2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress

User Portrait Technology Based on Stacking Mode

Yirui Wu
College of Computer and Information,
Hohai University
Nanjing City, China
wuyirui@hhu.edu.cn

Pengyu Yu
College of Computer and Information,
Hohai University
Nanjing City, China
jadesperwalker@qq.com

Abstract—**Because of the differences in code styles and programming levels among developers, it is prone to code irregularities, poor readability, and security vulnerabilities. Although developers can see their problems in the test report, it is difficult to guarantee that they will not make mistakes on the same problem. In this article, we propose a way to build a programming level for developers. The proposed method explores the Stacking model for building a programming level for developers. First, cluster the problems that occurred during the development process, give weight to the category information, score the developer's programming level, and divide the user groups into different categories. Then use the word vector to extract the features of the code defect, generate a feature matrix, and finally pass the feature matrix to the Stacking classifier to classify the defect information and update the developer's programming level portrait. Experimental results show that it is effective in predicting defect code information. In addition, a comparative study with the state-of-the-art method shows that the method is superior to existing methods in terms of classification rate, recall, precision and F-measure.**

Keywords—code defect, clustering, stacking model, user portrait and code defect prediction

I. Introduction

In the traditional software development process, the problem of code defects is usually sent to the corresponding developers, and the analysis of the developer's programming level and the correlation of code problems is not done from the perspective of the developer. As a result, it's easy for developers to still commit the same problem. The main reason is that there is no correlation between the code problem and the developer's own ability, which causes the developer to ignore the code problem and not improve his own problem. In addition, when evaluating the programming level of developers, often relevant personnel need to manually collect all the defect issues, and classify and score the defect issues. Then score the developer's programming level based on the code problems the developer has encountered. The process is cumbersome and the degree of automation is low. In order to solve the above problems, this work focuses on the classification of code defect information and the construction of developer user portraits to help developers improve their programming skills.

User portrait is a model that describes user information from multiple dimensions. User portraits are based on the attributes and behaviors of users in real life, abstracting tags from multiple dimensions to restore the user's true appearance as much as possible. After the user is tagged, the enterprise can accurately locate the specific user according to the tag, so as to customize corresponding policies for different users and reduce the cost of pushing.

Marquar et al. [1] used a multi-label classification method to predict the user's gender and age. Cai Guoyong et al. [2] analyzed the emotional connection between graphics and text based on convolutional neural network, and it has a good effect on the emotion prediction of graphics and text fusion. Torres-Valencia et al. [3] used support vector machines to analyze the emotional characteristics of users. Kuzma et al. [4] extract user preference characteristics based on neural network models. Mueller et al. [5] used multiple word structure features to gender-identify Twitter username information.

In 2001, Webb GI et al. [6] applied machine learning algorithms to the construction of user portrait models, and proposed that when constructing user portraits, problems such as large data sets and user characteristic attributes that change over time are required. In 2003, Degemmis M et al. [7] used a text classification algorithm to extract the user's interest habits, and successfully applied user portraits to personalized recommendations. In 2010, Zheng Baoxin et al. extracted targeted user feature information and targeted promotion of opponent game players. In 2015, Liu Hai et al. used a clustering algorithm to classify users, dig out potential connections between consumers and products, and conduct accurate marketing for consumers. Gu et al. proposed a method of modeling linguistic and psychological characteristics to dig out the relationship between the user's personality characteristics and their behaviors, so that business organizations can better serve the user population. Wu Tongshui et al. used a decision tree algorithm to analyze airline customers, which enables airlines to take corresponding improvement measures for customer churn.

Chen Yan et al. [9] believe that when constructing user portraits, two aspects of the timeliness of user data and the dynamics of portrait data should also be considered. The user's behavior will change over time. When constructing a portrait, if you select some older user data and do not update the portrait, the value of the portrait will be difficult to reflect.

