Software Change Prediction with Homogeneous Ensemble Learners on Large Scale Open-Source Systems

Megha Khanna, Srishti Priya, Diksha Mehra

Software Change Prediction with Homogeneous Ensemble Learners on Large Scale Open-Source Systems

Megha Khanna, Srishti Priya and Diksha Mehra

Sri Guru Gobind Singh College of Commerce, University of Delhi, Delhi, India

meghakhanna86@gmail.com
{shrishtipriya.18,dikshamehra.18}@sggscc.ac.in

Abstract. Customizability, extensive community support and ease of availability have led to the popularity of Open-Source Software (OSS) systems. However, maintenance of these systems is a challenge especially as they become considerably large and complex with time. One possible method of ensuring effective quality in large scale OSS is the adoption of software change prediction models. These models aid in identifying change-prone parts in the early stages of software development, which can then be effectively managed by software practitioners. This study extensively evaluates eight Homogeneous Ensemble Learners (HEL) for developing software change prediction models on five large scale OSS datasets. HEL, which integrate the outputs of several learners of the same type are known to generate improved results than other non-ensemble classifiers. The study also statistically compares the results of the models developed by HEL with ten non-ensemble classifiers. We further assess the change in performance of HEL for developing software change prediction models by substituting their default base learners with other classifiers. The results of the study support the use of HEL for developing software change prediction models and indicate Random Forest as the best HEL for the purpose.

Keywords: Empirical Validation, Ensemble Learners, Large-scale OSS, Software Change Prediction.

1. Introduction

OSS follows the principle of open exchange and community-oriented development. These systems are in continuous development adhering to the dynamic requirements of the users [1]. Regular modifications and capability upgrades of these systems add to their complexity and size. In such a scenario, it is critical to continuously monitor and maintain these systems effectively, so that their quality does not degrade with time. Management and maintenance of these software systems require resources like time, effort and cost. However, considering the limited availability of these resources, they need to be used judiciously. One of the effective methods for sensible use of resources and guaranteeing effective software quality is the prediction of change-prone parts in OSS. In the event that we can predict the parts which are more inclined to changes, they can be examined thoroughly. These parts may be redesigned appropriately or rigorously verified to ensure good quality software. Therefore, constraint resources may be directed to these change-prone parts as they are likely to change because of fault correction or additional functionality requested by the users [2]. Thus, software change prediction (SCP) aids timely delivery and cost-effective management of software systems.

In recent years, various learning techniques have been assessed for SCP. Many studies have predicted the change-prone classes of software systems using statistical methods [3] and machine-learning (ML) techniques [2,4,5]. A recent review by Malhotra and Khanna [1] on SCP ascertained that models developed by ensemble learners exhibit better performance as compared to models developed using classifiers of other categories. Ensemble learners aggregate the output of various base learners to give an effective prediction model. They are further classified into HEL and heterogeneous ensemble learners [6]. In HEL, the same base learner is used for developing several models. However, the diversity is ensured by using varied datasets for training (for example, Bagging). On the other hand, in heterogeneous ensembles, different base learners are used to develop models using the same training dataset. The model outputs the combined outcome of base models through voting or stacking. As compared to heterogeneous ensembles, HEL can employ a larger number of base models. For instance, Random Forest (an HEL) can easily aggregate 100 decision trees developed on variants of training sets, but it is difficult to build 100 base models using diverse algorithms for aggregation using voting or stacking. Thus, this study investigates the effectiveness of SCP models developed using HEL.

We investigate the performance of eight HEL for developing SCP models on datasets obtained from five large-scale OSS. The HEL investigated were AdaBoost (AB), Bagging (BG), Dagging (DG), Decorate (DC), MultiBoostAB (MB), Random Forest (RF), Random SubSpace (RSS) and Rotation Forest (ROF). The models were developed using ten-fold cross validation and inter-version validation. We also statistically compare the

effectiveness of SCP models developed using HEL with models developed using classifiers that belong to other categories such as decision tree, Bayesian learners etc. The study also analyses the change in performance of the HEL when their default base learners are replaced. The following research questions are explored in this study:

RQ1: What is the performance of SCP models developed with HEL using ten-fold cross validation?
SCP models were developed using HEL and the performance of the models was assessed using Area Under the Receiver Operating Characteristics Curve (AUC), F-measure (F1-score) and Mathew's Correlation Coefficient (MCC). The models were ranked in accordance with their performance using Friedman test. A post-hoc Wilcoxon test with Bonferroni correction was also conducted.

RQ2: What is the comparative performance of SCP models developed using HEL in RQ1 with non-ensemble learners?
We compare the performance of top 3 HEL performers (obtained in RQ1) with ten learners that belong to other categories (Classification and Regression Trees (CART), Instance-based learner (IB), J48, JRip, Logistic Regression (LR), Multilayer Perceptron (MLP), Naive Bayes (NB), OneR and Sequential Minimal Optimization (SMO)) for developing the SCP models (ten-fold cross validation). We refer to these algorithms as non-ensemble learners. The comparison is statistically performed by analyzing AUC, F1-score and MCC measures.

RQ3: What is the performance of SCP models developed with HEL using inter-version validation? Are these SCP models better than inter-version models developed using non-ensemble learners?
The question evaluates the effectiveness of HEL for developing SCP models using inter-version validation. Thereafter, the performance of the developed SCP models is statistically ranked using the Friedman test. We also statistically compare the pairwise difference (Wilcoxon test) in the performance of SCP models developed by the investigated HEL and those developed using the ten non-ensemble learners over three performance measures (AUC, F1-score and MCC) using inter-version validation.

