Expert Systems With Applications 176 (2021) 114899

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# A novel hybrid ensemble model based on tree-based method and deep learning method for default prediction

Hongliang He [a], Yanli Fan [b,*]

[a] School of Electronic and Computer Engineering, Peking University
[b] Shanghai University of Finance and Economics, School of Public Economics and Administration

ARTICLE INFO

*Keywords:*
Feature generation
Deep learning
Ensemble learning
Default prediction
Binary classification

ABSTRACT

Default prediction plays an important role in emerging financial market, so it has attracted extensive attention from financial industry and academic community. A slight improvement in default prediction performance can avoid huge economic losses. Many existing studies have used feature selection to improve the performance of default prediction models but paid limited attention to feature generation. Additionally, deep learning methods have been gradually explored for classification problems. In this study, a novel hybrid ensemble model is proposed to improve the performance of default prediction. First, a tree-based method (i.e., LightGBM) is used to learn new feature interactions and enhance the representation of original features. Second, a deep learning method (i.e., Convolutional Neural Network) is used as feature generation method to generate deeper feature interactions. Moreover, the structure of Inner Product-based Neural Network (IPNN) is used as deep learning classifier to learn feature interactions and reach a good trade-off between predictive accuracy and complexity. Third, ensemble learning method is used to combine the deep learning classifier with tree-based classifiers to obtain superior predictive results. Finally, two default datasets and four evaluation metrics are used to measure the predictive performance. The experimental results show that each component of the proposed model has significant improvement on overall performance.

## 1. Introduction

In recent years, consumer credit and consumption loans have been extensively accepted with the rapid development of consumer financial market. Besides traditional bank loan, some online lending platforms have established making the consumer financial market more various and complex. Moreover, borrower's repayment behavior will be influenced by various subjective and objective factors, such as occupation, income, macroeconomic fluctuations, etc. Big data and artificial intelligence make it possible to adapt to this era, identifying the potential risks by improving default prediction techniques.

Credit scoring and default prediction have been important research hotspot in recent decades with the emerging consumer financial market and improving artificial intelligence techniques. Huge economic losses can be avoided even if the accuracy of the model is slightly improved in default prediction and credit scoring (Hand & Henley, 1997; Dahiya, Handa, & Singh, 2016). According to the China Banking and Insurance Regulatory Commission, non-performing loans of China's commercial banks reached 2.84 trillion yuan at the end of the third quarter of 2020, and improved 98.7 billion yuan from the end of the previous quarter. The non-performing loan ratio of commercial banks was 1.96%, and improved 0.02% from the end of the previous quarter. This type of loans poses a great threat to sustainable development of banks and overall financial market. In recent decades, researchers have made endless explorations on credit evaluation models. Some researchers focused on constructing approaches or models to extract information from data and then used for predicting the probability of default in practical financial decisions. Binary classification problems were originally studied based on statistical models (Altman, 1968; Zavgren, 1985), such as logistic regression (LR; Hand & Kelly, 2002), linear discriminant analysis (LDA; Fisher, 1936). However, most statistical methods are based on some specific assumptions (e.g., independence between variables and normal distribution of the data). These assumptions are too strict to satisfy real-world conditions.

Unlike statistical models, machine learning methods do not require assumptions or prior knowledge, and they can automatically extract useful information from dataset. Therefore, machine learning methods have received a lot of attention and extensive application in binary

* Corresponding author at: School of Public Economics and Administration, Shanghai University of Finance and Economics, Shanghai 200433, China.
*E-mail addresses:* hehl@pku.edu.cn (H. He), fanyanli@163.sufe.edu.cn (Y. Fan).

https://doi.org/10.1016/j.eswa.2021.114899
Received 12 November 2019; Received in revised form 14 January 2021; Accepted 6 March 2021
Available online 13 March 2021

classification problems. Decision Trees (DT; Li, Ying, Tuo, & Li, 2004), Support Vector Machines (SVM; Huang, Chen, Hsu, Chen, & Wu, 2004), Random Forests (RF; Friedman, 2016), and Gradient Boosting Decision Tree (GBDT; Friedman, 2016) are representative machine learning methods which are extensively used in default prediction. In many application scenarios, high-dimensional, multi-categorical, and sparse data is common and it cannot adequately express the real contents of data. In this occasion, a single machine learning method is sometimes inadequate to get competitive performance. Therefore, how to construct a machine learning method that not only has excellent feature representation ability but also improves predictive performance has become a common concern for academia and industry.

Feature generation plays an important role in improving model performance. However, traditional feature generation is mainly a time-consuming manual operation process and it requires specific domain knowledge. This limitation makes it difficult to deploy in practical application. Recently, some researchers have explored approaches to automatically generate useful features, such as Factorization Machine (FM; Rendle, 2010), Field-aware Factorization Machine (FFM; Juan, Zhuang, Chin, & Lin, 2016), GBDT + LR (He et al., 2014), and Product-based Neural Networks (PNN; Qu et al., 2016). However, most of these approaches are applied to Click-Through-Rate (CTR) prediction in e-commerce field, and are rarely used in default prediction.

Previous studies (Zięba, Tomczak, & Tomczak, 2016; Liu et al., 2019; Zhang, Han, Xu, & Wang, 2019b) have shown that tree-based methods and deep learning methods can automatically generate useful features. However, there are still some limitations remain to be solved. For example, tree-based methods cannot learn the feature combination that rarely appears or never appears in training data. Although deep learning methods can extract feature combination through inner product of the hidden vectors, they have the limitation of extracting low-level feature combinations. Therefore, integration of tree-based methods and deep learning methods can not only improve the ability to characterize the relationship between features, but also improve the representative ability of features.

This study proposed a hybrid ensemble model combining tree-based method and deep learning method for default prediction. In the proposed model, new features are automatically generated based on partition path of the tree-based method (i.e., LightGBM; Ke et al., 2017), and the deep learning method (i.e., Convolutional Neural Network (CNN; LeCun, Bottou, Bengio, & Haffner, 1998)). The embedding of new features and original features are used as input to the subsequent deep learning model (i.e., Inner Product-based Neural Network (IPNN; Qu et al., 2016)). Then, IPNN is combined with various machine learning classifiers (i.e., RF, XGBoost, and LightGBM) to construct an ensemble model for final prediction results. The experimental results show that the proposed model has superior predictive performance and stronger robustness in different datasets compared to baselines.

The remainder of the study is organized as follows. Section 2 provides a literature review of feature generation and classifier ensemble. Section 3 elaborates the proposed hybrid ensemble default prediction model. Section 4 describes the experimental design. Section 5 presents the experimental results and the comparison of baseline classifiers. Section 6 describes the conclusions and future work.

## 2. Related work

### 2.1. Feature generation

Feature generation is a process of generating new features based on existing features, which can enhance the representation of original features and improve the performance of machine learning methods. Feature generation is more important than model selection to some extent (Domingos, 2012).

However, traditional manual-operated feature generation relies on professional domain knowledge, restraining generated features to some subjective factors. It makes the manually generated features not conducive to expansion. Therefore, some researchers have set out to explore feature generation methods that can automatically generate new features. These methods can freedom the constraints from subjective factors and increase the efficiency of generating effective features. To explore new features, Rendle (2010) proposed FM that extracts feature combinations by using factorized parameters between feature variables. Juan, Zhuang, Chin, & Lin, 2016 proposed FFM by introducing the concept of field to extend FM model. The characteristic of FFM is that the features with same nature are attributed to the same field. He et al. (2014) proposed a new hybrid model combining GBDT and LR. The experimental results showed that transforming real-valued input features with GBDT significantly improve the predictive accuracy of the linear classifiers. Serrano-Cinca and Gutiérrez-Nieto (2016) designed a decision support system based on multivariate regression and chi-squared automatic interaction detector (CHAID) decision tree for P2P lending. The experimental analysis showed that nonlinear data mining technology can improve the performance of profit scoring system.

In recent years, deep learning methods have received great attention due to its excellent performance and strong ability to learn complex and nonlinear relationships from dataset. Kim & Cho, 2018 proposed a deep dense convolutional network (DenseNet) for default prediction in P2P lending to automatically generate features and improve the predictive performance. The experimental results showed that DenseNet can automatically extract useful features from Lending Club data, avoid over-fitting, and achieve optimal performance compared with deep CNN and other machine learning methods. Bastani, Asgari, and Namavari (2019) proposed a two-stage scoring approach and applied wide and deep learning method to build the predictive models in both stages with its memorization and generalization abilities. Empirical studies have shown that the proposed approach achieved excellent performance in Lending Club data. Zhang et al. (2019b) proposed a model based on homogeneity-oriented behavior analysis (HOBA) and deep learning framework, which can construct high-level new features and achieve excellent predictive performance. However, some researchers have pointed out the limitation that newly generated feature set is usually too large and it has to be optimized (Bastani et al., 2019; Zhang et al., 2019b; Kim & Cho, 2018). They also argued that the models combining deep learning methods with traditional data mining methods can be an exploring and promising research perspective in future research.

