DOI: 10.24412/2413-2527-2023-335-51-59

Software Defect Prediction System Based on Well-Tuned Random Forest Technique

F. H. Khorsheed, N. J. Ibrahim
University of Diyala
Baqubah, Iraq
farah_hatam@uodiyala.edu.iq, nebras.jalel@uodiyala.edu.iq

Q. Saihood
Ashur University College
Baghdad, Iraq
qusaysaihood@au.edu.iq

Abstract. **Software quality is the main criterion for increasing user demand for software. Therefore, software companies seek to ensure software quality by predicting software defects in the software testing phase. Having an intelligent system capable of predicting software defects helps greatly in reducing time and effort consumption. Despite the great trend to develop software defect prediction systems based on Machine Learning techniques in last few years, the accuracy of these systems is still a major challenge.**

Therefore, in this study, a software defect prediction system based on three stages is presented to improve the prediction accuracy. First stage, data pre-processing is performed, which includes (data cleaning, data balance, data normalization, and feature selection). Second stage the hyperparameters of ML are tuned using Grid Search technique. Finally, a well-tuned ML technique is implemented to predict software defects.

Performance experiments were carried out on the JM1 dataset where the proposed system achieved promising results in predicting software defects. Among ML techniques used, a well-tuned RF technique outperformed the rest of the used ML techniques, in addition to the techniques mentioned in previous works, with an accuracy of 88,26 %. This study proves that the selection of important features and efficient hyperparameter tuning of ML techniques significantly improve the accuracy of software defect prediction.

Keywords: **machine learning, Random Forest, software defects, feature selection, prediction.**

Introduction

The world is witnessing a great development in computer technology, smart phones, and the Internet of Things. This development leads to the development of software and an increase in demand for it because software is a means of connecting humans and electronic devices. Today people live in the age of software where there are millions of programs developed daily [1, 2]. The most important phase of Software Development Life Cycle (SDLC) is the testing phase in which future software defects are checked before exporting the software for use [3]. A software defect is a software error that leads to incorrect results that may be caused by errors in the source code. The occurrence of software defects in software in the future negatively affects the quality and reliability of the software [4]. Also, the process of repairing them is costly and may withdraw all versions from the market, and thus cause costly losses for the company producing the program. These flaws may also cause serious security holes that hackers can exploit as a way to exploit your request and sometimes steal important information or money [5].

The biggest challenge for software companies and programmers is to predict future software defects in real-life scenarios [6]. Therefore, early prediction of software defects is seen as the most important research field since the beginning of the software era [4]. To ensure software quality many quality attributes and metrics have been developed with many quality assurance techniques, but still the important question of how to ensure that software will possess good quality is an open issue. Finding out which units are prone to failure is closely related to the quality of the program. Defects prediction involves early detection of those dangerous modules of the program that are prone to errors and impair quality [7]. Detecting defective modules at the early stage is vital as the cost of rectification increases in later stages of the development life cycle. Software metrics extracted from historical software data are used to predict defective units [8, 9]. Therefore, defective modules must be detected and removed 100% to ensure high quality software.

In recent years, Artificial Intelligence (AI) methods have shown a great ability to predict hidden patterns in huge data. There have been many studies that dealt with the use of Machine Learning (ML) and Deep Learning (DL) techniques in predicting software defects in the software testing phase. Many ML and DL techniques such as (Logistic Regression (LR) technique, Decision Tree (DT) technique, Support Vector Machine (SVM) technique, Random Forest (RF) technique, K-Nearest Neighbor (KNN) technique, Multi-Layer Perceptron (MLP) technique, Artificial Neural Network (ANN), and Convolutional Neural Network (CNN)) were built to predict software defects [10]. Although there are many studies dealing with the use of AI methods to predict software defects, there are several shortcomings. One of the most important shortcomings is the accuracy of these methods. Until now, the accuracy of predicting software defects using ML and DL techniques is still a great challenge [7, 11–13]. In addition to identifying the features that affect the process of predicting software defects. Efficient selection of important features greatly influences the prediction of software defects with high accuracy [7].

Therefore, this study aims to build a particular system to predict software defects effectively. The proposed system consists of three stages: In the first stage, data pre-processing is implemented, which includes filling in missing values, balancing data categories, data normalization, and selection of important features affecting the prediction process. In the second stage, Grid search technique is applied to effectively adjust the hyperparameters of ML techniques to improve their performance. Finally, six different ML techniques (RF, DT, SVM, LR, KNN, and MLP) are built for predicting software defects. Performance experiments for this study were performed on the JM1 dataset which is one of 12 online open-source software defect datasets provided by the NASA Software Engineering Repository [14]. The performance of the proposed system was evalu-

ated using common ML evaluation metrics (Accuracy, Precision, Sensitivity, Specificity, F Score). In addition to calculating the confusion matrix that shows actual and predictor values.

The rest of the article is organized as follows: In the second section, related work is summarized. In the third section, the general approach of the proposed system is summarized, which includes all steps of data pre-processing and hyperparameter tuning, in addition to mentioning ML techniques used briefly. In the fourth section, evaluation matrices are mentioned to evaluate the performance of ML techniques used in this work. In the fifth section, the results are mentioned and discussed, in addition to comparing them with the results of related work. Finally, the conclusion and future work summarizes the contributions of the work and outlines future work in Section sixth.

Related works

Recent years have seen a significant increase in the use of AI methods in predicting software defects. Various ML and DL techniques have been used to explore hidden patterns in the software's source code. Early software defects prediction helps to ensure software quality. In this section, several studies using ML and DL techniques for software defect prediction are summarized.

P. D. Singh and A. Chug [15] analyzed 7 datasets from NASA Promise dataset repository using DT, Naive Bayes (NB), Linear classifier (LC), ANN and Particle Swarm Optimization (PSO) for predicting software defects. 10-fold cross validation was used to evaluate the techniques used. The LC technique achieved a better performance than the rest of the techniques used when it was used on each of the datasets (JM1, KC2, PC1, and AT). While the DT technique outperformed the rest of the techniques with two data sets (CM1 and KC1).