RQ4: Does the change in base learners significantly improve the performance of models developed by HEL?
The question ascertains if there is any change in performance of SCP models developed using HEL when their default base learners are modified. For seven of HEL's investigated in the study (except RF), we developed SCP models (ten-fold cross validation) by replacing their base learners with ten classifiers. The classifiers evaluated as base learners were the non-ensemble learners investigated in RQ2. We statistically rank the performances (using AUC, F1-score & MCC) of different base learners for each HEL using the Friedman test. The top three base learners that attained the highest ranks were then compared with the default base learners of the HEL using Wilcoxon test.

The results of the study confirm the efficacy of HEL in the domain of SCP. The developed models can be used by software practitioners for effective management of software resources by focusing them on the problematic change-prone classes. The organization of the paper includes a broad discussion of the related literature studies in Section 2. Section 3 includes the various variables used in the research, the data collection procedure, performance measures and statistical tests. Section 4 provides an overview of the HEL and non-ensemble classifiers analyzed. Section 5 and 6 elaborates on the results of the study and the threats to validity respectively. Section 7 discusses the conclusions drawn and prospective future work.

2. Related Work

Various studies in the domain of software quality predictive modeling have ascertained the superiority of ensemble learners as compared to other non-ensemble algorithms [2,4,11]. The characteristics of some of the prominent literature studies of SCP and Software Fault Prediction (SFP), which is a related area of SCP have been listed in Table 1. The table enlists the total number of datasets used for validation and specifies the percentage of large datasets among them (in brackets). It also lists the HEL investigated, statistical test used, whether the study has evaluated different base learners, domain and the performance measure used in each study.

Ensemble learners have proven to be effective for yielding not only improved SFP models but also SCP models. In this context, Catolino and Ferrucci [4], Malhotra and Khanna [2], Malhotra and Bansal [10] assessed the performance of HEL for SCP. As depicted in Table 1, Rathore and Kumar [6], Yucular [8], Kaur and Kaur [9] have also evaluated the effect of change in base learners of HEL while developing SFP models. However, only Zhu et al. [5] assessed the performance of HEL for SCP, by changing the underlying base learners. Kumar et al. [12] evaluated the SCP models using heterogeneous ensemble learners. The study by Aljamaan and Alazba [7] validated tree-based HEL for SFP, advocating the use of these techniques in the domain. Amongst the studies

listed in Table 1, only a few percentage of the total datasets that were evaluated by the researchers were large scale OSS. To the author's best knowledge, none of the studies have examined the performance of SCP models using inter-version validation. Also, the change in base learner have been neglected in most of the SCP studies. Motivated by these research gaps, we realized that there is still a need for an extensive evaluation of HEL on large scale OSS. In the presented work, we evaluated eight HEL for SCP and also investigated the effect of change in their default base learners on their predictive capability.

Table 1. Characteristics of literature studies

Study Name	No. of dataset(% of large)	Name of HEL used	Statistical test used	Evaluation of different base learners	Domain (SCP/SFP)	Performance Measures used
Malhotra & Khanna [2]	6 (0%)	RF, AB, LB, BG	Friedman & Wilcoxon	No	SCP	Precision, Recall, AUC, Accuracy, Balance & G-mean
Catolino & Ferruci [4]	8 (12%)	AB, RF, BG	Wilcoxon	No	SCP	Precision & Recall
Zhu et al. [5]	8 (25%)	BG	Scott-knott	Yes	SCP	Recall, F1-score, MCC & AUC
Rathore [6]	28 (35%)	DG, DC, MB, AB, ROF, ES, GR	Friedman & Wilcoxon	Yes	SFP	Precision, Recall, AUC, Specificity, G-mean1 & Gmean2
Aljamaan & Alazba [7]	11 (54%)	RF, ET, AB, GB, HG, XGB, CB	Wilcoxon	No	SFP	Accuracy & AUC
Yucular [8]	15 (20%)	AB, LB, MB, BG, RF, DG, ROF	-	Yes	SFP	AUC & F1-score
Kaur & Kaur [9]	9 (0%)	BG, BOOSTING, RF	Wilcoxon	Yes	SFP	AUC
Malhotra & Bansal [10]	11 (100%)	BG, RF, LB, AB	Friedman	No	SCP	AUC, G-mean & Balance

AB- AdaBoost, **BG**- Bagging, **CB**- CatBoost, **DC**- Decorate, **DG**- Dagging, **ES**- Ensemble Selection, **ET**- Extra Trees, **GB**- Gradient Boosting, **GR**- Grading, **HG**- Hist Gradient Boosting, **LB**- Logit Boost, **MB**- MultiBoostAB, **RF**- Random Forest, **ROF**- Rotation Forest, **XGB**- XGBoost

3. Research Background

This section discusses the independent and the dependent variables of the study followed by the data collection procedure. It also states the performance measures and the statistical tests used in the study.

3.1. Independent Variables

Previous studies have already validated the relationship among OO metrics and change-proneness [2,3]. For our study, we have used eleven Object Oriented (OO) metrics as the independent variables. We use the popular Chidamber and Kemerer metrics suite [13] which consists of Coupling Between Objects (CBO), Number of Children (NOC), Response For a Class (RFC), Depth of Inheritance Tree (DIT), Lack Of Cohesion in Methods (LCOM) and Weighted Methods of Class (WMC). We also used two OO metrics proposed by Lorenz & Kidd [14], i.e. Number of Instance Methods (NIM) and Number of Instance Variables (NIV). Other metrics used were Number of Private Methods (NPM) and Number of Public Methods (NPRM) of QMOOD metrics suite [7] and Lines of Code (LOC) metric. The metrics mentioned were computed using 'Understand' tool (https://www.scitools.com/).

