Journal of Information Science and Engineering 39, 1185-1207 (2023)
DOI: 10.6688/JISE.202309_39(5).0011

SADEM: An Effective Supervised Anomaly Detection Ensemble Model for Alert Account Detection

Hui-Kuo Yang+, Bing-Li Su and Wen-Chih Peng
Department of Computer Science
National Yang Ming Chiao Tung University
Hsinchu, 300093 Taiwan
E-mail: hgyang@gmail.com+; billy4195.su@gmail.com; wcpeng@cs.nctu.edu.tw

Anomaly detection has been an important research topic for a long time and has been applied to many real-world applications. However, due to the high cost of manually getting the instance label, researchers mostly resort to unsupervised or semi-supervised learning approaches. The supervised learning method has rarely been used in anomaly detection tasks. In this paper, we proposed a supervised learning ensemble method to detect alert accounts among transaction data. We solve the problem of low-confident predictions when the anomalies reside within normal data points. The ensemble model comprises the LightGBM and Multi-layer Perceptron (MLP) to synergize machine learning and neural network models. The proposed model preserves the result of high-confident predictions and improves the performance of low-confident predictions with the new features generated from encoding the leaf node of GBDT (Gradient Boosting Decision Tree). Our experiments on a real-world dataset show the effectiveness of the model when compared with the state-of-the-art methods.

Keywords: alert account detection, anomaly detection, imbalanced classification, supervised learning, low-confident predictions

1. INTRODUCTION

Anomaly detection is an important function in many critical tasks [1, 2], in which the failure to detect anomaly cases could cause a huge impact or loss. The goal of the anomaly detection is to pick up the anomaly instances among the normal ones. The definition of anomaly varies from application to application. It could be a network packet sent from a malicious program to launch intrusive attacks in cyberspace [3, 4], or a series of fraudulent transaction requests in the financial industry [5]. Because of the cost of quality labels from the expert, most anomaly detection tasks do not have fully labeled ground truth. Their works focus on the unsupervised or semi-supervised method, which does not require too many labels.

Received August 17, 2022; accepted November 8, 2022.
Communicated by Shyi-Ming Chen.
+Corresponding author.

In this paper, we will focus on a specific case of anomaly detection in banking, **alert account detection**. Alert accounts are those blocked by banks from certain operations because of their suspected involvement in a crime or a lawsuit. The alert accounts will be collected monthly and sent to the bank by the law enforcement agency. In certain cases, the list would be given to the bank earlier once the law enforcement agency finds it necessary and urgent to investigate and stop crimes. Therefore, the bank can obtain a list of accounts being labeled as alert accounts at least once a month. From the perspective of the bank, if they could detect alert accounts earlier than receiving notifications from the law enforcement agency, they can actively control the risk themselves to reduce the loss in advance, and further improve customer experiences.

As illustrated in Fig. 1, we hope to detect the alert accounts before we get them from the crimal investigators of the government so that we can actively prevent any illegal money transfer to protect potential victims. Thus we set our goal as finding a top-K watch list that is most likely to contain the accounts in the list obtained from the law enforcement agency. The number K is defined by our bank client through their consideration of the human resources involved in account validation procedures. The generated list will be manually checked by the bank clerks using phone calls or other means; therefore, the validation of the list of K accounts should be an affordable workload for the bank. In our setting, the K is larger than the average number of alert accounts in a month to attain better coverage.

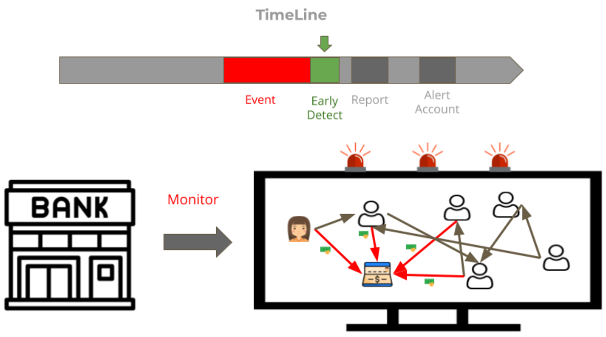
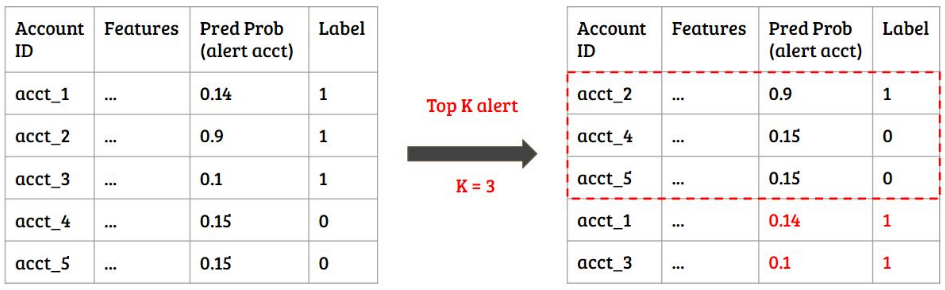


Fig. 1. The motivation to early detect the alert account.

Following the scenario mentioned above, we next highlight challenges for alert account detection:

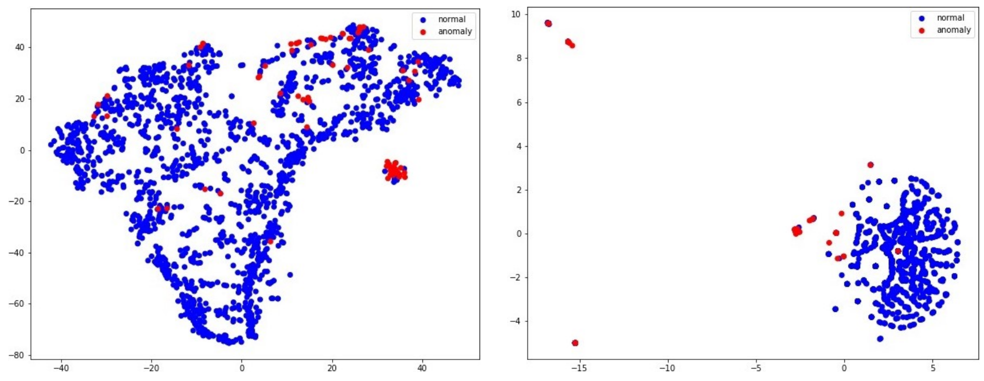
- **Low-confident predictions**
  A trivial way to get a K watch list is to select K accounts with the highest probabilities from the predicted outputs of the best classification model as alert accounts. Since the alert account is scarcely seen, the model would give high scores only to a small number of alert accounts. In contrast, other alert accounts get very low scores, which render the problem of low-confident predictions, as shown in Fig. 2. Thus, only a few instances are recognized as anomalies, and the remaining undiscovered anomalies are low-confident


Top K alert
K = 3

Account ID	Features	Pred Prob (alert acct)	Label
acct_1	...	0.14	1
acct_2	...	0.9	1
acct_3	...	0.1	1
acct_4	...	0.15	0
acct_5	...	0.15	0

Account ID	Features	Pred Prob (alert acct)	Label
acct_2	...	0.9	1
acct_4	...	0.15	0
acct_5	...	0.15	0
acct_1	...	0.14	1
acct_3	...	0.1	1

Fig. 2. The toy example of low-confident predictions problem.