Although many domestic and foreign scholars have conducted in-depth research on user portraits, and their application fields are also very wide, they are mostly used in the personal customization, business analysis, precision marketing, and user statistics of user groups. The lack of programming is rarely involved and not researched enough. In addition, in the traditional evaluation of developer programming level, manual collection and experience evaluation methods are often used, the degree of automation is too low and it cannot objectively reflect the programming level of developers. Therefore, it is necessary to develop a method that is applied to the portrait of the developer user, which can extract the characteristics of the defect information and predict the developer's programming level.

Therefore, in this work, we propose a new method based on the Stacking model and K-means clustering algorithm for building developer user portraits. K-means clustering algorithm can automatically classify the defects of developers. Then the relevant personnel give weights to the results of clustering, and then score the developers' programming ability according to the weight and defect information. Scoring builds a portrait of the developer's programming level. The SVR (Support Vactor Regerssion) algorithm, the RF (Random Forest) algorithm, and the GBDT (Gradient Boosting Decision Tree) algorithm are combined into a Stacking model, and then the defects of the developer are predicted, and the developer portrait is updated again. This is to remind developers where to pay attention to when programming, reducing the problems that developers have during the programming process [10,11].

II. Proposed Method

In this work, there are three parts: clustering process, training process and prediction process as shown in Fig.1. For the clustering process, first perform data preprocessing on the defect information, remove redundant and irrelevant data, and then use the improved K-means algorithm to classify all the defect information, then label the defect information with a category label and give weight to the category information . The developer's defect information and the category of the defect information are scored, and finally the programming level of the developer is classified to divide the user groups of different categories. For the training process, feature extraction is first performed on the defect information with categories, and then the feature matrix is input into the Stacking algorithm for training to obtain a Stacking classification model. For the prediction process, first extract the characteristics of the defect information, then input it into the Stacking classification model for classification, evaluate the prediction ability of the Stacking classification model, and finally re-score the developer's programming level and update the user group.

A. User Group Analysis Based on Improved K-means

For the division of developer user groups, we use the K-means algorithm and optimize the K-means algorithm to enable it to quickly and efficiently select clustering centers and clustering results. For the selection of the initial center and the optimization of the complexity of the K-means algorithm, the maximum distance principle is adopted, and the object with the largest distance difference is used as the cluster center. The specific optimization steps are as follows.

1) For the data set $S\_n = \{x\_1, x\_2, \dots, x\_n\}$, calculate the distance between any two points, and select the two points p and q with the largest distance from them. Note the first cluster center,

$$\mathrm{x}1 = x_p \tag{1}$$

the second Clustering center.

$$x2 = x_q \tag{2}$$

2) For a given sample point x_i and cluster centers x1 and x2. if,

$$|\mathrm{x}_i - x1| < |\mathrm{x}_i - \mathrm{x}2| \tag{3}$$

Then divide x_i into the class with x1 as the cluster center, otherwise divide it into the class with x2 as the cluster center, and finally divide the entire data set into S1 and S2 classes.

3) First calculate the Euclidean distance from the sample points in the S1 set to the cluster center x1, and select the sample point with the largest distance from it, satisfying $d1 = max\,\{|x_i - x1|, x_i \in S1\}$, and then calculate the The Euclidean distance from each sample point to x2, we get $d2 = max\,\{|x_i - x2|, x_i \in S2\}$. Note $d3 = max\,\{d1, d2\}$, the sample point that meets this condition is x3, and x3 is used as the third initial cluster center;

4) Repeat the above steps *2)* and *3)* until k initial cluster centers are found;

5) Calculate the Euclidean distance between any two initial cluster centers, and record it as $\mathrm{d}(x_i, x_j)$;

6) Traverse the sample points in the data set S, and calculate the distance from each sample point to each cluster center. For a given sample point x_j, the cluster center with the smallest distance is selected, and the sample points are divided into corresponding classes. For a given sample point x and two cluster centers x_p and x_q, if

$$\mathrm{d}(\mathrm{x}, x_p) \le \frac{1}{2} d(x_p, x_q) \tag{4}$$

then

$$\mathrm{d}(\mathrm{x}, x_p) \le d(\mathrm{x}, x_q) \tag{5}$$

7) Repeat steps *5)* and *6)*, iterating multiple times until the objective function value converges, then the clustering process ends.

