Theft of Electricity Detection Based on Supervised Learning

Authors: Yichi Yang, Guangzi Xu

School: Hangzhou Foreign Languages School

Hangzhou, Zhejiang, China

Instructors: Yifei Xu, Shuiguang Deng

August 2017

Theft of Electricity Detection Based on Supervised Learning

Yichi Yang, Guangzi Xu

Abstract

With the rapid development of infrastructure including power grids, managers in power industry these days are faced with an increasingly severe problem of theft of electricity. Theft of electricity has negative effects on many socioeconomic aspects, including impacting stable growth of economic for power enterprises and social development. The traditional anti-theft means require officers checking the integrity of kilowatt-hour meter and the correctness of wiring house by house, which requires enormous manpower and material resources. With the advancement of information collecting technology, power enterprises now possess relatively complete database of power consumption. As a result, performing data mining on existing database and identifying abnormal users has become a hot topic in the field of information technology.

The purpose of this paper is to identify abnormal users with machine learning algorithms, providing power enterprises with means to detect theft of electricity at lower cost. The contributions of this paper are listed as follows:

1) Propose various features, providing means to describe users' behaviors. First, monistic features (mean, difference, coefficient of variation, range, standard deviation), binary features (cosine similarity, Pearson product-moment correlation coefficient) and multivariate features are calculated based on user power consumption (monthly, seasonally, yearly, per holiday, per workday). Then principal component analysis is used to perform dimensionality reduction on the dataset. The dataset is finally scaled and oversampled to form the feature dataset.
2) Study various classification algorithms, optimize hyperparameters, providing models to detect theft of electricity. In this paper, the mechanism behind support vector machine, back-propagation neural network, random forest and XGBoost is studied and the hyperparameters are optimized.
3) Compare and evaluate different classifiers, and provide suggestions for real-world application. In this paper, different classifiers are evaluated with different metrics and the best classifier is recommended based on real-world dataset and different application scenarios.

Keywords: theft of electricity, supervised learning, user behavior

Contents

1. Introduction

With the rapid development of China's power industry, the issue of electricity theft has become increasingly prominent. Avoiding or reducing electricity charge illegally seriously disrupts market order and affects the normal profitability of power enterprises. When performing such actions, electricity thieves often temper with kilowatt-hour meters and wires, which causes security risks, for example, in the form of fire hazards[1]. In addition, theft of electricity leads to power shortage and power failure, bringing inconvenience to normal users.

Despite techniques involved in theft of electricity becoming more concealed, statistics inevitably record evidence of illegal behaviors. Thanks to the advancement of information collecting technology, power enterprises now possess relatively complete database of power consumption. As a result, performing data mining on existing database and identifying abnormal users has become a hot topic in the field of information technology[2, 3].

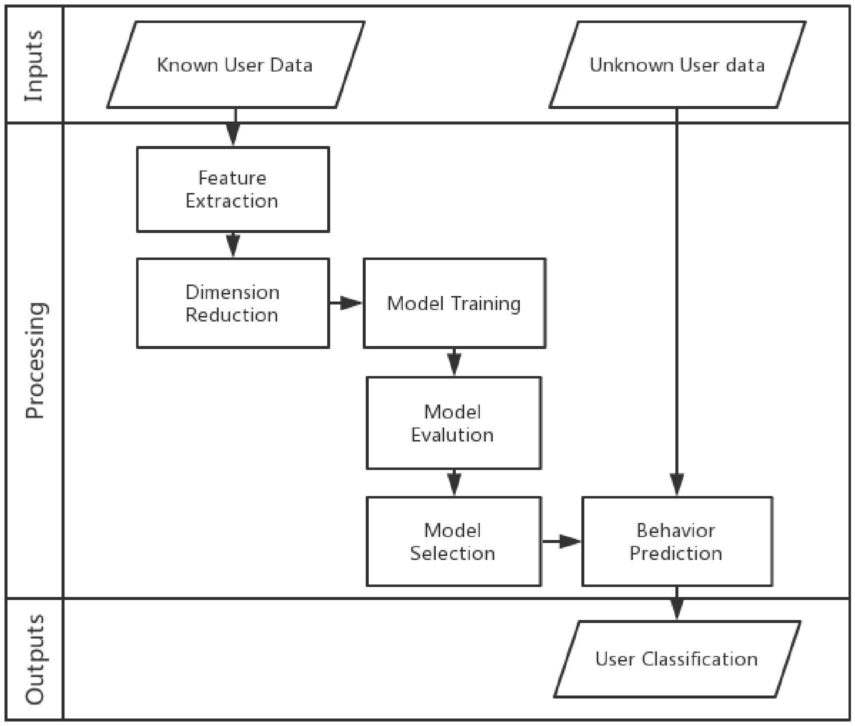

Fig. 1. Theft of electricity detection model

As shown in Fig 1, in this paper, a theft of electricity detection model is proposed based on supervised learning. The model includes inputs, processing, and outputs. In the processing section, various features and high-performance classification models are proposed, which help power enterprises to identify illegal users and reduce the operating cost.

The contributions of this paper are listed as follows:

1) Propose various features and models to describe users' behaviors. First, monistic features (mean, difference, coefficient of variation, range, standard deviation), binary features (cosine similarity, Pearson product-moment correlation coefficient) and multivariate features are computed based on user power consumption (monthly, seasonally, yearly, per holiday, per workday). Then principal component analysis is used reduce the data dimension. The dataset is finally scaled and oversampled to form the feature dataset.
2) Study various classification approaches, optimize hyperparameters, providing models to detect theft of electricity. In this paper, the mechanism of support vector machine, back-propagation neural network, random forest and XGBoost is studied and the hyperparameters are optimized.
3) Compare and evaluate different classifiers, and provide suggestions for real-world application. In this paper, different classifiers are evaluated with different metrics and the best classifier is recommended based on real-world dataset and different application scenarios.

The content of this article is arranged as follows:

In chapter 2, features are proposed to describe user behaviors; dimensionality reduction is performed with principal component analysis; evaluation metrics are listed and their meaning in the context of theft of electricity is discussed. In chapter 3, different classification algorithms are analyzed. In chapter 4, the experiment is discussed. The experiment includes datasets processing, hyperparameters optimization, and results analysis. Chapter 5 is the conclusion, and possible future improvements are proposed.

2. User Behavior Description

2.1 Feature Extraction

User behaviors can be described with features. To form a dataset consisting of various features, a power consumption list of **H** days for each user is derived from the raw dataset. The power consumption list for nth user is denoted by a vector $x_n=\{x_n^{(h)},h\in \text{Dates}\}$, where Dates is a set of the date with consumption records. A dataset containing all users daily power consumption vectors can be denoted by $X=\{x_n,n=1,2,3,\ldots,N\}$, where **N** is the number of users in total. Monthly, seasonally, yearly, per holiday, per workday, etc. power consumption datasets are then obtained based on the daily power consumption dataset. Features are calculated with following statistical indicators, where a_i is a data element, while T,K are data vectors. According to the number of involved features, features are clustered into monistic features, binary features, and multivariate features.

$$A_{(\text{Coefficient of Variation})}=\frac{\sqrt{\sum_{i=1}^{n}(a_i)^2}}{\frac{1}{n}\sum_{i=1}^{n}a_i} \tag{1}$$

$$B_{(\text{Standard Deviation})}=\sqrt{\sum_{i=1}^{n}(a_i)^2} \tag{2}$$

$$C_{(\text{Range})}=\text{Max}(a_i)\text{-Min}(a_i) \tag{3}$$

$$D_{(\text{Mean})}=\frac{1}{n}\sum_{i=1}^{n}a_i \tag{4}$$

$$E_{(\text{Cosine Similarity})}=\frac{T\cdot K}{\|T\|\|K\|} \tag{5}$$

$$F_{(\text{PPMCC})}=\frac{\text{cov}(T,K)}{\sigma T\sigma K} \tag{6}$$

Where PPMCC is short for Pearson product-moment correlation coefficient.

