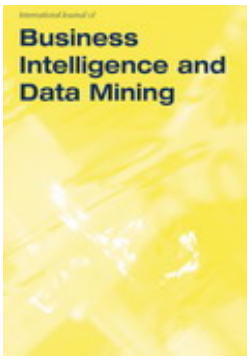


International Journal of Business Intelligence and Data Mining

ISSN online: 1743-8195 - ISSN print: 1743-8187
https://www.inderscience.com/ijbidm

Performance evaluation of oversampling algorithm: MAHAKIL using ensemble classifiers

C. Arun, C. Lakshmi

DOI: 10.1504/IJBIDM.2022.10043149

Article History:
Received: 26 August 2021
Accepted: 17 September 2021
Published online: 30 November 2022

Performance evaluation of oversampling algorithm: MAHAKIL using ensemble classifiers

C. Arun*

Department of Computational Intelligence,
School of Computing, SRMIST,
Chennai, Tamil Nadu, India
Email: arunc@srmist.edu.in
*Corresponding author

C. Lakshmi

School of Computing,
SRM Institute of Science and Technology, India
Email: lakshmic@srmist.edu.in

Abstract: Class imbalance is a known problem that exists in real-world applications, which consists of disparity in the existence of sample counts of different classes that results in biased performance. The class imbalance issue has been catered by many sampling techniques which may either fall into an oversampling approach that solves issues to a greater extent or under sampling. MAHAKIL is a diversity-based oversampling approach influenced by the theory of inheritance, in which minority samples are synthesised in view of balancing the class using Mahalanobis distance measure. In this study the performance of MAHAKIL algorithm has been tested using various ensemble classifiers which are proved to be effective due to its multi hypothesis learning approach and better performance. The results of the experiment conducted on 20 imbalanced software defect prediction datasets using six different ensemble approaches showcase XGBoost provides better performance and reduced false alarm rate compared to other models.

Keywords: class imbalance; software fault prediction; synthetic samples; over sampling techniques; MAHAKIL; false alarm rate; evolutionary algorithm; ensemble; inheritance.

Reference to this paper should be made as follows: Arun, C. and Lakshmi, C. (2023) 'Performance evaluation of oversampling algorithm: MAHAKIL using ensemble classifiers', *Int. J. Business Intelligence and Data Mining*, Vol. 22, Nos. 1/2, pp.1–15.

Biographical notes: C. Arun received his BTech in Information Technology and ME in Software Engineering and currently pursuing his PhD. He is currently working as an Assistant Professor in SRM Institute of Science and Technology, India having more than ten years of experience and one year industry experience. His research interest includes software testing, software architecture, quality analysis, and machine learning.

C. Lakshmi holds a PhD in Computer Science and Engineering from SRM University and is a Professor in the Department of Computational Intelligence, School of Computing, SRM Institute of Science and Technology. Her main area of research interest includes pattern recognition, image processing,

machine learning and web services. She is a Life Time member of the Indian Society for Technical Education (ISTE), International Association of Computer Science and Information Technology, Singapore and International Association of Engineers-IAENG, Hong Kong. She has published several papers in well-known peer-reviewed journals.

1 Introduction

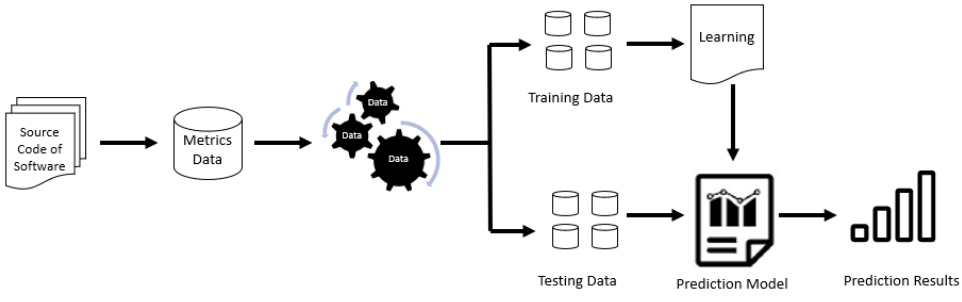
Predictive analytics is a type of data analytics aimed at prediction of future events or unknown events given with current data or historical data using machine learning models. Software defect prediction (SDP) is kind of predictive analytics (Rathore and Kumar, 2019; Song et al., 2011) which primarily focus on the likelihood prediction of faulty modules among the given set of components based on the underlying characteristics of a software modules (Wang and Yao, 2013; Menzies et al., 2007). SDP models use various quantitative metrics calculated across the software modules as an input towards predicting the defective ones. As stated by Pareto principle 80% of the defects is due to 20% of the problem, if we can predict those 20 predict of the component then we cater the efforts of testing towards those components which in turn minimise the cost and efforts required for testing activity. Software metrics has high correlation towards the quality of the software component (Shatnawi and Li, 2008; Shatnawi et al., 2006). Various studies have been carried out to understand the relationship between the software metrics (Kitchenham, 2010; Hall et al., 2012) and defects in which metrics proposed by Chidamber and Kemerer's object oriented (CK OO) (Li and Henry, 1996; Tang et al., 1999; Chidamber et al., 1998), McCabe, Halstead, static code, process, code change, representative process, and other metrics were analysed towards its impact in identify the defective modules. Studies concluded that object oriented, and process metrics have high probability in identifying the fault prone modules. Performance of the SDP is hindered due to the imbalance nature of the fault prediction dataset. Class imbalance is classical machine problem that exists in many real-world applications such as rare disease prediction, fraudulent transaction identification, Spam detection, intrusion detection, anomaly detection and so on (Joshi et al., 2001; Nazir, 2009; Bikku et al., 2019). Due to intrinsic nature of the software component the defect prediction dataset also falls into the imbalance category where only few faulty instances available compared to the number of non-faulty instances (Kubat and Matwin, 1997; Laurikkala, 2011; Arun and Lakshmi, 2020). Machine learning models assume the class of data are equally balance in case of inequality the result will be biased towards majority class i.e., non-defective ones. Because of the imbalance nature of SDP dataset (NASA Data Repository, http://mdp.ivv.nasa.gov; PROMISE Data Repository http://openscience.us/repo/; Software Defect Datasets, https://ieee-dataport.org/), the conventional machine learning models are biased towards the majority classes which results in higher false alarm rate. Ensemble techniques attempt to improve the performance of prediction model by combining several base classifier models to provide an optimum prediction model.