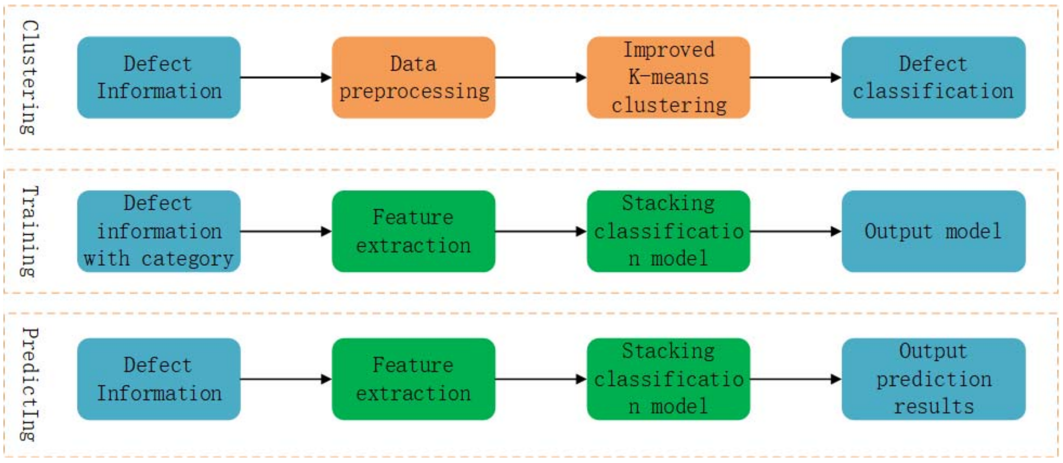


Fig. 1. User portrait technology workflow based on Stacking model.

B. Stacking-based Code Defect Prediction Model

When the prediction capability of a single model encounters a bottleneck, it can be used as a basic model and the ensemble learning method can be used to further improve the prediction effect [12]. In order to further improve the accuracy of the code prediction model, we propose a two-layer ensemble learning algorithm model based on the Stacking algorithm, as shown in Fig.2. In the first layer model, we use the GBDT algorithm, Random Forest algorithm, and SVR algorithm with better prediction capabilities to fuse, and fully mine the feature information related to code defects. In the second model, we used the LR (Logistic Regression) algorithm to reduce overfitting problems during training.

For the training stage, in the first layer model, the Stacking model uses the K-fold cross-validation method in the prediction process of each base classifier. The training sample set is divided into k copies. And one is selected as the verification set. The others are selected as training sets. Since the validation sets are different from each other, and each fold is predicted on the base classifier, for a base classifier, after using K-fold cross-validation, k unique prediction results can be obtained. It can be found that after the stitching of the k prediction results is the prediction result of the entire training sample set. In the second layer model, the prediction results of each base classifier are merged side by side to obtain a feature matrix, and the real result of the training set is used as the output matrix of the model, which is brought into the base classification model of the second layer. Train to get the final Stacking classification model.

For the test phase, in the first layer model, since the test sample set is predicted at every fold in the training phase, for a base classifier, the prediction results of k test sets can be obtained. Then add their parts and take the average to get the average prediction result of the test set. In the second layer model, the prediction results of each base classifier are merged side by side to obtain a feature matrix, and the stacking model obtained during the training phase is used to test the test sample set.

The algorithm described in the above steps is shown in Table I.