2.1.1 Monistic Features

1) Yearly Power Consumption

Coefficient of variation $A_1^{(1)}(A_{1\text{-}2014}^{(1)},A_{1\text{-}2015}^{(1)},A_{1\text{-}2016}^{(1)})$, standard deviation $B_1^{(1)}(B_{1\text{-}2014}^{(1)},B_{1\text{-}2015}^{(1)},B_{1\text{-}2016}^{(1)})$, range $C_1^{(1)}(C_{1\text{-}2014}^{(1)},C_{1\text{-}2015}^{(1)},C_{1\text{-}2016}^{(1)})$, mean $D_1^{(1)}(D_{1\text{-}2014}^{(1)},D_{1\text{-}2015}^{(1)},D_{1\text{-}2016}^{(1)})$ of yearly power consumption. 3+3+3+3=12 features in total.

2) Holiday and Non-working Day Power Consumption

Coefficient of variation $A_2^{(1)}$, standard deviation $B_2^{(1)}$, range $C_2^{(1)}$, mean $D_2^{(1)}$ of the ratio of holiday power consumption to daily power consumption; Coefficient of variation $A_2^{(2)}$, standard deviation $B_2^{(2)}$, range $C_2^{(2)}$, mean $D_2^{(2)}$ of the ratio of working day power consumption to non-working day power consumption. 4+4=8 features in total.

3) Difference Between First 5 Months and Last 5 Months

Yearly power consumption difference between first 5 months and last 5 months $H_8^{(1)}(H_{8\text{-}2014}^{(1)},H_{8\text{-}2015}^{(1)},H_{8\text{-}2016}^{(1)})$, coefficient of variation $A_8^{(1)}$, standard deviation $B_8^{(1)}$, range$C_8^{(1)}$, mean$D_8^{(1)}$ of yearly power consumption difference. 3+4=7 features in total.

2.1.2 Binary Features

1) Monthly Power Consumption Correlation Between Two Years

Cosine similarity $E_3^{(1)}(E_{3\text{-}2014\&2015}^{(1)},E_{3\text{-}2015\&2016}^{(1)})$, PPMCC $F_3^{(1)}(F_{3\text{-}2014\&2015}^{(1)},F_{3\text{-}2015\&2016}^{(1)})$ between two years' monthly power consumption. PPMCC difference $G_3^{(1)}=\left|E_{3\text{-}2014\&2015}^{(1)}-E_{3\text{-}2015\&2016}^{(1)}\right|$. 2+2+1=5 features in total.

2) Holiday Power Consumption Correlation Between Two Years

Cosine similarity $E_4^{(1)}(E_{4\text{-}2014\&2015}^{(1)},E_{3\text{-}2015\&2016}^{(1)})$, PPMCC $F_4^{(1)}(F_{4\text{-}2014\&2015}^{(1)},F_{4\text{-}2015\&2016}^{(1)})$ between two years' holiday power consumption. PPMCC difference $G_4^{(1)}=\left|E_{4\text{-}2014\&2015}^{(1)}-E_{4\text{-}2015\&2016}^{(1)}\right|$. 2+2+1=5 features in total.

3) Seasonal Power Consumption Correlation Between Two Years

Cosine similarity $E_5^{(1)}(E_{5\text{-}2014\&2015}^{(1)},E_{5\text{-}2015\&2016}^{(1)})$, PPMCC $F_5^{(1)}(F_{5\text{-}2014\&2015}^{(1)},F_{5\text{-}2015\&2016}^{(1)})$ between two years' seasonal power consumption. PPMCC difference $G_5^{(1)}=\left|E_{4\text{-}2014\&2015}^{(1)}-E_{4\text{-}2015\&2016}^{(1)}\right|$. 2+2+1=5 features in total.

4) Power Consumption Correlation Between a Particular User and Normal Average

Normal average power consumption is the average consumption of all legal users. The difference between unknown users' power consumption data and normal average power consumption data can be used to identify abnormal users.

a) Cosine similarity $E_6^{(1)}(E_{6\text{-}2014}^{(1)},E_{6\text{-}2015}^{(1)},E_{6\text{-}2016}^{(1)})$, PPMCC $F_6^{(1)}(F_{6\text{-}2014}^{(1)},F_{6\text{-}2015}^{(1)},F_{6\text{-}2016}^{(1)})$ between monthly power consumption of a particular user and normal average.

b) Cosine similarity $E_6^{(2)}(E_{6\text{-}2014}^{(2)},E_{6\text{-}2015}^{(2)},E_{6\text{-}2016}^{(2)})$, PPMCC $F_6^{(2)}(F_{6\text{-}2014}^{(2)},F_{6\text{-}2015}^{(2)},F_{6\text{-}2016}^{(2)})$ between seasonal power consumption of a particular user and normal average.

c) Cosine similarity $E_6^{(3)}(E_{6\text{-}2014}^{(3)},E_{6\text{-}2015}^{(3)},E_{6\text{-}2016}^{(3)})$, PPMCC $F_6^{(3)}(F_{6\text{-}2014}^{(3)},F_{6\text{-}2015}^{(3)},F_{6\text{-}2016}^{(3)})$ between holiday power consumption of a particular user and normal average.

6+6+6=18 features in total.

2.1.3 Multivariate Features

Gradient of power consumption indicates consumption difference between two consecutive days, which reflects the trend of variation in power consumption. Gradient difference is the difference between a particular user and the average gradient of all legal users, which can be used to identify abnormal users.

a) Coefficient of variation $A_7^{(1)}$, standard deviation $B_7^{(1)}$, range$C_7^{(1)}$, mean $D_7^{(1)}$ of the list of power consumption gradients.

b) Coefficient of variation $A_7^{(2)}$, standard deviation $B_7^{(2)}$, range $C_7^{(2)}$, mean$D_7^{(2)}$ of the list of gradient differences.

c) Cosine similarity $E_7^{(3)}$, PPMCC $F_7^{(3)}$ between a particular user's list of power consumption gradients and the average list of gradients.

4+4+2=10 features in total.

2.2 Dimensionality Reduction

Principal component analysis (PCA) is a dimensionality reduction algorithm. After performing PCA, a high dimensional dataset is converted into a smaller dataset with a set of linearly uncorrelated principal components using orthogonal linear transformation. In the procedure of PCA, principal components with small variances are discarded to reduce the dimension of the dataset while keeping most of the information[4, 5].

Let there be a dataset $X=[x_1\ x_2\ x_3\ldots\ldots x_n]$ with n features, $x_1, x_2,x_3\ldots\ldots x_n$. m principal components of the dataset X, $PC_1, PC_2,PC_3\ldots\ldots PC_m$, can be calculated as shown below, where $A_i^T=[x_{i1}\ x_{i2}\ x_{i3}\ldots\ldots x_{in}]$ is the loading of the i^th principal component.

$$\begin{cases} PC_1=A_1^T X \\ PC_2=A_2^T X \\ PC_3=A_3^T X \\ \quad\cdots\cdots \\ PC_m=A_m^T X \end{cases} \tag{7}$$

2.3 Evaluation Metrics Analysis

A classification model's performance can be evaluated with different metrics. Accuracy, precision and recall are used to evaluate classifiers in this experiment. Accuracy, precision and recall are calculated as follows, where TP denotes true positives, FN denotes false negatives, FP denotes false positives, and TN denotes true negatives.

$$\text{precision}=\frac{TP}{TP+FP} \tag{8}$$

$$\text{recall}=\frac{TP}{TP+FN} \tag{9}$$

$$\text{accuracy}=\frac{TP+TN}{TP+FN+FP+TN} \tag{10}$$

In the scenario of theft of electricity detection, accuracy is the ratio of correct classification, precision is the ratio of true illegal users in all predicted illegal users, and recall is the ratio of predicted illegal users in all true illegal users. To accurately identify illegal users without disturbing normal users by labeling them as illegal users, precision is the main metric used to evaluate a classifier in this experiment.