2 Related works

SDP models aims to predict the fault prone components of software module using their underlying characteristics (Rathore and Kumar, 2019; Song et al., 2011). Prediction model is trained using those properties of the software components which contain the fault information of those components too. Early detection of faulty components helps the quality assurance teams to streamline their efforts which in turn minimise the cost, time and effort of testing activities there by enhance the quality of software.

Figure 1 depict the generalised software fault prediction process which comprises of three major components are: metrics dataset, performance model and evaluation (Kitchenham, 2010; Li and Henry, 1993). Software fault dataset is constructed by collecting information related to faults from various sources such as repositories, change logs, bug tracking module and so on (Tang et al., 1999). Once the fault information extracted the values for various software metrics on component level, process, project level is extracted. Collected data is given to pre-processing module to handle data inconsistency and other issues related to data. Prep-processed data is partitioned into training and testing set and provided to the prediction model. The prediction model is mode of statistical and machine learning techniques which understand the characteristics from the training model and perform prediction on the testing data. Finally, the performance of the prediction model is evaluated using various statistical measures.

Figure 1 Generic software prediction process (see online version for colours)



Performance of the prediction model depends on the quality and quantity of the dataset. Performance of the SDP is hindered by the class imbalance issue which exists in the SDP dataset (Provost, 2000). Generally, the number of faulty components in the software component is very minimal compare to the non-faulty one which creates an imbalance in the dataset. Hence the outcome of the prediction model biased towards the non-faulty components, even though the prediction model provides higher accuracy the class level accuracy with respect the faulty components is low which proved to be ineffective prediction model (Nazir, 2019). Henceforth numerous techniques have been proposed to counter the class imbalance issue. Class imbalance issues have been sorted out by proposing many techniques which are mainly classified into three categories: sampling-based, algorithmic-based and hybrid approach (Rathore and Kumar, 2019).

Sampling-based approach works on the data level which tries to create a balance between the classes using two major techniques namely over sampling and under sampling. Over sampling approach balance the dataset by synthesising the minority samples using various statistical methods or by averaging the two nearest data points of

minority class. Oversampling approaches such as ROS (Rathore and Kumar, 2019), SMOTE (Chawla et al., 2002), B-SMOTE (Han et al., 2005), ADASYN (He et al., 2008), and variations of SMOTE (Barua et al., 2014). ROS approach tries to balance the dataset by randomly replicating the samples of minority class till the desired level of balance reach because of duplication it suffers from the overfitting issue. SMOTE (Chawla et al., 2002) synthesis the minority samples by identifying the nearest neighbour of a minority sample and drawing a line between them and pick points along the lines. B-SMOTE (Han et al., 2005) synthesis samples using the minority samples which lies on the boundary and are difficult to classify.

Under sampling techniques on the other create desired balance in the dataset by removing samples from the majority class iteratively. The sample are removed either randomly or the sample which is either noisy or safe to discard are identified and eliminated. Oversampling results in overfitting on the other hand undersampling suffers from loss of information. Kubat and Matwin proposed an undersampling techniques by combining Tomek link and CNN (Vergin Raja Sarobin et al., 2021; Reghukumar et al., 2021) approach to identify samples to be discarded. Laurikkala proposed an undersampling which computes three nearest neighbour for each sample, and discard the sample if three neighbours are minority and sample consider is from majority class.