3.2. Dependent Variable

The dependent variable ascertains the probability of change of a class in the upcoming version of the software product [2,5]. We use the binary dependent variable with two possible values "yes" or "no", referring to whether a class changed in the newer version of the product or not.

3.3. Data Collection & Validation

In order to empirically validate our results, we collected data from five large-scale Java OSS namely- Vuze, PlantUml, LogicalDOC, Seata (Simple Extensible Autonomous Transaction Architecture) and MPXJ. Vuze is a software used to search and download torrent files. PlantUml allows creation of UML diagrams using a simple textual description language. Seata is a distributed transaction solution that brings high performance under a microservices architecture. LogicalDOC is a document management platform. MPXJ is a file handling library for Java. Two consecutive stable versions of all these datasets were acquired from http://sourceforge.net/. The analysed versions were designated as "old" and "new" according to their release date.

These OSS were chosen based on the following criterion- (i) The common classes (data points) of each software should be 800 or more, (ii) The percentage of changed classes in "old" and "new" versions should be 20% and above, (iii) The software system should belong to varied domains like community oriented and industry oriented.

Table 2 displays the number of classes and the size (in KLOC) of the older and newer version, the common classes of both the versions (data points) and the percentage change of classes in the two versions of each dataset. As can be seen from the table, the number of classes in the investigated datasets range from 911-3616, indicating the large-size of datasets. The table also depicts the versions of datasets taken for validation while performing inter-version validation. The versions used for validation are the successive versions of those used for training.

In order to compute the dataset, at first, the OO metrics (mentioned in Section 3.1) were computed for the older version of each of the dataset. We use the Understand tool (https://www.scitools.com) for extracting the metrics. Secondly, the common classes of the two versions (the older version and newer version) of the dataset were compared to identify changes in the classes. Thereafter, interfaces and methods were excluded. Additionally, the metrics include various anonymous and unknown classes which were also discarded. Finally, change statistics were computed for each of the common classes in the metrics. These common classes are the data points. Change statistics include the number of inserted, deleted and modified source code lines for each data point. After computing the change statistics, we introduce a binary variable 'ALTER'. ALTER is the dependent variable of our study. For each data point, if the change statistics computed gave a non-zero positive value, the ALTER was marked as "yes", otherwise "no".

We examine the five datasets and develop models using 10-fold cross-validation and inter-version validation. In cross validation, the dataset is divided into several sub-parts, in this case ten. Then, nine of these divided datasets are used for training the model, while the remaining one is used for testing the model. This process is repeated ten times, so that each dataset part is used for testing at least once.

Inter-version validation refers to the validation of training model developed using version 'v' of a software using any of the later versions of the same software. The difference in inter-version validation is that it takes the dataset obtained from later versions of the software into account to be used as testing data rather than dividing a single dataset into training and validation data (i.e. in k-fold cross validation). For inter-version validation, we first develop an SCP model, let's say S1 using the dataset obtained from analyzing the change and metrics of version v1 & v2. Thereafter, S1 is validated using the dataset obtained from analyzing the change and metrics of version v2 & v3 (later versions of corresponding software).

Table 2. Software details

Name of Software	Classes	Data Points	Training data (% change)	Time Gap	Size (in KLOC)	Validation data (% change)
Vuze	3590-3616	2559	5.7.4 - 5.7.5 (33%)	3 months	625-632	5.7.5 – 5.7.6 (3%)
PlantUml	2966-2772	2329	1.2020.10 - 1.2020.22 (31%)	7 months	265-212	1.2020.23 – 1.2021.1 (12%)
LogicalDOC	1948-1370	1117	8.3.4 - 8.4.2 (23%)	6 months	229-159	8.4.2 – 8.5.2 (22%)
Seata	1123-1229	906	v1.2 – v1.4 (35%)	7 months	67-74	v1.4.0 – v1.4.1 (5%)
MPXJ	911-942	821	8.0.0 – 8.2.0 (26%)	8 months	175-181	8.3.0 – 8.5.0 (26%)

Training data is the dataset used for ten-fold cross validation .

3.4. Performance Measures

All the models developed in the study were analyzed based on performance measures described below. The greater the value of these measures, the better is the performance of developed models. We selected these performance measures as they are robust, stable and give effective results even with imbalanced data [5,8,12].

- *F1-score:* It is measured as the harmonic mean of precision and recall.
- *AUC:* It is the area under the Receiver Operator Characteristic (ROC) curve. The curve is plotted for true-positive rate (y-axis) vs. false-positive rate (x-axis).
- *Mathew's Correlation Coefficient (MCC):* It is a symmetric measure which gives an unbiased result than other measures in an imbalanced data sample. The formula of MCC incorporates true positives, false positives, true negatives and false negatives. The range of MCC values lies from -1 to +1.

3.5. Statistical Tests

We use two non-parametric tests i.e. Friedman test and Wilcoxon signed rank test to statistically evaluate the results of our study. Friedman test is used to rank the performance of SCP models developed by HEL (in RQ1 & RQ3) on the basis of performance measure values (AUC, MCC and F1-score) across all the datasets. The test statistic is based on chi-square distribution. We further employed Wilcoxon test which is used to pairwise assess two classifiers and check if there is a significant difference in their performance. The comparison done between the two classifiers depends on the pairwise difference obtained on the values of performance measures. Bonferroni correction was used with Wilcoxon test where the chosen α value (0.05) is divided by the total number of pairwise comparisons evaluated. This correction is used to reduce the number of false positives i.e. type 1 error in statistical analysis.