3. User Behavior Classification

To detect theft of electricity, different classifiers are used to classify user behaviors. In this chapter four algorithms—support vector machine, backpropagation neural network, random forest, and XGBoost—and the function of their hyperparameters are analyzed.

3.1 Support Vector Machine

Support vector machine (SVM) can detect nonlinear patterns in a high dimension dataset with small sample size, and be used in other machine learning problems such as function fitting[6]. As shown in Fig 2, when the dataset is linearly separable, SVM searches for a hyperplane which has the maximum distance (or margin) to nearest data points from both classes, so future data points can be classified into one of the two classes depending its position relative to the hyperplane.

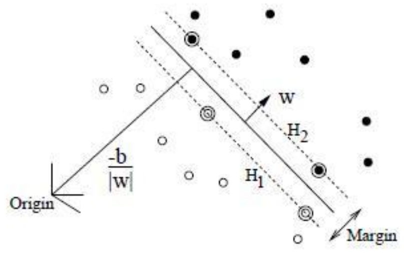

Fig. 2. Diagram of SVM

The problem of maxing the margin can be rewritten as constrained optimization problem (11), which then can be solved to find the optimal parameters, w and b, and then the hyperplane can be determined.

$$\begin{cases} \frac{\min\|w\|^2}{2} \\ \text{s.t.} \quad y_i(w\cdot x_i+b)-1\geq 0 \end{cases} \tag{11}$$

When a dataset cannot be easily separated, SVM can expand the feature space by projecting the dataset into higher dimensions using nonlinear transformation (kernel methods). The dataset can then be separated in higher dimensional space. Common kernel function include linear kernel and radial basis function (RBF).

3.2 Backpropagation Neural Network

Backpropagation neural network is a kind of neural network trained with backpropagation algorithm, and is the most widely used neural network today[7]. A backpropagation neural network has one input layer, one output layer, and one or more hidden layer(s). Its training process mainly includes two steps, forward transmission of the feed and backpropagation of the error. The structure of a backpropagation neural network is shown below in Fig 3.

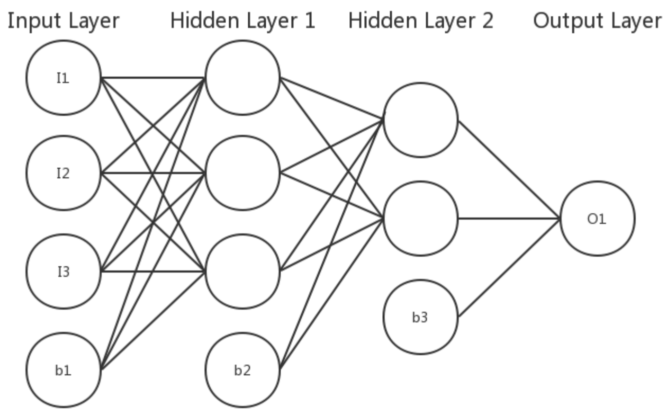

Fig. 3. Diagram of Backpropagation Neural Network

Backpropagation is commonly used in conjunction with gradient descent algorithm to optimize the weights between neurons and biases. In a backpropagation neural network with m-2 hidden layers and activation function f(·), in each step Δw_{ij}^{k} is subtracted from w_{ij}^{k}, the weight connecting ith neuron in (k-1)th layer and jth neuron in kth layer, to correct the error, where E is the cost function. Δw_{ij}^{k} is given by $\Delta w_{ij}^{k} = \eta \frac{\partial E}{\partial w_{ij}^{k}} = \eta \cdot \frac{\partial E}{\partial x_j^k} \cdot z_i^{k-1}$, where η is the training step size. $\frac{\partial E}{\partial x_j^k}$ can be calculated by solving following recursive formula (12), where x is the input of a neuron, z is the output of neuron, and y is the expected output.

$$\begin{cases} \frac{\partial E}{\partial x_j^k} = \left(z_j^m - \mathrm{y_j}\right) \cdot f'\left(x_j^m\right), (k = m) \\ \frac{\partial E}{\partial x_j^k} = d_j^{k+1} \cdot w_{ij}^{k+1} \cdot f'\left(x_j^k\right), (k < m) \end{cases} \qquad (12)$$

When the error is lower than a predetermined acceptable value (or training steps exceed a set limit), training is completed, otherwise the process of feed forward transmission and error backpropagation is repeated, and weights and biases are corrected each time. Both the structure of the network (the number of hidden layers and the number of neurons per layer) and step size η affect the performance of a backpropagation neural network.

3.3 Random Forest

Random forests are an ensemble learning method. When constructing a random forest, a random sample is selected with replacement from the original dataset with a sample size of N. The process is repeated n time and n samples are selected. In all features, m features are randomly selected. A decision tree is then fitted to the dataset with n samples and m features. This process is repeated k times to generate k decision trees, forming a random forest.

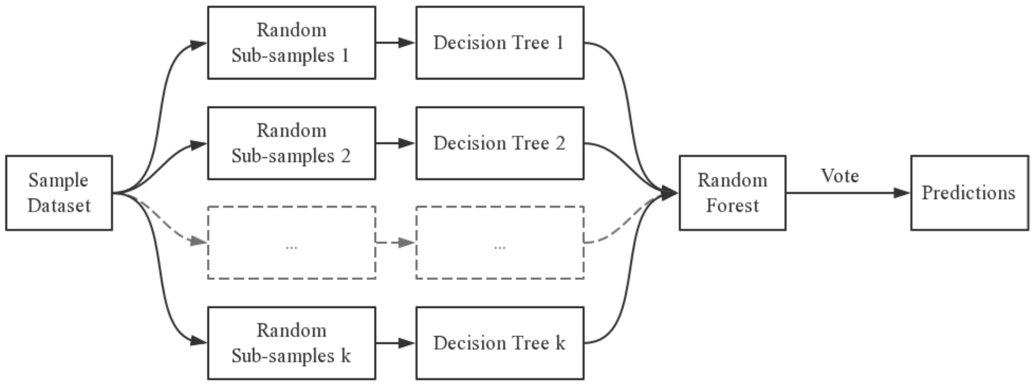

Fig. 4. Diagram of Random Forest

As shown by Fig 4, when making predictions for unseen sample, the decision is made by taking majority vote in all decision trees. Random forests are insensitive to multicollinearity and is robust to missing or unbalanced data [8]. Max tree depth controls the complexity of decision trees, and the number of decision tress controls the size of the random forest, which are two crucial hyperparameters.

3.4 XGBoost

XGBoost is an open source implementation of gradient boosting decision tree (GBDT). It overcomes traditional GBDT’s shortcomings, such as difficulty to parallelize and having high complexity. Despite being GBDT, XGBoost has a cost function with regularization term, and supports customizing classifier and cost function. It has advantages in being easy to parallelize, having low complexity, and having low possibility of overfitting [9-11].