Algorithm-based approach creates the balance between the class by modifying the learner's algorithm instead of altering data. Algorithmic level approach based on techniques such as decision tree, SVM, Bayesian have been proposed to alleviate the bias. SVM method modifies the kernel function by increasing the cost of misclassifies are entities which are difficult to classify. Finally, hybrid approach which combines both data and algorithmic level to counter the imbalance issues. Hybrid technique such as boosting, bagging, SMOTE+Tomek (Rathore and Kumar, 2019), SOME+ENN (Rathore and Kumar, 2019) alleviate the imbalance issue by either stacking multiple learners or combing the performance average of multiple parallel runs or combing the oversampling and undersampling techniques.

The primary oversampling approaches such as ROS, RUS were able to generate a required number of samples, but fails to perform better when predicting the faulty components. Random oversampling approach generates synthetic sample by duplicating the existing minority sample, due to which algorithm suffers in over generalisation problem while predicting the faulty ones. Similarly, the random undersampling algorithm balance the dataset by eliminating some of the samples from minority class which results in loss of information which leads to overfitting while performing the prediction process. In this study, we attempt to analyse the performance of MAHAKIL algorithm with different oversampling approaches, since this algorithm generate diverse sample which helps in alleviating the class imbalance issue also reduce the possibility of over-generalisation and over fitting.

3 Material and method

20 benchmark SDP datasets (NASA Data Repository, http://mdp.ivv.nasa.gov; PROMISE Data Repository http://openscience.us/repo/; Software Defect Datasets, https://ieee-dataport.org/) obtained from various repositories such as promise, NASA, and IEEE Dataport Repositories have been used to construct the experiment which suffers from the class imbalance issue largely. Imbalance ratio of the dataset varies from 6

percentile to 27 percentage. The quality of a software component is measure with respect to various characteristics by using different class of metrics such as code metrics, process metrics, project metrics, change code metrics, CK metrics and so on. There were about 84 different such kind metrics are in use in the software development. Based on the studies conducted by Chidamber et al. (1998), Li and Henry (1993), Ohlsson et al. (1998), Tang et al (1999), Glasberg et al. (1999), Shatnawi et al. (2006), Kpodjedo et al. (2009), Radjenovic et al. (2013), Rathore and Gupta (2012), its observed CK and OO (Tang et al., 1999) are highly aligned towards fault prediction efficiency. Hence those metrics are primarily considered for the fault prediction model.

Metrics are the quality indicators which defines the quality of a software component and the process used (Sutton, 1903; Ohlsson et al., 1998; NASA Data Repository, http://mdp.ivv.nasa.gov; PROMISE Data Repository http://openscience.us/repo/). Dataset contains the value for these software metrics which is calculated across class, modules code metrics (Table 1) of the projects that are written in Java. Table 1 provides the description of the various static code metrics that represented in the datasets. The bug represent the status of defects, if it contains the defect the numbers of defect found in the module is mentioned else zero is mentioned. A summary of the datasets (Table 2), including the name and its release version, number of modules, number of non-defects, number of defects and percentage of defective samples.

Table 1 Details of static code metrics

Abbreviation	*Description*
WMC	Weighted method per class
DIT	Depth of inheritance tree
NOC	Number of children
CBO	Coupling between objects classes
RFC	Response for a class
LCOM	Lack of cohesion in methods
CA	Afferent coupling
CE	Efferent coupling
NPM	Number of public methods
LCOM3	Lack of cohesion in methods, differ from LCOM
LOC	Lines of code
DAM	Data access metrics
MOA	Measure of aggregation
MFA	Measure of functional abstraction
CAM	Cohesion among methods of class
IC	Inheritance coupling
CBM	Coupling between methods
AMC	Average methods complexity
MAX_CC	Maximum value of CC methods of Investigate class
AVG_CC	Arithmetic mean of the CC value in the Investigate class
DEFECTS	Counts of bugs detected in the class

Table 2 Summary of the 20 imbalanced datasets from promise repository

S. no.	*Dataset*	*# modules*	*# defective*	*# n-defective*	*Defects (%)*
1	ant-1.3	125	20	105	0.160
2	ant-1.4	178	40	138	0.225
3	ant-1.5	293	32	261	0.109
4	ant-1.6	351	92	259	0.262
5	arc	234	27	207	0.115
6	camel-1.4	875	145	730	0.166
7	camel-1.6	965	188	777	0.195
8	ivy-1.4	241	16	225	0.066
9	ivy-2.0	352	40	312	0.114
10	jedit-4.0	306	75	231	0.245
11	jedit-4.1	312	79	233	0.253
12	jedit-4.2	367	48	319	0.131
13	log4j-1.0	135	34	101	0.252
14	pbeans2	51	10	41	0.196
15	readktor	176	27	149	0.153
16	Synapse-1.0	157	16	141	0.102
17	systemdata	65	9	56	0.138
18	tomcat	858	77	781	0.090
19	xerces-1.2	440	71	369	0.161
20	xerces-1.3	453	69	384	0.152