4. Research Methodology

In this section, we briefly introduce the various HEL used in the study to develop SCP models. We selected these HEL as they encompass a diverse category of ensemble techniques. For instance, AB and MB are boosting classifiers, while BG and DG belong to the class of bagging learners. DC uses artificial training instances, while RSS chooses random features for model development. Lastly, ROF and RF techniques are aggregate of decision trees.

1. *AdaBoost (AB)* – It is a boosting technique that tries to improve the classification performance by training a sequence of weak learners. In this iterative technique, every following weak learner is trained to focus on the feature that was missed by the previous learner [10].
2. *Bagging (BG)* - It is a method in which the sample data is divided into independent subsets of data using bootstrap. The individual datasets are then evaluated with a weak-learner and their result is aggregated using the voting method [15].
3. *Dagging (DG)* - It is a method in which the sample data is divided into disjoint subsets (i.e. independent datasets are generated without replacement). The final result is evaluated by combining the output of weak learners on disjoint datasets using a voting scheme [6].
4. *Decorate (DC)* - This technique builds different intermediate prediction models by using specially constructed artificial training samples. The predictions from the weak learners are then integrated into one by the mean combination rule [6].
5. *MultiBoostAB (MB)* - It is an extension of AdaBoost method. It reduces the prediction bias and discrepancy in the final model by incorporating wagging techniques [16].
6. *Random Forest (RF)*- This technique consists of a number of decision trees, making sure that each one of the individual trees is distinct. Each tree is built on a subset of data points (with replacement) and the nodes of the tree use random features which are selected without replacement [17].
7. *Random SubSpace (RSS)* - It selects random subsets containing particular features of a sample dataset. The result is then predicted by the majority vote of the models created using these subsets [18].
8. *Rotation Forest (ROF)* - This method utilizes Principal Component Analysis (PCA) algorithm to choose features and data of the training sample to generate individual decision trees. The classification of each decision tree is aggregated to give the final result by the voting method [6].

Table 3 depicts the parameter settings of HEL used in this study. These are the default parameter settings of the WEKA tool.

Table 3. Parameter details of HEL

HEL	Default base learners	Parameter Values
AB	Decision Stump	Batch Size =100, Iterations =10, Weight Threshold =100
BG	REPtree	Batch Size=100, Iterations =10
DG	Decision Stump	Batch Size =100, Number of Folds =10
DC	J48	Batch Size =100, Desired Size =15, Iterations =50
MB	Decision Stump	Batch Size =100, Iterations =10, Weight Threshold =100
RF	Random Tree	Batch Size=100, Max Depth =0, Iterations=100
RSS	REPtree	Batch Size =100, Iterations =10, Sub Space Size= 0.5
ROF	J48	Batch Size =100, Group Size=3, Iterations=10

The non-ensemble classifiers analysed in the study were- CART, IB, J48, JRip, LR, MLP, NB, OneR and SMO. These methods were chosen as they belong to different classification categories. CART and J48 belong to decision trees, NB and LR belong to statistical models, JRip and OneR belong to rule-based, IB comes under K-nearest neighbor, SMO comes under support vector machine and lastly MLP belongs to neural network. For these non-ensemble classifiers, we have used only the default parameters of the WEKA tool.

5. Result Analysis and Discussion

We discuss the results of RQ's of our study in this section.

5.1. *RQ1. What is the performance of SCP models developed with HEL using ten-fold cross validation?*

We assess the performance of the SCP models developed using HEL by analyzing AUC, F1-score and MCC values. The models were developed using ten-fold cross validation. Fig. 1 depicts a stacked graph of average performance measure values (across all the five investigated datasets).

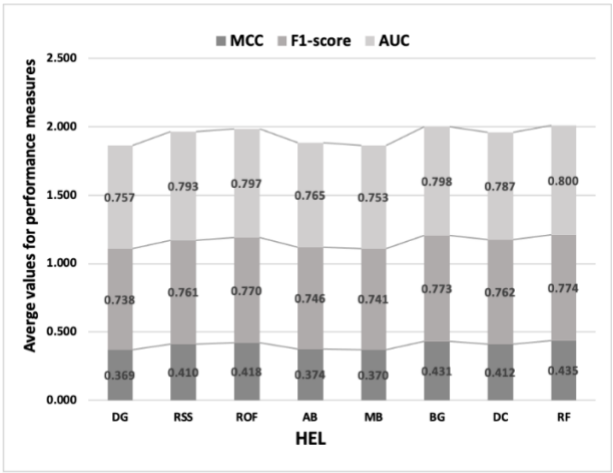

Fig. 1. Graph representing the average values of HEL for AUC, F1-score and MCC.

The AUC values of SCP models on the five datasets were in the range from 0.714 - 0.825, indicating their effectiveness. Similarly, F1-score values and MCC values were in the range of 0.714 - 0.774 and 0.370 - 0.435 respectively. According to the figure, the SCP models developed using RF, BG and ROF were the top 3 performers as they depicted the best average cumulative values for all the three performance measures. DG showed a decrease of 3% and 9% for average F1-score and MCC values over all datasets in comparison to the other average values obtained by HEL, thus demonstrating poor performance. Similarly, MB obtained the lowest value for AUC measure. Nevertheless, all the models developed using HEL exhibited acceptable values. These results support the use of HEL for determining change-prone classes in large OSS. We statistically analyzed the performance of the developed SCP models using the Friedman test. The test was conducted on the performance measure values (AUC, MCC and F1-score) obtained by the models on the five datasets. In all the three cases, Friedman test results

were found significant at α = 0.05. This indicates a significant difference in the performance of the investigated HEL for developing SCP models. The models developed by RF and BG ensemble classifiers obtained the best Friedman ranks using AUC, F1-score and MCC values. The next best ranks were obtained by ROF for AUC and F1-score measures and RSS for MCC measure respectively. The models developed by MB, DG and AB obtained the worst ranks.