Different from traditional GBDT, XGBoost defines an objective function (13) as the cost function for supervised learning.

$$\mathrm{Obj}=\sum_{i=1}^{n} l\left(y_i,\hat{y}_i\right)+\sum_{i=1}^{n}\Omega(f_k) \tag{13}$$

The regularization term, $\sum_{i=1}^{n}\Omega(f_k)$, controls the complexity of the model, preventing it from overfitting.

$$\Omega(f_t)=\gamma T+\frac{1}{2}\lambda\sum_{j=1}^{T} w_i^2 \tag{14}$$

XGBoost reduces the value of objective function by performing additive training. In each training step, the set of classification and regression trees (CART) remains unchanged, and a new CART is added to the set to achieve a maximum reduction in objective function. Shown below is the objective function in tth step (15), simplified by a second-order Taylor expansion and omitting the constant, where $g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i,\hat{y}_i)$ and $h_i = \partial^2_{\hat{y}_i} l(y_i,\hat{y}_i)$. It can be seen XGBoost can use any second order differentiable function as its cost function.

$$\mathrm{Obj}^{(t)}=\sum_{i=1}^{n}\left[g_i f_t(x_i)+\frac{1}{2}h_i f_t^2(x_i)\right]+\Omega(f_t) \tag{15}$$

After expending the regularization term, the objective function in tth step can be rewritten as function (16).

$$\mathrm{Obj}^{(t)}=\sum_{j=1}^{T}[G_j w_j+\frac{1}{2}\left(H_j+\lambda\right)w_i^2]+\gamma T \tag{16}$$

For a given t, the objective function in tth step is a quadratic function on w_j, from which we can find its minimum value (17).

$$\mathrm{Obj}^{*}=-\frac{1}{2}\sum_{j=1}^{T}\frac{G_j}{H_j+\lambda}+\gamma T \tag{17}$$

And the decrement of the objective function when splitting a node in a CART, or splitting gain, can be given as follows：

$$\text{Gain}=\frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda}+\frac{G_R^2}{H_R+\lambda}-\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right]-\gamma \tag{18}$$

When creating a CART, splitting possibilities are traversed, and the one with maximum gain is selected. When making predictions for unseen sample, the decision is made by averaging outputs of all CARTs. Additionally, XGBoost also features row subsampling (random instance sampling) and column subsampling (random feature sampling), which increases training speed and avoids overfitting. Learning rate, the number of CARTs, and the maximum depth of CARTs, etc. affect the classification performance of XGBoost.

4. Experiment and Result Analysis

4.1 Raw User Datasets

Raw user datasets consist of users' daily power consumption records in a certain place from Jan. 2014 to Oct. 2016. The ratio of illegal users to normal users is 3799 to 40419. Raw dataset 1 contains power consumption data, while raw dataset 2 contains labels indicating if a user is a normal user or illegal user, as shown in Table 1 and Table 2.

Table 1. Raw dataset 1

Field	Description	Example
CONS_NO	User ID	0E13DA5C01DEEFA7 E11B5DD0BAA1542A
DATA_DATE	Date	2014/1/1
KWH_READING	Reading at the End of the Day	626.64
KWH_READING1	Reading at the Beginning of the Day	625.75
KWH	Daily Power Consumption	0.89

Table 2. Raw dataset 2

Field	Description	Example
CONS_NO	User ID	0E13DA5C01DEEFA7E11 B5DD0BAA1542A
LABEL	0: Normal User; 1: Illegal User	0

Records in dataset 1 are sorted by date, and spans 1034 days. Each user has one unique user ID, while having power consumption records of multiple days. After removing null and invalid records, there is a total of 42026 valid users in the dataset. Combined with dataset 2, dataset 1 describes two types of users: class 0 indicates normal users; class 1 indicates illegal users. The number of users in class 0 is 38264, and the number of users in class 1 is 3762, with a ratio of 10.2:1.

4.2 Dataset Pre-processing and Cross Validation Method Selection

4.2.1 Dataset Processing

In the training process, data are processing with the following steps: data cleaning, feature extraction, sampling, and scaling.

1）Data Cleaning

First, null and invalid records are removed. Raw dataset 1 includes user ID, date, reading at the end of the day, reading at the beginning of the day, and daily power consumption (as shown in Table 1). Since daily power consumption is the difference between the reading at the end of the day and the reading at the beginning of the day, reading at the end of the day and the reading at the beginning of the day are redundant and then is removed.

After cleaning, the dataset only includes user ID, date, and daily power consumption. The dataset is then sorted by user ID and date before performing feature extraction.

2）Feature Extraction

Based on the methods mentioned in chapter 2, two datasets with different features are generated. Dataset 1 has 42026 samples and 70 features. 70 Features include all monistic, binary and multivariate features. Dataset 2 has 42026 samples and 5 features. 5 features are mean, minimum, maximum, variance and medium of daily power consumption, respectively.

3）Sampling

Because the dataset is biased (class 0: class 1 = 10.2:1), samples in class 1 are over sampled to reduce negative effects on classification performance. After over sampling, class 0 and class 1 have the same sample size.

3）Scaling

Different features have different variances. To prevent features with significantly larger variance from having stronger effects in training, features are scaled using formula (19), where Fea is the original feature, and $\mathrm{Fea}_{\mathrm{norm}}$ is the feature normalized.

$$\mathrm{Fea}_{\mathrm{norm}}=\frac{\mathrm{Fea}\text{-}\mathrm{Min}(\mathrm{Fea})}{\mathrm{Max}(\mathrm{Fea})\text{-}\mathrm{Min}(\mathrm{Fea})} \tag{19}$$

4.2.2 Cross Validation Method Selection

Cross validation is a technique used to evaluate the model’s ability to be generalized. A common cross validation method is random splitting, where data is randomly split into two parts. One part is used to train the classification model, while the other part is held out for testing the model after training. One drawback of this method is that not all the data are used to train the classification model.

To use data sufficiently, the strategy called stratified k-folds is used in our experiment. When using stratified k-folds to evaluate the model, data other than evaluation set are split into k folds, and each fold contains roughly the same proportions of the two types of class labels. When training, k-1 folds are used to train the model, while 1 fold is used for testing. This is repeated k times. Therefore every fold is used as test set once, and average accuracy, precision and recall of the model are calculated for evaluation [12].

4.3 Hyperparameter Analysis and Tuning

4.3.1 Support Vector Machine Analysis and Tuning

C and kernel affect the performance of support vector machine.

1) C

C is the penalty for misclassifying an instance. When C is too small, the model tends to underfit, reducing its precision; when C is too large, the model tends to overfit. Grid search is a popular way to determine optimal hyperparameters. With other hyperparameters fixed at default, grid search is performed, the optimal value is determined to be 1000.

2) Kernel

When a dataset cannot be easily separated, SVM projects the dataset into higher dimensions using kernel methods, and then separates data in higher dimensional space. One kernel commonly used is RBF. However, after experiments, linear kernel is chosen for its higher speed and precision.

4.3.2 Backpropagation Neural Network Analysis and Tuning

learning_rate_init and hidden_layer_sizes are two hyperparameters that affect convergence speed and training results for a backpropagation neural network.

1) learning_rate_init

Gradient descent algorithm is commonly used in training backpropagation neural network. learning_rate_init is the initial learning rate, which controls the step-size in updating the weights and influences the convergence speed and training results. In general, learning_rate_init is negatively correlated with the training time. Namely, when learning_rate_init is too small, time spent training the model is too long; when learning_rate_init is too big, model tends to oscillate and does not convergence. In order to find optimal learning_rate_init, the following experiment is conducted to tune the hyperparameter. With hidden layers remain unchanged, different values for learning_rate_init, 0.0005, 0.001, 0.003, 0.005, 0.007, 0.01, are tested, and corresponding result are shown in Table 3 and Fig 5.