### 2.2. Classifier ensemble

Classifier ensemble refers to the process of constructing multiple base classifiers into an ensemble model using some ensemble strategies. Classifier ensemble models can combine advantages of different base classifiers to achieve better predictive performance and they have received extensive attention from academia and industry in the past few decades.

Finlay et al. (2011) and Lessmann, Baesens, Seow, and Thomas (2015) systematically analyzed the classifier ensemble model to compare the advantages and disadvantages of various ensemble strategies through multiple sets of comparative experiments. Marqués, García, and Sánchez (2012) explored the performance of different base classifiers under different ensemble strategies. Similarly, Abellán and Castellano (2017) explored how to choose the appropriate base classifiers in ensemble model. These researches have provided a good exploration and inspiration for selecting ensemble strategies. Moreover, Ala'raj and Abbod (2016a) proposed a combination approach based on classifier consensus that combines multiple classifier systems of different classification algorithms. Based on this, Ala'raj and Abbod (2016b) further proposed a hybrid ensemble model that combines two data preprocessing methods (i.e., Gabriel Neighbourhood Graph Editing and Multivariate Adaptive Regression Splines). The experimental results showed that the performance of these proposed models is superior to that of comparative models. Xia, Liu, Da, and Xie (2018) proposed a

heterogeneous ensemble credit model that integrates bagging method with stacking method. Xia, Zhao, He, Li, and Niu (2020) proposed a tree-based dynamic heterogeneous ensemble method for credit scoring and default prediction. Experimental results showed that the model outperforms other comparative models. All these researches demonstrated the effectiveness and efficiency of ensemble methods from different perspectives.

In our previous research, various classifier ensemble methods have been explored and verified to achieve superior performance (He, Zhang, & Zhang, 2018; Zhang, He, & Zhang, 2018; Zhang, He, & Zhang, 2019a). Among them, tree-based classifiers are demonstrated as ideal base classifiers for ensemble model. Moreover, the enhanced multi-population niche genetic algorithm (EMPNGA) is proposed to select appropriate base classifiers in the candidate classifier repository, and it is further used for constructing ensemble model. Experimental results indicated that the predictive performance of ensemble model is better than that of single classifiers. Moreover, feature selection methods have been carefully studied in Zhang et al. (2019a). Another effective method, feature generation method, has increasingly attracted attention from researchers, and there is some exploration and development can be done to improve the performance for predictive performance.

This study combines tree-based methods and deep learning methods to construct a novel hybrid ensemble model for feature generation and classifier ensemble. The tree-based method (i.e., LightGBM) and the deep learning method (i.e., CNN) are used to generate new features for subsequent tree-based classifiers and deep learning classifiers, and stacking method is used to construct an ensemble model for final prediction results.

## 3. The proposed hybrid ensemble model

This study proposed a novel hybrid ensemble model based on tree-based methods and deep learning methods to improve the predictive performance for default prediction. The main components of the proposed model are feature generation based on tree-based method and deep learning method, classifier setting and classifier ensemble. Its framework is shown in Fig. 1, where $n$ represents the number of instances in dataset, $T$, $M$ and $D$ represent the number of original features and newly generated features generated by LightGBM and CNN respectively, $clf_i(1 \le i \le K+L)$ represents the $i$-th base classifier, $P_i(1 \le i \le K+L)$ represents the $i$-th prediction result generated by $i$-th base classifier. The specific description of each component is as follows.

### 3.1. Feature generation

#### 3.1.1. Feature generation based on tree-based method

Feature generation for traditional default prediction mainly relies on expert domain knowledge and it requires a significant cost of time for review. This condition makes it difficult for model generalization. To compensate for this, a feature generation method (i.e., LightGBM) which

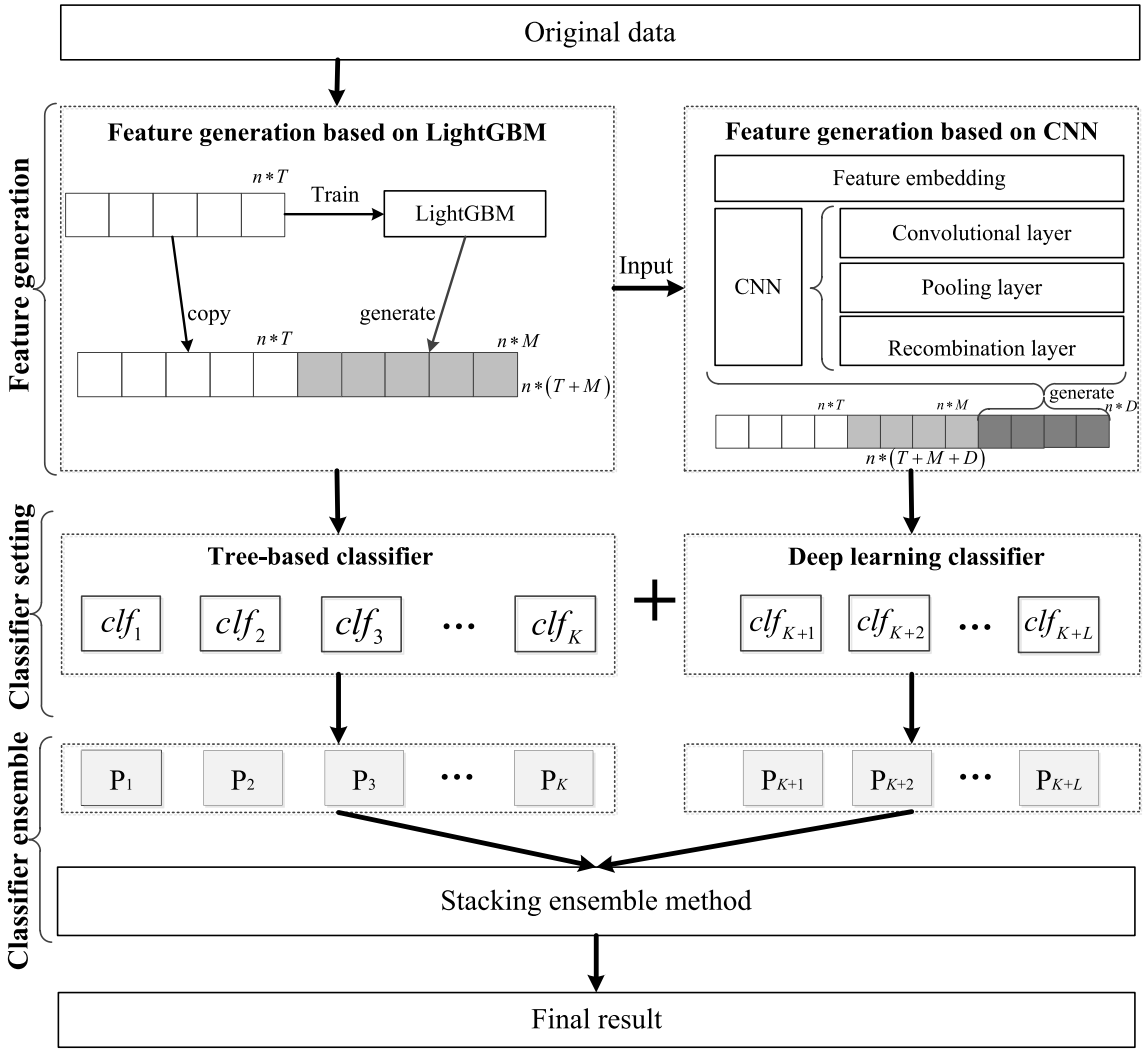


**Fig. 1.** Framework of the proposed model.

is improved from GBDT algorithm is used to automatically generate new features. It has been widely used in CTR prediction and demonstrated to improve the predictive performance.

GBDT is an ensemble method based on decision tree and boosting framework. Traditional boosting framework is iterated to weight erroneous samples, while each iteration in GBDT is calculated to reduce the last residual and it establishes a new tree model in the gradient of the residual reduction to eliminate the residual. The model generated by each iteration has the least loss in training set. The iterative strategy of GBDT is beneficial for discovering effective features and their feature interactions. In practical applications, He et al. (2014) first used GBDT in CTR prediction to automatically extract effective features and feature interactions. It shows that using GBDT to transform real-valued input features can significantly improve the predictive performance of probabilistic linear classifiers.

LightGBM applies Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to improve the performance of traditional GBDT (Ke et al., 2017). More specifically, LightGBM uses histogram-based algorithms (Li, Wu, & Burges, 2007) to segment continuous feature values into discrete bins to speed up training and reduce memory usage. It uses a leaf-wise (best-first) decision tree growth strategy that will pick the leaf nodes with the largest delta loss. Experimental results show that the leaf-wise algorithm can reduce more losses than the level-wise algorithm when growing the same leaf. Therefore, the proposed model used LightGBM to generate new features for subsequent tree-based classifiers. Each element of the new feature vector corresponds to the leaf node of the tree in LightGBM. The length of the new feature vector is equal to the accumulated numbers of leaf nodes contained by all the trees of LightGBM, and the value at each leaf node is 0 or 1. That is, when an instance falls on a leaf node of a tree, the element value of new feature vector corresponding to the leaf node is 1, and the value of element corresponding to other leaf nodes is 0.