Performance of the prediction model have been evaluated using various statistical metrics such as accuracy, precision, recall, AUC, ROC, FPR, FNR, sensitivity, F-score and false alarm rate (Rathore and Kumar, 2019). These metrics are calculated based on the values of confusion matrix which is presented in Table 3. The performance of the conventional techniques is assessed by accuracy metric, but the SDP doesn't fit into that because of class imbalance issue which results in poor fault prediction efficiency. Hence metrics such as recall, precision, F-score and false alarm rate considered to validate the model. Recall (pd) is the measure tells the module contained fault is classified correctly. False alarm rate (pf) is the measure where non-faulty modules are wrongly classified as faulty one which ultimately over compensate the process of fault prediction (Buckland and Gey, 1994; Joshi et al., 2001). Hence the ideal measure to evaluate the performance are defined in equations (1), (2), (3) and (4).

$$\text{Recall(pd)} = \text{TP}/(\text{TP}+\text{FN}) \tag{1}$$

$$\text{Precision} = \text{TP}/(\text{TP}+\text{FN}) \tag{2}$$

$$\text{F-score(F1)} = 2*((\text{Recall}*\text{Precision})/(\text{Recall}+\text{Precision})) \tag{3}$$

$$\text{False alarm(pf)} = \text{FP}/(\text{TP}+\text{FN}) \tag{4}$$

Table 3 Confusion matrix

	Predicted positive	*Predicted negative*
Positive	TP	FN
Negative	FP	TN

4 Experiment setup

This section discusses the details of experimental setup which includes different ensemble-based algorithms consider for the study of MAHAKIL performance using different levels of balancing used to predict the faulty components using source code metrics.

4.1 Experiment design

Objective of the work is to provide a comparative study of the ensemble learning model combined with over sampling approach called MAHAKIL proposed by Ebo Bennin et al (2017). MAHAKIL was inspired from the theory of inheritance by Walter Sutton and Theodor Boveri (Sutton, 1903) which synthesis new generation samples by obtaining the traits from both parents. MAHAKIL introduce synthetic samples using diversity-based measure called MD. Mahalanobis (1936) proposed the distance metric which computes the diversity of the sample towards its distribution. The Minority samples are separated from the majority and the MD is calculated for each minority sample and the dataset is divided into two bins (c1, c2) based on the distance value. c1 contain the samples with value less than the midpoint and c2 contain the sample with value greater than midpoint and the sample in each bin are labelled. The synthetic samples introduced by computing the average of the samples with same label in each bin. The process repeated till the required balance attained. Performance of the MAHAKIL oversampling testing is evaluated using traditional machine learning models such as C4.5, NNET, KNN, SVM and RF (Ebo Bennin et al., 2017). Its observer that RF and KNN with MAHAKIL outperform all other models but, still there is room for improvement in terms of false alarm rate.

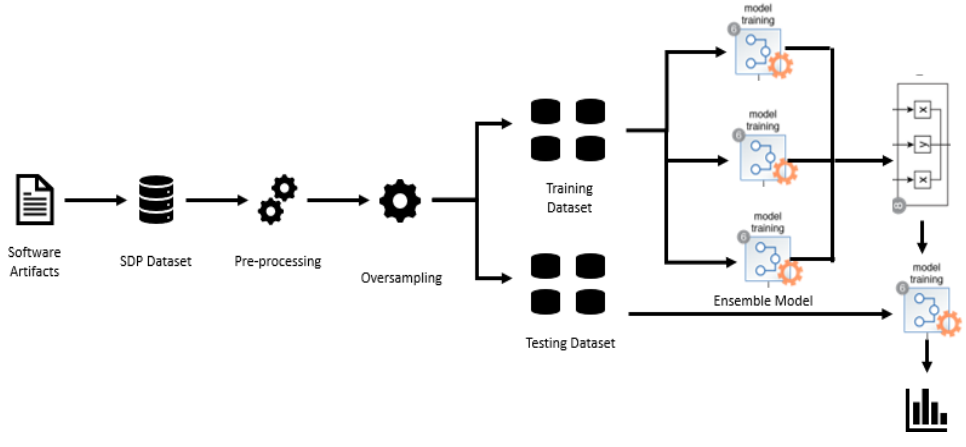
4.2 Algorithms

Ensemble techniques are known to provide better accuracy compared to traditional models, so in this work we attempt to evaluate the performance of MAHAKIL using five XGBoost, gradient boosting, ada boosting, bagging classifier, and LightGBM classifier model which is depict in Figure 2. Ensemble classifier is an attempt to improve the performance of the learning by training the sample with multiple base classifiers and average the performance across them. For computing the final prediction performance generally two methods adopted are: bagging and boosting.

Bagging perform classification by dividing the training set into multiple bins and perform classification on each bin parallelly and compute the average in order to obtain the final result. Boosting model uses the weak learner models sequentially and tends increase the cost of weak learner or adjusting the weights of the samples which are

difficult to predict. Also, in some cases stacking applied where the dataset is parallelly trained using heterogenous models and combine the accuracy of all model to obtain the final result.

