150	0.1104	0.1060
Linear-SVM	0.0881	0.0717	0.0817	0.0768	0.0724
RBF-SVM	0.0971	0.0833	0.0833	0.0839	0.0847
XGBoost	0.0773	0.0737	0.0737	0.0818	0.0766
DNN	0.0897	0.0862	0.0812	0.0897	0.0935
IF	0.5782	0.5643	0.5593	0.5667	0.5680
DAE	0.7122	0.7000	0.7100	0.7032	0.7097
u	0.2300	0.2186	0.2224	0.2222	0.2228

D.4. Effects of Parameter T on the Attack Algorithm Performance

As the number of base classifiers T should be moderately sized and preferably even (Li and Zhou 2007), we primarily analyze the changes of the attack performance for the Linear-STBA (Co-Forest) at $T = 4, 6, ...,$ and 12. Table A4 shows the associated SEIs of the eight target models on the two datasets, and Figure A4 displays the average u of the SEIs. We find that the performance of the attack algorithm is the best for $T = 6$.

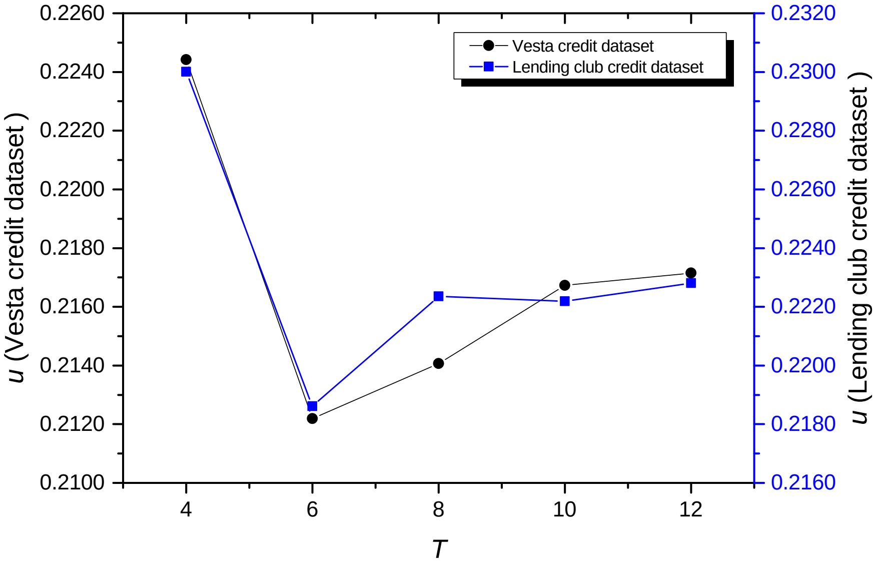


Figure A4 **Varying u for Different Values of the Number of Base Classifiers T on Two Experimental Datasets**