Fig. 2 shows the boosted tree model with 3 subtrees, each of which has 8 leaf nodes. For an instance X, it ends up at leaf 9 in the first subtree, leaf 10 in the second subtree, and leaf 13 in the third subtree. Therefore, the generated features (i.e., $M$ in Fig. 1) can be represented by a binary vector [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0], where the first 8 entries indicate the leaves of the first subtree, the middle 8 entries indicate the leaves of the second subtree, and the remained 8 entries indicate the he leaves of the third subtree.

### 3.1.2. Feature generation based on CNN

Besides LightGBM, the other feature generation method based on a deep learning method (i.e., CNN) is applied to generate new features for further deep learning prediction. It has been applied and verified to obtain good performance in CTR (Liu et al., 2019). In advance, feature embedding is used to transfer high-dimensional sparse features into low-dimensional dense ones for subsequent deep learning classifier.

After one-hot encode processing, the categorical features in each instance generate high-dimensional sparse feature vectors. If these high-dimensional sparse features are directly used as input to train the deep learning model, a large number of parameters will be generated. It is not beneficial for classifier training and prediction. Feature embedding, as an efficient method for mapping high-dimensional sparse features into low-dimensional dense ones, is applied to process features for facilitating the training of subsequent deep learning classifier. In each instance, every categorical feature (also known as field) $i$ $(1 \le i \le n_c)$ is represented as a low-dimensional vector $e_i \in R^{1\times k}$, where $n$ is the number of categorical features, $k$ is the embedding size. After feature embedding, high-dimensional sparse features are mapped to embedding matrix $E = \left(e_1^T, e_2^T, e_3^T \cdots, e_{n_c}^T\right)^T$, where $E \in R^{n_c \times k}$.

After feature embedding, a feature generation method based on CNN is applied to generate new features for subsequent deep learning classifier. As shown in Fig. 3, the feature generation method used in this study mainly consists of three layers: convolutional layer, pooling layer, and recombination layer, which is similar to the framework used by Liu et al. (2019) in CTR prediction.

First, a convolutional layer is generated to capture the interaction of neighbor features. Specifically, each instance's embedding matrix $E \in R^{n_c \times k}$ is reshaped to get $E^1 \in R^{n_c \times k \times 1}$, which is used as the input matrix of the first convolutional layer. A convolutional layer is obtained by convolving a matrix $WC^1 \in R^{h^1 \times 1 \times 1 \times m_c^1}$ with nonlinear activation functions, where $h^1$ is the height of first convolutional weight matrix, $m_c^1$ is the number of feature maps in the first convolutional layer. Assuming that the output of the first convolutional layer is $C^1 \in R^{n \times k \times m_c^1}$, each index output of the $i$-th feature map can be expressed as Eq. (1), where $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $p$ and $q$ are the row and column index of the $i$-th feature map.

$$C^1_{p,q,i} = tanh\left(\sum_{m=1}^{1}\sum_{j=1}^{h^1} E^1_{p+j-1,q,m} WC^1_{j,1,1,i}\right) \tag{1}$$

After the first convolutional layer, the pooling layer is used to preserve the most important feature interactions and reduce the number of parameters. This study used max-pooling as pooling function. Assuming

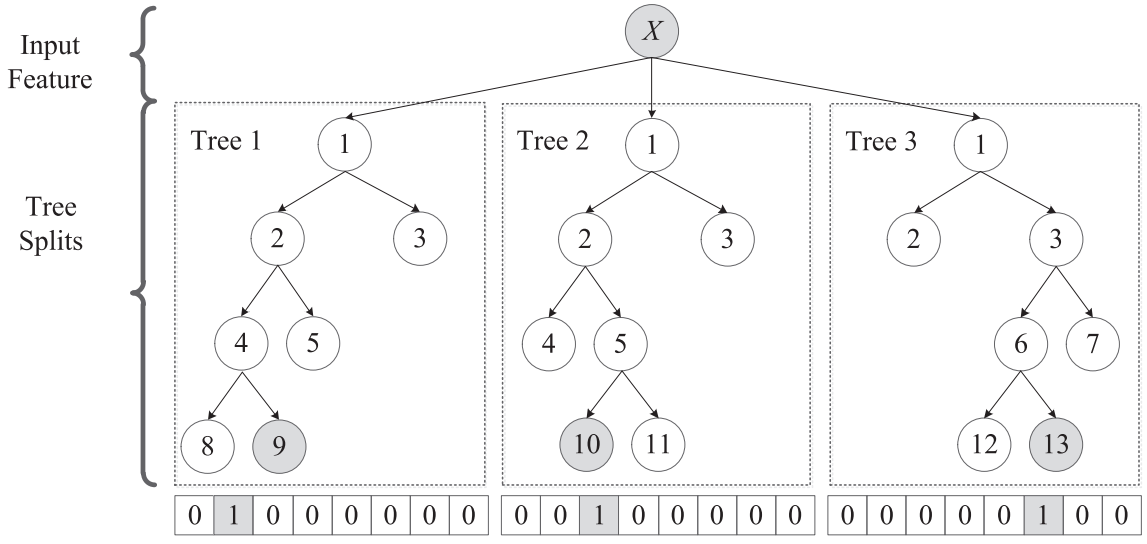


**Fig. 2.** Feature generation with LightGBM.

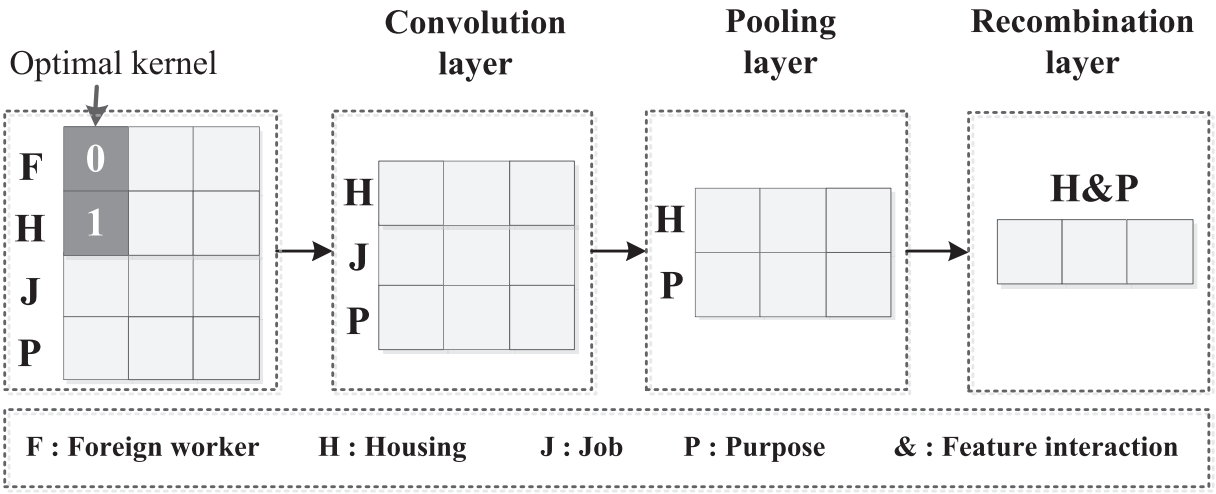


**Fig. 3.** Feature generation based on CNN.

that the output of the first pooling layer is $S^1 \in R^{\left(\frac{n_c}{h_p}\right)\times k\times m_c^1}$, each index output of the $i$-th feature map can be expressed as Eq. (2). The result of the $i$-th pooling layer is the input of the $(i+1)$-th convolutional layer, that is, $E^{i+1} = S^i$.

$$S^1_{p,q,i} = max\left(C^1_{p\cdot h_p,q,i}, C^1_{p\cdot h_p+1,q,i}, \cdots, C^1_{p,q,i}\right) \tag{2}$$

To overcome the limitations of CNN framework which easily ignores global non-neighbor feature interactions, a fully connected layer (i.e., recombination layer) is obtained to combine local neighbor feature patterns and generate new features. $N_i = \frac{n_c}{h_p^i m_r^i}$ features are denoted by $R^1$ in the first recombination layer, and $R^1$ is expressed as Eq. (3), where $R^1_w \in R^{(n_c/h\cdot k\cdot m_c^1)\times(n_c/h\cdot k\cdot m_r^1)}$ indicates the weight matrix, $R^1_r \in R^{n_c/h\cdot k\cdot m_r^1}$ indicates the bias, and $m_r^1$ is the number of new features' map in the first recombination layer.

$$R^1 = tanh\left(S^1 \cdot R^1_w + R^1_b\right) \tag{3}$$

In feature generation framework, these above layers (i.e., convolutional layer, pooling layer, and recombination layer) are cascaded multiple times to generate overall new features $R \in R^{N\times k}$, as shown in Eq. (4), where $N = \sum_{i=1}^{t} N_i$, $t$ is the round number.

$$R = \left(R^1, R^2, \cdots, R^t\right) \tag{4}$$

Therefore, the original features $E \in R^{n\times k}$ and the new features $R \in R^{N\times k}$ are concatenated to obtain the new features $E_{new} \in R^{(n+N)\times k}$, as shown in Eq. (5).

$$E_{new} = \left(E^T, R^T\right)^T. \tag{5}$$

### 3.2. Classifier setting

In previous studies, tree-based learning classifiers and deep learning classifiers have been verified to have good performance in default prediction (He et al., 2018; Bastani et al., 2019). In this Sub-section, some tree-based machine learning classifiers and deep learning classifiers used in this study will be introduced in detail.