A. Iqbal, et al. [7] introduced a comprehensive study of a variety of ML techniques to predict software defects so that researchers can later use this study as a basic plan in their future research. NB, MLP, Radial Basis Function (RBF), SVM, KNN, kStar (K*), One Rule (OneR), PART, DT, RF, and ensemble methods were applied to 12 datasets from NASA with and without using feature selection techniques. The performance of the techniques used was evaluated using the commonly used performance metrics Accuracy, Precision, Recall, F Score, MCC, and ROC. The results of this study show that the prediction accuracy of the used techniques is still not good enough, in addition to that the used datasets suffer from a large variation in the size of the categories.

A. Iqbal and S. Aftab [13] tried to solve the data imbalance using oversampling technique. In addition to suggesting use feature selection techniques and Multi-Layer Perceptron (MLP) technique for software defect prediction. Performance experiments were conducted on 12 datasets provided by NASA and in two different directions: with and without the over-sampling technique. In addition, the performance of the proposed system was compared with a set of common ML classifiers using common performance measures, as it was noted that the proposed system with the over-sampling technique performs well compared to other ML classifiers.

B. Khan, et al. [11] compared the performance of seven ML techniques (MLP, SVM, J48, RBF, RF, Hidden Markov Model (HMM), Dependency Decision Tree (CDT), KNN, Average Dependency Estimator (A1DE), and NB) based on seven different datasets provided by NASA. Performance of ML techniques was evaluated using various measures such as MAE, RAE, and RMSE. and RRSE, recall, and accuracy. The performance of ML techniques is evaluated using different measures (Accuracy, Recall, Relative Absolute Error (RAE), Root Mean Squared Error (RMSE), Root Relative Squared Error (RRSE), and Mean Absolute Error (MAE)). Among the seven used ML techniques RF technique has the highest accuracy rate of 88.32%. The outputs of this research can be used as a reference point for new research.

R. Vats and A. Kumar [16] explored several supervised and unsupervised ML techniques to determine the best method for predicting software defects. Performance experiments were conducted on 9 datasets from NASA, where experiments showed that supervised ML techniques are more suitable than unsupervised ML techniques for predicting software defects.

R. Shrimankar, et al. [17] proposed to use ensemble methods (XGBoost, AdaBoost, and Gradient Boost) as well as base ML methods (LR, MLP, NB, SVM, RF, DT, and KNN) to analyze the efficiency of ML techniques in software defect prediction. Performance experiments have been conducted on 12 datasets provided by NASA, where XGBoost method slightly outperformed other used methods on some of used datasets.

Previous work indicates that achieving high accuracy in predicting defects is still a major challenge, especially when using the JM1 dataset as shown in table 1. Also, there is no specific ML technique that significantly outperforms other techniques in predicting software defects, as well as the importance of identifying a feature selection technique that is appropriate with the used ML technique. In addition, most of the datasets with software defects suffer from a large disparity between categories.

Methodology

The proposed general framework includes the following stages: In the first stage, data pre-processing is well implemented, which includes cleaning the data by making sure that the data is free of any inconsistencies, balancing classes within the data set, normalizing the data, and determining the importance of each feature using DT technique as a feature selection technique. In the second stage, using the Grid search technique to effectively adjust the hyper-parameters of the ML technique used to improve its performance. In the third stage, apply six different ML techniques (RF, DT, SVM, LR, KNN, and MLP) to predict software defects. Finally, the use of common performance metrics (Accuracy, Precision, Sensitivity, Specificity, F Score, ROC curve, and Confusion matrix) to evaluate the ML technologies used as shown in Figure 1.

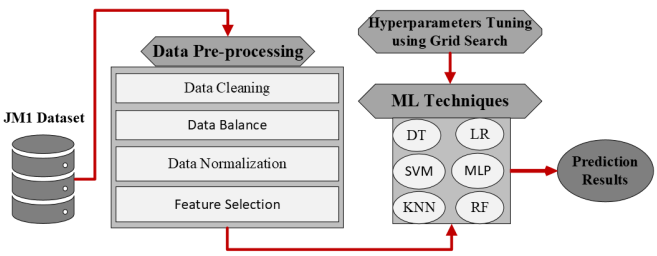

Fig. 1. The Proposed General Framework for Software Defect Prediction

Table 1

Summary of Related Work

Study	Year	Dataset	FS Techniques	Data Balance	ML techniques	JM1 ac., %
P. D. Singh and A. Chug [15]	2017	CM1, JM1, KC1, KC2, PC1, AT, KC1 LC	No	No	DT, NB LC, ANN, PSO	80,64
A. Iqbal, et al. [7]	2019	CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, PC5	Yes	No	NB, MLP, RBF, SVM, KNN, K*, OneR, PART, DT, and RF	80,61
A. Iqbal and S. Aftab [13]	2020	CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, PC5	Yes	Yes	NB, MLP, RBF, SVM, KNN, K*, OneR, PART, DT, RF, MLP-FS, and MLP-FS-ROS	80,44
B. Khan et al. [11]	2021	CM1, JM1, KC3, MC1, MC2, PC1, PC2, PC3	No	No	MLP, SVM, J48, RBF, RF, HMM, CDT, KNN, A1DE, NB	82,02
R. Vats and A. Kumar [16]	2021	CM1, JM1, KC3, MC1, MC2, PC1, PC2, PC3, PC4	No	No	Bagging, AdaBoost, RF, K-Mean, and K-harmonic Mean (KHM)	88,00
R. Shrimankar, et al. [17]	2022	CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, PC5	No	No	XGBoost, AdaBoost, Gradient Boost, LR, MLP, NB, SVM, RF, DT, KNN	80,00

JM1 DATASET

There are 12 online open-source software defect datasets provided by the NASA promise software engineering repository. In this study performance experiments were performed on one of the toughest datasets provided by NASA known as JM1 [18]. The JM1 dataset consists of 10 885 samples and 21 features in addition to the target category (Defect or Not Defect). The JM1 dataset was developed using the C programming language and is a real-time predictive terrestrial system that uses mimicries to generate predictions. Table 2 describes the details of the JM1 dataset.