(a) Original features

(b) Leaf features

Fig. 3. Using t-SNE to visualize the distribution of the alert account and the normal account in the feature space. The red points are the anomaly (alert accounts) and the blue points are the normal (normal accounts).

predictions. It would hurt the accuracy of model predictions by generating many false alert accounts, and overlooking most true alert accounts, thus decreasing the practical value of the model and increasing the labor cost for manual account validation.

- **Imbalanced class dataset**
  With the help of label information, we could use supervised learning to build a classification model. However, since the anomaly behavior is rare, a huge difference in the number of data points between alert and normal accounts would be the case. The imbalanced class dataset can result in a trained model being deeply biased toward the major class as it only learns to predict inputs as normal accounts since the normal class is the most seen in the dataset.

- **Anomalies mixed with normal data**
  The behavior of the alert account is, to some degree, similar to the normal accounts. The alert accounts would undergo the same operations as the normal accounts, such as depositing, withdrawing, and transferring. When visualiz-

ing these behaviors, as shown in Fig. 3 (a), we could observe that the anomaly (red) points are very close to normal (blue) points or reside within normal points. Unsupervised models could barely identify the anomaly points because their distance to the normal points is probably shorter than that between normal points and thus are indistinguishable. Anomalies mixed with normal data would lead to an inseparable feature space. A feature space transformation to higher dimensions is often necessary to overcome it, thus adding model complexity and the risk of overfitting.

In this paper, we have proposed an ensemble classification model that consists of LightGBM (LGBM) [6] and Boost Predictor, which further consists of Multilayer Perceptron (MLP) [7] and Logistic Regression [8]. Since the number of alert accounts reported each month is small, we choose LGBM as the machine learning model that could match the scale of our dataset. LGBM is used to generate new features to improve representation learning for anomaly identification. It then enables a neural network for this dataset to map the original feature vectors to a different space. We would argue that using features of leaf nodes from a tree-based model (*e.g.*, LightGBM as in our work) would have a higher chance for these new features to be more distinguishable. As shown in Fig. 3 (b), when turning to the feature space generated using leaf nodes of a tree-based model, anomalies are far away from normal points when compared with Fig. 3 (a). Our experiments show the effectiveness of the proposed model.

The contributions of this paper are summarized as follows:

- **Present the low-confident predictions problem and design a boost predictor to solve it**
  Instead of selecting K accounts that contain low-confident predictions, we design a boost predictor, which takes advantage of a neural network so that it could learn the non-linear and complex feature combinations to improve the low-confident predictions.

- **Enhance anomaly detection with the strength of the imbalanced classification model**
  Since the anomalies are scarce compared with the amount of normal data, we tackle this problem with the strength of the imbalanced classification model, using the sampling method and cost-sensitive learning. The sampling method selects an adequate number of normal accounts for LightGBM to train, while the cost-sensitive method tries to adjust the loss weight to recognize the anomaly points better.

- **Propose to learn separable features from a tree-based model**
  We use the result of a supervised machine learning model LightGBM [6], in learning the representation to solve that alert accounts are indistinguishable from normal accounts. With the help of LightGBM, we transform original features into leaf-node based feature space. In the new feature space, the anomalies are easier to be identified. Furthermore, we could utilize a neural network to train the model on the transformed dataset while LightGBM enriches by adding distinguishable features.

The rest of the paper is organized as follows. In Section 2, we review papers in anomaly detection and imbalanced classification. In Section 3, the proposed model SADEM is elaborated in detail. We present the experiment results in Section 4. Section 5 concludes our work.

2. RELATED WORK

The alert account detection task could be discussed in two perspectives, **(1) anomaly detection task**; which is to discover anomaly accounts using a seldom labeled dataset, and **(2) imbalanced classification problem**; which is with full-labeled dataset. In the anomaly detection task, some state-of-the-art methods include unsupervised and semi-supervised learning. We discuss some state-of-the-art methods focusing on supervised learning approaches for the imbalanced classification problem.

2.1 Anomaly Detection

Due to the high cost of acquiring labels among a massive number of data instances, most of the anomaly detection tasks only contain very few ground truths. Therefore, the previous works mainly focus on unsupervised learning and semi-supervised learning.

2.1.1 Unsupervised learning

The unsupervised learning methods include one-class SVM [9], and SVDD [10]. One-class SVM used a curve while SVDD [10] used a sphere to split the normal and anomaly points. They transformed the input data into a new feature space before splitting the data points. However, their learning of representation was usually independent of the target task that was to find out the anomalies, as addressed in [11]. Deep-SVDD [12] and One-Class Neural Network(OCNN) [13] proposed their respective objectives to learn the representation for the target task in model training. IsolationForest [14, 15] built a forest consisting of a random split decision tree to recognize the anomalies. Following the main structure of IsolationForest, [16] later used data-dependent information to build isolation forest efficiently. [3, 4, 17, 18] used the bottleneck autoencoder to learn the hidden representation.

2.1.2 Semi-supervised learning

There may be partial labels available in the dataset when training a model to recognize anomalies. With the help of label information, the model could better identify anomalies very close to normal data points in the feature space. In [19], they used active learning and involved domain experts to collaborate during the training process to get new labels. In this way, data points that were likely to be anomalies or the points that were indistinguishable for their model would have a higher chance of being selected for further scrutiny. The new labeled points would then be added to the training dataset to improve the existing model. [20] used only the known labels for training a model, then iteratively selected a new anomaly data and put it to the unlabeled data points until the stop criterion was met. In [21], a deviation metric was considered for directly learning anomaly scores deviating from average scores of normal data points by leveraging a few labeled anomalies. The semi-supervised learning performs better than unsupervised learning for datasets in certain situations. Such as when the dataset contains ground truth obtained with the collaboration of domain experts or when the partially labeled dataset contains undiscovered anomalies that need to discover. However, it does not fit our alert account detection task because the dataset is fully labeled.$^{+}$Corresponding author.

2.2 Imbalanced Classification

In imbalanced classification, we treat the problem as to train a model to recognize the given samples between imbalanced classes. We discuss three types of methods in this category.