#### 3.2.1. Tree-based classifier

With the development of machine learning research, more and more researches have studied the application and extension of machine learning classifiers in default prediction and various tree-based classifiers have been verified to have good prediction performance, such as RF, XGBoost, and LightGBM (He et al., 2018; Ma et al., 2018). In this study, three tree-based classifiers, (i.e., RF, XGBoost, and LightGBM) are used as base classifiers with the features generated by LightGBM. Since LightGBM algorithm has been described in Sub-section 3.1, RF and XGBoost are described in this Sub-section.

RF is a tree-based bagging model proposed by Breiman (2001). It randomly extracts $m$ sub-samples and $k$ sub-features from the original dataset, forming multiple sets of sub-data for training multiple decision trees. Then, it applies voting method to combine the classification results of each decision tree and generate final classification results. In this regard, RF algorithm has two advantages. First, it has fast training speed and it is easy to calculate in parallel. And then, it can effectively prevent from overfitting.

XGBoost is another classical tree-based classifier proposed by Chen and Guestrin (2016). Its core idea is to carry out second-order Taylor expansion of the objective function on the basis of GBDT. XGBoost has attracted increasingly attention and been widely used in many data mining competitions recently due to its excellent performance. First, it adds complexity of the tree model to regular term of the objective function, effectively avoiding overfitting. Second, it uses second derivative to speed up convergence of the model while training. Third, it introduces a feature sub-sampling method that effectively reduces overfitting and computation.

#### 3.2.2. Deep learning classifier

Deep learning method, owing to its powerful function, has been attracted more and more attention from researchers recently. This study applied IPNN (Qu et al., 2016) combining with the tree-based classifiers (i.e., RF, XGBoost, and LightGBM) to obtain good prediction performance and strong robustness for default prediction. The structure of IPNN is used as the deep learning classifier to learn feature interactions and reach a good trade-off between accuracy and complexity. IPNN combines the learning capability of FM and Multi-Layer Perception (MLP; West, 2000) so that it can obtain better prediction performance. The structure of IPNN model is shown in Fig. 4.

In IPNN framework, the latest features are operated in two ways: the first way is to multiply these features directly by the constant "1", that is, directly copy it to part Z of the Product Layer; the second way is to perform pairwise feature interactions based on the operation of the tensor inner product to obtain part P of the Product Layer. Then, part Z and part P are concatenated to form the Product Layer and it is used as the input of the subsequent deep learning classifier. Part Z and Part P are expressed as Eqs. (6) and (7) respectively, where $E_i$ is the embedding of the $i$-th field, $< E_i, E_j >$ indicates the inner product of $E_i$ and $E_j$.

$$Z = (E_1, E_2, \cdots, E_{N+n}) \tag{6}$$

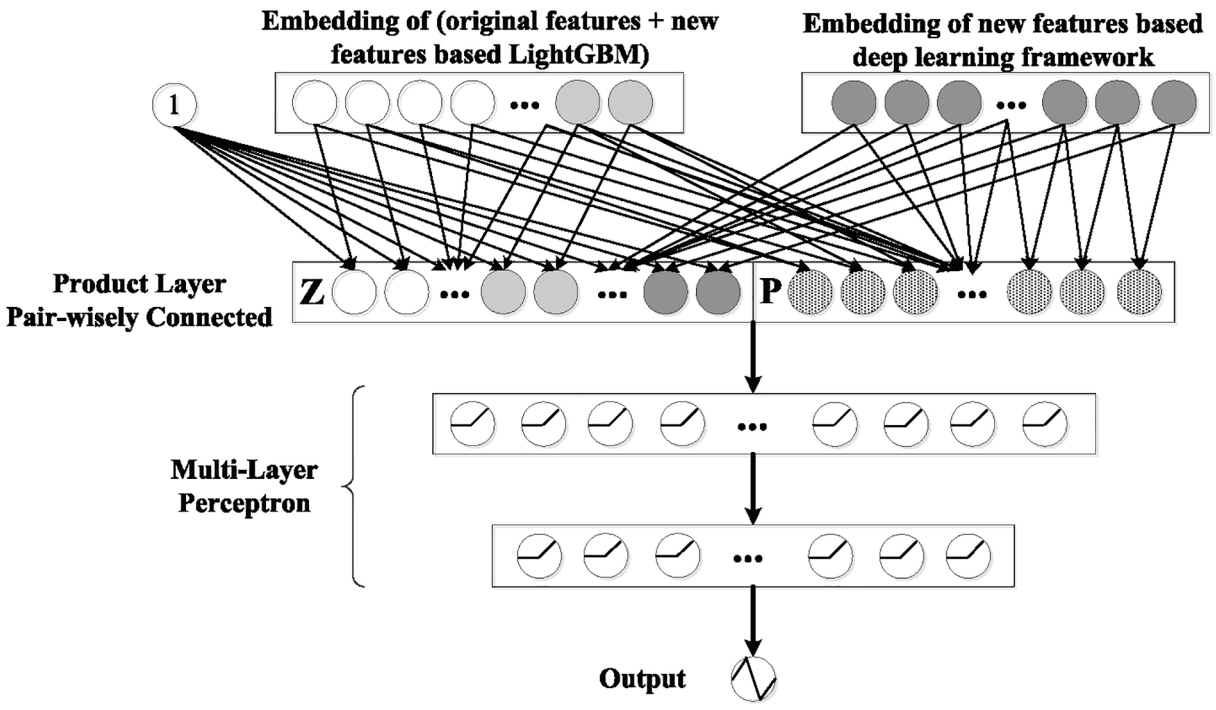


**Fig. 4.** The structure of IPNN.

$$P = (< E_1, E_2 >, \cdots, < E_{n+N-1}, E_{n+N} >) \tag{7}$$

For MLP with $d$ hidden layers, the initial input $I_1$ can be expressed as Eq. (8). The output and input for the $i$-th hidden layer are denoted as $O_i$ and $I_i$, which are expressed as Eqs. (9) and (10). Therefore, the final prediction can be obtained according to Eq. (11).

$$I_1 = (P, flatten(Z)) \tag{8}$$

$$O_i = relu(W_i I_i + B_i) \tag{9}$$

$$I_{i+1} = O_i \tag{10}$$

$$\hat{y} = sigmoid(W_d O_d + B_d) \tag{11}$$

Where function *flatten*(*) is used to turn a multidimensional input into a one-dimensional input, $W_i$ and $B_i$ are the weight matrix and bias of the $i$-th hidden layer in MLP, respectively.

In this study, cross entropy is used as the objective function (i.e., loss function) of the deep learning model, and it is expressed as Eq. (12) where $y$ and $\hat{y}$ denote the true value and the predicted probability respectively$y \in \{0,1\}$, and $0 \le \hat{y} \le 1$.

$$\mathscr{L}(y, \hat{y}) = -y log\hat{y} - (1-y) log(1-\hat{y}) \tag{12}$$

### 3.3. Classifier ensemble

Recent researches have shown that ensemble models and classifiers generally outperform single classifiers in binary classification (Finlay et al., 2011; Lessmann et al., 2015). For example, RF, XGBoost, and LightGBM are ensemble classifiers which are ensembled and optimized from decision tree. These ensemble classifiers are demonstrated to have excellent predictive performance in default prediction. In our previous works (He et al., 2018; Zhang et al., 2019a), stacking method is verified to be one of the best way to integrate heterogeneous classifiers. In this study, in order to further improve the performance and robustness of the proposed model, RF, XGBoost, and LightGBM are used as the base classifiers to ensemble with the deep learning model. Stacking method is used as ensemble method for this study and the framework of stacking method for ensemble classifiers is shown in Fig. 5.

In the training stage of stacking method, $K$ classifiers are validated through $N$-fold cross-validation (e.g., Fig. 5 is an example of 5-fold cross-validation) to obtain prediction probability of the complete training set (i.e., Prob in Fig. 5). These prediction probabilities are concatenated as new training set for training meta-classifier in second layer. In the testing stage of stacking method, each base classifier obtains prediction probability by predicting the testing set corresponding to each cross-validation of the training stage. The $N$ probabilities corresponding to each base classifier is averaged to obtain the average prediction probability of each classifier. Furthermore, the concatenated average prediction probabilities ($m$*$k$) are input into trained *meta*-classifier to obtain final prediction result. Finally, final performance of the model corresponding to different evaluation metrics is obtained based on final prediction result.