Table 2

Description of JM1 dataset features

No.	Feature	Type	Description
1	loc	numeric	Halstead line count of code
2	v(g)	numeric	Halstead's «cyclomatic complexity»
3	ev(g)	numeric	Halstead's «essential complexity»
4	iv(g)	numeric	Halstead's «design complexity»
5	n	numeric	Halstead's total operators + operands
6	v	numeric	«volume»
7	l	numeric	unique «program length»
8	d	numeric	unique «difficulty»
9	i	numeric	total «intelligence»
10	e	numeric	total «effort»
11	b	numeric	module
12	t	numeric	time estimator
13	lOCode	numeric	line count
14	lOComment	numeric	count of lines of comments
15	lOBlank	numeric	count of blank lines
16	lOCodeAndComment	numeric	lines of comments + line count of code
17	uniq_Op	numeric	operators
18	uniq_Opnd	numeric	operands
19	total_Op	numeric	operators
20	total_Opnd	numeric	operands 0,875 branch-Count: numeric % of the flow graph
21	branchCount	numeric	the flow graph
22	defects	{false,true}	has/has not one or more reported defects

DATA PRE-PROCESSING

The performance of ML techniques depends mainly on the quality of the data provided to it. Therefore, data pre-processing is an essential and important stage in ML and data mining processes to ensure data quality. Oftentimes, real world data has many issues such as data inconsistency, noise, may contain missing values, or may not be suitable for ML techniques [19]. If ML techniques are applied directly to this raw data, the performance of the model will be negatively affected, and it may be learned incorrectly [20]. Therefore, data pre-processing is carried out to prepare the data well by cleaning the data and filling in the missing values, as well as transforming the data to a form appropriate with the used ML techniques [21]. Data pre-processing in this work includes four stages: data cleaning, data balancing, data normalization, and feature selection.

DATA CLEANING

The first stage of data pre-processing is the data cleaning process in which the data is addressed from noise, inconsistent data, redundant data, and missing values [19]. The JM1 dataset contains some missing values for some features (uniq_Op, uniq_Opnd, total_Op, total_Opnd, BranchCount) which are addressed by filling them by calculating the mean value of the column to which the missing value belongs.

Data balancing

The JM1 dataset contains 10 885 samples, of which 8 779 are of the non-defect category and 2 106 are of the defect category, which indicates a significant deviation between the two data categories. To address the problem of categories imbalance in the JM1 dataset, The Synthetic Minority Over-Sampling Technique (SMOTE), a technique used to increase small category size [22], was used. The size of the small category was increased by generating additional samples using the SMOTE technique to be both categories equal in size.

Data normalization

The dataset usually contains a discrepancy in the range of numeric data between the different columns. When some columns have a very high range, while others may have a very low range, this can negatively affect the performance of the ML technique [23]. Therefore, data normalization is used to normalize the data to be between a uniform range while maintaining the differences in the ranges of values and not losing information [24]. In this work, the Min-max Scaler method has been applied to numeric features to convert them to range between 0 and 1 by following the equation next [25]. The scale of data for a few features may be essentially distinctive from those of others, which may hurt the execution of our models. It is particularly the case with algorithms that depend on a degree of separation, such as Neural Networks and KNN. It is additionally accommodating for optimizing machine learning processes like gradient descent and empowers convergence to happen faster, and it can offer assistance to improve the execution and speed of the execution of algorithms. Since the data is as of now scaled-down, complex calculations basically required to optimize algorithms are quicker. in expansion to that, it can moreover be accommodating when comparing diverse datasets or models in terms of their performances [22].

$$D = \frac{(X - \min(X))}{(max(X) - min(X))},$$

where D is the normalized value, X is the original value of a feature, min (X) is smallest value of a feature, max (X) is biggest value of a feature.

Feature selection

Feature selection is the process of selecting a set of input data features that have a significant impact on the target. Determining the important and appropriate Features able to improve the performance of the ML technique and reduce the time it takes to train [26]. The DT technique has a feature importance property that can be used to calculate feature importance [27]. The importance of the JM1 dataset Features was calculated using DT technique and then features with less importance were removed.

ML techniques

ML is part of the science of AI, which are statistical methods that enable a computer or any device to build its own concept based on the data provided to it in the training phase. There are several types of ML depending on the type of learning: supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning [28]. In this work, 6 supervised ML techniques were used.

DT

It is one of the most popular non-parametric supervised ML techniques that can be used for both classification and regression tasks. The structure of DT technique is like an upside-down tree where the first node represents the root and then it is split into inner nodes. Each inner node refers to the feature, the branches refer to the rules and the leaves refer to the result of the techniques [28].

RF

RF is one of the most powerful and popular ML techniques that can be used for both classification and regression tasks. RF technique is one of the ensemble methods based on bagging method which aggregates a set of decision trees. Each decision tree within the forest is trained on a subset of the data, and thus the final decision is made using tree-majority voting. RF is mostly used to solve some problems of DT technique and some individual ML techniques [7, 11, 28]:

- in most cases, it gives better classification accuracy than DT technique;
- provides an effective way to deal with lost data;
- solving the problem of overfitting in DT technique;
- more stable than DT technique.

Random forest could be a Supervised Machine Learning Calculation that's utilized broadly in Classification and Regression issues. It builds decision trees on distinctive tests and takes their larger part vote for classification and normal in case of regression [25]. A random forest may be a machine learning technique that's won't to solve regression and classification problems. It utilizes ensemble learning, that is a technique that mixes several classifiers to produce solutions to complicated problems.

Features of a Random Forest algorithm:

- it's more accurate than the decision tree algorithm;
- it provides an effective way of handling missing data;
- it can produce a reasonable prediction without hyper-parameter tuning;
- it solves the issue of overfitting in decision trees;
- in every random forest tree, a subset of features is selected randomly at the node's splitting point.

SVM

SVM is one of the most important ML techniques suitable for both small and complex data. SVM can be used to solve both classification and regression problems, but it is usually used to solve classification problems. SVM technique finds the best separation line between data categories by maximizing margins and determining the best separation of data [19, 28].