Figure 2 Experimental setup (see online version for colours)



4.2.1 AdaBoost

AdaBoost refers to adaptive boosting ensemble model designed for binary classification which performs prediction using multiple iteration to create a strong learner. Weak learner is tweaked in favour of instance which are misclassified or difficult to classify in previous iteration. In each phase of training, a new weak learner is added to the ensemble, and a weighting vector is adjusted to emphasis on instances that were misclassified in previous rounds.

4.2.2 Gradient boosting

Gradient boosting is ensemble model which try to increase the prediction performance of the model by adding a new predictor in view of correcting the predecessor. Initial prediction starts with base learner with equal weights and model with false prediction is identified and prediction is assigned to learner with higher weightage.

4.2.3 XGBoost

XGBoost is a decision tree-based ensemble technique where each regression tree maps an input sample to one of its leaf's that contains a continuous score. XGBoost improves the performance of the prediction by adding loss function to computer difference in prediction outcome and penalty term by adding weights to misclassified instances. The process continues iteratively to make final prediction by adding new tress.

4.2.4 LightGBM

LightGBM is a boosting model based on tree learning algorithm which grows vertically i.e., leaf-wife or breadth first approach. In which the leaf with large loss or difficult in prediction is allowed to grow, hence loss can be lowered.

4.2.5 Bagging classifier

A bagging classifier is an ensemble meta-estimator base model in which random subset of original dataset is obtained and fir to the base classifier model. The final outcome is obtained by aggregating the results of base classifier either by voting or averaging method and arrive at the final outcome.

4.3 Construction of ensemble learner

The dataset obtained from the repositories often contain noisy, incomplete and inconsistent which is given to the pre-processing model for data cleaning. Cleaned data is given as input to the oversampling module which balance the dataset by synthesising minority samples using MAHAKIL approach in multi fold fashion. The balancing approach has been adopted with three different variations of 30, 40 and 50 percentage, since the highest level of balancing existing in the original dataset was 26 percentage and maximum balance to obtain is 50 percentage. Balanced dataset in partitioned into training and test dataset and training dataset was given as input to the Ensemble learning framework which learns from the input sample and the model performance is verified by validating the model using testing dataset. In order to analyse the performance of the selected ensemble algorithms, precision, F-score, and false alarm rate are used as evaluation criteria and all experiments were tested using a five-fold cross validation strategy.

5 Result and discussion

Experiment is conducted using MAHAKIL algorithm along with five prominent ensemble model to study the performance of MAHAKIL with ensemble model also the performance of MAHAKIL algorithm with traditional classifier models such as naive Bayesian and decision tree. The major concern with the SDP is false alarm rate due which the efforts catered towards QA activities is wasted. Based on the percentage of balancing and the variation in the sample count, we have picked five different datasets to showcase the performance of MAHAKIL when integrated with ensemble models. All the ensemble techniques and MAHAKIL algorithm is implemented using python libraries such as scikit-learn, lightgbm, and XGBoost. Table 4 show the precision, f-measure and false alarm scores generated from five learners with different level of balancing.

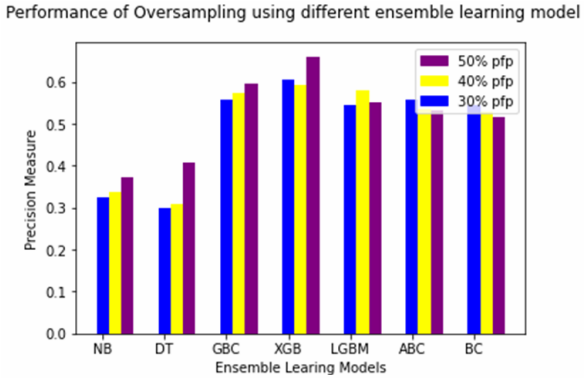
Table 4 Precision, F-score and false alarm of five ensemble model with various balancing levels