Additionally, we employed the post-hoc Wilcoxon test with Bonferroni correction (α= 0.05) to pairwise compare the performance of RF with other HEL (Table 4). RF outperformed (denoted by ⇑ in Table 4) all the other seven HEL, thereby, making it a desirable HEL for developing SCP models. However, its superiority was not significant.

Table 4. Wilcoxon test with Bonferroni correction of RF with all other HEL

Performance measures	AB	BG	DG	DC	MB	ROF	RSS
F1-score	⇑	⇑	⇑	⇑	⇑	⇑	⇑
AUC	⇑	⇑	⇑	⇑	⇑	⇑	⇑
MCC	⇑	⇑	⇑	⇑	⇑	⇑	⇑

⇑ : better

5.2. *RQ2. What is the comparative performance of SCP models developed using HEL in RQ1 with non-ensemble learners?*

We assess the performance of the models developed using HEL in RQ1 by comparing them with ten non-ensemble classifiers using AUC, F1-score and MCC values. Fig. 2 displays the average values of the performance measures obtained by the HEL and non-ensemble classifiers across all the datasets. As depicted in figure, the models developed using the eight HEL obtained higher average values than the ten non-ensemble classifiers for AUC, F1-score and MCC values. The average AUC values for SCP models obtained by the non-ensemble classifiers on the five datasets were in the range of 0.587 - 0.750. Similarly, average F1-score values were in the range of 0.425-0.766 and average MCC values were in the range of 0.199-0.414. These values were lower than those reported by the models developed in RQ1. With respect to AUC, models developed by non-ensemble classifiers exhibited a 12% decrease as compared to models developed by HEL. Similarly, F1-score values attained by the non-ensemble classifiers decreased by 7 % and MCC values depicted a 15% decrement. Thus, SCP models developed using HEL show an improvement over the models developed by non-ensemble learners.

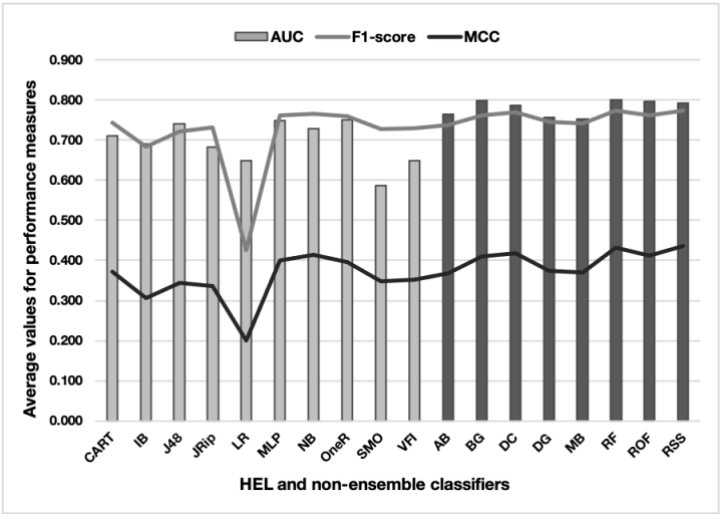

Fig 2. Graph representing average AUC, F1-score and MCC values for HEL and non-ensemble classifiers

We also statistically evaluated the results using Wilcoxon test with Bonferroni correction at α= 0.05 by pairwise comparing the best three performing HEL (obtained by Friedman test in RQ1) with ten non-ensemble classifiers on the three performance measures. Table 5 depicts the Wilcoxon test results and indicates that HEL perform notably better by consistently obtaining higher values than non-ensemble classifiers across all the datasets. In all the 90 pairwise comparisons performed, there was only one exception in which a non-ensemble classifier i.e. J48

obtained a higher MCC value than RSS (HEL). However, these results were not significant when Bonferroni correction was used.

Table 5. Comparing HEL & non-ensemble classifiers using AUC, F1-score and MCC

	MLP	SMO	NB	IB	VFI	CART	J48	JRIP	ONER	LR
AUC										
RF	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑
BG	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑
ROF	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑
F1-score										
RF	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑
BG	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑
ROF	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑
MCC										
RF	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑
BG	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑
RSS	⇑	⇑	⇑	⇑	⇑	⇑	⇓	⇑	⇑	⇑

⇑ : better, ⇓ : worse

5.3. *RQ3. What is the performance of SCP models developed with HEL using inter-version validation? Are these SCP models better than inter-version models developed using non-ensemble learners?*

We assess the performance of SCP models developed using HEL and non-ensemble learners using inter-version validation by evaluating AUC, F1-score and MCC measures. Table 6 and 7 depict the average values of performance measures obtained on five datasets by HEL and non-ensemble learners respectively. The average AUC values for SCP models developed by HEL were in the range 0.727-0.764. The average values for F1-score and MCC were in the range of 0.769-0.790 and 0.160-0.255 respectively. On the other hand, the average performance measure values for non-ensemble learners were in the range 0.619-0.759 (AUC), 0.544-0.806 (F1-score) and 0.150-0.210 (MCC), which were considerably lower than those obtained by models developed using HEL.