LR

LR is one of the simplest classification techniques for ML based on statistical background. LR is a statistical method for analyzing a data set in which there are one or more independent variables that determine the result. Every user who uses LR technique needs to know log probabilities, the key concept behind a LR technique. LR estimates the probability that an event, such as a defect or a non-defect, will occur based on a given data set of independent variables. Since the outcome is a probability, the dependent variable is constrained between 0 and 1 [29].

KNN

KNN is a non-parametric ML technique that can be used for classification and regression tasks [19]. The way the KNN technique works is very simple and effective to determine if a new sample is a software defect or not. The distance between the sample to be classified and all points of the training data set of software defects is measured using one of the distance scales such as the Euclidean distance, Manhattan distance, and Minkowski distance. Next, the number *K* representing the number of nearest neighbors is determined, which determines the nature of the new sample (defect or not) according to the nature of its nearest neighbors [29]. In this work, the distance between the samples of the training set and the samples of the test set was measured using a Minkowski distance.

MLP

MLP is a fully connected feedforward ANN that can be used for both classification and regression tasks. The MLP network consists of three layers, an input layer used to enter the training data and a layer or hidden layers used to process the input data. And the last layer is the output layer, which is used to obtain the required outputs [19].

TUNING HYPERPARAMETERS OF ML TECHNIQUES

ML techniques have hyperparameters that help fine-tune their performance. Fine tuning of hyperparameters of ML techniques helps to stabilize performance, improve accuracy, and reduce model complexity. Usually, hyperparameters of ML techniques are manually tune, which is cumbersome and time consuming. Therefore, the Grid Search technique was used to tune the hyperparameters of the ML techniques used in this work. Grid search technique is fed with the hyperparameters and their potential value, and then the 10-fold validation method is used to train the technique with all the possible values of the hyperparameters. Finally, the best values of the hyperparameters with the best performance of the ML techniques are determined.

PERFORMANCE MEASUREMENT

The performance of used ML techniques was evaluated by calculating the confusion matrix in addition to using six common performance evaluation criteria (Accuracy, Precision, Sensitivity, Specificity, F1 Score, and ROC Curve). The confusion matrix contains the actual values and predicted values from which the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) are calculated. The «False» (not defect) class is represented as negative '0', while the «True» (defect) class is represented as positive '1'.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}},$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}},$$

$$\text{F1 Score} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}}.$$

RESULTS AND DISCUSSION

The proposed RF technique, as well as other ML techniques used in this study, were built using the scikit-learn library. Performance experiments were conducted on the JM1 dataset, where the dataset was separated 80 % for training ML techniques and 20 % for testing them.

The proposed RF technique achieved the best performance with an accuracy of 88,26%, along with a precision of 87,76 %, Sensitivity of 88,47 %, Specificity of 88,06 %, and F1-Score of 88,11 % compared to the other techniques used as shown in Table 3. Then, KNN technique ranked second with a competitive performance with an accuracy of 85,76 %, along with a precision of 81,60 %, Sensitivity of 91,71 %, Specificity of 80,0 %, and F1-Score of 86,36 %. Also, the DT technique achieved a good performance with an accuracy of 82,17 %, along with along with a precision of 81,54 %, Sensitivity of 82,39 %, Specificity of 81,96 %, and F1-Score of 81,97 %, while the performance of the MLP, LR, and SVM techniques was modest with an accuracy of 70,50 %, 66,88 %, and 65,51 %, respectively.

Table 3

Performance comparison of ML techniques used in this study

Technique	**Accuracy, %**	**Precision, %**	**Sensitivity, %**	**Specificity, %**	**F1 Score, %**
RF	88,26	87,76	88,47	88,06	88,11
DT	82,17	81,54	82,39	81,96	81,97
SVM	65,51	68,14	56,10	74,62	61,54
KNN	85,76	81,60	91,71	80,00	86,36
LR	66,88	68,10	61,43	72,15	64,59
MLP	70,50	70,55	68,67	72,26	69,60

The performance of ML techniques was also compared by drawing the ROC curve as shown in Figure 2. In addition, a confusion matrix was calculated for each of ML techniques used to show the correct and incorrect predictions for both classes as in Figures 3.