TABLE I. ALGORITHM OF USER PREDICTION MODEL BASED ON STACKING ALGORITHM

Algorithm of user prediction model based on Stacking algorithm

```
Input:
Training set: T = $((x_1,y_1),(x_2,y_2),\dots,(x_n,y_n))$
Base classifier: GBDT, RF, SVR
Meta Classifier: LR
Output:
Classification model: Stacking Model
$T_1,T_2,\dots,T_k = k\_fold(T,k)$
meta_T={}
foreach $T_j$ in $\{T_1,T_2,\dots,T_k\}$:
GBDT[j] = GBDT(T-$T_j$)
SVR[j]=SVR(T-$T_j$)
RF[j]=RF(T-$T_j$)
foreach $x_i$ in $T_j$:
    $y_{i1}$=GBDT[j].predict($x_i$)
    $y_{i2}$=SVR[j].predict($x_i$)
    $y_{i3}$=RF[j].predict($x_i$)
    meta_T.append(($y_{i1},y_{i2},y_{i3}$), $y_1$)
endfor
endfor
Stacking = LR(meta_T)
```

The design steps of the user prediction model based on the Stacking algorithm are as follows:

1) Sample the code defect data after clustering to obtain the code defect data training set and test set, and then use the cross-validation method to divide the code defect data training set into 5 parts;

2) Using random forest algorithm, GBDT algorithm, and SVR algorithm to predict the cross-validation training set respectively, the prediction results of the three basic models can be obtained, and then the prediction results and corresponding feature labels are combined to obtain the meta feature vector;

3) The meta-feature vector is used as the new training data set, and the meta-feature vector is trained by the LR algorithm to obtain the final Stacking classification model;

4) Use the Stacking classification model to predict the test samples.

III. EXPERIMENTAL RESULTS

In this work, we will analyze the user group results of the improved K-means algorithm and analyze the code defect prediction model based on the Stacking algorithm

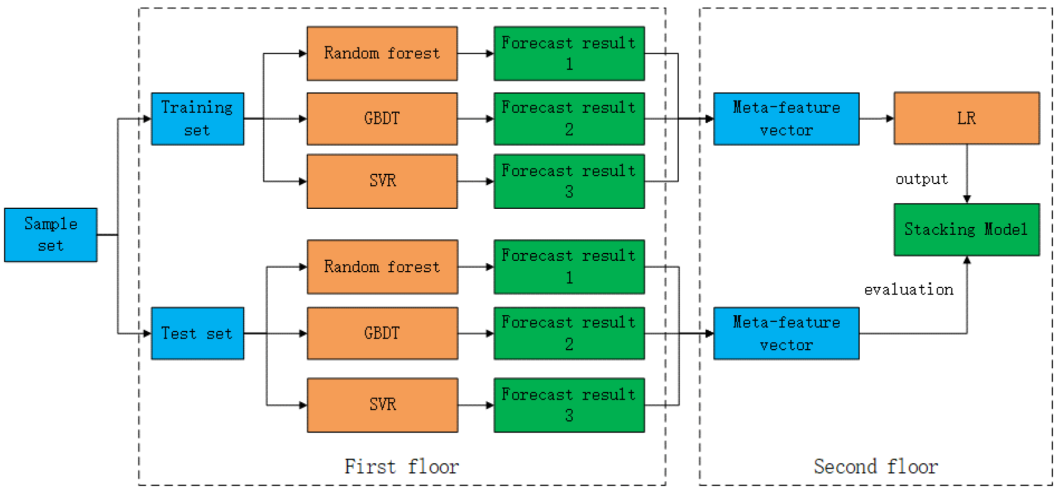


Fig. 2 Framework diagram of code defect prediction model based on Stacking model

A. User Group Analysis Based on Improved K-means

The clustering algorithm is an unsupervised learning algorithm [13]. When classifying defect information, the number of classes and the meaning of each class cannot be well determined. Therefore, we use SSE (Sum of the Squared Errors) to evaluate the number of categories. The calculation method of SSE is shown in equation (6).

$$\mathrm{SSE} = \sum_{i=1}^{k} \sum_{x \in C_i} dist(x, m_i) \qquad (6)$$

Where k represents the number of clusters, C_i represents the i-th cluster, x represents the sample points in C_i, m_i is the cluster center of C_i, and dist represents the Euclidean distance from the sample points to the cluster center.