% balance		30 % balance			40% balance			50% balance		
Measures		Precision	F_score	F_alarm	Precision	F_score	F_alarm	Precision	F_score	F_alarm
ant-1.3	NB	0.389	0.413	0.103	0.379	0.413	0.082	0.372	0.319	0.090
	DT	0.400	0.444	0.098	0.400	0.444	0.078	0.367	0.371	0.098
	GBC	0.330	0.286	0.095	0.333	0.286	0.095	0.500	0.500	0.095
	XGB	0.650	0.333	0.048	0.500	0.333	0.048	0.500	0.333	0.048
	LGBM	0.432	0.341	0.000	0.587	0.522	0.077	0.333	0.286	0.095
	BC	0.400	0.302	0.081	0.390	0.311	0.076	0.433	0.348	0.076
	ABC	0.512	0.371	0.048	0.667	0.571	0.048	0.667	0.571	0.048
Camel 1.4	NB	0.400	0.205	0.041	0.400	0.205	0.041	0.333	0.114	0.027
	DT	0.148	0.143	0.144	0.167	0.154	0.130	0.267	0.271	0.151
	GBC	0.625	0.270	0.021	0.625	0.270	0.021	0.500	0.256	0.034
	XGB	0.500	0.121	0.014	0.500	0.121	0.014	0.500	0.121	0.014
	LGBM	0.476	0.400	0.075	0.476	0.400	0.075	0.375	0.340	0.103
	BC	0.370	0.233	0.058	0.429	0.270	0.053	0.425	0.289	0.060
	ABC	0.417	0.244	0.048	0.417	0.244	0.048	0.400	0.205	0.041
Ivy 2.0	NB	0.625	0.625	0.048	0.500	0.333	0.032	0.500	0.333	0.032
	DT	0.273	0.316	0.095	0.300	0.333	0.063	0.400	0.444	0.095
	GBC	0.620	0.400	0.038	0.500	0.500	0.063	0.500	0.500	0.063
	XGB	0.667	0.571	0.032	1.000	0.400	0.000	0.453	0.400	0.000
	LGBM	0.250	0.167	0.048	0.455	0.526	0.095	0.455	0.526	0.095
	BC	0.646	0.610	0.043	0.674	0.562	0.033	0.671	0.512	0.033
	ABC	0.400	0.444	0.095	0.444	0.471	0.079	0.444	0.471	0.079

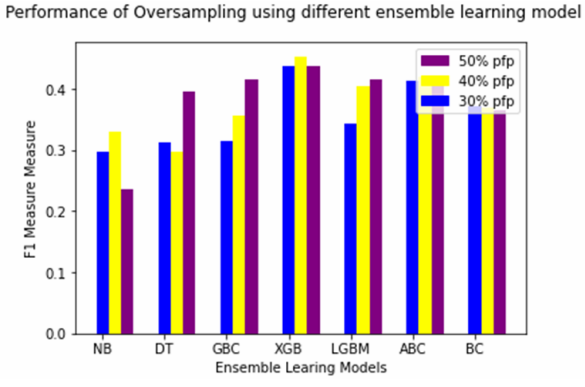
Table 4 Precision, F-score and false alarm of five ensemble model with various balancing levels (continued)

% balance		*30 % balance*			*40% balance*			*50% balance*		
Measures		*Precision*	*F_score*	*F_alarm*	*Precision*	*F_score*	*F_alarm*	*Precision*	*F_score*	*F_alarm*
Jedit 4.2	NB	0.333	0.154	0.091	0.218	0.214	0.090	0.317	0.311	0.086
	DT	0.333	0.316	0.078	0.286	0.235	0.078	0.375	0.333	0.078
	GBC	0.200	0.133	0.063	0.333	0.250	0.063	0.333	0.250	0.063
	XGB	0.714	0.286	0.031	0.333	0.154	0.031	0.333	0.154	0.031
	LGBM	0.516	0.548	0.373	0.333	0.250	0.063	0.333	0.250	0.063
	BC	0.278	0.154	0.041	0.280	0.193	0.045	0.319	0.199	0.052
	ABC	0.614	0.588	0.031	0.500	0.250	0.078	0.500	0.500	0.078
Xerces-1.3	NB	0.125	0.091	0.091	0.125	0.483	0.095	0.167	0.100	0.065
	DT	0.333	0.345	0.065	0.400	0.313	0.169	0.636	0.560	0.052
	GBC	0.571	0.381	0.039	0.571	0.381	0.039	0.667	0.522	0.039
	XGB	0.571	0.381	0.039	0.571	0.381	0.039	0.571	0.381	0.039
	LGBM	0.625	0.455	0.039	0.625	0.455	0.039	0.500	0.462	0.078
	BC	0.523	0.344	0.047	0.520	0.360	0.048	0.554	0.344	0.042
	ABC	0.429	0.286	0.052	0.429	0.286	0.052	0.333	0.200	0.052

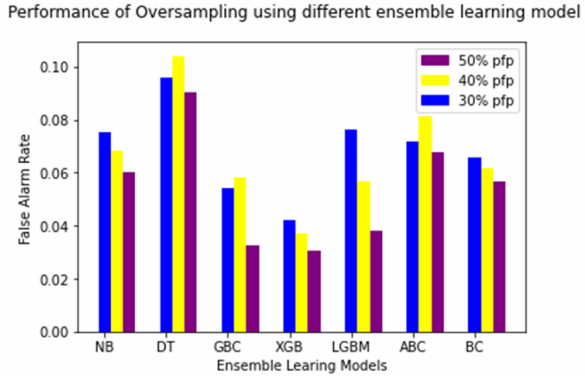
Figure 3 (a) Precision score comparison (b) F1-score measure comparison (c) False alarm rate comparison (see online version for colours)



(a)



(b)



(c)

From Table 4 its evident the ensemble model performs better with increasing in the level of balancing on an average the accuracy of all the ensemble models is greater than 80%, since the data is imbalance in nature accuracy measure doesn't suits the model hence precision, f-score and false alarm rate is considered. From the results it's evident that

amount and quality of data plays a major role in defining the performance of the prediction model, since the performance increase with increase with amount of balancing percentage. Models provides better performance when the dataset balanced at 100% i.e., each class contain 50% of entries.