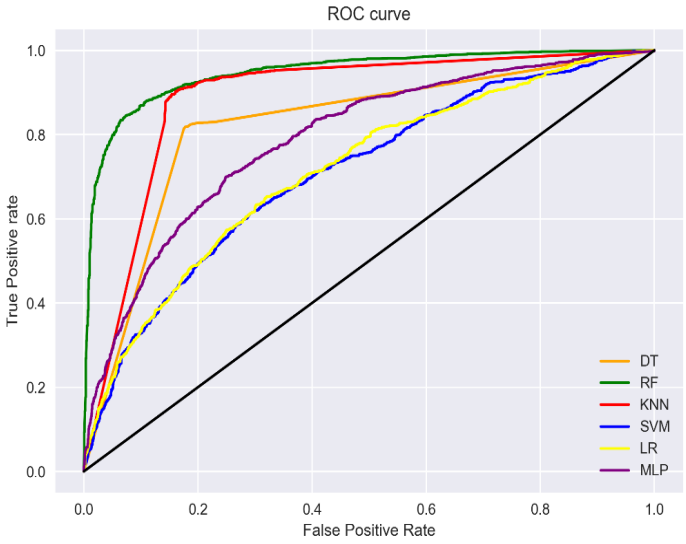

Fig. 2. ROC Curve of used ML techniques

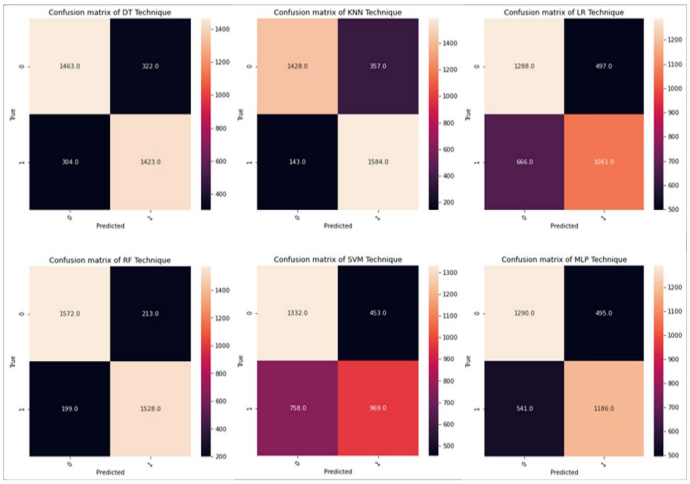

Fig. 3. Confusion matrix of used ML techniques

The proposed system also proved superior when compared with the achieved results of related work as shown in Table 4. The superiority of the RF technique over the RF techniques used in studies [7, 11, 13, 16] as well as over all other techniques used in previous works. Also, the superiority of KNN technique used in this study over the KNN techniques used in the studies [7, 11, 13, 17]. Also, the DT technique used in this study achieved remarkable superiority over the DT techniques used in previous works [7, 15, 17]. The proposed system in this study achieved promising results in predicting software defects, superior to previous works. This study proved that preparing the data well and selecting the effective features helps to improve the prediction accuracy. Also, choosing the appropriate ML technique for this task, in addition to fine tuning its hyperparameters, greatly improves the prediction accuracy.

Table 4

Comparison of the proposed work with related work

Study	**Accuracy, %**
[15]	80,64
[7]	80,61
[13]	80,44
[11]	82,02
[16]	88,00
[17]	80,00
Our DT	82,17
Our KNN	85,76
Our RF	88,26

Conclusion and Future Works

Predicting software defects at the software testing stage is very important to ensure the quality of software before it is offered to users. Therefore, this study aims to build an intelligent system to automatically predict software defects in the software testing phase based on ML techniques. The proposed system consists of three basic stages: data pre-processing, hyper-parameter tuning, and application of ML techniques. In the data pre-processing stage, the data was cleaned, balanced, and normalized, in addition to using the DT technique to select the important features. In the second stage, the Grid search technique is applied to tune the hyperparameters of the ML techniques. In the final stage, ML techniques are applied, and their performance is compared using various scales. Where the RF technique achieved promising results with an accuracy of 88,26 %, superior to other techniques used. RF technique has been shown to be highly effective in predicting software defects when used with an appropriate FS technique. This system was tested only on JM1 dataset, so there is a possibility to test it on other datasets provided by NASA in future works. Individual classifiers can also be combined using ensemble methods to improve performance.

References

1. Hassan F., Farhan S., Fahiem M. A., Tauseef H. A Review on Machine Learning Techniques for Software Defect Prediction, *Technical Journal*, 2018, Vol. 23, No. 02, Pp. 63–71.

2. Ayon S. I. Neural Network Based Software Defect Prediction Using Genetic Algorithm and Particle Swarm Optimization, *Proceedings of the First International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, May 03–05, 2019*. Institute of Electrical and Electronics Engineers, 2019, 4 p. DOI: 10.1109/ICASERT.2019.8934642.

3. Jayanthi R., Florence L. Software Defect Prediction Techniques Using Metrics Based on Neural Network Classifier, *Cluster Computing*, 2019, Vol. 22, Suppl. Is. 1, Pp. 77–88. DOI: 10.1007/s10586-018-1730-1.

4. Catal C., Diri B. A Systematic Review of Software Fault Prediction Studies, *Expert Systems with Applications*, 2009, Vol. 36, Is. 4, Pp. 7346–7354. DOI: 10.1016/j.eswa.2008.10.027.

5. Rashid M., Kaur L. Finding Bugs in Android Application Using Genetic Algorithm and Apriori Algorithm, *Indian Journal of Science and Technology*, 2016, Vol. 9, Is. 23, 5 p. DOI: 10.17485/ijst/2016/v9i23/94572.

6. Zhou Y., Shan C., Sun S., et al. Software Defect Prediction Model Based On KPCA-SVM, *Proceedings of the 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Leicester, United Kingdom, August 19–23, 2019*. Institute of Electrical and Electronics Engineers, 2019, Pp. 1326–1332. DOI: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00244.

7. Iqbal A., Aftab S., Ullah I., et al. A Feature Selection Based Ensemble Classification Framework for Software Defect Prediction, *International Journal of Modern Education and Computer Science*, 2019, Vol. 11, No. 9, Pp. 54–64. DOI: 10.5815/ijmecs.2019.09.06.

8. Aftab S., Ahmad M., Hameed N., et al. Rainfall Prediction in Lahore City Using Data Mining Techniques, *International Journal of Advanced Computer Science and Applications*, 2018, Vol. 9, Is. 4, Pp. 254–260. DOI: 10.14569/IJACSA.2018.090439.

9. Herzig K., Just S., Rau A., Zeller A. Predicting Defects Using Change Genealogies, *Proceedings of the 24th International Symposium on Software Reliability Engineering (ISSRE), Pasadena, CA, USA, November 04–07, 2013*. Institute of Electrical and Electronics Engineers, 2013, Pp. 118–127. DOI: 10.1109/ISSRE.2013.6698911.

10. Pachouly J., Ahirrao S., Kotecha K., et al. A Systematic Literature Review on Software Defect Prediction Using Artificial Intelligence: Datasets, Data Validation Methods, Approaches, and Tools, *Engineering Applications of Artificial Intelligence*, 2022, Vol. 111, Art. No. 104773, 33 p. DOI: 10.1016/j.engappai.2022.104773.

11. Khan B., Naseem R., Shah M. A., et al. Software Defect Prediction for Healthcare Big Data: An Empirical Evaluation of Machine Learning Techniques, *Journal of Healthcare Engineering*, 2021, Vol. 2021, Art. No. 8899263, 16 p. DOI: 10.1155/2021/8899263.

12. Elsabagh M. A., Farhan M. S., Gafar M. G. Cross-Projects Software Defect Prediction Using Spotted Hyena Optimizer Algorithm, *SN Applied Sciences*, 2020, Vol. 2, Is. 4, Art. No. 538, 15 p. DOI: 10.1007/s42452-020-2320-4.