2.2.1 Data level

This type of method focuses on data distributions. They use sampling [22–28] to balance the class distributions. [22, 24] generate the synthetic minority points to augment the dataset. They use interpolation to acquire new points from interpolated neighbor points. The undersampling methods [25, 28] reduce the data points in the majority class. The goal of these two types of sampling is to balance the class distributions in the dataset. Although these methods could resolve the imbalanced class problem, the oversampling methods would cause overfitting by repeatedly visiting the same minority points. For the undersampling methods, the main drawback is information loss.

2.2.2 Algorithmic level

Cost-sensitive learning [6, 29–35] is used here to emphasize the misclassification of the minority class. XGBoost [33] and LightGBM [6] are both the gradient boosting decision tree(GBDT) [36] models. GBDT is one kind of ensemble model composed of a bunch of decision trees, they use the weight to control the imbalanced classes, and the contribution of different classes in the loss function depends on the number of data points in that class.

2.2.3 Hybrid

There are some models [37–39] that combine the sampling and cost-sensitive learning together. Besides, several deep learning methods have been proposed [35, 40–45]. They tend to combine the sampling and the cost-sensitive learning methods together to build an end-to-end neural network model. Because the number of alert accounts is small, it could only be trainable by the traditional machine learning model, otherwise, the deep learning model would overfit by updating loss function contributed from the minority class repeatedly. After observing our dataset, the alert accounts are fairly close to the normal accounts. The anomaly detection methods may not apply to our dataset because they only use few or no label information in their models. The imbalanced learning methods would be more suitable for our problem because they consider the label information. However, the lack of alert accounts data has forced us to try machine learning models in our dataset, and the low-confident prediction is the main problem addressed in our work.

3. PROPOSED MODEL

3.1 Problem Definition

In this paper, our goal is to generate a watch list of accounts that is suspicious to be alert accounts. Given the dataset $D = \{(X_1, y_1), (X_2, y_2), \dots (X_N, y_N)\}$, where X are the input features of the accounts and $y \in \{0, 1\}$ are the alert account indicator. The proposed model **Supervised Anomaly Detection Ensemble Model (SADEM)** outputs the watch list accounts $W_{alert} = \{w_1, w_2, \dots, w_K\}$, where w is the account id that is suspected to be an alert account, and K is the length of the watch list, which is specified depending on the requirement of real scenarios.

We solved our problems mentioned above using $SADEM(X; \kappa)$, in which κ is the parameters of the model $SADEM(X; \kappa)$ that we will learn. Let $G : (Y, \widehat{Y}) \to [0, 1]$ be the grading function used to evaluate model performance. Y is the ground truth, and $\widehat{Y}$ is the model output. For example, $G = (|Y \cap \widehat{Y}|)/(|Y|)$ is the recall often used to evaluate model performance in classification problems. With these notations in place, we can formulate our problem as below

$$\kappa^* = \arg\max_{\kappa} G(Y, \widehat{Y} = \{y_i; \forall i \in SADEM(X; \kappa)\}).$$

When κ^* is attained, it is our optimizer and $G^* = \max G$. In reality, it is difficult to attain such an optimal point, and we train our model to obtain a set of evaluations $\{G_{\kappa'}\}$, where κ' is the learned model parameters, and we have the approximation to G^* as below

$$\sup\{G_{\kappa'}\} \approx G^*.$$

3.2 Supervised Anomaly Detection Ensemble Model (SADEM)

Our model aims to consider the low-confident predictions in an imbalanced class learning problem. The overview of the proposed model is shown in Fig. 4. The input is downsampled account information which is the result of random under-sampler (RUS), and the output of **SADEM** is the top K watch list W_{alert}, which is the combination of the set of High-Confident Predictions (HCP) and the set of Augmented Predictions (AP), as described in Eq. (1). HCP are the accounts in the predictions with a high probability of being the anomalies, and the AP is the suspicious accounts that are generated by the ensemble model composed of ML model and NN model.

$$\begin{aligned} W_{alert} &= SADEM(X; \kappa) \\ &= \{HCP, AP\} \end{aligned} \tag{1}$$

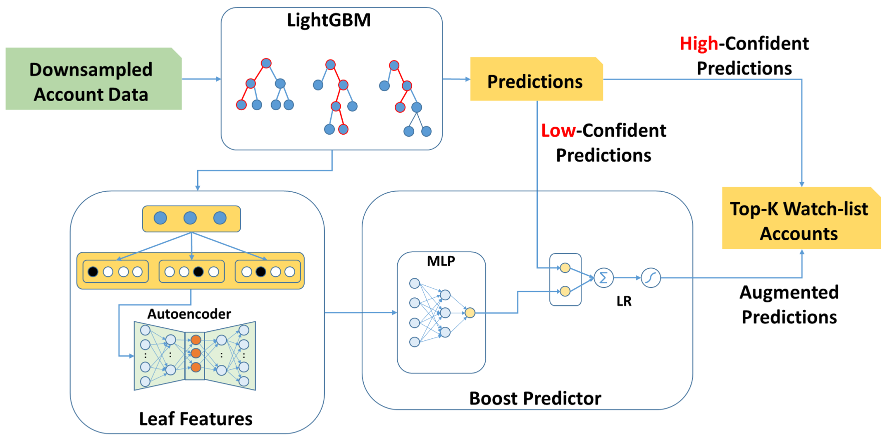


Fig. 4. The SADEM model.

The model is composed of the machine learning model and the deep learning model to be a supervised imbalanced classification model. The **random under-sampler(RUS)** balances the data distribution in the training dataset. **Light-GBM(LGBM)** [6] generates high-confident predictions and transforms the input into leaf features. The **Leaf Features** are the enriched features that fit into the neural network model training. The **Boost predictor** takes the low-confident predictions and uses the leaf features as input to improve the performance of low-confident predictions. In the following subsections, we explain the implementation of RUS, LGBM, Leaf Features, and Boost Predictor.