## 4. Experimental design

### 4.1. Data description

In the experiment, two real-world default datasets (i.e., DefaultData dataset and PPDai dataset) were used to compare the performance between the proposed model and other comparative models. DefaultData dataset is obtained from the UCI machine learning library (Asuncion & Newman, 2007). PPDai dataset is obtained from customer data from a Chinese P2P lending platform named PaiPaiDai .A brief description of these datasets is shown in Table 1. DefaultData dataset contains 30,000 instances including 23,364 non-default instances and 6,636 default instances. Each instance contains 24 features (15 numeric features and 9 categorical features). PPDai dataset contains 55,596 instances including 48,413 non-default instances and 7,183 default instances. Each instance contains 29 features (22 numeric features and 7 categorical features). Summary description of the used dataset including all features, and their corresponding types and definitions is shown in Appendix 1. For more detailed information, the raw datasets are provided in figshare repository (https://doi.org/10.6084/m9.figshare.13573871.v1).

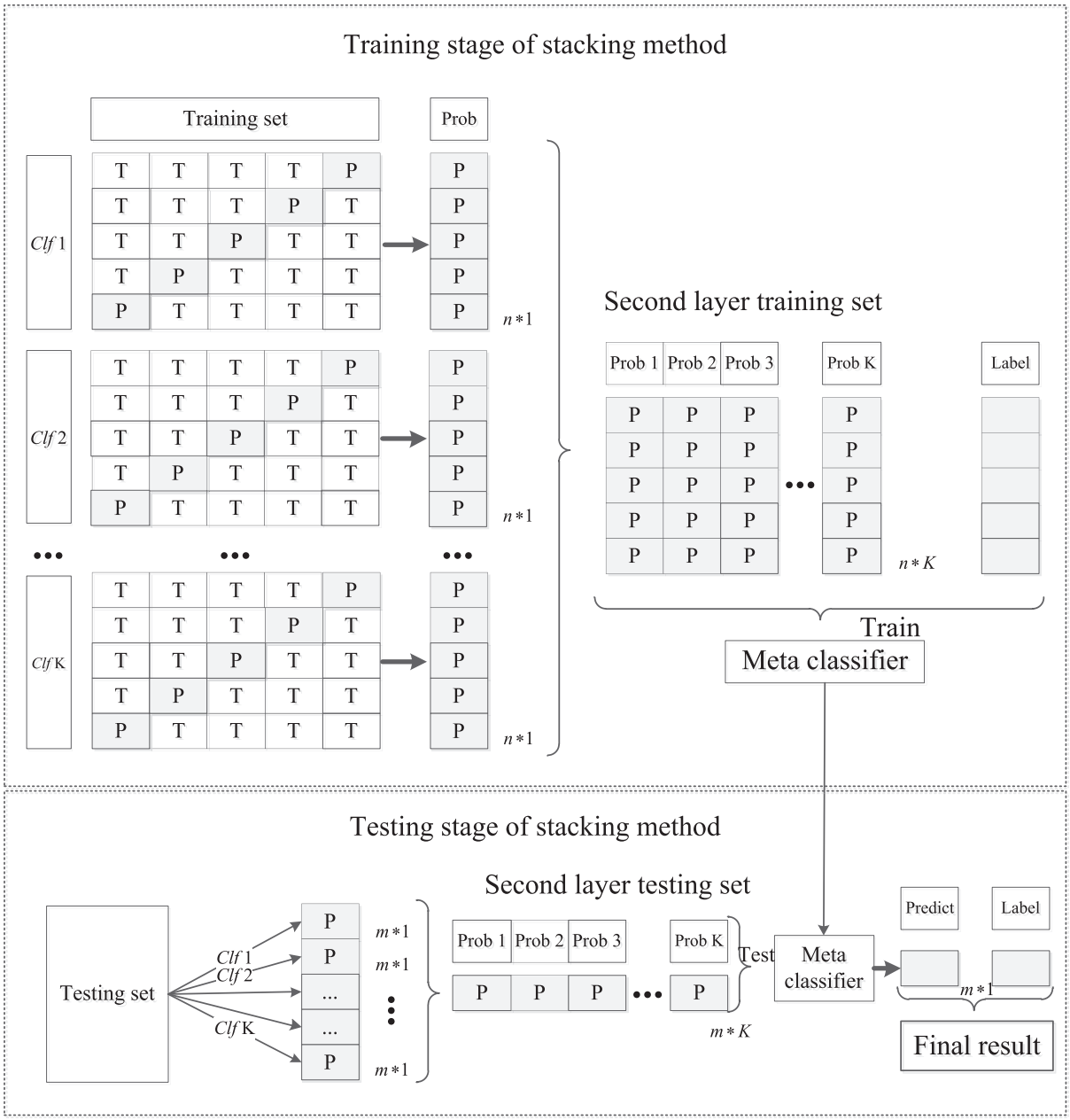


**Fig. 5.** The framework of stacking method.

**Table 1**
Description of the two datasets used in the study.

| Dataset | Number of instances | Non-default | Default | Numeric features | Categorical features | Total feature |
|---|---|---|---|---|---|---|
| DefaultData | 30,000 | 23,364 | 6,636 | 15 | 9 | 24 |
| PPDai | 55,596 | 48,413 | 7,183 | 22 | 7 | 29 |

### 4.2. Evaluation metrics

Prediction probability (i.e., probability of default in this study) for binary classification problems ranges from 0 to 1. The closer the prediction probability is to 1, the greater the probability of default. In the experiment, a specific threshold (set as 0.5 by default) is set to distinguish between positive class and negative class, and the description of confusion matrix is shown in Table 2.

Four commonly used evaluation metrics based on confusion matrix, that is, Accuracy, Area Under ROC Curve (AUC), Logistic Loss, and

**Table 2**
Confusion matrix.

| | | Predicted | |
|---|---|---|---|
| | | Positive | Negative |
| Real | Positive | True Positives(TP) | False Negatives(FN) |
| | Negative | False Positives(FP) | True Negatives(TN) |

Kolmogorov-Smirnov statistic (KS) are applied to evaluate the predictive performance of classifiers and proposed model.

Accuracy: It is the most commonly used metric for classification problems. It represents the ratio of true-predicted number of instances to the total instances. The expression of Accuracy is shown in Eq. (13).

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}. \tag{13}$$

AUC: It is a comprehensive evaluation metric based on Receiver Operating Characteristic (ROC) curve, which is equal to the area under ROC curve. The closer AUC value comes to 1, the better the overall performance a model has. ROC curve is obtained based on the false positive rate (FPR) and the true positive rate (TPR), as shown in Eqs. (14) and (15).

$$FPR = \frac{FP}{FP + TN}. \tag{14}$$

$$TPR = \frac{TP}{TP + FN}. \tag{15}$$

Logistic Loss: It is a likelihood estimate of the predicted probability and can be used to evaluate the stability of the model. It can be depicted in Eq. (16) where $N$ represents the number of testing instances, $y_i$ and $p_i$ denote the true value and the predicted probability$y_i \in \{0, 1\}$ and $0 \le p_i \le 1$.

$$Loss_{logistic} = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(p_i) + (1 - y_i) log(1 - p_i)). \tag{16}$$

KS: It is a cumulative distribution function based on TPR and FPR, and it is used to evaluate the ability of the model to identify positive and negative instances. Its expression is shown as Eq. (17).

$$KS = Max(TPR - FPR) \tag{17}$$

### 4.3. Data preprocessing

Real data always has some limitations, such as missing values and complicated data structure affecting the results and performance of the experiments, so preprocessing is a crucial step before constructing and training the model. First, missing values are preprocessed in this study. Average value of the intact instances is used to replace missing variables in numerical features, and a new category value is generated to replace missing variables in category features. This study normalizes the numerical features as Eq. (18), making it range between 0 and 1 to unify the dimensions between different features. Then, dummy method is used to expand the categorical features. That is, a categorical feature with $n$ attributes is transferred to $n$-dimensional features with only 0 and 1 attributes.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{18}$$

Where $x$ and $x'$ represent the feature value before and after normalization, and $x_{max}$ and $x_{min}$ represent the maximal and minimal value of the original feature, respectively.

In this study, the experiments are repeated 50 times based on different training sets and testing sets. The experiments set different running seeds to minimize the effect of the variability so as to reach a reliable result. Moreover, each dataset is divided into two parts, 80% is used for training and the remaining 20% is for testing, which is consistent with various previous researches (Wang, Ma, Huang, & Xu, 2012; Tsai, 2014; He et al., 2018). 10-fold cross-validation is used in the experiments.