Table 3. Backpropagation neural network's performance for different learning_rate_init and hidden_layer_sizes

hidden_layer_sizes	learning_ rate	Accuracy	Precision	Recall	Time(s)
85	0.0005	80.26%	79.55%	81.75%	215.52
85	0.001	82.58%	81.81%	83.85%	172.06
85	0.003	82.90%	81.83%	84.60%	160.63
85	0.005	79.92%	78.74%	82.24%	170.63
85	0.007	76.82%	73.32%	84.41%	142.14
85	0.01	74.08%	71.80%	79.79%	128.15
30	0.003	75.10%	73.68%	78.22%	88.70
50	0.003	79.26%	79.05%	79.91%	118.30
70	0.003	81.95%	81.10%	83.51%	135.99
100	0.003	83.58%	81.94%	86.13%	231.29
110	0.003	84.90%	83.11%	87.71%	222.24

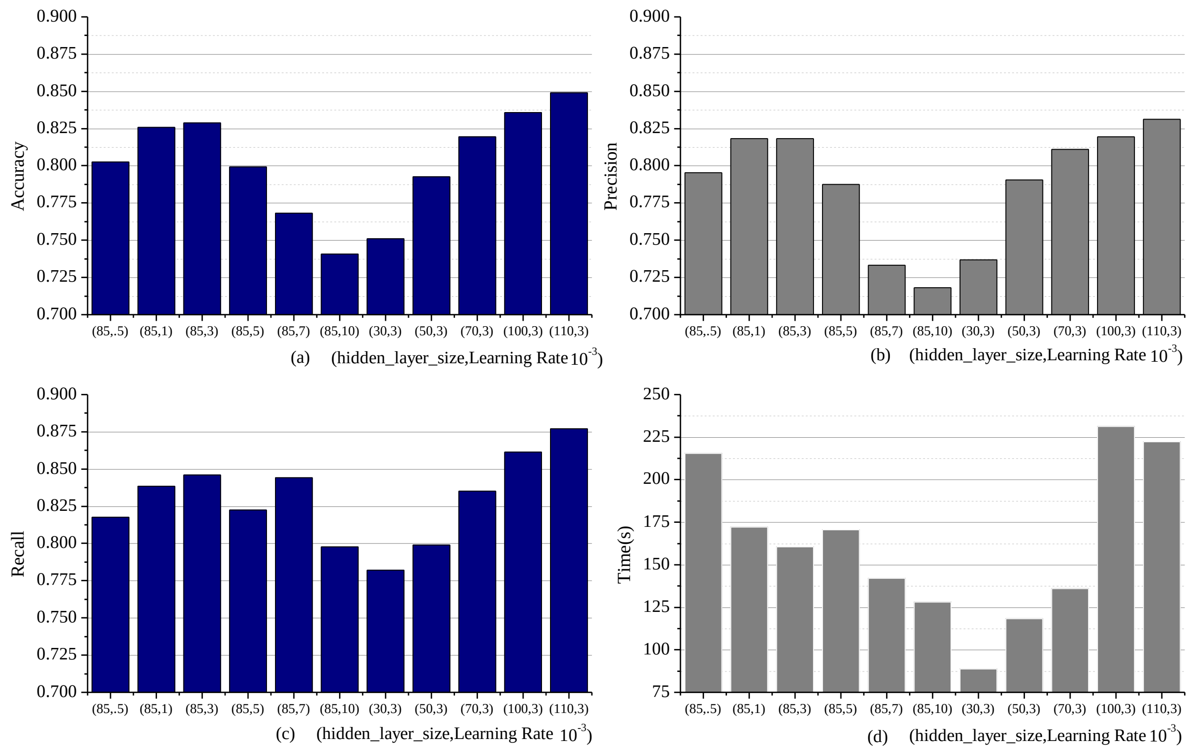

Fig. 5. Backpropagation neural network's performance for different learning_rate_init and hidden_layer_sizes

As shown by Table 3, it can be concluded when learning_rate_init is set to 0.003 or 0.001, the model obtains highest precision. However, setting learning_rate_init to 0.001 results in negligible performance improvement, and requires more training time, so 0.003 is selected as best learning_rate_init.

2) hidden_layer_sizes

Numbers of neurons in each hidden layer affect convergence speed and training results. Increment in the number of hidden layers and number of neurons per hidden layer results in increment in precision and training time, and the increments is minimum after passing the point of diminishing returns. To determine the optimal structure of the neural network, following experiment is conducted.

With other hyperparameters at default, number of hidden layers is set to 2 and different numbers of neurons per layer are tested. As shown by Fig 5, when the number of neurons per layer is smaller than 85, the increment in precision is significant. However, when it is larger than 85, the training time increases significantly. Consequently, it can be concluded that 85 is the best number of neurons per layer.

4.3.3 Random Forest Analysis and Tuning

To tune random forest, experiments are conducted to determine the optimal value of max_depth and n_estimators.

1) max_depth

max_depth is the maximum depth limit for decision trees. When the value is too large, the complexity of the model and the risk of overfitting increases. When the value is too small, underfitting may occur, which will reduce the precision. To find optimal limit for tree depth, different settings are testes.

As shown by Table 4 and Fig 6, when max_depth is set to None (no tree depth limit), best precision is achieved. Hence, None is selected for max_depth.

Table 4. Random Forest's performance for different n_estimators and max_depth

n_estimators	max_depth	Accuracy	Precision	Recall	time (s)
10	10	82.03%	80.35%	84.80%	17.75
20	10	83.11%	81.63%	85.47%	32.64
30	10	83.22%	81.49%	86.00%	46.14
40	10	83.13%	81.49%	86.00%	64.45
50	10	83.83%	82.29%	86.05%	77.65
40	20	97.75%	96.54%	86.21%	86.40
40	30	98.66%	98.28%	99.05%	90.84
40	None	98.68%	98.32%	99.05%	91.85
30	30	98.57%	98.13%	99.06%	68.31
20	30	98.55%	98.06%	99.04%	45.95
10	30	98.40%	97.79%	99.05%	22.37
5	30	97.03%	95.23%	99.03%	10.75
10	None	98.52%	98.01%	99.05%	22.85