3.3 Random Under-Sampler (RUS)

A random under-sampler is used to reduce the number of normal accounts (majority class) in the training dataset. Although the number of anomalies is suitable for the machine learning model to train rather than the neural network model, the huge amount of normal data will bias the model to only learn the normal data points. The imbalanced problem will be tackled by sampling and cost-sensitive learning to reserve the original property and enable the model to identify minority class. The dataset can be split into two parts, $D = \{D_N, D_A\}$, the D_N stands for the set of normal data, and the D_A is the set of anomaly data. The imbalanced distribution of data is described as $|D_A| \ll |D_N|$. We undersample the D_N to D'_N such that $|D_N| > |D'_N|$, and $|D'_N| = r \times |D_A|$ as Eq. (3) shows, where $RUS(D_N)$ is the random under-sampler for D_N . r is the hyper-parameter to control the ratio to be chosen in the majority class. The undersampled dataset D' consists of D'_N and D_A as shown in Eq. (4).

$$D = \{D_N, D_A\} \tag{2}$$

$$D'_N = RUS(D_N) \quad \text{and} \quad |D'_N| = r \times |D_A| \tag{3}$$

$$D' = \{D'_N\} \cup \{D_A\} \tag{4}$$

3.4 LightGBM & High-Confident Prediction

After undersampling, we need to put an appropriate model that fits our requirements. Here, we choose LightGBM(LGBM) [6] as the ML model for training on the undersampled dataset. The reasons for choosing LightGBM are (1) LGBM is one kind of GBDT [36], and it has been the well-known winner of a lot of ML competitions[1]; (2) it supports cost-sensitive learning using class weights on loss; and (3) it can extract the learned information to be the leaf features for our needs. We keep the output of LightGBM model only for high-confident predictions. For low-confident predictions, since there are still anomalies undiscovered in those predictions, thus they will get other processes by Boost Predictor to acquire more convincing and better results.

3.4.1 LightGBM

LightGBM is a lightweight version of the GBDT (Gradient Boosting Decision Tree). It is a histogram-based algorithm aiming to reduce memory usage and improve the speed of model training and inference. GBDT is an ensemble model that iteratively adds a new decision tree $T_j(X_i)$ to improve the previous model $h^{(j-1)}(X_i)$ as shown in Eq. 5 until the stop criterion is met, *i.e.*, the total loss $\sum_{i=1}^{\|D\|} loss(\widehat{y}_i, y_i)$ is minimized.

$$h^{(j)}(X_i) = h^{(j-1)}(X_i) + T_j(X_i) = h^{(j-1)}(X_i) + y_{ij} \tag{5}$$

We apply GBDT in alert account detection to solve the imbalanced problem. Besides under-sampling, LGBM automatically adjusts the weights of different classes in the training dataset. The weights of different classes in the loss are related to the number of data samples due to the imbalanced nature of the dataset, and it would significantly increase the performance of identifying the minority class. Moreover, LightGBM uses a leaf-wise tree growth approach, and the number of leaves for each decision tree in the forest is the same. The output dimension of each decision tree is thus the same after one-hot encoded so that we could treat it as an intermediate representation learned by LGBM. In each decision tree T_j, the prediction of the input features $X_i \in D$ is y_{ij}. The prediction from overall LGBM given X_i is the combination of all the decision trees in the model y_i^{LGBM} in Eq. (6), where N is the number of decision trees in the model.

$$\begin{aligned} y_i^{LGBM} &= LGBM(X_i) \\ &= \sum_{j=1}^{N} y_{ij} \end{aligned} \tag{6}$$

3.4.2 High-confident prediction (HCP)

The LGBM outputs the predicted score between 0 and 1 of each input $X_i \in D$ as the probability of being an anomaly. The higher score generated by LGBM

[1] https://github.com/microsoft/LightGBM/blob/master/examples/README.md#machine-learning-challenge-winning-solutions

means the higher confidence of the prediction to be an anomaly. Therefore, we only keep the high-confident predictions (HCP) while the remaining low-confident predictions will need more processes. Those low-confident predictions will be boosted by using the tree structure learned from LGBM and the output scores together to train a Boost Predictor. HCP is obtained by setting a boundary for the model output with a threshold τ to separate high-confident predictions from low-confident predictions, as in Eq. (7).

$$HCP = \{y_i > \tau | y_i \in y^{LGBM}\} \tag{7}$$

3.5 Leaf Features

In order to improve the low-confident predictions, we utilize a neural network model for its learning strength. As mentioned before, our main problem is that the lack of anomaly points is the cause of poor performance. Once LGBM is trained on the training dataset, the structure of decision trees in LGBM could be seen as refined features. We can view different inputs leading to different combinations of decision trees resulting from LGBM. The information obtained from LGBM includes feature selection, data sampling, and complicated feature combinations. Inspired by [46], we use leaf features to enrich the dataset and apply a neural network to obtain improved results. Leaf features are the predicted leaf indexes of all decision trees in LGBM; the process of generating leaf features is described below.

3.5.1 Predict leaves

In order to extract the learned structure of LGBM, we use the leaf indexes as the transformed features. To obtain the leaf indexes, we combined the results of decision trees as the output features. We obtain leaf index by the tree operation $T.pred\_leaf(X)$. As shown in Eq. (8), the leaf index of the decision tree T_j for the given input $X_i \in D$ is L_{ij}, and M is the number of leaves in T_j.

$$L_{ij} = T_j.pred\_leaf(X_i), \quad \text{where} \quad 1 \leq L_{ij} \leq M,\ L_{ij} \in \mathbb{N} \tag{8}$$

Overall output features of LGBM are described in Eq. (9). The input X_i will be transformed to L_i, which is the combination of all leaf indexes L_{ij} from each decision tree T_j, where $1 \leq j \leq N$ (N is the number of trees in LGBM).

$$\begin{aligned} L_i &= LGBM.pred\_leaf(X_i) \\ &= \{L_{i1}, L_{i2}, ..., L_{iN} |\ L_{ij} = T_j.pred\_leaf(X_i)\} \end{aligned} \tag{9}$$

3.5.2 Encode leaves

The generated leaf indexes are categorical. The difference between leaf index one and leaf index 3 in tree T_j is meaningless. Therefore, we need to transform it into an encoded format. Here we choose one-hot encode to represent leaf index features. One-hot encode is the most famous method to encode categorical features. It can encode categorical data into numeric value without any information loss. The leaf index features L_i will then be transformed to $L_i^o \in l^{M\times N}, l = \{0,1\}$ which is the concatenation of all leaf features L_{ij}^o as shown in Eq. (10).

$$L_i^o = (L_{i1}^o, ..., L_{ij}^o),\ 1 \le j \le N \tag{10}$$

Each leaf feature L_{ij}^o is encoded from the leaf index L_{ij} by a one-hot encoder as shown in Eq. (11).

$$\begin{aligned} L_{ij}^o &= OneHotEncoder(L_{ij}) \\ &= (l_1, ..., l_M), \quad \text{where} \quad l_t \in \{0,1\},\ \sum_{t=1}^{M} l_t = 1 \end{aligned} \tag{11}$$

3.5.3 Reduce dimensions

After encoding the leaf features, the dimension of the features will be $M\times N$. It will become very large if we adjust the N and M to the larger numbers. The larger dimensions in input features will cause the model to need more anomaly points in the dataset because the model will contain more parameters to be trained, thus requiring more training time. One solution is to reduce the dimensions of the input features. We choose autoencoder here for dimension reduction, which we are inspired by [47]. We design a bottleneck autoencoder to reduce the dimension of features. The autoencoder is one kind of neural network model that consists of an encoder and a decoder, and the goal is to learn a low-dimensional representation with minimized reconstruction error. Assume that the encoded feature is $L_i^e \in \mathbb{R}^d$, where d is the reduced dimension, and the encoder is used to encode the input feature L_i^o to the latent features L_i^e as shown in Eq. (12).

$$L_i^e = s_E(W_E L_i^o + b_E), \tag{12}$$

where s_E is the activation function, and W_E and b_E are parameters and bias for encoder network, respectively.