13. Iqbal A., Aftab S. A Classification Framework for Software Defect Prediction Using Multi-Filter Feature Selection Technique and MLP, *International Journal of Modern Education and Computer Science*, 2020, Vol. 12, No. 1, Pp. 18–25. DOI: 10.5815/ijmecs.2020.01.03.

14. Sayyad Shirabad J., Menzies T. J. The PROMISE Repository of Software Engineering Databases, *School of Information Technology and Engineering, University of Ottawa*. Available at: http://promise.site.uottawa.ca/SERepository (accessed 15 Jul 2023).

15. Singh P. D., Chug A. Software Defect Prediction Analysis Using Machine Learning Algorithms, *Proceedings of the 7th International Conference on Cloud Computing, Data Science and Engineering — Confluence, Noida, India, January 12–13, 2017*. Institute of Electrical and Electronics Engineers, 2017, Pp. 775–781. DOI: 10.1109/CONFLUENCE.2017.7943255.

16. Vats R., Kumar A. Software Defects Prediction Using Supervised and Unsupervised Machine Learning Approaches: A Comparative Performance Analysis, *International Journal of Research in Engineering, Technology and Science*, 2021, Vol. 13, Is. 8, 14 p.

17. Shrimankar R., Kuanr M., Piri J., Panda N. Software Defect Prediction: A Comparative Analysis of Machine Learning Techniques, *Proceedings of the 2022 International Conference on Machine Learning, Computer Systems and Security (MLCSS), Bhubaneswar, India, August 05–06, 2022*. Institute of Electrical and Electronics Engineers, 2022, Pp. 38–47. DOI: 10.1109/MLCSS57186.2022.00016.

18. Can H., Jianchun X., Ruide Z., et al. A New Model for Software Defect Prediction Using Particle Swarm Optimization and Support Vector Machine, *Proceedings of the 25th Chinese Control and Decision Conference (CCDC), Guiyang, China, May 25–27, 2013*. Institute of Electrical and Electronics Engineers, 2013, Pp. 4106–4110. DOI: 10.1109/CCDC.2013.6561670.

19. Saihood Q., Sonuç E. The Efficiency of Classification Techniques in Predicting Anemia Among Children: A Comparative Study, *Emerging Technology Trends in Internet of Things and Computing (TIOTC 2021): Revised Selected Papers of the First International Conference, Erbil, Iraq, June 06–08, 2021*. Cham, Springer Nature, 2022, Pp. 167–181. DOI: 10.1007/978-3-030-97255-4_12.

20. Naz H., Ahuja S., Nijhawan R., Ahuja N. J. Impact of Data Pre-Processing in Information Retrieval for Data Analytics. In: *Kumar A., et al. (eds.) Machine Intelligence, Big Data Analytics, and IoT in Image Processing: Practical Applications*. Beverly (MA), Scrivener Publishing, 2023, Pp. 197–224. DOI: 10.1002/9781119865513.ch9.

21. Chu X., Ilyas I. F., Krishnan S., Wang J. Data cleaning: Overview and emerging challenges, *SIGMOD '16: Proceedings of the International Conference on Management of Data, San Francisco, CA, USA, June 26–July 01, 2016*. New York, Association for Computing Machinery, 2016, Pp. 2201–2206. DOI: 10.1145/2882903.2912574.

22. Chawla N. V., Bowyer K. W., Hall L. O., Kegelmeyer W. P. SMOTE: Synthetic Minority Over-Sampling Technique, *Journal of Artificial Intelligence Research*, 2002, Vol. 16, Pp. 321–357. DOI: 10.1613/jair.953.

23. Sola J., Sevilla J. Importance of input data normalization for the application of neural networks to complex industrial problems, *IEEE Transactions on Nuclear Science*, 1997, Vol. 44, Is. 3, Pp. 1464–1468. DOI: 10.1109/23.589532.

24. Singh D., Singh B. Investigating the Impact of Data Normalization on Classification Performance, *Applied Soft Computing*, 2020, Vol. 97, Part B, Art. No. 105524, 23 p. DOI: 10.1016/j.asoc.2019.105524.

25. Khaire U. M., Dhanalakshmi R. Stability of Feature Selection Algorithm: A Review, *Journal of King Saud University — Computer and Information Sciences*, 2022, Vol. 34, Is. 4, Pp. 1060–1073. DOI: 10.1016/j.jksuci.2019.06.012.

26. Hota H. S., Shrivas A. K. Decision Tree Techniques Applied on NSL-KDD Data and Its Comparison with Various Feature Selection Techniques. In: *Kundu M. K., et al. (eds.) Advanced Computing, Networking and Informatics — Volume 1: Advanced Computing and Informatics Proceedings of the Second International Conference on Advanced Computing, Networking and Informatics (ICACNI-2014), Kolkata, India, June 24–26, 2014*. Cham, Springer International Publishing, 2014, Pp. 205–211. DOI:10.1007/978-3-319-07353-8_24.

27. Aggarwal S. Machine Learning Algorithms, Perspectives, and Real-World Application: Empirical Evidence from United States Trade Data, *International Journal of Science and Research*, 2023, Vol. 12, Is. 3, Pp. 292–313. DOI: 10.21275/SR23305084601.

28. Zhang W., Huang G., Zheng K. Application of Logistic Regression and Machine Learning Methods for Idiopathic Inflammatory Myopathies Malignancy Prediction, *Clinical and Experimental Rheumatology*, 2023, Vol. 41, Is. 2, Pp. 330–339. DOI: 10.55563/clinexprheumatol/8ievtq.

29. Uddin S., Haque I., Lu H., et al. Comparative Performance Analysis of K-Nearest Neighbour (KNN) Algorithm and Its Different Variants for Disease Prediction, *Scientific Reports*, 2022, Vol. 12, Art. No. 6256, 11 p. DOI: 10.1038/s41598-022-10358-x.