Performance of ensemble model is better compared with traditional classification models. Figures 3(a), 3(b), and 3(c) shows the average performance of ensemble model with respect to three different performance measures such as precision, f-score and false alarm. The results indicate the XGBoost and bagging classifier provides better performance when compare with other ensemble model for SDP dataset. From the experimental study its evident that XGBoost performs better compare to other models.

Since ensemble model is combination of diverse set of learners which tries to optimise the performance of predictions. During learning process, the model performs high has been preferred over other in subsequent learning process i.e., the method which fails to classify the data accurately in the next level data passed on to the model which performs better and hence overall accuracy increases. Also, ensemble model employs drift and different weighing policies for the data or model which is difficult to classify.

XGBoost is tree-based model which add more and more tree sequentially when there is a residual error or when model perform poor, hence overall accuracy improved thereby reducing the overfit. Hence it's evident from the results that XGBoost algorithm outperforms other ensemble and traditional classification models.

6 Conclusions and future work

In this paper, a comparative study of MAHAKIL algorithm using different ensemble algorithm with three different percentage of balancing level is carried out. All the experiments were conducted on 20 different SDP benchmark dataset obtained from various repositories which aids in finding the best ensemble model in the case of SDP. The performance of the experimental setup was compared and analysed based on precision, F1-score and false alarm rate measure. MAHAKIL algorithm with ensemble models provides better performance when compared traditional machine learning models. The results projected for five different datasets with varied level of balancing from minimum to highest level balancing that exists. Even though obtained better performance but, still room for improvement in the case of false alarm rate is greater concern. The performance of the model can be improved by fine tuning the parameters of learner model to getter better results, since model is trained with default settings. In future the work can be extended to compare the performance of ensemble model with deep learning model in view of reduced false alarm rate.

References

Arun, C. and Lakshmi, C. (2020) 'Class imbalance in software fault prediction data set', in *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, Vol. 1056, pp.745–757, Springer, Singapore.

Barua, S., Islam, M.M., Yao, X. and Murase, K. (2014) 'MWMOTE – majority weighted minority oversampling technique for imbalanced data set learning', *IEEE Trans. Knowledge and Data Eng.*, February, Vol. 26, No. 2, pp.405–425.

Bikku, T., Sambasiva Rao, N. and Akepogu, A.R (2019) 'A novel multi-class ensemble model based on feature selection using Hadoop framework for classifying imbalanced biomedical data', *International Journal of Business Intelligence and Data Mining* (*IJBIDM*), December, Vol. 14, Nos. 1/2, pp.25–39.

Buckland, M.K. and Gey, F.C. (1994) 'The relationship between recall and precision', *J. Amer. Soc. Inform. Sci.*, Vol. 45, No. 1, pp.12–19, 1994.

Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. (2002) 'SMOTE: synthetic minority over-sampling technique', *Journal of Artificial Intelligence Research*, Vol. 16, pp.321–357, https://doi.org/10.1613/jair.953.

Chidamber, S., Darcy, D. and Kemerer, C. (1998) 'Managerial use of metrics for object oriented software: an exploratory analysis', *IEEE Trans Softw. Eng.*, Vol. 24, No. 8, pp.629–639.

Ebo Bennin, K., Keung, J., Phannachitta, P., Monden, A. and Mensah, S. (2017) 'MAHAKIL: diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction', *IEEE Trans. Softw. Eng.*, Vol. 44, pp.534–550.

Glasberg, D., Emam, K.E., Melo, W. and Madhavji, N. (1999) *Validating Object-oriented Design Metrics on a Commercial Java Application*, National Research Council Canada, Institute for Information Technology, pp.99–106.

Hall, T., Beecham, S., Bowes, D., Gray, D. and Counsell, S. (2012) 'A systematic review of fault prediction performance in software engineering', *IEEE Trans Softw. Eng.*, Vol. 38, No. 6, pp.1276–1304.

Han, H., Wang, W.Y. and Mao, B.H. (2005) 'Borderline-smote: a new over-sampling method in imbalanced data sets learning', in *Advances in Intelligent Computing*, Vol. 3644, pp.878–887, Springer-Verlag, Hefei, China.

He, H. et al. (2008) 'Adasyn: adaptive synthetic sampling approach for imbalanced learning', in *Proc. IEEE Int. Joint Conf. Neural Netw. IEEE World Congr. Comput. Intell.*, pp.1322–1328.

Joshi, M.V., Kumar, V. and Agarwal, R.C. (2001) 'Evaluating boosting algorithms to classify rare classes: comparison and improvements', in *Proc. IEEE Int. Conf. Data Mining*, pp.257–264.