The decoder will learn to use the latent features L_i^e to rebuild the original input $\widetilde{L}_i^o \in \{0,1\}^{M\times N}$ as shown in Eq. (13).

$$\widetilde{L}_i^o = s_D(W_D L_i^e + b_D), \tag{13}$$

where s_D is the activation function, W_D and b_D are parameters and bias for the decoder network, respectively. The objective is to minimize the reconstruction error $\|\widetilde{L}_i^o - L_i^o\|$.

3.6 Boost Predictor & Augmented Predictions

To solve the low-confident predictions problem, we proposed **Boost Predictor**, which utilizes both neural networks and cost-sensitive learning to tackle it. Boost Predictor takes the leaf features and the low-confident predictions produced by LGBM as inputs and generates the **Augmented Predictions (AP)** for adding to the overall model output.

3.6.1 Boost predictor

Although the neural network is popular for its strong learning capability, its restriction is the prerequisite of a large-sized dataset for learning. In previous sections, we have mentioned how we solve the minority problems by extending features to large spaces. However, the data distribution is still imbalanced. As a result, we need the help of cost-sensitive learning to handle the imbalanced problem. Boost Predictor consists of a **weighted Multi-layer Perceptron (wMLP)** and **weighted Logistic Regression(wLR)**. The neural network can capture complicated feature interactions through the complex structure and the non-linear activation function. The predictions of wMLP are defined in Eq. (14).

$$\begin{aligned} p_i &= wMLP(L_i^e) \\ &= q_i^{(k)}, \quad \text{where} \quad 0 \le q_i^{(k)} \le 1, \end{aligned} \tag{14}$$

p_i is the output of the wMLP with the input of encoded leaf features for the i-th data point, and k is the layer index in wMLP. Each layer in the wMLP is defined in Eq. (15). The output of the last layer in the wMLP will be taken as the model prediction.

$$q_i^{(k)} = \begin{cases} s(W^{(1)}L_i^e + b^{(1)}), & if\ k = 1 \\ s(W^{(k)}q_i^{(k-1)} + b^{(k)}), & if\ k \ge 2 \end{cases} \tag{15}$$

$W^{(k)}$ and $b^{(k)}$ are the parameters and bias for the k-th layer of wMLP, respectively, and s is the sigmoid function in wMLP. The difference between wMLP and MLP is its error function. The original error function of MLP is cross-entropy as shown in Eq. (16).

$$Err = \frac{1}{m}\sum_{i=1}^{m}[-y_i log(p_i) - (1-y_i) log(1-p_i)] \tag{16}$$

But for wMLP, the error function is a weighted cross-entropy in which the weights are adjusted for different classes. In this paper, the weighted cross-entropy is defined in Eq. (17), in which λ is a weight factor that controls the weight of loss from normal and anomaly classes.

$$wErr = \frac{1}{m}\sum_{i=1}^{m}[-y_i log(p_i)\lambda - (1-y_i) log(1-p_i)(1-\lambda)] \tag{17}$$

However, the number of anomaly points is still insufficient for a neural network to train a strong model. Under this situation, we design the *Boost Predictor* that combines the predictions of LGBM and wMLP by stacking. Then we use a logistic regression model, which takes the predictions from LGBM and wMLP as inputs to generate augmented predictions. The reason to choose logistic regression is that first, it seeks to capture the non-linear relation of features using the sigmoid function; second, it is less complex which is suitable to train a model with only two classes.

3.6.2 Augmented predictions

The output after going through the boost predictor is called *Augmented Predictions.* It is the improved versions of predictions from the initial predictions of LGBM in which high-confident predictions(HCP) are excluded. Augmented predictions are used to replace the low-confident predictions so that they are combined with HCP to form a revised complete top-K predictions as shown in Eq. (1).

4. EXPERIMENT

In our experiment, the transaction dataset is from an Asia-based regional bank whose service targets include individuals and corporations, with service domains extended to digital banking and AI-enabled financial services. Our research aims to detect any account suspicious of criminal activities from transaction records and put it into an alert list. By conducting a series of online experiments, we have validated the effectiveness of our model. We also compare our approach with state-of-the-art competitors, including anomaly detection and imbalanced classification algorithms.

4.1 Transaction Dataset

The transaction records span the period from July 2018 to May 2019. The input data include transaction features aggregated by user and date, such as the number of transactions in the last 24 hours, the amount of money transferred out in the last 24 hours, *etc.* Furthermore, the items in a profile of an account owner, *i.e.*, age, gender, yearly income range, occupation, *etc.*, are also included. We split the dataset into a training set and a testing set by the record date. The date in the training set is from July 2018 to December 2018, and the testing set dates from January 2019 to May 2019. The total number of individual accounts in the training set is 900,000.

We evaluate the performance of our algorithm by $recall@K$, which is the ratio of detected alert accounts over true alert accounts in the top K output list. From the bank's perspective, they request to identify as many anomalies as possible in a restricted resource. Advised by the bank domain experts, the number of K is set as 350, which shows the realistic condition for the workload allocation of the bank. We use the predictions each month of the testing set to evaluate the performance.

4.2 Competing Models

We choose ten well-known models for comparison, three of which are from anomaly detection models, and the rest are from imbalanced classification models. Besides that, we add two variants of our algorithm to the comparison list.

4.2.1 Anomaly detection

- *IForest* [15] is an ensemble model that utilizes a random split tree to detect an anomaly. It uses the path length to find the leaf as the anomaly score.

- *Deep-SVDD* [12] is a one-class SVM-based model that uses a neural network to replace the kernel function.
- *Autoencoder* [4] is a model that utilizes a bottleneck autoencoder to learn the representation of normal data points. It uses reconstruction error as the anomaly score.