DOI: 10.24412/2413-2527-2023-335-51-59

Система прогнозирования дефектов программного обеспечения на основе хорошо отлаженной техники «случайного леса»

Ф. Х. Хоршид, Н. Дж. Ибрагим
Университет Диялы
Баакуба, Ирак
farah_hatam@uodiyala.edu.iq, nebras.jalel@uodiyala.edu.iq

К. Сайхуд
Университетский колледж Ашура
Багдад, Ирак
qusaysaihood@au.edu.iq

***Аннотация*. Качество программного обеспечения является основным критерием для повышения спроса пользователей на программное обеспечение. Поэтому компании, занимающиеся программным обеспечением, стремятся обеспечить качество программного обеспечения путем прогнозирования его дефектов на этапе тестирования. Наличие интеллектуальной системы, способной прогнозировать дефекты программного обеспечения, значительно снижает затраты времени и усилий. Несмотря на широкую тенденцию разработки систем прогнозирования дефектов программного обеспечения на основе техники машинного обучения в последние несколько лет, точность этих систем по-прежнему является серьезной проблемой.**

В данном исследовании для повышения точности прогноза представлена система прогнозирования дефектов программного обеспечения, состоящая из трех этапов. На первом этапе выполняется предварительная обработка данных, которая включает в себя очистку данных, баланс данных, нормализацию данных и выбор признаков. На втором этапе гиперпараметры настраиваются по методике Grid Search. Наконец, хорошо отлаженная техника машинного обучения реализована для предсказания дефектов программного обеспечения.

На базе набора данных JM1 были проведены эксперименты, в ходе которых предлагаемая система дала многообещающие результаты в прогнозировании недостатков программного обеспечения. Среди используемых методов хорошо настроенный метод Random Forest с точностью 88,26 % превзошел остальные используемые методы машинного обучения. Проведенное исследование доказывает, что выбор важных особенностей и эффективная гиперпараметрическая настройка методов машинного обучения значительно улучшают точность прогнозирования дефектов программного обеспечения.

Ключевые слова: **машинное обучение, случайный лес, дефекты программного обеспечения, выбор признаков, прогнозирование.**

Литература

1. A Review on Machine Learning Techniques for Software Defect Prediction / F. Hassan, S. Farhan, M. A. Fahiem, H. Tauseef // Technical Journal. 2018. Vol. 23, No. 02. Pp. 63–71.

2. Ayon, S. I. Neural Network Based Software Defect Prediction Using Genetic Algorithm and Particle Swarm Optimization // Proceedings of the First International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT) (Dhaka, Bangladesh, 03–05 May 2019). — Institute of Electrical and Electronics Engineers, 2019. — 4 p. DOI: 10.1109/ICASERT.2019.8934642.

3. Jayanthi, R. Software Defect Prediction Techniques Using Metrics Based on Neural Network Classifier / R. Jayanthi, L. Florence // Cluster Computing. 2019. Vol. 22, Suppl. Is. 1. Pp. 77–88. DOI: 10.1007/s10586-018-1730-1.

4. Catal, C. A Systematic Review of Software Fault Prediction Studies / C. Catal, B. Diri // Expert Systems with Applications. 2009. Vol. 36, Is. 4. Pp. 7346–7354. DOI: 10.1016/j.eswa.2008.10.027.

5. Rashid, M. Finding Bugs in Android Application Using Genetic Algorithm and Apriori Algorithm / M. Rashid, L. Kaur // Indian Journal of Science and Technology. 2016. Vol. 9, Is. 23. 5 p. DOI: 10.17485/ijst/2016/v9i23/94572.

6. Software Defect Prediction Model Based On KPCA-SVM / Y. Zhou, C. Shan, S. Sun, [et al.] // Proceedings of the 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI) (Leicester, United Kingdom, 19–23 August 2019). — Institute of Electrical and Electronics Engineers, 2019. — Pp. 1326–1332. DOI: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00244.

7. A Feature Selection Based Ensemble Classification Framework for Software Defect Prediction / A. Iqbal, S. Aftab, I. Ullah, [et al.] // International Journal of Modern Education and Computer Science. 2019. Vol. 11, No. 9. Pp. 54–64. DOI: 10.5815/ijmecs.2019.09.06.

8. Rainfall Prediction in Lahore City Using Data Mining Techniques / S. Aftab, M. Ahmad, N. Hameed, [et al.] // International Journal of Advanced Computer Science and Applications. 2018. Vol. 9, Is. 4. Pp. 254–260. DOI: 10.14569/IJACSA.2018.090439.

9. Predicting Defects Using Change Genealogies / K. Herzig, S. Just, A. Rau, A. Zeller // Proceedings of the 24th International Symposium on Software Reliability Engineering (ISSRE) (Pasadena, CA, USA, 04–07 November 2013). — Institute of Electrical and Electronics Engineers, 2013. — Pp. 118–127. DOI: 10.1109/ISSRE.2013.6698911.

10. A Systematic Literature Review on Software Defect Prediction Using Artificial Intelligence: Datasets, Data Validation Methods, Approaches, and Tools / J. Pachouly, S. Ahirrao, K. Kotecha, [et al.] // Engineering Applications of Artificial Intelligence. 2022. Vol. 111. Art. No. 104773. 33 p. DOI: 10.1016/j.engappai.2022.104773.

11. Software Defect Prediction for Healthcare Big Data: An Empirical Evaluation of Machine Learning Techniques / B. Khan, R. Naseem, M. A. Shah, [et al.] // Journal of Healthcare Engineering. 2021. Vol. 2021. Art. No. 8899263. 16 p. DOI: 10.1155/2021/8899263.

12. Elsabagh, M. A. Cross-Projects Software Defect Prediction Using Spotted Hyena Optimizer Algorithm / M. A. Elsabagh, M. S. Farhan, M. G. Gafar // SN Applied Sciences. 2020. Vol. 2, Is. 4. Art. No. 538. 15 p.
DOI: 10.1007/s42452-020-2320-4.