First, select all the defect information appearing in the developer for one year as the input features of the clustering algorithm, and then set the range of the number of clustering clusters k to an integer value between 2 and 20. Then perform a cluster analysis on each cluster number. The value of the clustering error SSE is obtained by calculation. Finally, the relationship between the clustering error SSE and the number of clustering clusters k is shown in the form of a line chart, as shown in Fig.3.

In the evaluation of the number k of clusters, the number of clusters can be determined by the elbow method. When the number of clusters gradually increases, the sample will be divided more and more finely, the degree of aggregation of each category will gradually increase, and the clustering error SSE will gradually decrease.

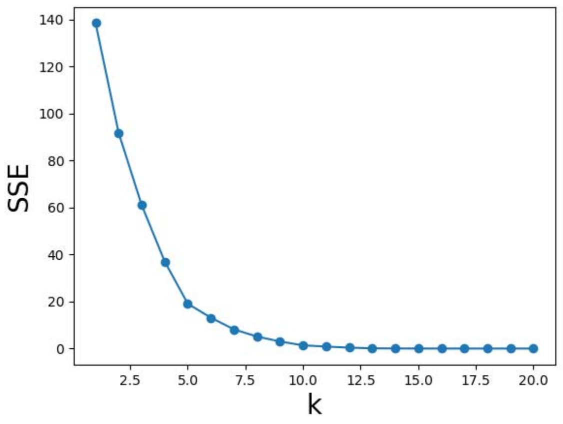


Fig. 3. Polyline relationship diagram of clustering error SSE and the number of clustering clusters k

It can be found that when the number of clustering clusters k is smaller than the number of real clustering clusters, as the value of k increases, the clustering error will decrease rapidly. When the number of clustering clusters k is larger than the number of real clusters, as the value of k increases, the clustering errors tend to be flat. When the number of clustering clusters is 5, it is the true number of clustering clusters. However, when the number of clustering clusters is set to 5 for clustering, there is very little information in one of the categories. Therefore, we set the number of cluster clusters to 4, and then use the improved K-means algorithm to divide the four types of defect information, as shown in Fig.4.

1	io,sendsshcommand,sshutil,reftype,fileutil,class,removedir,inputstream
2	character,140,space,due,useless,org,util
3	field,baseentity,unused,variable,local,named,one2many,string
4	name,testdir,file,dependencyanalysisproject,init,example,src,com

Fig. 4. Classification of defect information

It can be found that the first type is mainly defect information such as the io operation. When an error occurs in this type, it will directly cause the function of the project to be unavailable, which belongs to the severity level. The second type is mainly the defect information related to the guide space and the character space. For this type of error, the code operation is not standardized, and the irrelevant third-party component packages introduced are low-level errors. The third type is mainly some unused variables and named related variables or strings. For this kind of defects, mainly after the code is written, some variables are not used, which is a medium-level error. The fourth category is mainly file processing and some function initialization operations. For this type of error information, mainly the initialization of functional modules and problems of test analysis are high-level errors.