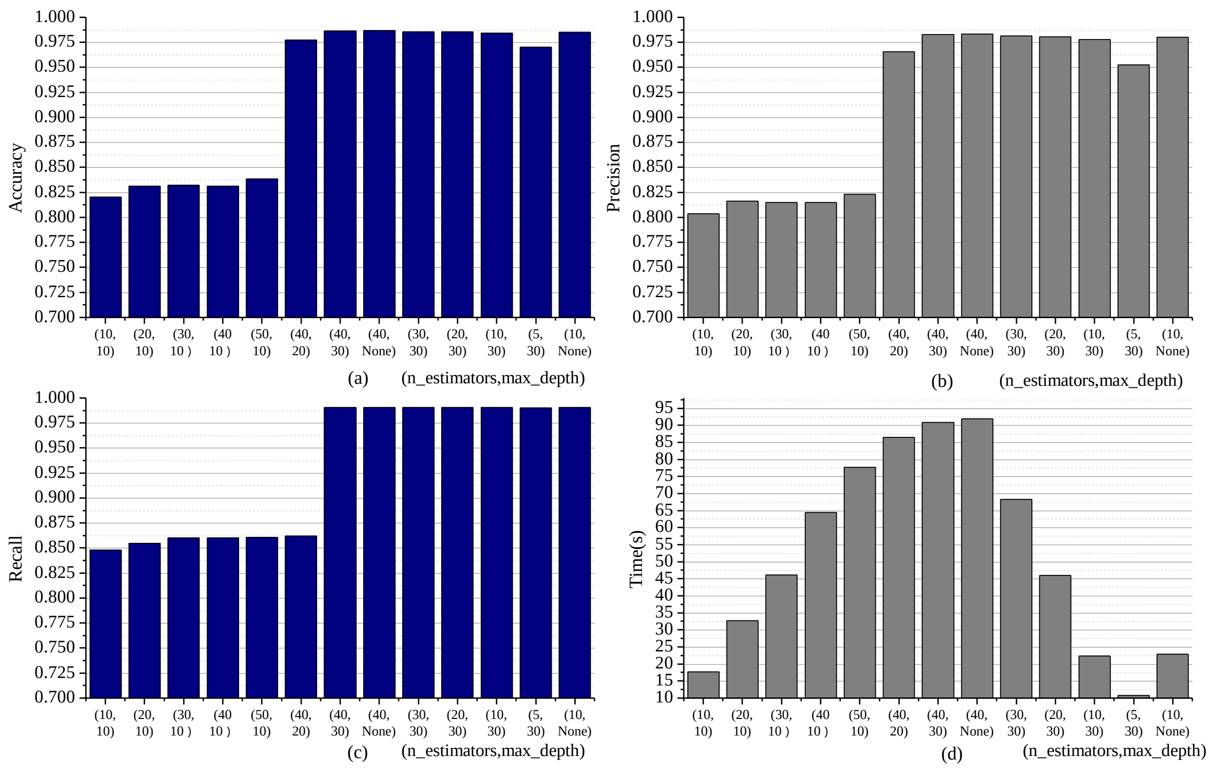

Fig. 6. Random Forest's performance for different n_estimators and max_depth

2) n_estimators

n_estimators is the number of CARTs used in a random forest. When n_estimators is too small, the model shows a significantly lower precision; when n_estimators is too big, precision and training time are both increased. To determine its optimal value, with other hyperparameters at default and max_depth set to None (determined in previous experiment), 10, 20, 30, 40, and 50 are tested.

As shown by Fig 6, with other hyperparameters remain unchanged, when n_estimators exceeds 10, the increment in precision in small. To save time while maintaining precision, 10 is selected for n_estimators.

4.3.4 XGBoost Analysis and Tuning

Hyperparameters are set before starting the training process, and directly affect the performance of the model. In XGBoost, important hyperparameters include learning_rate and n_estimators for controlling the learning rate and number of iterations; max_depth, min_child_weight and gamma for controlling CART growth; subsamples and colsample_bytree for controlling row and column subsampling; and reg_lambda for controlling regularization.

1) learning_rate and n_estimators

learning_rate is the shrinkage of weight when generating new CARTs. n_estimators is the number of CARTs in the ensemble. Normally, the smaller the learning_rate, the more robust the classifier is, and the longer training takes. To find the balance between the training time and performance and to find the optimal hyperparameters in controllable time, learning_rate is set to a higher value initially, so other hyperparameters can be quickly tuned. After other hyperparameters are set to optimal values, learning_rate is decreased and n_estimators increased.

2) max_depth 、 min_child_weight and gamma

XGBoost is an ensemble learning method consisting of CARTs. max_depth, min_child_weight, and gamma control the structure of CARTs in the ensemble. max_depth limits the maximum depth of CARTs; min_child_weight limits the minimum sum of instance weight in a leaf node; gamma limits the minimum gain in leaf node partition. Increasing min_child_weight gamma make the classifier more conservative, while increasing max_depth makes the classifier more complex and more likely to overfit. To determine optimal values, a grid search is performed, and best values for max_depth, min_child_weight and gamma are found to be 9,1 and 0, respectively, as shown in Table 5, Table 6 and Fig 7.

Table 5. XGBoost's performance for different max_depth and min_child_weight

max_depth	min_child_weight	accuracy	precision	recall
3	1	87.62%	84.25%	92.52%
3	3	86.85%	83.65%	91.59%
3	5	86.29%	83.23%	90.88%
5	1	95.42%	92.49%	98.87%
5	3	94.87%	91.64%	98.75%
5	5	94.29%	90.87%	98.48%
7	1	96.81%	94.73%	99.14%
7	3	96.29%	93.81%	99.12%
7	5	95.87%	93.11%	99.08%
9	1	97.14%	95.33%	99.14%
9	3	96.57%	94.29%	99.14%
9	5	96.25%	93.74%	99.13%

Table 6. XGBoost's performance for different gamma

gamma	accuracy	precision	recall
0	97.14%	95.33%	99.14%
0.1	96.96%	95.00%	99.14%
0.2	96.68%	94.51%	99.11%
0.3	96.37%	94.00%	99.06%
0.4	96.33%	93.94%	99.06%

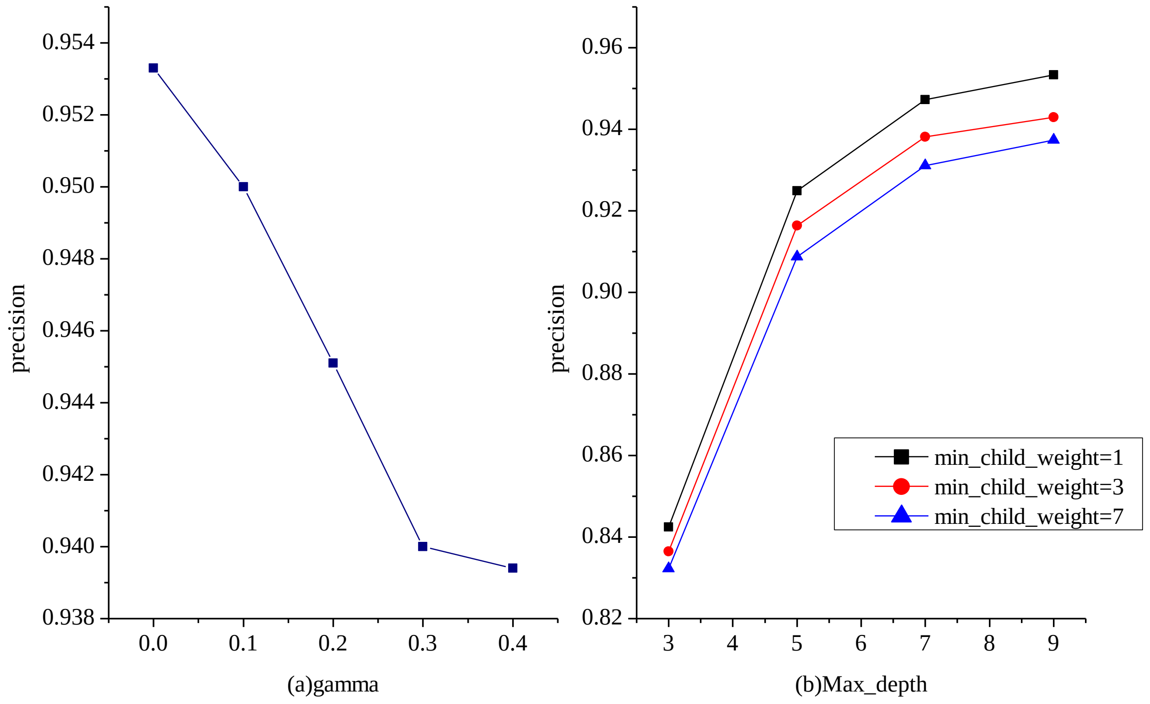

Fig. 7. XGboost's performance for different max_depth, min_child_weight and gamma

3) subsample and colsample_bytree

XGBoost features row subsampling and column subsampling to avoid overfitting and reduce computational complexity. subsample is the ratio of instance subsampling, and colsample_bytree is the ratio of column subsampling when constructing CARTs. To determine their optimal values, a grid search is performed, and best values for subsampling and colsample_bytree are found to be 0.9 and 0.8, respectively, as shown by Table 7, and Fig 8.