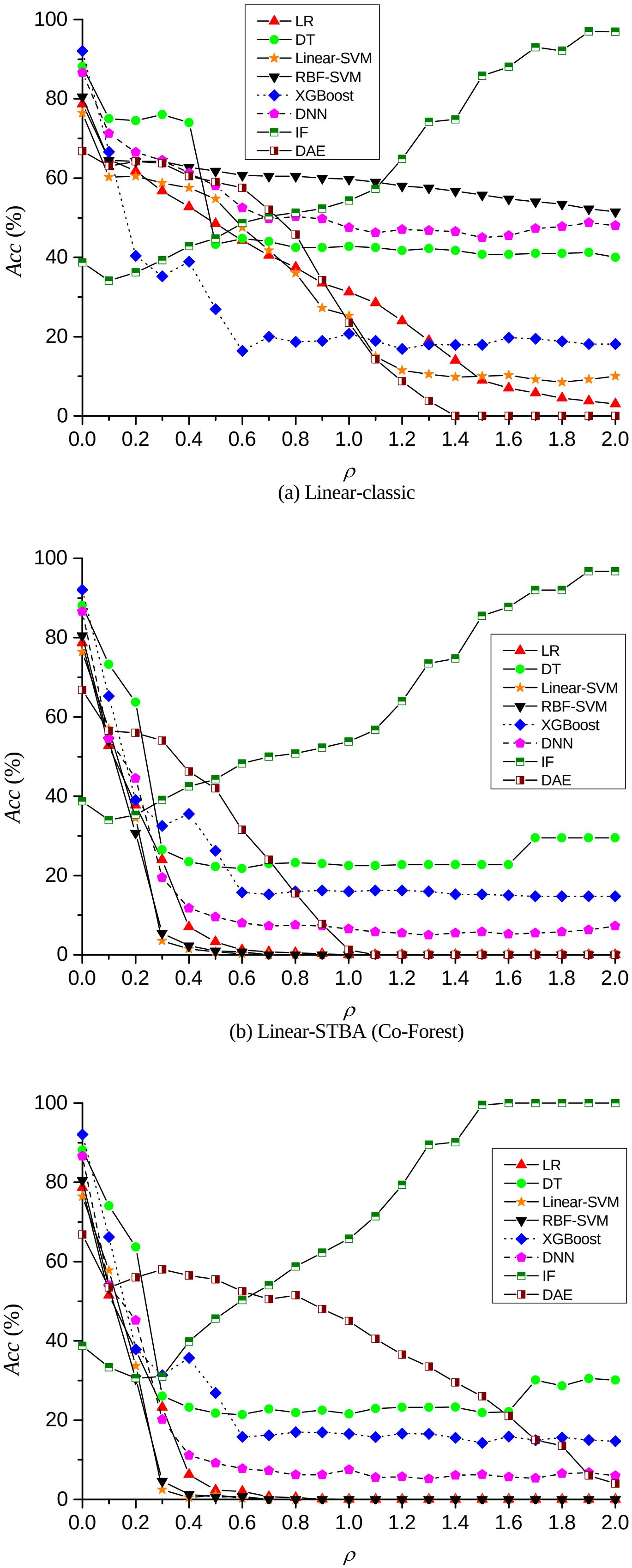
Appendix E: Comparison of Computational Time for Different Versions of the STBA Algorithm

Due to the various substitute models, there are differences in the computational time of the Linear-STBA and RBF-STBA. Table A5 shows the time needed to construct 100 fraudulent transaction samples in the test sets into adversarial examples using the Linear-STBA (Co-Forest) and RBF-STBA (Co-Forest) on the two experimental datasets. The time spent for the RBF-STBA (Co-Forest) to generate adversarial examples is obviously longer than for Linear-STBA (Co-Forest). The main cause for the large differences in the computational time of the two versions of the attack algorithm is that the Linear-STBA (Co-Forest) only needs to solve problem (6) once to construct all the adversarial examples, whereas the RBF-STBA (Co-Forest) has to solve problem (9) repeatedly when constructing each adversarial example. However, in actual credit card fraud, the computational time of the attack algorithm may not be a constraint. For the fraudsters, if an attack algorithm can increase the success rate of fraud, it is acceptable even if the attack algorithm requires more computational time.

It is worth mentioning that a similar conclusion can be obtained by comparing the computational time of Linear-STBA (FlexMatch) and RBF-STBA (FlexMatch).

Table A5 **Computational Time for Different Versions of the STBA Algorithm (s)**

Dataset	Linear-STBA (Co-Forest)	RBF-STBA (Co-Forest)
Vesta credit dataset	1.53	45.70
Lending club credit dataset	2.56	31.63



(a) Linear-classic

(b) Linear-STBA (Co-Forest)

(c) Linear-STBA (FlexMatch)