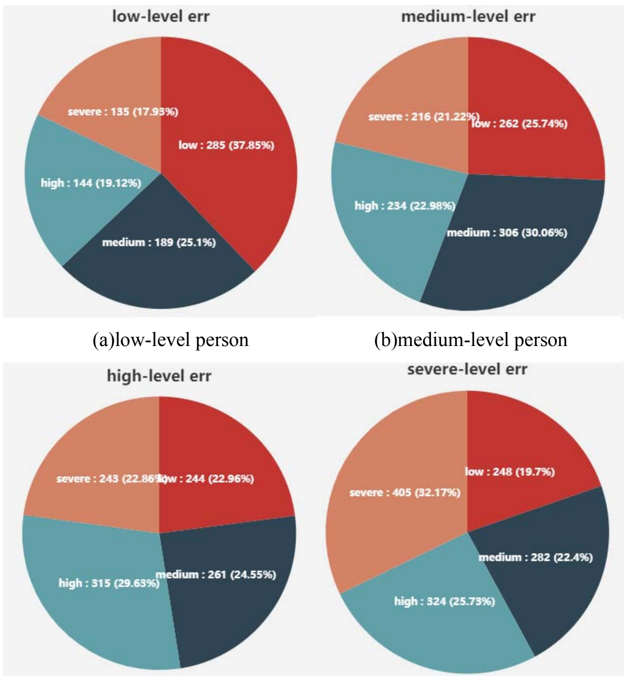
Therefore, the developer's defect information can be classified and counted, and various types of defect information can be given weights. Finally, the developer's coding level is evaluated according to the various types of scores. A category with a high proportion is defined as a category where users are prone to problems, as shown in equation (5).

$$\omega_1 * low : \omega_2 * middle : \omega_3 * high : \omega_4 * serious \quad (5)$$

Where ω_1, ω_2, ω_3, and ω_4 are the weights of low, medium, high, and severe respectively. According to past experience, the weights of each level are set to 1, 3, 9, and 27 respectively.

Finally, according to the proportion of each level, developers can be divided into those prone to low-level errors, intermediate-level errors, high-level errors, and severe-level errors. Fig.5 shows the errors of personnel at all levels.

It can be found that when the defect type is given weight information, the system will be more fair in scoring the developer's programming, and it is no longer just to measure a developer's programming level based on the number of defect types. For developers prone to low-level problems, it is necessary to improve their programming habits. Have a good programming practice, on the one hand can enhance the readability of your own code, and on the other hand, it is convenient for others to quickly understand the function of the code during maintenance. Developers who are prone to intermediate-level problems need to use them in a timely manner when declaring variables. If they are not used after declaration, they need to be deleted when submitting code. For C and C ++ programming languages, variables need to be released in a timely manner when they are no longer used. For developers who are prone to high-level problems, you need to pay attention to the function completeness and initialization issues. For developers prone to serious level problems, you need to pay attention to whether the software crashes or exits abnormally due to io operation or parameter interface call errors.



(a)low-level person (b)medium-level person

(c)high-level person (d)severe-level person

Fig. 5. The errors of personnel at all levels.

B. *Stacking-based Code Defect Prediction Model*

For an ensemble learning algorithm, if the prediction effect of the ensemble model is better than that of any basic model, the construction of the ensemble model is considered successful. The standard metrics (ie recall, precison and F-measure) are used to calculate the metrics defined in equations (6)-(8). True (TP) is the number of correct classifications and belongs to the positive category; true negative (TN) is the number of incorrect classifications and belongs to the negative category. False positive (FP) and false negative (FN) are in the positive and negative categories, respectively the number of classification errors. With these definitions, we define the following metrics:

$$\text{precision} = \frac{TP}{TP + FP} \tag{7}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{8}$$

$$\text{F1} - \text{score} = \frac{2 * Precision \ * \ Recall}{Precision \ + \ Recall} \tag{9}$$

We first used the base classification algorithm to perform category prediction of defect information. The base classification algorithms mainly include LR algorithm, SVR algorithm, RF algorithm and GBDT algorithm. Then, the above-mentioned base classification algorithm is integrated through Stacking to obtain a Stacking integrated classification algorithm, and the Stacking integrated classification algorithm is used to predict the category of defect information. The evaluation results of the base classification algorithm and the Stacking integrated classification algorithm on the classification and prediction of defect information are shown in Table II.