13. Iqbal, A. A Classification Framework for Software Defect Prediction Using Multi-Filter Feature Selection Technique and MLP / A. Iqbal, S. Aftab // International Journal of Modern Education and Computer Science. 2020. Vol. 12, No. 1. Pp. 18–25. DOI: 10.5815/ijmecs.2020.01.03.

14. Sayyad Shirabad, J. The PROMISE Repository of Software Engineering Databases / J. Sayyad Shirabad, T. J. Menzies // School of Information Technology and Engineering, University of Ottawa. Available at: http://promise.site.uottawa.ca/SERepository (дата обращения 15.07.2023).

15. Singh, P. D. Software Defect Prediction Analysis Using Machine Learning Algorithms / P. D. Singh, A. Chug // Proceedings of the 7th International Conference on Cloud Computing, Data Science and Engineering — Confluence (Noida, India, 12–13 January 2017). — Institute of Electrical and Electronics Engineers, 2017. — Pp. 775–781.
DOI: 10.1109/CONFLUENCE.2017.7943255.

16. Vats, R. Software Defects Prediction Using Supervised and Unsupervised Machine Learning Approaches: A Comparative Performance Analysis / R. Vats, A. Kumar // International Journal of Research in Engineering, Technology and Science. 2021. Vol. 13, Is. 8. 14 p.

17. Software Defect Prediction: A Comparative Analysis of Machine Learning Techniques / R. Shrimankar, M. Kuanr, J. Piri, N. Panda // Proceedings of the 2022 International Conference on Machine Learning, Computer Systems and Security (MLCSS) (Bhubaneswar, India, 05–06 August 2022). — Institute of Electrical and Electronics Engineers, 2022. — Pp. 38–47. DOI: 10.1109/MLCSS57186.2022.00016.

18. A New Model for Software Defect Prediction Using Particle Swarm Optimization and Support Vector Machine / H. Can, X. Jianchun, Z. Ruide, [et al.] // Proceedings of the 25th Chinese Control and Decision Conference (CCDC) (Guiyang, China, 25–27 May 2013). — Institute of Electrical and Electronics Engineers, 2013. — Pp. 4106–4110.
DOI: 10.1109/CCDC.2013.6561670.

19. Saihood, Q. The Efficiency of Classification Techniques in Predicting Anemia Among Children: A Comparative Study / Q. Saihood, E. Sonuç // Emerging Technology Trends in Internet of Things and Computing (TIOTC 2021): Revised Selected Papers of the First International Conference (Erbil, Iraq, 06–08 June 2021). — Cham: Springer Nature, 2022. — Pp. 167–181. — (Communications in Computer and Information Science, Vol. 1548).
DOI: 10.1007/978-3-030-97255-4_12.

20. Impact of Data Pre-Processing in Information Retrieval for Data Analytics / H. Naz, S. Ahuja, R. Nijhawan, N. J. Ahuja // Machine Intelligence, Big Data Analytics, and IoT in Image Processing: Practical Applications / A. Kumar, [et al.] (eds.). — Beverly (MA): Scrivener Publishing, 2023. — Pp. 197–224. DOI: 10.1002/9781119865513.ch9.

21. Data cleaning: Overview and emerging challenges / X. Chu, I. F. Ilyas, S. Krishnan, J. Wang // SIGMOD '16: Proceedings of the International Conference on Management of Data (San Francisco, CA, USA, 26 June–01 July 2016). — New York: Association for Computing Machinery, 2016. Pp. 2201–2206. DOI: 10.1145/2882903.2912574.

22. SMOTE: Synthetic Minority Over-Sampling Technique / N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer // Journal of Artificial Intelligence Research. 2002. Vol. 16. Pp. 321–357. DOI: 10.1613/jair.953.

23. Sola, J. Importance of input data normalization for the application of neural networks to complex industrial problems / J. Sola, J. Sevilla // IEEE Transactions on Nuclear Science. 1997. Vol. 44, Is. 3. Pp. 1464–1468. DOI: 10.1109/23.589532.

24. Singh, D. Investigating the Impact of Data Normalization on Classification Performance / D. Singh, B. Singh // Applied Soft Computing. 2020. Vol. 97, Part B. Art. No. 105524. 23 p. DOI: 10.1016/j.asoc.2019.105524.

25. Khaire, U. M. Stability of Feature Selection Algorithm: A Review / U. M. Khaire, R. Dhanalakshmi // Journal of King Saud University — Computer and Information Sciences. 2022. Vol. 34, Is. 4. Pp. 1060–1073.
DOI: 10.1016/j.jksuci.2019.06.012.

26. Hota, H. S. Decision Tree Techniques Applied on NSL-KDD Data and Its Comparison with Various Feature Selection Techniques // H. S. Hota, A. K. Shrivas // Advanced Computing, Networking and Informatics — Volume 1: Advanced Computing and Informatics Proceedings of the Second International Conference on Advanced Computing, Networking and Informatics (ICACNI-2014) (Kolkata, India, 24–26 June 2014) / M. K. Kundu, [et al.] (eds.). — Cham: Springer International Publishing, 2014. — Pp. 205–211. — (Smart Innovation, Systems and Technologies, Vol. 27). DOI:10.1007/978-3-319-07353-8_24.

27. Aggarwal, S. Machine Learning Algorithms, Perspectives, and Real-World Application: Empirical Evidence from United States Trade Data // International Journal of Science and Research. 2023. Vol. 12, Is. 3. Pp. 292–313.
DOI: 10.21275/SR23305084601.

28. Application of Logistic Regression and Machine Learning Methods for Idiopathic Inflammatory Myopathies Malignancy Prediction / W. Zhang, G. Huang, K. Zheng // Clinical and Experimental Rheumatology. 2023. Vol. 41, Is. 2. Pp. 330–339. DOI: 10.55563/clinexprheumatol/8ievtq.

29. Comparative Performance Analysis of K-Nearest Neighbour (KNN) Algorithm and Its Different Variants for Disease Prediction / S. Uddin, I. Haque, H. Lu, [et al.] // Scientific Reports. 2022. Vol. 12. Art. No. 6256. 11 p.
DOI: 10.1038/s41598-022-10358-x.