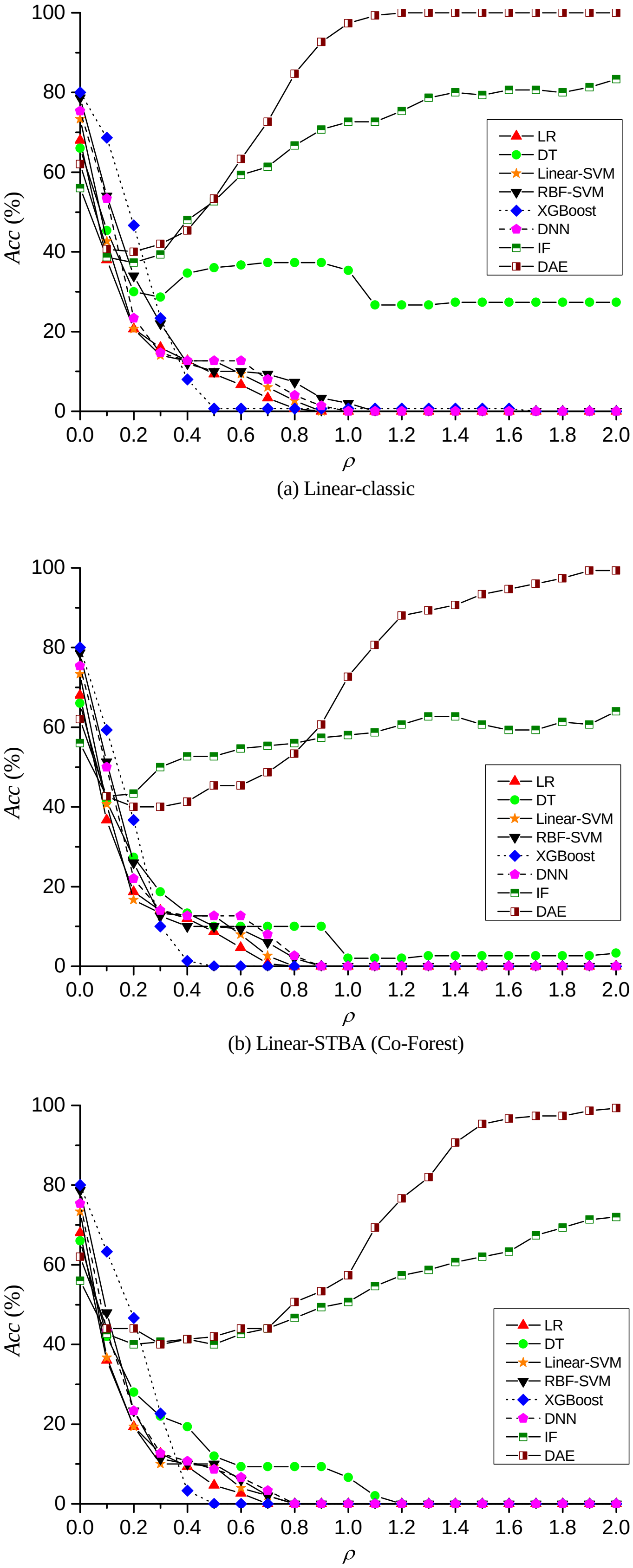
Figure A5 Security Evaluation Curves of Eight Target Models Under Attacks of the (a) Linear-classic, (b) Linear-STBA (Co-Forest) and (c) Linear-STBA (FlexMatch) on the Vesta Credit Dataset

Appendix F: Security Evaluation Curves and SEIs of Eight Target Models

Here, we report the experimental results not shown in Sections 3.3–3.5, including Figures A5–A8 and Table A6.

Table A6 SEIs of Eight Target Models Under the Attacks of Linear-STBA (FlexMatch) and RBF-STBA (FlexMatch)

Target model	Vesta credit dataset			Lending club credit dataset		
	Linear-STBA (FlexMatch)	RBF-STBA (FlexMatch)	Difference	Linear-STBA (FlexMatch)	RBF-STBA (FlexMatch)	Difference
LR	0.0817	0.0973	-0.0155	0.0633	0.0787	-0.0153
DT	0.3020	0.2369	0.0652	0.0915	0.0638	0.0277
Linear-SVM	0.0670	0.1048	-0.0378	0.0677	0.0710	-0.0033
RBF-SVM	0.0657	0.0593	0.0064	0.0820	0.0793	0.0027
XGBoost	0.2369	0.1341	0.1028	0.0770	0.0570	0.0200
DNN	0.1369	0.1514	-0.0145	0.0849	0.1042	-0.0193
IF	0.6853	0.6883	-0.0031	0.4367	0.4759	-0.0392
DAE	0.9772	0.3974	0.5798	0.6473	0.7453	-0.0980