TABLE II. EVALUATION RESULTS TABLE

method	precision	recall	F-measure
LR	0.742	0.749	0.735
SVR	0.727	0.672	0.794
RF	0.867	0.872	0.863
GDBT	0.868	0.891	0.846
Stacking	**0.923**	**0.912**	**0.935**

It can be found that for a single learner, the performance of the random forest and GDBT algorithm on F1 value, precison and recall is better than SVR and LR, showing the advantages of integrated learning in classification prediction. Compared with the detection effect of other learners, the detection results of the SVR algorithm on accuracy are lower, the main reason may be that when processing a large amount of Chinese and English defect information, the parameter setting of the SVR model is problematic. Therefore, a lot of practice and exploration are needed in the future to make the SVR model's effectiveness in predicting defect information improve. For the Stacking integration algorithm, the F1 value, precison, and recall are better than the base classification algorithm, and it has a good ability to predict the category of defect information.

For the Stacking integration model used above, the most important parameters affecting the structure of the model are the number of individual learners, the maximum depth of GBDT, and the maximum depth of RF.

In ensemble learning, the number of individual learners has a significant impact on the performance of the ensemble model. When the number of samples in the training set is very large, multiple individual learners can be selected to learn the training samples, so that the integrated model can have better results in prediction. Where the number of individual learners is the number of K-fold cross-validation. We use 3-fold, 4-fold, and 5-fold individual learners to perform prediction experiments on defect code information. During the experiment, the individual learner was iterated 300 times, and the accuracy of the Stacking model was calculated with an interval of 50 times. The accuracy of the Stacking integration model under different iterations is shown in Fig.6.

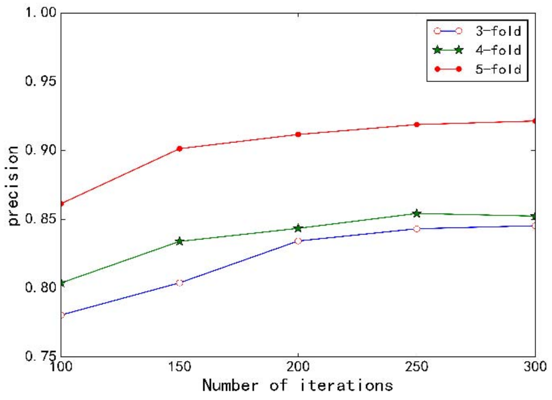


Fig. 6. Accuracy of Stacking model under different iterations

It can be found that with a certain number of iterations, the more individual learners, the higher the accuracy of the Stacking integration model. With a certain number of individual learners, the number of iterations and accuracy is a gradually increasing relationship, and eventually stabilizes.

For the two basic models of GBDT and RF, if the two basic models are debugged to the best performance first, and then integrated through the Stacking algorithm, the resulting Stacking integration model is not necessarily the optimal model. Therefore, the parameters of these two models need to be combined and debugged together to obtain the optimal Stacking integration model. For the GBDT base model, the main influence parameter is the maximum depth depth_G, and for the RF base model, the main influence parameter is the maximum depth depth_R. The accuracy of GBDT's maximum depth_G selection and RF maximum depth depth_R selection in different combinations is shown in Table III.

TABLE III. ACCURACY OF DIFFERENT PARAMETER COMBINATIONS

	depth_R=10	depth_R=15	depth_R=20	depth_R=25
depth_G=5	0.8614	0.9246	0.9142	0.9028
depth_G=6	0.8687	0.9265	0.9178	0.9135
depth_G=7	0.8543	0.9215	0.9035	0.9146

It can be seen that the optimal parameter combination of GBDT and RF is [depth_G = 6, depth_R = 15]. It is better that the maximum depth of GBDT and the maximum depth of RF are not larger. The accuracy will decrease before reaching a certain threshold.

According to the above experiments, it can be found that the Stacking algorithm can give full play to the advantages of each basic model, and can effectively make up for the shortcomings of a single basic model in some aspects. As a result, the Stacking integration model has a good effect in all aspects, and the final prediction result is as close to the real situation as possible.