4.2.2 Imbalanced classification

- *LightGBM(LGBM)* [6] is an ensemble model based on decision trees. The implementation includes sampling and cost-sensitive methods which use a histogram mechanism.
- *XGBoost* [33] is a gradient boosting decision tree model that utilizes a pre-sorting mechanism to speed up computation.
- *Weighted-MLP(wMLP)* [35] is a neural network method that takes weighted cross-entropy as a loss function for misclassification of the minority class.
- *Leaf-wMLP* is a weighted MLP model in which the input is transformed from leaf features generated by LGBM. It demonstrates the effect of leaf feature transformation.
- *RUS-LGBM* is a model based on LGBM, but with preprocessed dataset. The majority class is downsampled to balance the uneven population distribution among classes.
- *SMOTE-LGBM* [22] is an LGBM-based model with preprocessed dataset by SMOTE to oversample the minority class to increase the number of their data points.
- *TU-LGBM* [28] is another LGBM-based model with preprocessed dataset by TU, which is a trainable undersampling method that trains a sampler by a recurrent neural network.

4.2.3 Reduced SADEM

- *SADEM-NoAuto* is based on our proposed model with the autoencoder being removed. It used one-hot encoded leaf features as the input of our Boost Predictor module.
- *SADEM-NoCost* is also based on our proposed model, removing the cost-sensitive learning. All the class weights are equally set to 1 in Boost Predictor.

4.3 Implementation Detail

4.3.1 SADEM

In the implementation, we set the hyper-parameters of SADEM as follows. The r of RUS is 100. We randomly undersample the normal data according to that ratio. The threshold τ is 0.9. We only extract predicted probability higher than

0.9 to be high-confident predictions. The weight factor λ is 0.9995 following the ratio '`anomaly : normal=1 : 2000`'. We use the official package[2] of LightGBM for Python interface as our LGBM implementation. The setting of hyper-parameters for LightGBM are '`num_leaves = 31`', '`n_estimators = 50`', and the rest remain the default settings. For one-hot encoder and Logistic Regression, we use the implementation in scikit-learn [48]. TensorFlow [49] is used to develop our neural network modules, including Autoencoder and MLP. The encoder of the Autoencoder has two hidden layers, and their sizes are 100 and 250. The decoder dimensions are the reverses of the encoder. The wMLP contains two hidden layers, with the hidden units for each layer being 100 and 80, respectively. All the neural networks use a fully-connected layer with a sigmoid function as the activation function. Our objective is optimized by the Adam algorithm [50]. The length of the watch list, K, is 350, and so is the setting of K in our evaluation metric *recall*@K. We set K to a specific value to meet the constraint of maximum workload per inspector in a real-world scenario.

4.3.2 Reduced SADEM

The implementation of SADEM-NoAuto is identical with SADEM. The only difference is the removal of leaf features encoding with an autoencoder. We instead use the one-hot encoded leaf index as the input for Boost Predictor. The reduced SADEM-NoCost is done by disabling the cost-sensitive learning in Boost Predictor of full SADEM. We remove the weight factor term in the loss function for wMLP.

4.3.3 Competing models

For IForest [15], we use '`IsolationForest`' in scikit-learn [48] for our experiment. For Deep-SVDD, we use the implementation from the author's GitHub repository[3]. We use the exact implementation of LGBM and Autoencoder as in SADEM, and both of them output the top 350 data points with the lowest reconstruction errors. For wMLP, we developed a neural network with the same model shape and training parameters as the one we did for SADEM. For RUS-LGBM, we randomly downsample normal data points to 100 times the number of anomalies and then use them for training LGBM. For SMOTE-LGBM, we use the implementation of the Python package '`imbalanced-learn`' [51] to sample our training set and train the LGBM on the sampled dataset. For TU-LGBM, we use the implementation of the sampler from the author's GitHub repository for data sampling, and then train the LGBM to detect alert accounts.

4.4 Results

There are three parts of evaluations: (a) comparisons with baseline anomaly detection models; (b) comparisons with baseline imbalance classification models; and (c) validation of the effectiveness of our model design.

[2]https://pypi.org/project/lightgbm/
[3]https://github.com/lukasruff/Deep-SVDD-PyTorch

Table 1. Recall@350 of anomaly detection vs. SADEM.

	Test Time Period (2019)					
Method	*Jan*	*Feb*	*March*	*Apr*	*May*	*Avg.*
IForest	0.2400	0.2917	0.4098	0.3068	0.2800	0.3057
Deep-SVDD	0.2800	0.1250	0.0984	0.2500	0.1400	0.1787
Autoencoder	0.3600	0.2500	0.3443	0.4205	0.3200	0.3389
SADEM	0.7600	0.7500	0.7705	0.7841	0.7800	0.7689

4.4.1 Anomaly detection models

We compare our model with state-of-the-art anomaly detection algorithms, which do not utilize any label information in model training. We observed that those unsupervised learning models perform poorly in our dataset in which the data points of alert accounts mixed with those of normal accounts. Deep-SVDD [12] performs even worse and we suspect the reason is their learning for mixed-type data points. The objective of Deep-SVDD is to minimize the volume of hypersphere enclosing the transformed data points, in which it assumes the normal data points are enclosed within the hypersphere while the anomaly ones fall outside of it. This learning goal does not guarantee a better representation to distinguish anomaly from normal data points, especially when they are uniformly mixed in original data space. As we can observe from Table 1, on contrary to Deep-SVDD, the Autoencoder [4] algorithm focuses on learning better representation of normal data in that it can better distinguish the normal data from unseen data, which are the anomaly data points in our dataset. With the combining strength in both classification and representation learning, our model SADEM outperforms all the other anomaly detection methods by up to 59%.

Table 2. Recall@350 of imbalanced classification v.s. SADEM.

	Test Time Period (2019)					
Method	*Jan*	*Feb*	*March*	*Apr*	*May*	*Avg.*
LGBM	0.7000	0.6667	0.7049	0.7386	0.7200	0.7060
XGBoost	0.5600	0.4167	0.6066	0.5454	0.5800	0.5417
wMLP	0.3525	0.3438	0.4098	0.3537	0.4100	0.3740
Leaf-wMLP	0.4400	0.5417	0.5738	0.4773	0.4200	0.4905
RUS-LGBM	0.7200	0.625	0.7213	0.7272	0.7600	0.7107
SMOTE-LGBM	0.7000	0.6250	0.6885	0.7272	0.72	0.6921
TU-LGBM	0.6400	0.5833	0.5246	0.6818	0.58	0.6019
SADEM	0.7600	0.7500	0.7705	0.7841	0.7800	0.7689

Table 3. Recall@350 of SADEM and the reduced versions.

	Test Time Period (2019)					
Method	*Jan*	*Feb*	*March*	*Apr*	*May*	*Avg.*
SADEM-NoAuto	0.7400	0.7083	0.7377	0.7614	0.7800	0.7455
SADEM-NoCost	0.6600	0.6250	0.6721	0.6932	0.6800	0.6661
SADEM	0.7600	0.7500	0.7705	0.7841	0.7800	0.7689

Table 4. Recall of high-confident predictions (HCP) and augmented pedictions (AP) in top 350 restriction.