Table 7. XGBoost's performance for different subsample and colsample_bytree

subsample	colsample_bytree	accuracy	precision	recall
0.6	0.6	96.16%	93.58%	99.13%
0.6	0.7	96.16%	93.58%	99.13%
0.6	0.8	96.34%	93.90%	99.12%
0.6	0.9	96.34%	93.90%	99.12%
0.7	0.6	96.32%	93.88%	99.10%
0.7	0.7	96.32%	93.88%	99.10%
0.7	0.8	96.40%	94.00%	99.12%
0.7	0.9	96.40%	94.00%	99.12%
0.8	0.6	96.28%	93.82%	99.09%
0.8	0.7	96.28%	93.82%	99.09%
0.8	0.8	96.40%	94.01%	99.11%
0.8	0.9	96.40%	94.01%	99.11%
0.9	0.6	96.37%	93.97%	99.10%
0.9	0.7	96.37%	93.97%	99.10%
0.9	0.8	96.54%	94.28%	99.09%
0.9	0.9	96.54%	94.28%	99.09%

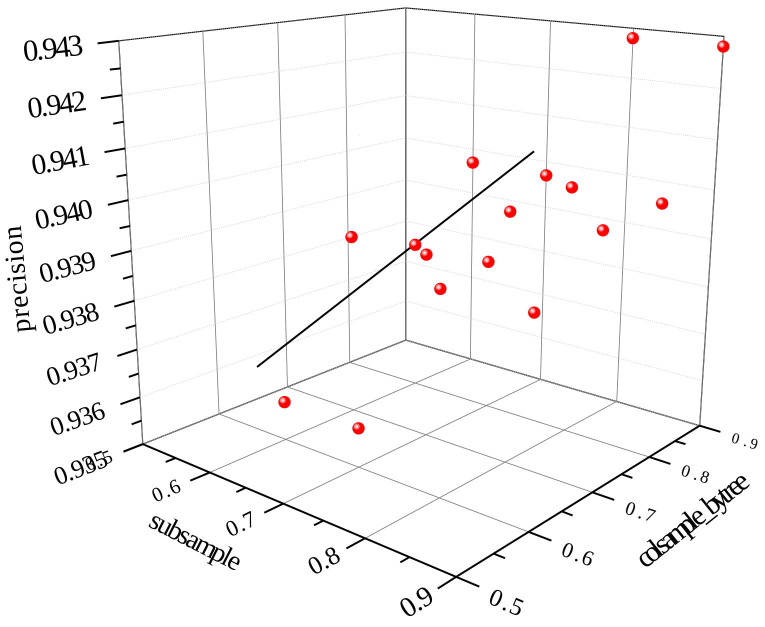

Fig. 8. XGBoost's performance for different subsample and colsample_bytree

4) reg_lambda

reg_lambda is the weight for L2 regularization term. Higher reg_lambda value makes the model more conservative. In the experiment, the optimal reg_lambda value 5E-05 was selected by using grid search (see Table 8 and Fig 9).

Table 8. XGBoost's performance for different reg_lambda

reg_lambda	accuracy	precision	recall
0.00E+00	96.47%	94.13%	99.12%
5.00E-06	96.47%	94.13%	99.12%
1.00E-05	96.47%	94.13%	99.12%
5.00E-05	96.53%	94.23%	99.12%
1.00E-04	96.51%	94.20%	99.12%
1.00E-03	96.47%	94.13%	99.12%
1.00E-01	96.51%	94.23%	99.09%
3.00E-01	96.50%	94.17%	99.12%
5.00E-01	96.43%	94.06%	99.12%
6.00E-01	96.51%	94.21%	99.12%
7.00E-01	96.50%	94.18%	99.12%
1.00E+00	96.35%	93.92%	99.11%
5.00E+00	95.20%	92.07%	98.91%
1.00E+02	71.20%	71.12%	71.34%

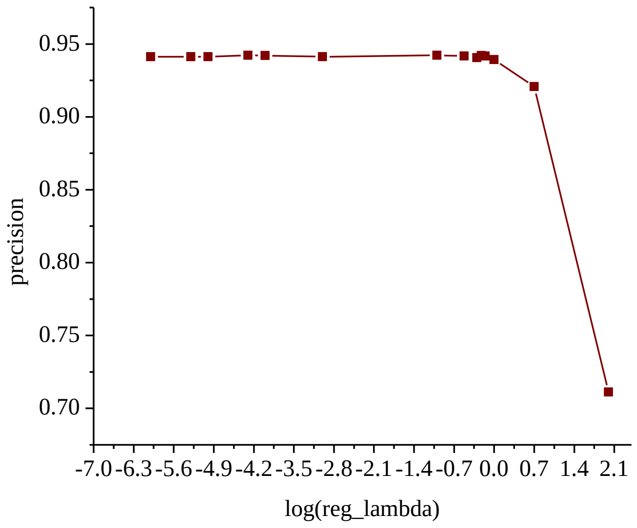

Fig. 9. XGBoost's performance for different reg_lambda

4.4 Model Evaluation

In our experiments, classifiers are trained with dataset 1 (70 features) and dataset 2 (5 features), and stratified k-folds method is used for cross validation. Accuracy, precision and recall are used to evaluate classifiers. Results are presented in Table 9.

Table 9. Model performance evaluation table

	Model	BP Neural Network		SVM		Random Forest		XGBoost	
	Dataset	1	2	1	2	1	2	1	2
Before Tuning	Accuracy	72.15%	60.13%	N/A	N/A	98.53%	96.55%	73.37%	68.56%
	Precision	75.14%	63.16%	N/A	N/A	98.05%	94.89%	73.31%	69.37%
	Recall	66.98%	50.43%	N/A	N/A	99.03%	98.40%	73.51%	66.43%
	Time(s)	27.66	94.20	N/A	N/A	19.47	5.29	30.29	4.26
After Tuning	Accuracy	83.94%	60.13%	67.21%	50.08%	98.53%	96.55%	97.72%	96.67%
	Precision	82.68%	63.16%	69.34%	86.88%	98.05%	94.89%	96.47%	94.47%
	Recall	85.98%	50.43%	61.72%	20.14%	99.03%	98.40%	99.06%	99.14%
	Time(s)	109.62	94.20	29546.11	84.86	19.47	5.29	592.23	71.02

Note: since SVM is unable to complete training in a reasonable amount of time with default hyperparameters, there are no records for SVM before training.

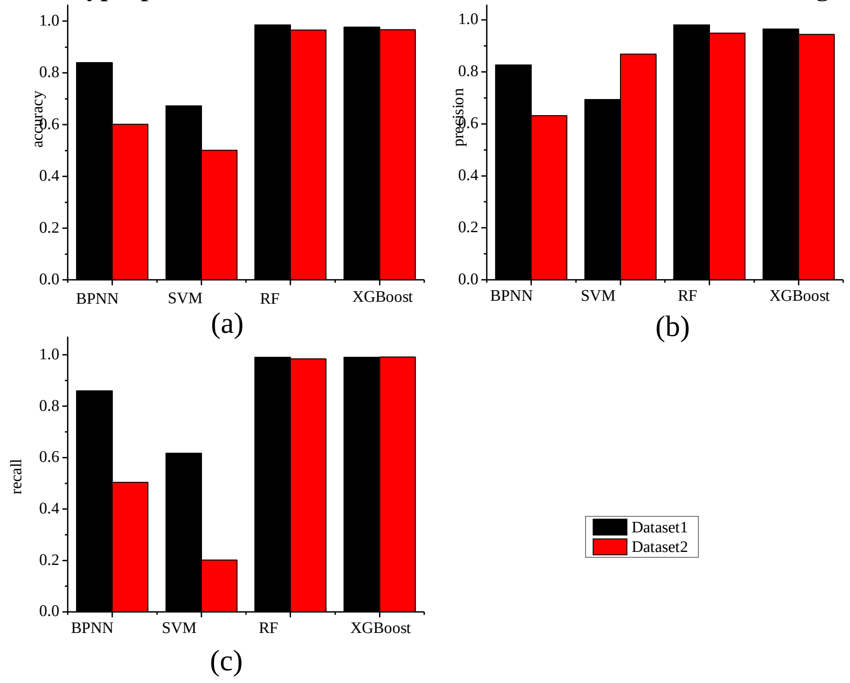

Fig. 10. Model performance evaluation graph

In the scenario of theft of electricity detection, accuracy is the ratio of correct classification, precision is the ratio of true illegal users in all predicted illegal users, and recall is the ratio of predicted illegal users in all true illegal users. To accurately identify illegal users without disturbing normal users by labeling them as illegal users, precision is the main metric used to evaluate a classifier in this experiment.