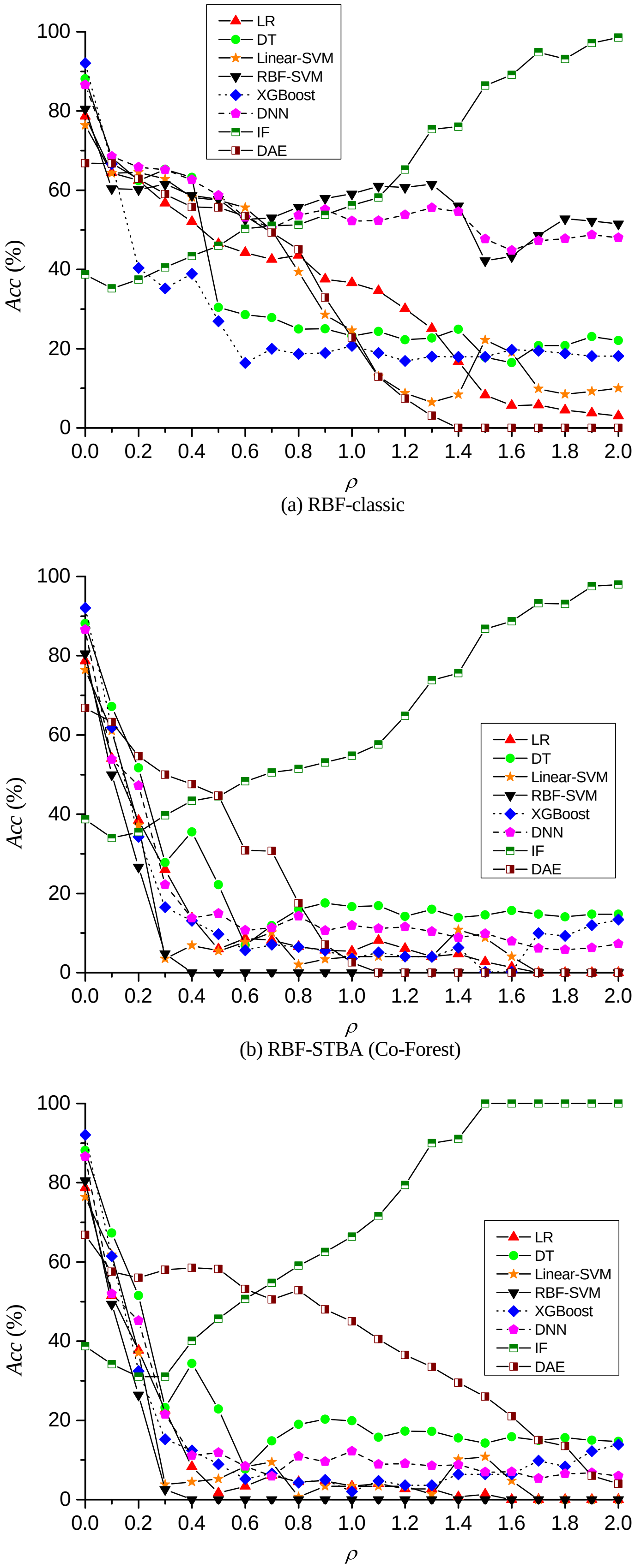


(a) Linear-classic

(b) Linear-STBA (Co-Forest)

(c) Linear-STBA (FlexMatch)

Figure A6 Security Evaluation Curves of Eight Target Models Under Attacks of the (a) Linear-classic, (b) Linear-STBA (Co-Forest) and (c) Linear-STBA (FlexMatch) on the Lending Club Credit Dataset