Table 6. Performance measure values of HEL

	AB	BG	DG	DC	MB	RF	RSS	ROF
AUC	0.727	0.756	0.732	0.764	0.729	0.759	0.758	0.749
F1-score	0.769	0.786	0.780	0.778	0.789	0.781	0.790	0.789
MCC	0.160	0.204	0.175	0.195	0.210	0.255	0.207	0.208

Table 7. Performance measure values of non-ensemble learners

	MLP	SMO	NB	IB	VFI	CART	J48	JRip	OneR	LR
AUC	0.743	0.619	0.745	0.704	0.692	0.715	0.705	0.643	0.620	0.759
F1-score	0.797	0.815	0.806	0.766	0.544	0.781	0.779	0.792	0.792	0.802
MCC	0.196	0.206	0.200	0.208	0.150	0.196	0.192	0.184	0.184	0.210

Further, in order to determine the best HEL for developing SCP models using inter-version validation, we statistically assessed their performances using Friedman test. Though the test results were not found significant for AUC measure and F1-score, the results were significant for MCC at α=0.05. RF, BG and RSS were the top three performers obtaining the highest values for MCC.

We further used Wilcoxon test with Bonferroni correction to validate the performance of eight HEL with the ten non-ensemble learners on MCC values. Fig. 3 shows the number of non-ensemble learners that performed better (shown below the axis) and worse (shown above the axis) than HEL when validated using inter-version validation. The figure illustrates that the SCP models developed using HEL performed better than those developed

using non-ensemble learners in the majority of cases (54%), however Boosting techniques (AB and MB) and DG showed poor performance measure values than most of the investigated non-ensemble learners. Similar trend was observed when Wilcoxon test was conducted using AUC and F1-score values. Wilcoxon results also depicted that the MCC values for SCP models developed using RF were the best amongst all non-ensemble learners (Fig. 3) as the models developed using RF were superior than all the other investigated non-ensemble learners.

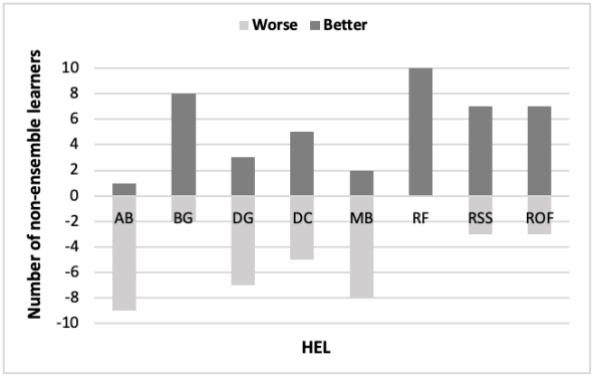

Fig.3. Wilcoxon results for MCC values (Inter-version validation)

5.4. *RQ4. Does the change in base learners significantly improve the performance of models developed by HEL?*

We assess the performance of HEL by changing their default base learners for developing SCP models using ten-fold cross validation. We used the non-ensemble classifiers (used in RQ2) as the various base learners. However, it may be noted that since RF is the aggregation of multiple decision trees where each decision tree uses different features, we could not alter its base learner. For all the other seven investigated HEL, we altered the base learners to evaluate the change in their performance.

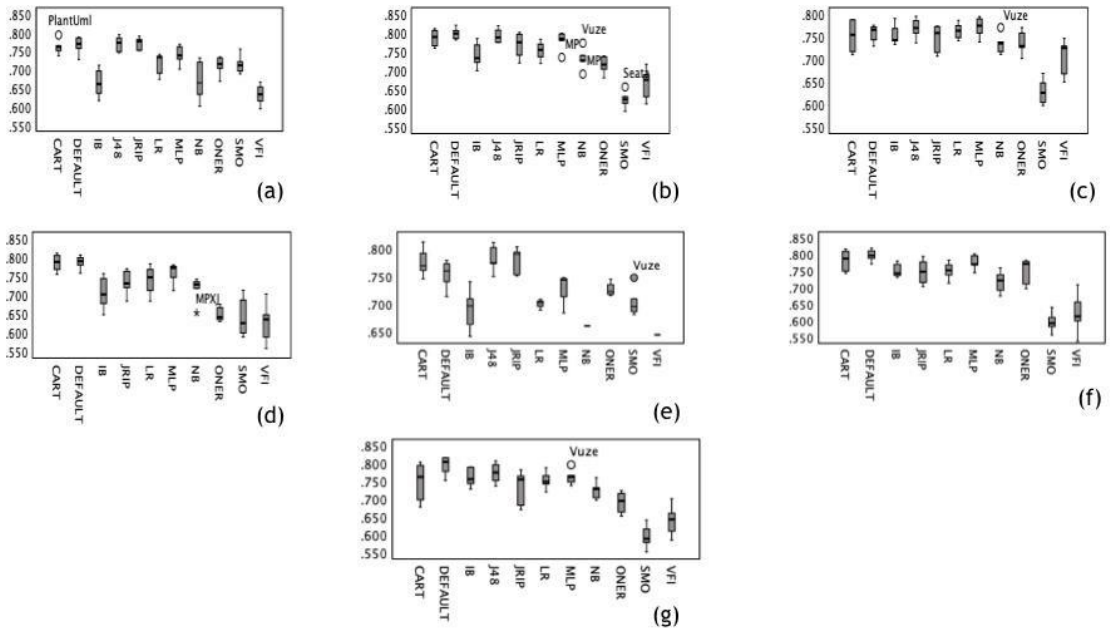

Fig. 4. AUC box plots for different base learners - (a) AB, (b) BG, (c) DG, (d) DC, (e) MB, (f) ROF, (g) RSS

Fig. 4 shows the AUC box-plots for the developed SCP models by changing the base learners of investigated HEL. The figure depicts that majority of the investigated HEL show best results for SCP models with their default base learners. It was also observed that most of the HEL show improved performances with J48, JRip and CART as their base learners in comparison to their default base learner (although, the percentage change was less than 1%). However, all the HEL with SMO and VFI as their base learners obtain the lowest values. There was an average

decrease of 16% for the AUC value, 33% for the MCC value and 17% for F1-score for SMO and VFI in comparison to default base learner.