	Test Time Period (2019)					
Method	*Jan*	*Feb*	*March*	*Apr*	*May*	*Avg.*
HCP	0.6000 (30/50)	0.5000 (12/24)	0.6066 (37/61)	0.5909 (52/88)	0.6400 (32/50)	⟍
AP	0.4000 (8/20)	0.5000 (6/12)	0.4167 (10/24)	0.4722 (17/36)	0.3889 (7/18)	⟍
SADEM	0.7600 (38/50)	0.7500 (18/24)	0.7705 (47/61)	0.7841 (69/88)	0.7800 (39/50)	0.7689

4.4.2 Imbalance classification models

We compare with supervised learning models focusing on the class imbalanced problem. LGBM [6] is less complex than the neural network model in terms of model complexity, and it performs better when they both apply the cost-sensitive approach. As shown in Table 2, LGBM achieves a better result than the weighted-MLP(wMLP) [35]. As for the data level methods, RUS-LGBM undersamples normal data points randomly, which helps to balance the loss update of both classes and reduce the size of the dataset. Its performance is even better than the original LGBM. As for the SMOTE-LGBM, adding more anomalies by interpolation would potentially increase difficulties in identifying the anomalies when the anomalies uniformly mixed with normal data points. TU-LGBM [28] trains an RNN to learn how to undersample the dataset, but it requires longer training epochs to train the sampler, and the trained model suffers overfitting and results in low performance in testing. When comparing wMLP with Leaf-wMLP, we could find that the Leaf-wMLP is better than wMLP, which is also an evidence that the leaf features are a better representation than original data features as used for distinguishing anomaly data points. Our method adopts distinguishable data features, and we utilize the strong approximation property of a neural network to learn such data representation and adjust the weights of learning criteria between classes so that the model gains improved performance. Our improvement is about 6% when compared with the best baseline model RUS-LGBM.

4.4.3 Effectiveness

In Table 3, we compare the SADEM with reduced versions that aim to check the effectiveness of the model design. The comparison with SADEM-NoAuto shows that Autoencoder helps the Boost Predictor achieve better results because reducing the feature dimension lowers the risk of overfitting. Without cost-sensitive learning, SADEM-NoCost performs worse than SADEM, and it is even worse than LGBM. The reason for this result is that in SADEM-NoCost, both the majority and the minority classes share equal weight to penalize the error when the model falsely makes a prediction. However, it would discourage the model from learning the data features of the minority class due to the population imbalance. We could further observe two groups of the minority class, as shown in Fig. 3 (b): the easy group and the tricky group. The easy group comprises distinguishable points, *i.e.*, anomaly points distancing from normal points, while the tricky group includes the indistinguishable ones, *i.e.*, those lying mixed within normal data points. The SADEM-NoCost model had learned well on the anomaly points in the easy group. Therefore the model outputted only partial cases of high-confident predictions of LGBM mainly from them. Since such cases had been removed from entering low-confident prediction learning (See Fig. 4), the Boost Predictor suffered from even imbalanced data distribution. It then led to a worse result for SADEM-NoCost. Thus we incorporate weight factor λ to control the portion of loss contribution from normal and anomaly classes to counteract this effect. The SADEM-NoAuto and SADEM-NoCost help validate the importance of dimension reduction and cost-sensitive learning in an imbalanced dataset classification problem.

In Table 4, the first two rows are the recall of HCP and AP in the final top 350 watch list. This table demonstrates that both HCP and AP contribute to the final result, and it shows the effectiveness of the Augmented Predictions. Furthermore, in the third row, SADEM, the combination of the output of HCP and AP, achieves a better result than LGBM, as shown in Table 2.

5. CONCLUSION

In this paper, we propose the model SADEM that can solve the alert account detection problem in the banking industry. We proposed an ensemble model to take advantage of machine learning algorithms and neural network models. The low-confident predictions are enhanced by the Boost Predictor, which learns the better representation of leaf features from the gradient boosting decision tree models. Indeed, it helps to distinguish between anomalies and normal ones. We further strengthen our model to tackle the imbalanced dataset problem with the sampling method and cost-sensitive learning. It has shown improvements in online experiments with a real-world dataset from the regional banking industry. Compared with other competitors, *i.e.*, anomaly detection models and imbalanced classification models, the result shows that our model SADEM improves performance by up to 59% at *Recall*@350.

REFERENCES

1. D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, Vol. 22, 2019, pp. 949-961.

2. R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv Preprint*, 2019, arXiv:1901.03407.
3. Y. Yu, J. Long, and Z. Cai, "Network intrusion detection through stacking dilated convolutional autoencoders," *Security and Communication Networks*, Vol. 2017, 2017.
4. R. C. Aygun and A. G. Yavuz, "Network anomaly detection with stochastically improved autoencoder based models," in *Proceedings of IEEE 4th International Conference on Cyber Security and Cloud Computing*, 2017, pp. 193-198.
5. M. Renström and T. Holmsten, "Fraud detection on unlabeled data with unsupervised machine learning," Master Thesis, KTH, School of Engineering Sciences in Chemistry, Biotechnology and Health (CBH), Biomedical Engineering and Health Systems, Health Informatics and Logistics. 2018.
6. G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, 2017, pp. 3146-3154.
7. F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," Technical Report, Cornell Aeronautical Lab Inc., Buffalo, NY, 1961.
8. J. A. Nelder and R. W. Wedderburn, "Generalized linear models," *Journal of the Royal Statistical Society: Series A (General)*, Vol. 135, 1972, pp. 370-384.
9. J. Jiang and L. Yasakethu, "Anomaly detection via one class svm for protection of scada systems," in *Proceedings of International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 2013, pp. 82-88.
10. D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, Vol. 54, 2004, pp. 45-66.
11. K. M. Ting, B.-C. Xu, T. Washio, and Z.-H. Zhou, "Isolation distributional kernel: A new tool for kernel based anomaly detection," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020, pp. 198-206.
12. L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *Proceedings of International Conference on Machine Learning*, 2018, pp. 4393-4402.
13. R. Chalapathy, A. K. Menon, and S. Chawla, "Anomaly detection using one-class neural networks," *arXiv Preprint*, 2018, arXiv:1802.06360.
14. F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proceedings of the 8th IEEE International Conference on Data Mining*, 2008, pp. 413-422.
15. F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, Vol. 6, 2012, pp. 1-39.
16. H. Xiang, Z. Salcic, W. Dou, X. Xu, L. Qi, and X. Zhang, "Ophiforest: Order preserving hashing based isolation forest for robust and scalable anomaly detection," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, 2020, pp. 1655-1664.
17. J. Chen, S. Sathe, C. Aggarwal, and D. Turaga, "Outlier detection with autoencoder ensembles," in *Proceedings of SIAM International Conference on Data Mining*, 2017, pp. 90-98.