IV. CONCLUSION AND FUTURE WORK

In this work, we propose a stacking model-based developer user portrait model to classify developers' programming levels. First, the developer's defects are processed, the redundant and irrelevant information in the defect information is removed, and then the improved K-means algorithm is used to classify the defect information and give weight to the category information to divide the developer user group. The feature matrix is extracted from the typed defect information, then the feature matrix is input into the Stacking classification model for training, and finally the defect information without categories is used to evaluate the Stacking classification model. The experimental results show that the method is superior to the existing methods in terms of classification rate, recall, precision and F-measures. However, the developers should be portrayed from multiple perspectives to make the portrait information richer and the guidance of the portrait stronger [14]. This is our future work to extend the proposed method to improve results.

ACKNOWLEDGMENT

This work was supported by National Key R&D Program of China under Grant 2018YFC0407901, the Fundamental Research Funds for the Central Universities under Grant B200202177, the Natural Science Foundation of China under Grant 61702160, the Natural Science Foundation of Jiangsu Province under Grant BK20170892.

REFERENCES

[1] ZHAO T, ZHANG Y, ZHANG D X. Analysis of big data application technology and prospect of intelligent distribution network[J]. Power System Technology, 2014, 38(12): 3305-3312.

[2] CAI G Y,XIA B B. Sentiment prediction of graphic and text fusion media based on convolutional neural network [J]. Computer Application,2016,36(02):428-431+477.

[3] Torres-Valencia, Cristian, álvarez-López, Mauricio, Orozco-Gutiérrez, álvaro. SVM-based feature selection methods for emotion recognition from multimodal data[J]. Journal on Multimodal User Interfaces,11(1):9-23,2017.

[4] Kuzma M,Andrejková,Gabriela.Predicting user's preferences using neural networks and psychology models[J].Applied Intelligence,2016,44(3):526-538.

[5] ZENG J. Research on radvizvisualization technology measurement model[D]. Beijing: Beijing Jiaotong University, 2011.

[6] Webb G I, Pazzani M J, Billsus D. Machine Learning for User Modeling[J]. User Modeling and User-Adapted Interaction, 2001, 11(1-2): 19-29.

[7] Degemmis M, Lops P, Semeraro G, et al. Extraction of User Profiles by Discovering Preferences through Machine Learning[M]. Springer Berlin Heidelberg, 2003.

[8] ZHEN B X. Promotion of mobile game products based on user portrait and signaling mining technology [C]. Guangdong Communications Society,2010:133-136.

[9] SONG M Q,CHEN Y,ZHANG R. Review of User Portrait Studies [J]. Information Science,2019,37(04):171-177.

[10] X. Zhou and Q. Jin, "A heuristic approach to discovering user correlations from organized social stream data," Multim. Tools Appl., vol. 76, no. 9, pp. 11487–11507, 2017.

[11] W. Liang, X. Zhou, S. Huang, C. Hu, X. Xu, and Q. Jin, "Modeling of cross-disciplinary collaboration for potential field discovery and recommendation based on scholarly big data," Future Gener. Comput. Syst., vol. 87, pp. 591–600, 2018.

[12] X. Zhou, N. Y. Yen, Q. Jin, and T. K. Shih, "Enriching user search experience by mining social streams with heuristic stones and associative ripples," Multim. Tools Appl., vol. 63, no. 1, pp. 129–144, 2013.

[13] X. Xu, Q. Wu, L. Qi, W. Dou, S.-B. Tsai, and M. Z. A. Bhuiyan, "Trust-aware service offloading for video surveillance in edge computing enabled internet of vehicles," IEEE Trans-actions on Intelligent Transportation Systems, 2020, DOI: 10.1109/TITS.2020.2995622.

[14] X. Xu, D. Zhu, X. Yang, S. Wang, L. Qi, and W. Dou, "Concurrent practical byzantine fault tolerance for integration of blockchain and supply chain," ACM Transactions on Internet Technology (TOIT), DOI: 10.1145/3395331.