To rank the performance of SCP models developed using HEL with different base learners, we performed the Friedman test. Table 8 shows the best three ranks obtained by various base learners for all the HEL on three performance measures (AUC, F1-score and MCC). For each HEL, we evaluated eleven possibilities, one with their default base learner and other ten with the rest of the base learners used in this study. The Friedman test result was found significant at α= 0.05. According to Friedman results on AUC, F1-score and MCC values, in addition to corresponding default base learners, J48, JRip and CART were designated as the best base learners for most of the investigated HEL. The MLP technique also exhibited effective results when used as a base learner for DG and MB. On the other hand, SMO, VFI and OneR were found to be the worst base learners.

Furthermore, we used post-hoc Wilcoxon test with Bonferroni correction to validate the performance of HEL with their default base learner and the top three base learners (according to the Friedman test). The test evaluates a total of 63 pairwise comparisons of HEL with varied base learners for the three performance measures (AUC, F1-score and MCC). According to the test results, the performance of SCP models developed using HEL with different base learners showed an improvement in only 8 of the 21 cases when AUC was evaluated. AB, DG and MB progressively increased their AUC performance when their default base learners were altered. But in majority of the cases, there was a decline in the AUC performance of other HEL when their base learners were changed. Similar results were observed when Wilcoxon test was performed using F1-score and MCC values. Thus, changing the default base learner does not necessarily improve the performance of HEL for developing SCP models.

Table 8. Top ranks of base learners according to Friedman test

Friedman ranks for AUC				**Friedman ranks for F1-score**			**Friedman ranks for MCC**		
HEL	**Rank 1**	**Rank 2**	**Rank 3**	**Rank 1**	**Rank 2**	**Rank 3**	**Rank 1**	**Rank 2**	**Rank 3**
AB	JRip	DFT	J48	JRip	J48	DFT	JRip	J48	CART
BG	DFT	J48	CART	DFT	J48	CART	J48	DFT	CART
DG	MLP	J48	LR	DFT	J48	MLP	MLP	J48	JRip
DC	DFT	J48	CART	DFT	CART	J48	CART	DFT	J48
MB	J48	JRip	CART	JRip	MLP	DFT	JRip	CART	MLP
ROF	DFT	J48	CART	DFT	CART	J48	DFT	CART	J48
RSS	DFT	J48	MLP	DFT	CART	IB	CART	DFT	J48

DFT: Default Base Learner

5.5. Discussion of Results

The results of the study indicate the effectiveness of the investigated HEL for developing SCP models. The performance of HEL was evaluated on AUC, F1-score and MCC for five large OSS by ten-fold cross validation and inter-version validation. For ten-fold cross validation, the mean AUC value exhibited by all the HEL across the datasets was 0.778 with a standard deviation (SD) of 0.02. Similarly, the mean F1-score and MCC values were 0.758 (SD = 0.02) and 0.402 (SD = 0.06) respectively. The values obtained by all the HEL for the three performance measures are close to the mean value, signifying that all the ensemble learners are competent and effective for developing SCP models [19,20]. Similar results were obtained using inter-version validation for which the mean AUC and F1-score were 0.747 and 0.783 respectively. However, there was a slight decline in the MCC values obtained using inter-version validation (mean MCC=0.202).

It was observed that the SCP model developed using RF was the most efficient as it obtained the highest values for all the investigated performance measures. The maximum values obtained by models developed using RF (ten-fold cross validation) for AUC, F1-score and MCC were 0.825, 0.796, and 0.518 respectively across all the datasets. Even for inter-version validation, the average AUC value obtained by RF exhibited a 2% increase and MCC exhibited a 31% rise compared to the other HEL. RF works on the principle of bagging using random feature selection method. Since, RF is a combination of decision trees which are developed using varied samples of the dataset (chosen randomly with replacement) and varied predictors (chosen randomly without replacement from the original set of features), it provides an edge over other HEL [21]. Also, RF can handle large amount of data and tends to decrease overfitting [17]. These characteristics of RF make it an ideal technique for developing SCP classification models.

According to Friedman test results (both ten-fold & inter-version), SCP models developed by BG, ROF and RSS also yielded effective results as they were amongst the top-3 HEL in majority of the cases. ROF technique involves the application of BG and feature selection to perform PCA which is used to build decision trees. This

ensures accuracy and diversity of the individual decision trees [22]. BG tends to reduce variance as it performs sampling of data with replacement. Just as the variance decreases, overfitting also decreases which increases the accuracy of BG algorithm [23].

AB and MB (boosting techniques) and DG performed poorly in comparison to other investigated HEL. Boosting techniques (AB and MB) work on weak base learners which are sensitive to noise. Boosting techniques also give more weight to misclassified data and hence if the data contains outliers, it will tend to increase overfitting [23]. This could be a possible reason for their poor results.

The results obtained by SCP models developed using HEL on inter-version validation were comparable to the results obtained by ten-fold cross validation . For instance, in LogicalDOC the AUC and F1-score values of SCP models developed using RF (ten-fold cross validation) were 0.785 and 0.796 respectively. On the other hand, the inter-version model on LogicalDOC (using RF) exhibited an AUC and F1-score value 0.707 and 0.758 respectively. Though, there was a slight decrease (up to 10%) in these performance measure values when using inter-version, they can be considered at par with ten-fold cross validation results.