### 4.4. Experiment parameters setting

To ensure the effectiveness and comparability of the experiment, parameters are required to be preset. In the experiment, the number of new features generated by LightGBM was set to 40; and the embedding size of feature embedding was set to 8. In CNN, the cascaded times was set to 4; the width of filter in each convolutional layer was set to (7, 7, 7, 7); the number of filters in each convolutional layer was set to (14, 16, 18, 20); the feature maps of generated features was set to (3, 3, 3, 3); the width of pooling layer was set to (2, 2, 2, 2); the value of dropout was 0. Gradient optimization method used in the experiment is adaptive moment estimation (adam), referring to the parameter settings of IPNN (Qu et al., 2016). LR and an MLP were performed with Python function "Scikit-learn" (Swami & Jain, 2013). When experimenting with different datasets, all of the features are used to prevent information losses.

## 5. Experimental results

In this section, experimental results are presented to compare the performance of the proposed model and other comparative classifiers. Three key points are validated through empirical experiments. The first point is whether the proposed tree-based feature generation method (i.e., LightGBM) generates more efficient features for the subsequent classification model. The second point is whether the feature generation based on the deep learning method (i.e., CNN) is beneficial for improving the performance of the subsequent classification model. The third point is whether the ensemble model constructed by the deep learning classifier (i.e., IPNN) and machine learning classifiers (i.e., RF, XGBoost, and LightGBM) is superior to the performance of their base classifiers.

In this study, all of the experiments used Python Version 3.6.8 on a PC with 3.2 GHz Intel CORE i7 processor. The PC has 32 GB of RAM, and runs the Microsoft Windows 7 operating system.

### 5.1. Benchmarking results

To measure the effectiveness of the methods used in this study, two representative benchmark classifiers, LR and MLP, are used to obtain benchmark results. LR (Hand & Kelly, 2002) is a kind of probabilistic nonlinear regression, which is a multivariate analysis method to study the relationship between binary (can be extended to multi-classification) observation results and some influencing factors. LR is considered to be the industry standard for the development of credit scoring model (Lessmann et al., 2015), and it is one of the most commonly used classifiers in practical default prediction; On the other hand, MLP is a neural network composed of fully connected layers with at least one hidden layer, and the output of each hidden layer is transformed by an activation function. It is a prevalent classifier which is used as the basis of many deep learning frameworks. Table 3 shows the prediction results of LR and MLP in two datasets based on four evaluation metrics, and these results are compared with the subsequent prediction results in following subsections.

As shown in Table 3, the predictive performance of LR under different evaluation metrics is much better than that of MLP in DefaultData dataset. Especially for AUC, LR is 5.62 higher than that of MLP. Moreover, the predictive performance of LR under different evaluation metrics is slightly better than that of MLP in PPDai dataset. These

**Table 3**
The Benchmarks results of LR and MLP in two datasets.

| DataSet | Classifier | Accuracy | AUC | Loss | KS |
|---|---|---|---|---|---|
| DefaultData | LR | 82.022 ± 0.453 | 76.557 ± 0.510 | 0.437 ± 0.008 | 0.414 ± 0.010 |
| | MLP | 79.480 ± 0.452 | 70.93 ± 0.502 | 0.617 ± 0.013 | 0.348 ± 0.014 |
| PPDai | LR | 87.047 ± 0.259 | 65.029 ± 0.183 | 0.368 ± 0.002 | 0.222 ± 0.004 |
| | MLP | 86.781 ± 0.398 | 63.604 ± 0.447 | 0.379 ± 0.004 | 0.201 ± 0.009 |

results indicate that MLP has long run to improve the predictive performance.

### 5.2. Comparison of prediction results before and after tree-based feature generation

LightGBM is applied as feature generation method, so this study compares the performance of LR and MLP before and after feature generation using LightGBM. It is used to verify whether LightGBM provides more effective features for subsequent classifiers. The comparison results are shown in Table 4.

Note: $LR_{fgl}$ indicates the predictive performance of LR after feature generation using LightGBM; $MLP_{fgl}$ indicates the predictive performance of MLP after feature generation using LightGBM.

Table 4 shows that the predictive performance of the classifiers after feature generation by LightGBM is better than that of the classifiers without feature generation. In DefaultData dataset, the Accuracy of LR improved 0.071 after feature generation by LightGBM, and the Accuracy of MLP improved 0.337. Efficient improvement is also generated in PPDai dataset with the Accuracy of LR improved 0.037 and the Accuracy of MLP improved 0.026 after feature generation using LightGBM. In DefaultData dataset, the AUC of LR improved 0.720 after feature generation by LightGBM, and the AUC of MLP improved 0.390. As for PPDai dataset, the AUC of LR improved 2.470 after feature generation by LightGBM, the AUC of MLP improved 3.250. Moreover, the Loss of LR and MLP is declined 0.003 and 0.010 respectively in DefaultData dataset, and declined 0.006 and 0.014 respectively in PPDai dataset after feature generation by LightGBM. The KS of LR and MLP improved 0.006 and 0.007 in DefaultData dataset, and improved 0.029 and 0.038 respectively in PPDai dataset after feature generation by LightGBM. In practical application, only a slight improvement in identifying great credit can reduce default rates, thus be beneficial for default risk management and control. The experimental results in Table 4 shows that the feature generation method (i.e., LightGBM) is beneficial for improving predictive performance in both dataset, demonstrating its efficiency and effectiveness.

### 5.3. Performance verification of deep learning methods

Besides LightGBM, a deep learning feature generation method based on CNN is applied to generate effective features for subsequent deep learning classifiers. This study compares the performance of IPNN classifier before and after using the deep learning-based feature generation method to verify whether it is beneficial for improving the predictive performance of deep learning classifier. Moreover, the performance of traditional MLP is obtained as the baseline.

Table 5 compares the performance of MLP, IPNN, and $IPNN_{fgc}$ to verify the effectiveness of $IPNN_{fgc}$ and the deep learning-based feature generation method. In DefaultData dataset, Accuracy of IPNN improves 2.063, AUC improves 6.836, KS improved 0.083, and Loss declines 0.172 compared to traditional MLP. In PPDai dataset, the performance of IPNN classifier has also been effectively improved compared to that of MLP. That is to say, the performance of IPNN classifier is superior to traditional MLP classifier under various evaluation metrics (i.e., Accuracy, AUC, KS, and Loss) in both DefaultData dataset and PPDai dataset, indicating that selecting IPNN as deep learning classifier is reasonable and effective. Compared to IPNN, $IPNN_{fgc}$ also outperforms IPNN in the two datasets. Accuracy of $IPNN_{fgc}$ is 0.300 higher than that of IPNN in DefaultData dataset. AUC and KS of $IPNN_{fgc}$ is 0.390 and 0.006 higher than that of IPNN respectively. Loss of $IPNN_{fgc}$ is 0.006 lower than that of IPNN. Similar results are obtained in another dataset. Accuracy of $IPNN_{fgc}$ is 0.134 higher than that of IPNN, AUC of $IPNN_{fgc}$ is 3.193 higher than that of IPNN, KS of $IPNN_{fgc}$ is 0.051 higher than that of IPNN, and Loss of $IPNN_{fgc}$ is 0.010 lower than that of IPNN in PPDai dataset. The performance results indicate that the feature selection method based on deep learning method is beneficial to improve the predictive performance of deep learning classifier.

**Table 5**
Comparison of the performance results between different deep learning methods.

| DataSet | Classifier | Accuracy | AUC | Loss | KS |
|---|---|---|---|---|---|
| DefaultData | MLP | 79.817 ± 0.570 | 71.324 ± 0.721 | 0.607 ± 0.014 | 0.355 ± 0.014 |
| | IPNN | 81.880 ± 0.412 | 78.160 ± 0.679 | 0.435 ± 0.011 | 0.438 ± 0.014 |
| | $IPNN_{fgc}$ | 82.180 ± 0.454 | 78.550 ± 0.553 | 0.429 ± 0.009 | 0.444 ± 0.011 |
| PPDai | MLP | 87.038 ± 0.258 | 66.848 ± 0.363 | 0.365 ± 0.004 | 0.239 ± 0.004 |
| | IPNN | 87.047 ± 0.218 | 67.850 ± 0.446 | 0.362 ± 0.006 | 0.258 ± 0.005 |
| | $IPNN_{fgc}$ | 87.181 ± 0.204 | 71.043 ± 0.364 | 0.352 ± 0.004 | 0.309 ± 0.005 |

*Note*: $IPNN_{fgc}$ indicates the predictive performance of IPNN after feature generation using CNN.

### 5.4. Comparisons of the proposed model and its base classifiers

The tree-based method (i.e., LightGBM) and the deep learning method (i.e., CNN) have been used and verified as effective feature generation methods. In order to verify the effectiveness of classifier ensemble process, final performance results of the proposed model are compared with the performance of base classifiers. The comparison results are shown in Table 6.