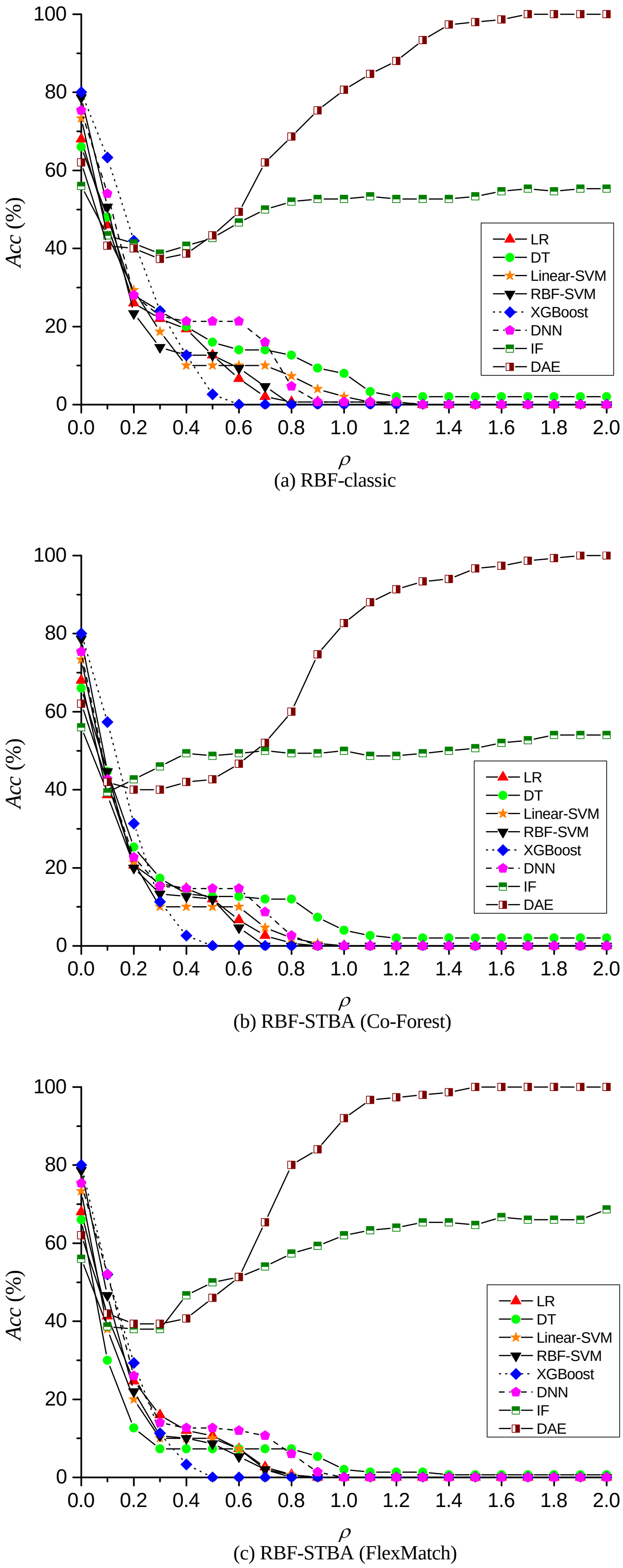


(a) RBF-classic

(b) RBF-STBA (Co-Forest)

(c) RBF-STBA (FlexMatch)

Figure A7 **Security Evaluation Curves of Eight Target Models Under Attacks of the (a) RBF-classic, (b) RBF-STBA (Co-Forest) and (c) RBF-STBA (FlexMatch) on the Vesta Credit Dataset**



(a) RBF-classic

(b) RBF-STBA (Co-Forest)

(c) RBF-STBA (FlexMatch)

Figure A8 **Security Evaluation Curves of Eight Target Models Under Attacks of the (a) RBF-classic, (b) RBF-STBA (Co-Forest) and (c) RBF-STBA (FlexMatch) on the Lending Club Credit Dataset**

References

Breiman L, Friedman J, Stone CJ, Olshen RA (1984) *Classification and Regression Trees* (London: Chapman and Hall/CRC).

Chen XW, Wasikowski M (2008) FAST: A roc-based feature selection metric for small samples and imbalanced data classification problems. *Proc. 14th ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining*, 124–132 (Las Vegas, NV: ACM).

Khosrojerdi S, Vakili M, Yahyaei M, Kalhor K (2016) Thermal conductivity modeling of graphene nanoplatelets/deionized water nanofluid by MLP neural network and theoretical modeling using experimental results. *Internat. Commun. Heat Mass Transfer* 74(5):11–17.

Li M, Zhou ZH (2007) Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. *IEEE Trans. Systems, Man, Cybern. A* 37(6):1088–1098.

Mead A, Lewris T, Prasanth S, Adams S, Alonzi P, Beling P (2018) Detecting fraud in adversarial environments: A reinforcement learning approach. *Proc. 16th Systems Information Engineering Design Symp.*, 118–122 (Charlottesville, VA: IEEE).

Smith AE, Coit DW, Liang YC (2009) Neural network models to anticipate failures of airport ground transportation vehicle doors. *IEEE Trans. Autom. Sci. Eng.* 7(1):183–188.

Van Vlasselaer V, Eliassi-Rad T, Akoglu L, Snoeck M, Baesens B (2016) Gotcha! Network-based fraud detection for social security fraud. *Management Sci.* 63(9):3090–3110.

Xiao J, Tian YH, Xie L, Jiang X, Huang J (2020) A hybrid classification framework based on clustering. *IEEE Trans. Ind. Informat.* 16(4):2177–2188.

Zhang B, Wang YD, Hou WX, Wu H, Wang JD, Okumura M, Shinozaki T (2021) Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Proc. 35th Internat. Conf. Neural Information Processing Systems*, 18408–18419 (ACM).