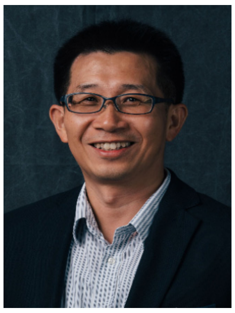
18. S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier detection using replicator neural networks," in *Proceedings of International Conference on Data Warehousing and Knowledge Discovery*, 2002, pp. 170-180.
19. C.-H. Mao, H.-M. Lee, D. Parikh, T. Chen, and S.-Y. Huang, "Semi-supervised co-training and active learning based approach for multi-view intrusion detection," in *Proceedings of ACM Symposium on Applied Computing*, 2009, pp. 2042-2048.
20. S. K. Wagh and S. R. Kolhe, "Effective intrusion detection system using semi-supervised learning," in *Proceedings of IEEE International Conference on Data Mining and Intelligent Computing*, 2014, pp. 1-5.
21. G. Pang, C. Shen, and A. van den Hengel, "Deep anomaly detection with deviation networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 353-362.
22. 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, 2002, pp. 321-357.
23. I. Mani and I. Zhang, "kNN approach to unbalanced data distributions: a case study involving information extraction," in *Proceedings of Workshop on Learning from Imbalanced Datasets*, Vol. 126, 2003, pp. 1-7.
24. N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newsletter*, Vol. 6, 2004, pp. 1-6.
25. X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 39, 2008, pp. 539-550.
26. H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, 2009, pp. 1263-1284.
27. M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1717-1724.
28. M. Peng, Q. Zhang, X. Xing, T. Gui, X. Huang, Y.-G. Jiang, K. Ding, and Z. Chen, "Trainable undersampling for class-imbalance learning," in *Proceedings of AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 4707-4714.
29. K. M. Ting, "A comparative study of cost-sensitive boosting algorithms," in *Proceedings of the 17th International Conference on Machine Learning*, 2000, pp. 983-990.
30. B. Zadrozny, J. Langford, and N. Abe, "Cost-sensitive learning by cost-proportionate example weighting," in *Proceedings of the 3rd IEEE International Conference on Data Mining*, 2003, pp. 435-442.
31. Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 18, 2005, pp. 63-77.

32. Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser, "SVMs modeling for highly imbalanced classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 39, 2008, pp. 281-288.
33. T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785-794.
34. H. Luo, X. Pan, Q. Wang, S. Ye, and Y. Qian, "Logistic regression and random forest for effective imbalanced classification," in *Proceedings of IEEE 43rd Annual Computer Software and Applications Conference*, Vol. 1, 2019, pp. 916-917.
35. Y. S. Aurelio, G. M. de Almeida, C. L. de Castro, and A. P. Braga, "Learning from imbalanced data sets with weighted cross-entropy function," *Neural Processing Letters*, Vol. 50, 2019, pp. 1937-1949.
36. J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, 2001, pp. 1189-1232.
37. N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Proceedings of European Conference on Principles of Data Mining and Knowledge Discovery*, 2003, pp. 107-119.
38. 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, 2010, pp. 552-568.
39. J. Zhang, X. Wu, and V. S. Shengs, "Active learning with imbalanced multiple noisy labeling," *IEEE Transactions on Cybernetics*, Vol. 45, 2014, pp. 1095-1107.
40. S. Pan and X. Zhu, "Graph classification with imbalanced class distributions and noise," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2013, pp. 1586-1592.
41. S. Pan, J. Wu, and X. Zhu, "Cogboost: Boosting for fast cost-sensitive graph classification," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 27, 2015, pp. 2933-2946.
42. Q. Dong, S. Gong, and X. Zhu, "Class rectification hard mining for imbalanced deep learning," in *Proceedings of IEEE International Conference on Computer Vision*, 2017, pp. 1851-1860.
43. Q. Dong, S. Gong, and X. Zhu, "Imbalanced deep learning by minority class incremental rectification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 41, 2018, pp. 1367-1381.
44. E. M. Hand, C. Castillo, and R. Chellappa, "Doing the best we can with what we have: Multi-label balancing with selective learning for attribute prediction," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018, pp. 6878-6885.
45. P. Jeatrakul, K. W. Wong, and C. C. Fung, "Classification of imbalanced data by combining the complementary neural network and smote algorithm," in *Proceedings of International Conference on Neural Information Processing*, 2010, pp. 152-159.

46. X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers *et al.*, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the 8th International Workshop on Data Mining for Online Advertising*, 2014, pp. 1-9.
47. Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, Vol. 184, 2016, pp. 232-242.
48. 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," *Journal of Machine Learning research*, Vol. 12, 2011, pp. 2825-2830.
49. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, 2016, pp. 265-283.
50. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv Preprint*, 2014, arXiv:1412.6980.
51. G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, Vol. 18, 2017, pp. 1-5.

Hui-Kuo Yang received his BS and MS degrees from the National Chiao Tung University, Taiwan, in 1996 and 1998, respectively. He then joined the Industrial Technology Research Institute, Hsinchu County, Taiwan, as a Software Engineer. He was involved in e-commerce and location-based projects, where he applied natural language processing for individual intention analysis. He is working toward his Ph.D. in Computer Science at National Yang Ming Chiao Tung University. His research interests include applications in data mining, and machine learning.

Bing-Li Su received his BS and MS degrees from the National Chiao Tung University, Taiwan, in 2014 and 2018. He is currently working at Synopsys as a Software Engineer, developing and maintaining debug component of the SystemVerilog simulator.



Wen-Chih Peng received his BS and MS degrees from the National Chiao Tung University, Taiwan, in 1995 and 1997, respectively, and his Ph.D. degree in Electrical Engineering from National Taiwan University, Taiwan, in 2001. Currently, he is a Professor in the Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan. Prior to joining the Department of Computer Science, National Yang Ming Chiao Tung University, he was mainly involved in projects related to mobile computing, data broadcasting, and network data management. He has served as a PC member in several prestigious conferences, such as IEEE International Conference on Data Engineering (ICDE), ACM International Conference on Knowledge Discovery and Data Mining (ACM KDD), IEEE International Conference on Data Mining (ICDM), and ACM International Conference on Information and Knowledge Management (ACM CIKM). His research interests include mobile data management and data mining. He is a member of the IEEE.