Table 6 shows that the proposed model is superior to other comparative base classifiers based on comprehensive evaluation metrics. More specifically, three tree-based ensemble classifiers (i.e., RF, XGBoost, and LightGBM) have achieved excellent predictive performance in both datasets with the new features generated by LightGBM. Moreover, the deep learning method (i.e., $IPNN_{fgc}$) outperforms these tree-based ensemble classifiers based on comprehensive evaluation metrics. It indicates the effectiveness of deep learning method from another aspect. Final prediction result of the proposed model is improved compared to the performance of four comparative base classifiers, indicating that the hybrid ensemble model combining tree-based method and deep learning method is efficient and effective.

**Table 4**
Comparison of prediction results before and after tree-based feature generation.

| DataSet | Classifier | Feature Generation | Accuracy | AUC | Loss | KS |
|---|---|---|---|---|---|---|
| DefaultData | LR | LR | 82.022 ± 0.453 | 76.557 ± 0.510 | 0.437 ± 0.008 | 0.414 ± 0.010 |
| | | $LR_{fgl}$ | 82.093 ± 0.624 | 77.277 ± 0.703 | 0.434 ± 0.008 | 0.420 ± 0.010 |
| | MLP | MLP | 79.480 ± 0.452 | 70.930 ± 0.502 | 0.617 ± 0.013 | 0.348 ± 0.014 |
| | | $MLP_{fgl}$ | 79.817 ± 0.570 | 71.324 ± 0.721 | 0.607 ± 0.014 | 0.355 ± 0.014 |
| PPDai | LR | LR | 87.047 ± 0.259 | 65.029 ± 0.183 | 0.368 ± 0.002 | 0.222 ± 0.004 |
| | | $LR_{fgl}$ | 87.084 ± 0.243 | 67.501 ± 0.188 | 0.362 ± 0.002 | 0.251 ± 0.007 |
| | MLP | MLP | 86.781 ± 0.398 | 63.604 ± 0.447 | 0.379 ± 0.004 | 0.201 ± 0.009 |
| | | $MLP_{fgl}$ | 87.038 ± 0.258 | 66.848 ± 0.363 | 0.365 ± 0.004 | 0.239 ± 0.004 |

**Table 6**
Comparisons between the proposed model and its base classifiers.

| DataSet | Classifier | Accuracy | AUC | Loss | KS | Avg rank |
|---|---|---|---|---|---|---|
| DefaultData | $RF_{fgl}$ | 81.637 ± 0.238 | 76.346 ± 0.417 | 0.445 ± 0.008 | 0.406 ± 0.008 | 5 |
| | $XGBoost_{fgl}$ | 82.039 ± 0.374 | 78.139 ± 0.492 | 0.430 ± 0.010 | 0.435 ± 0.010 | 3 |
| | $LightGBM_{fgl}$ | 81.828 ± 0.366 | 77.448 ± 0.420 | 0.436 ± 0.009 | 0.421 ± 0.008 | 4 |
| | $IPNN_{fgc}$ | 82.180 ± 0.454 | 78.550 ± 0.553 | 0.429 ± 0.009 | 0.444 ± 0.011 | 2 |
| | The proposed model | 82.410 ± 0.369 | 78.735 ± 0.352 | 0.426 ± 0.006 | 0.446 ± 0.007 | 1 |
| PPDai | $RF_{fgl}$ | 87.097 ± 0.226 | 69.547 ± 0.244 | 0.36 ± 0.009 | 0.286 ± 0.005 | 5 |
| | $XGBoost_{fgl}$ | 87.129 ± 0.242 | 71.042 ± 0.174 | 0.353 ± 0.007 | 0.308 ± 0.003 | 3.25 |
| | $LightGBM_{fgl}$ | 87.14 ± 0.254 | 70.982 ± 0.363 | 0.353 ± 0.006 | 0.308 ± 0.003 | 3.75 |
| | $IPNN_{fgc}$ | 87.181 ± 0.204 | 71.043 ± 0.364 | 0.352 ± 0.004 | 0.309 ± 0.005 | 2 |
| | The proposed model | 87.273 ± 0.202 | 71.947 ± 0.200 | 0.351 ± 0.006 | 0.320 ± 0.004 | 1 |

*Note*: AvgRank represents the average rank of each classifier in all evaluation metrics. $RF_{fgl}$, $XGBoost_{fgl}$, $LightGBM_{fgl}$ are the predictive performance of RF, XGBoost, and LightGBM respectively after feature generation using LightGBM; $IPNN_{fgc}$ indicates the predictive performance of IPNN after feature generation using CNN.

## 6. Conclusions and future work

Default prediction and credit scoring have attracted increasingly attention from all sectors of society, and they have gradually been important aspects for financial risk control in recent years. Correctly distinguishing default instances can effectively reduce the probability of default and bring substantial economic benefit for financial institutions. Emerging financial market requires more effective default prediction techniques, and developing artificial intelligence techniques provides a favorable condition for improving the performance of default prediction models. In this study, a hybrid ensemble model combining feature generation, deep learning, and ensemble learning is proposed to explore strategies to improve model performance from different perspectives. First, LightGBM is used to build new feature interactions to enhance feature expression. Second, $IPNN_{fgc}$ consisting of three components is applied in this study: feature embedding is applied to map high-dimensional sparse features into low-dimensional dense spaces; CNN is used to construct new feature interactions to reflect deeper correlation information between features; a deep learning classifier (i.e., IPNN) is used to predict instances with new features. Third, an ensemble model combining deep learning classifier (i.e., IPNN) and three tree-based classifiers (i.e., RF, XGBoost, and LightGBM) are used to obtain final prediction results. The experimental results show that: 1) tree-based feature generation (i.e., LightGBM) can effectively improve the predictive performance of the classifiers; 2) deep learning feature generation method based on CNN generates effective features for subsequent deep learning classifiers and benefits for improving the performance of default prediction; 3) ensemble method of the deep learning classifier (i.e., IPNN) and tree-based classifiers (i.e., RF, XGBoost, and LightGBM) are demonstrated to improve the final results of default prediction. This study provides a new perspective for feature generation and deep learning in default prediction. It not only explores a new probability for default research, but also benefits practical default prediction for financial institutions.

This study combines the advantage of tree-based models and deep learning models in feature generation and model construction. Some novel points have been covered in this study, but there is some further exploration can be done in the future work: 1) some advanced deep learning models such as ResNet and DenseNet have been developed. In the future work, these methods will be attempted to use for feature generation to further improve the predictive performance of deep learning classifiers. 2) hyperparameters method can improve the performance of prediction model to some extent, but the feature generation methods used in this study (i.e., LightGBM and CNN) have not been optimized for hyperparameters. Therefore, hyperparameters of feature generation methods will be optimized in the future work; 3) in order to avoid the transmission of errors in different stages of the proposed model, end-to-end method can be explored for default prediction.

## CRediT authorship contribution statement

**Hongliang He:** Conceptualization, Formal analysis, Methodology, Project administration, Supervision, Writing - original draft, Software, Validation. **Yanli Fan:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Writing - original draft.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgment

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

## Appendix 1. . Summary description of the used dataset

| DefaultData | Type | Definition |
|---|---|---|
| *LIMIT_BAL* | Numeric | Amount of the given credit: it includes both the individual consumer credit and his/her family (supplementary) credit |
| *SEX* | Categorical | Gender (1 = male; 2 = female) |
| *EDUCATION* | Categorical | Education (1 = graduate school; 2 = university; 3 = high school; 4 = others) |
| *MARRIAGE* | Categorical | Marital status (1 = married; 2 = single; 3 = others) |
| *AGE* | Numeric | Age |
| *PAY_0-PAY_6* | Categorical | History of past payment. *PAY_6* = the repayment status in September 2005; *PAY_5* = the repayment status in August 2005; …; *PAY_0* = the repayment status in April 2005. The measurement scale for the repayment status is: −1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; …; 8 = payment delay for eight months; 9 = payment delay for nine months and above |

(*continued on next page*)