Kitchenham, B. (2010) 'What's up with software metrics? A preliminary mapping study', *J. Syst. Softw*., Vol. 83, No. 1, pp.37–51.

Kpodjedo, S., Ricca, F., Antoniol, G. and Galinier, P. (2009) 'Evolution and search based metrics to improve defects prediction', in *2009 1st International Symposium on Search Based Software Engineering*, pp.23–32.

Kubat, M. and Matwin, S. (1997) 'Addressing the curse of imbalanced training sets: one-sided selection', in *Machine Learning-International Workshop then Conference*, Nashville, TN, USA, Morgan Kaufmann, pp. 179–186.

Laurikkala, J. (2011) 'Improving identification of difficult small classes by balancing class distribution', *Artificial Intelligence in Medicine*, Vol. 2101, pp.63–66, Springer-Verlag, Cascais, Portugal.

Li, W. and Henry, S. (1993) 'Object-oriented metrics that predict maintainability', *J Syst. Softw*., Vol. 23, No. 2, pp.111–122.

Li, W. and Henry, S. (1996) 'A validation of object-oriented design metrics as quality indicators', *IEEE Trans Softw Eng*., Vol. 22, No. 10, pp.751–761.

Mahalanobis, P.C. (1936) 'On the generalized distance in statistics', *Proc. Nat. Inst. Sci.*, Calcutta, Vol. 2, pp.49–55.

Menzies, T., Greenwald, J. and Frank, A. (2007) 'Data mining static code attributes to learn defect predictors', *IEEE Trans. Softw. Eng.*, January, Vol. 33, No. 1, pp.2–13.

NASA Data Repository [online] http://promise.site.uottawa.ca/SERepository/datasets-page.html (accessed 20 October 2021).

Nazir, A. (2019) 'A critique of imbalanced data learning approaches for big data analytics', *International Journal of Business Intelligence and Data Mining* (*IJBIDM*), 11 April, Vol. 14, No. 4, pp.419–457.

Ohlsson, N., Zhao, M. and Helander, M. (1998) 'Application of multivariate analysis for software fault prediction', *Softw Qual J*., Vol. 7, No. 1, pp.51–66.

PROMISE Data Repository [online] Software-Defect-Prediction/Data/dataatmaster· SinghJasmeet585/Software-Defect-Prediction·GitHub; https://www.kaggle.com/aczy156 /software-defect-prediction-nasa (accessed 20 October 2021).

Provost, F. (2000) 'Machine learning from imbalanced data sets 101', in *Proc. AAAI' Workshop Imbalanced Data Sets*, pp.1–3.

Radjenovic, D., Hericko, M., Torkar, R. and Zivkovic, A. (2013) 'Software fault prediction metrics: a systematic literature review', *Inf. Softw. Technol*., Vol. 55, No. 8, pp.1397–1418.

Rathore, S. and Gupta, A. (2012) 'Investigating object-oriented design metrics to predict fault-proneness of software modules', in *2012 CSI Sixth International Conference on Software Engineering* (*CONSEG*), pp.1–10.

Rathore, S.S. and Kumar, S. (2019) 'A study on software fault prediction techniques', *Artif Intell Rev*, Vol. 51, No. 2, pp.255–327.

Reghukumar, A., Jani Anbarasi, L., Prassanna, J., Manikandan, R. and Al-Turjman, F. (2021) 'Vision based segmentation and classification of cracks using deep neural networks', *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 29, Supp. 01, pp.141–156.

Shatnawi, R. and Li, W. (2008) 'The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process', *J Syst Softw*., November, Vol. 81, No. 11, pp.1868–1882.

Shatnawi, R., Li, W. and Zhang, H. (2006) 'Predicting error probability in the eclipse project', in *Proceedings of the International Conference on Software Engineering Research and Practice*, pp.422–428.

Software Defect Datasets [online] https://ieee-dataport.org/ (accessed 20 October 2021).

Song, Q.m Jia, Z., Shepperd, M., Ying, S. and Liu, J. (2011) 'A general software defect-proneness prediction framework', *IEEE Trans. Softw. Eng.*, Vol. 37, No. 3, pp.356–370.

Sutton, W.S. (1903) 'The chromosomes in heredity', *Biological Bulletin*, Vol. 4, No. 5, pp.231–250.

Tang, M., Kao, M.H. and Chen, M.H. (1999) 'An empirical study on object oriented metrics', in *Proceedings of the International Symposium on Software Metrics*, pp.242–249.

Vergin Raja Sarobin, M., Jani Anbarasi, L., Prassanna, J., Manikandan, R. and Al-Turjman, F. (2021) 'Swarm intelligence-based optimal device deployment in heterogeneous internet of things networks for wind farm application', *Int. J. Commun. Syst.*, 25 May, Vol. 34, No. 8, p.e4779.

Wang, S. and Yao, X. (2013) 'Using class imbalance learning for software defect prediction', *IEEE Trans. Rel.*, June, Vol. 62, No. 2, pp.434–443.