As shown in Table 9, random forest classifier and XGBoost classifier both achieve excellent classification results. In comparison, random forest classifier achieves slightly higher precision. The excellent performance and training speed of random forest may be related to its row and column subsampling. Row subsampling allows random forest classifier to have a faster training speed. Column subsampling can reduce variance and prevent over-fitting. Consequently the risk of overfitting is low even if the maximum depth of decision trees is no limited. Additionally, random forest is not very sensitive to the specific hyper-parameters used, so it can achieve good performance with hyperparameters set to default.

Backpropagation neural network is inferior to random forest and XGBoost in accuracy, precision and recall, and achieved significantly better performance when trained with dataset 1 than dataset 2. We think the phenomenon may be caused by the network structure of backpropagation neural network. The network structure has been proven to be able to implement complex nonlinear mapping, so it is suitable for modeling complex relation between features and labels when the dataset has a high feature count. On the other hand, the lack of a systematic method for selecting suitable network structure makes it difficult to determine the optimal network structure, which may be responsible for the poor performance of the backpropagation neural network in this experiment.

It can be seen from Table 9 that SVM is slow in training and poor in classification. The time complexity of SVMs is normally between $\mathrm{O}(\mathrm{n}^2)$ and $\mathrm{O}(\mathrm{n}^3)$, depending on how they are implemented. Compared with the random forests' complexity of $\mathrm{O}(\mathrm{n} \cdot \log(\mathrm{n}))$, SVMs are significantly more computational intensive, making it hard to scale to a large sample size.

Based on the background of theft of electricity detection where training and testing speed is important, random forest is the most suitable for its high training speed, high precision and ease of tuning. However, with suitable datasets and careful tuning, it may be possible to obtain higher precision with XGBoost.

5. Conclusions and Reflections

In this research, methods for feature extraction are analyzed; four classifiers including Backpropagation Neural Network, Support Vector Machine, Random Forest and XGBoost, are analyzed and trained; then the best performing algorithm and hyperparameters are selected to detect thefts of electricity. The major achievements of this research are listed as follows:

(1) Analyze feature extraction and dimensionality reduction algorithms such as Principal Component Analysis, process the original dataset and obtain two datasets, applied them to different algorithms and analyze the influence different datasets had on the results.

(2) Study various classifiers, analyzed their principles and implementation. Find the optimal hyperparameters. Achieve a precision of 82.68% for Backpropagation Neural Network, 86.88% for Support Vector Machine, 98.05% for Random Forest and 96.47% for XGBoost.

We propose a few future improvements:

(1) Features in dataset 1 include all monistic, binary and multivariate features proposed in chapter 2; features in dataset 2 include mean, minimum, maximum, variance and medium of daily power consumption. This paper studied the detection of electricity theft based on supervised learning from the theoretic perspective without considering the actual physical relationship between variables and theft of electricity. Future researches shall be able to select features more prudent based on the physical characters of theft of electricity.

(2) Oversampling is used to overcome the imbalance of classes in the dataset. However, oversampling produces replicated data, thus raising the possibility of overfitting. Future analysis may combine algorithms such as Borderline-SMOTE and ADASYN to generated new data, reducing the possibility of overfitting caused by oversampling.

(3) The accuracy of backpropagation neutral network remains low after tuning hyperparameters; the training speed of SVM classifier is slow, and the result of it is not satisfactory. Future studies can explore better method for searching hyperparameters, and

further tune hyperparameters of backpropagation neutral network in order to obtain better results. Additionally, future studies can explore better approaches of dimensionality reduction to reduce the computational complexity for training SVM classifier. Moreover, future studies may combine genetic algorithm or other algorithms to improve the training speed and the performance of backpropagation neutral network and support vector machine.

6. References

[1]   SALINAS S A, LI P. Privacy-preserving energy theft detection in microgrids: A state estimation approach [J]. IEEE Transactions on Power Systems, 2016, 31(2): 883-94.

[2]   TARIQ M, POOR H V. Electricity Theft Detection and Localization in Grid-tied Microgrids [J]. IEEE Transactions on Smart Grid, 2016,

[3]   GAUR V, GUPTA E. The determinants of electricity theft: An empirical analysis of Indian states [J]. Energy Policy, 2016, 93(127-136).

[4]   吴晓婷, 闫德勤. 数据降维方法分析与研究倡 [J]. 计算机应用研究, 2009, 26(8):

[5]   PENG C, KANG Z, CHENG Q. A fast factorization-based approach to robust PCA; proceedings of the Data Mining (ICDM), 2016 IEEE 16th International Conference on, F, 2016 [C]. IEEE.

[6]   MATHUR A, FOODY G M. Multiclass and binary SVM classification: Implications for training and classification users [J]. IEEE Geoscience and remote sensing letters, 2008, 5(2): 241-245.

[7]   曹峥. 反窃电系统的研究与应用 [D] [D]; 上海交通大学, 2011.

[8]   LIAW A, WIENER M. Classification and regression by randomForest [J]. R news, 2002, 2(3): 18-22.

[9]   CHEN T, GUESTRIN C. Xgboost: A scalable tree boosting system; proceedings of the Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, F, 2016 [C]. ACM.

[10]   GLEYZER S, MONETA L, ZAPATA O A. Development of Machine Learning Tools in ROOT; proceedings of the Journal of Physics: Conference Series, F, 2016 [C]. IOP Publishing.

[11]   CHEN K, LV Q, LU Y, et al. Robust regularized extreme learning machine for regression using iteratively reweighted least squares [J]. Neurocomputing, 2017, 230(345-358).

[12] REFAEILZADEH P, TANG L, LIU H. Cross-validation [M]. Encyclopedia of database systems. Springer. 2009: 532-538.

[13] HAN H, WANG W-Y, MAO B-H. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning [J]. Advances in intelligent computing, 2005, 878-887.

[14] HE H, BAI Y, GARCIA E A, et al. ADASYN: Adaptive synthetic sampling approach for imbalanced learning; proceedings of the Neural Networks, 2008 IJCNN 2008(IEEE World Congress on Computational Intelligence) IEEE International Joint Conference on, F, 2008 [C]. IEEE.

Acknowledgements

This is a collaborate research of Yichi Yang and Guangzi Xu.

Guangzi Xu studied the current situation of theft of electricity; studied various classifiers, analyzed their principles and implementation; and wrote chapter 1 (introduction) and capter3 (user behavior classification).

Yichi Yang studied feature extraction methods; conducted experiments and analyzed the results; and wrote chapter 2 (user behavior description), chapter 4 (experiment and result analysis), and chapter 5 (conclusions and reflections).

We sincerely thank Prof. Shuiguang Deng and Dr. Yifei Xu, who provided valuable suggestion during the course of this research. Without their help this research would not be possible.

学术诚信声明

本参赛团队声明所提交的论文是在指导老师指导下进行的研究工作和取得的研究成果。尽本团队所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果。若有不实之处，本人愿意承担一切相关责任。

参赛队员： 杨逸驰 徐光梓　　指导老师：徐亦飞 邓水光

2017 年 **8** 月 **30** 日