(*continued*)

| DefaultData | Type | Definition |
|---|---|---|
| *BILL_AMT1-BILL_AMT6* | Numeric | Amount of bill statement. *BILL_AMT1* = amount of bill statement in September 2005; *BILL_AMT2* = amount of bill statement in August 2005; …; *BILL_AMT6* = amount of bill statement in April 2005 |
| *PAY_AMT1-PAY_AMT6* | Numeric | Amount of previous payment. *PAY_AMT1* = amount paid in September 2005; *PAY_AMT2* = amount paid in August 2005; …; *PAY_AMT6* = amount paid in April 2005 |
| *default payment next month* | Binary classified | Default payment (1 = Yes, 0 = No) |
| PPDai | Type | Definition |
| *sex* | Categorical | Gender |
| *occupation* | Categorical | Occupation |
| *education* | Categorical | Education |
| *marriage* | Categorical | Marital status |
| *household* | Categorical | Types of registered permanent residence |
| *income* | Numeric | Average income |
| *outcome* | Numeric | Average outcome |
| *income_tm* | Numeric | Unix timestamp for average income |
| *outcome_tm* | Numeric | Unix timestamp for average outcome |
| *tm_encode_3* | Numeric | Unix timestamp for the bill |
| *prior_account* | Numeric | Amount of prior bill statement |
| *prior_repay* | Numeric | Amount of prior repayment |
| *credit_limit* | Numeric | Amount of the given credit |
| *account_balance* | Numeric | Current bill balance |
| *minimun_repay* | Numeric | Minimum payment on current bill |
| *consume_count* | Numeric | Number of consumptions |
| *account* | Numeric | Amount of current bill statement |
| *adjust_account* | Numeric | Adjustment amount of the bill |
| *circulated_interest* | Numeric | Cycle interest |
| *avaliable_balance* | Numeric | Available balance |
| *cash_limit* | Numeric | Cash advance limit |
| *repay_state* | Numeric | Repayment status |
| *browse_data* | Numeric | Total number of browse per user |
| *loan_time* | Numeric | Unix timestamp of releasing the loan |
| *time* | Numeric | *Loan_time - tm_encode_3* |
| *yu_e* | Numeric | *Income - outcome* |
| *yu_e_categery* | Categorical | Classified based on *yu_e* (1 = $yu\_e > 0$, 2 = $yu\_e = 0$, 3 = $yu\_e < 0$) |
| *yu_e_tm* | Numeric | *Income_tm - outcome_tm* |
| *yu_e_tm_categery* | Categorical | Classified based on *yu_e_tm* (1 = $yu\_e\_tm > 0$, 2 = $yu\_e\_tm = 0$, 3 = $yu\_e\_tm < 0$) |
| *label* | Binary classified | 1 = payment delay for>30 days; 0 = payment delay within 10 days; payment delay from 10 to 30 days is not included in this dataset |

Note: PPDai dataset has not released the concrete personal information due to confidentiality, but it does not affect the application in our study.

## References


Abellán, J., & Castellano, J. G. (2017). A comparative study on base classifiers in ensemble methods for credit scoring. *Expert Systems with Applications, 73*, 1–10.

Ala'raj, M., & Abbod, M. F. (2016a). Classifiers consensus system approach for credit scoring. *Knowledge-Based Systems, 104*, 89–105.

Ala'raj, M., & Abbod, M. F. (2016b). A new hybrid ensemble credit scoring model based on classifiers consensus system approach. *Expert Systems with Applications, 64*, 36–55.

Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance, 23*(4), 589–609.

Asuncion, A., & Newman, D. (2007). *UCI Machine Learning Repository*. Irvine, CA: School of Information and Computer Science, University of California. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Bastani, K., Asgari, E., & Namavari, H. (2019). Wide and deep learning for peer-to-peer lending. *Expert Systems with Applications, 134*(15), 209–224.

Dahiya, S., Handa, S. S., & Singh, N. P. (2016). Impact of baggingon MLP classifier for credit evaluation. In Proceedings 3rd International Conference on Computing for Sustainable Global Development. Delhi, India, March, 2016, pp. 3794-3800.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM, 55*(10), 78–87.

Finlay, S. (2011). Multiple classifier architectures and their application to credit risk assessment. *European Journal of Operational Research, 210*(2), 368–378.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. Annals of Human Genetics (in print), Annals of Human Genetics, UCL and Blackwell Publishing Ltd (online), 179-188.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics, 29*(5), 1189–1232.

Hand, D. J., & Henley, W. E. (1997). Statistical classification methods in consumer credit scoring: A review. *Journal of the Royal Statistical Society, 160*(3), 523–541.

Hand, D. J., & Kelly, M. G. (2002). Superscorecards. *Ima Journal of Management Mathematics, 13*(4), 273–281.

He, H., Zhang, W., & Zhang, S. (2018). A novel ensemble method for credit scoring: Adaption of different imbalance ratios. *Expert Systems with Applications, 98*, 105–117.

He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., … & Candela, J.Q. (2014). Practical lessons from predicting clicks on ads at Facebook. In Proceedings of the Eighth International Workshop on Data Mining for Online Advertising, New York, USA, August 24–27, 2014, pp. 1–9.

Huang, Z., Chen, H., Hsu, C. J., Chen, W. H., & Wu, S. (2004). Credit rating analysis with support vector machines and neural network: A market comparative study. *Decision Support Systems, 37*(4), 543–558.

Juan, Y., Zhuang, Y., Chin, W.S., & Lin, C.J. (2016). Field-aware factorization machines for CTR prediction. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, USA, September 15–19, 2016, pp. 43–50.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., … & Liu, T.Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In Proceedings of Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, December 4–9, 2017, pp. 3146–3154.

Kim, J.Y., & Cho, S.B. (2018). Deep dense convolutional networks for repayment prediction in peer-to-peer lending, In Proceedings of 13th International Conference on Soft Computing Models in Industrial and Environmental Applications, San sebastian, Spain, June 6–8, 2018, pp. 134–144.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.

Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136.

Li, P., Wu, Q., & Burges, C.J.C., (2007). Mcrank: Learning to rank using multiple classification and gradient boosting. In Proceedings of the 20th International Conference on Neural Information Processing Systems, December, 2007, pp. 897–904.

Li, X., Ying, W., Tuo, J., & Li, B. (2004). Applications of classification trees to consumer credit scoring methods in commercial banks. In Proccedings of IEEE International Conference on Systems, Man and Cybernetics, Hague, Netherlands, October 10–13, 2004, Vol. 5, pp. 4112–4117.

Ma, X., Sha, J., Wang, D., Yu, Y., Yang, Q., & Niu, X. (2018). Study on a prediction of P2P network loan default based on the machine learning LightGBM and XGboost algorithms according to different high dimensional data cleaning. *Electronic Commerce Research and Applications, 31*, 24–39.

Liu, B., Tang, R., Chen, Y., Yu, J., Guo, H., & Zhang, Y. (2019). Feature generation by Convolutional Neural Network for Click-Through Rate prediction. In Proceedings of

The World Wide Web Conference 2019, San Francisco, USA, May 13-17, 2019, pp. 1119-1129.
Marqués, A. I., García, V., & Sánchez, J. S. (2012). Exploring the behaviour of base classifiers in credit scoring ensembles. *Expert Systems with Applications, 39*(11), 10244–10250.
Qu, Y., Cai, H., Ren, K., Zhang, W., Yu, Y., Wen, Y., & Wang, J. (2016). Product-based neural networks for user response prediction. In Proceedings of 2016 IEEE International Conference on Data Mining, Barcelona, Spain, December 12-15, 2016, pp. 1149-1154.
Rendle, S. (2010). Factorization machines. In Proceedings of 2010 IEEE International Conference on Data Mining, Sydney, Australia, December 13-17, 2010, pp. 995-1000.
Serrano-Cinca, C., & Gutiérrez-Nieto, B. (2016). The use of profit scoring as an alternative to credit scoring systems in peer-to-peer (P2P) lending. *Decision Support Systems, 89*, 113–122.
Swami, A., & Jain, R. (2013). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research, 12*(10), 2825–2830.
Tsai, C. (2014). Combining cluster analysis with classifier ensembles to predict financial distress. *Information Fusion, 16*, 46–58.
Wang, G., Ma, J., Huang, L., & Xu, K. (2012). Two credit scoring models based on dual strategy ensemble trees. *Knowledge-Based Systems, 26*, 61–68.
Xia, Y., Liu, C., Da, B., & Xie, F. (2018). A novel heterogeneous ensemble credit scoring model based on bstacking approach. *Expert Systems with Applications, 93*, 182–199.
Xia, Y., Zhao, J., He, L., Li, Y., & Niu, M. (2020). A novel tree-based dynamic heterogeneous ensemble method for credit scoring. *Expert Systems with Applications, 159*, 113615. https://doi.org/10.1016/j.eswa.2020.113615
Zavgren, C. V. (1985). Assessing the vulnerability to failure of American industrial firms: A logistic analysis. *Journal of Business Finance & Accounting, 12*(1), 19–45.
Zhang, H., He, H., & Zhang, W. (2018). Classifier selection and clustering with fuzzy assignment in ensemble model for credit scoring. *Neurocomputing, 316*, 210–221.
Zhang, W., He, H., & Zhang, S. (2019a). A novel multi-stage hybrid model with enhanced multi-population niche genetic algorithm: An application in credit scoring. *Expert Systems with Applications, 121*, 221–232.
Zhang, X., Han, Y., Xu, W., & Wang, Q. (2019b). HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture. *Information Sciences*. https://doi.org/10.1016/j.ins.2019.05.023
Zięba, M., Tomczak, S. K., & Tomczak, J. M. (2016). Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. *Expert Systems with Applications, 58*, 93–101.

## *Further reading*

Breiman, L. (2001). Random forests. *Machine Learning, 45*, 5–32.
Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *In Proceedings of the 22nd ACM sigkdd International Conference on Knowledge Discovery and Data Mining*, 785–794.
West, D. (2000). Neural network credit scoring models. *Computers & Operations Research, 27*, 1131–1152.