As indicated by the values of performance measures (Fig. 2), SCP models (ten-fold cross validation) developed using HEL were found to be superior to the models developed using non-ensemble classifiers. There was a decrease of 6% in the values of the non-ensemble classifier obtaining the highest value for AUC as compared to the highest value of AUC obtained by the best HEL (RF). Likewise, a decrease of 1% and 5% was observed for F1-score and MCC values respectively. Unmistakably, the higher values obtained by HEL for all performance measures highlights their accuracy and efficiency for predicting change-prone classes. Even the HEL obtaining the lowest Friedman rank (MB) secured a minimum value of 0.753 over all datasets for AUC. This value is greater than the highest value obtained by the non-ensemble learner (LR) for AUC which was 0.750. The superiority of HEL was also confirmed when the SCP models were developed by inter-version validation. There was a slight decline in the performance of SCP models developed using non-ensemble learners in comparison to those developed using HEL. On an average, there was a 7%, 2% and 4% decline for AUC, F1-score and MCC values. This signifies the effectiveness of HEL for prediction tasks. It may be noted that the common errors in a model are often described in terms of two properties- bias and the variance [23]. Ensemble techniques aim to minimize variance and bias by combining various base models to build one optimal prediction model. The reduction in the variance element of generalization error improves the prediction capacity of the models developed by ensemble learners. Therefore, the results obtained by HEL for developing SCP models are robust as HEL reduces these errors and obtains an evenly spread values for all performance measures.

As observed in the results of RQ4, when the default base learners of the proposed HEL for developing SCP models are altered, there is a relative decline in their performance. With respect to AUC, there was a drop of 2-10% in the performances of HEL across all the datasets. MB depicted the lowest decrease of 2% and the highest decrease of 10% was obtained by RSS for AUC values. F1-score and MCC values also underwent a similar decrease. Since the percentage decrease was low, we statistically ranked all the base learners investigated for a specific HEL. J48, CART, JRip and MLP were ranked as the top base learners as they gave better values for the performance measures compared to the other investigated base learners. The results from the study suggested that when these base learners are used, there was a positive effect on the AUC values for AB, MB (boosting techniques) and DG.

Amongst the top three base learners, two of the learners (J48 and CART) were decision tree algorithms. These learners use greedy approach. They split the data on the best feature by considering the accuracy of all the available features [23]. As decision trees implicitly perform feature selection they are successful as base learners for HEL. However, in order to generalize these results, we need to ascertain the use of HEL with other base learners for developing SCP models on even larger datasets. Since the performance difference of HEL with altered base learners was not significant, researchers may use the default base learners for developing SCP models.

The SCP models developed in the study can be put into use by the software industry while allocating resources like time, effort and cost. An effective way for maintenance of large OSS is the prediction of change-prone classes so that more resources may allocated to these classes. To assess the efficacy of the developed SCP models, we carried out cost-benefit analysis on all five datasets using HEL techniques [25]. The cost/benefit gain is computed as the saving of resources if the developed SCP models are put into use instead of random testing. The higher the value of cost/benefit gain, the more successful is the SCP model. The percentage cost/benefit gain given by all HEL for the five datasets was found to be in the range of 26%-54%. This indicates optimum use of constraint resources if the developed models are put into effect by software managers.

6. Threats to Validity

The SCP models developed in the study using HEL have been statistically evaluated using Friedman and Wilcoxon test. This substantiates the conclusion validity of our results. Moreover, the performance of the models were evaluated on three performance measures- AUC, F1-score and MCC. This increases the credibility of the results.

The independent variables used in the study are the commonly used metrics in software engineering literature. These variables have already been validated as predictors in earlier studies [2, 3, 9], reducing the construct validity threat in the study. The results of the study do not take into account the confounding effect of size of the projects and other characteristics in development of the SCP models. However, this was not the intent of the study.

The results of the study are validated on five large OSS belonging to varied domains. However, researchers should perform empirical validation on OSS of different sizes (small, medium, large) along with the OSS developed using different languages like Python, JavaScript, C# to enhance the generalizability of obtained results.

7. Conclusion and Future work

The study performs an analysis of eight HEL namely - AB, BG, DG, DC, MB, RF, RSS, ROF to determine change-prone classes in five large-scale OSS (developed in Java language). SCP models were developed using ten-fold cross validation as well as inter-version validation. The effectiveness of the HEL was statistically evaluated using three performance measures - AUC, MCC and F1-score. The key results of the study are as follows-

- Each of the eight HEL analyzed in the study attained effective results for predicting change-prone classes. Particularly, RF was the best HEL as we observed an increase of 3% in AUC, 2% in F1-score and 9% in MCC values in the SCP models developed by RF using ten-fold cross validation as compared to the models developed by the other seven investigated HEL. Other HEL which showed promising results were BG, ROF and RSS.
- The results from the study indicated that SCP models (using ten-fold cross validation) developed by HEL are superior than those developed by non-ensemble classifiers. It was observed that the top three ranked HEL (RF, BG, ROF, RSS) when compared to the non-ensemble learners showed an improvement of up to 15%, 10% and 23% for AUC, F1-score and MCC values respectively.
- The outcomes of the study also showed that SCP models developed by HEL using inter-version validation methods have better performance than non-ensemble learners. HEL showed an increase of 7% in the AUC values as compared to the non-ensemble learners. In a similar manner, an increase of 2% and 5% was indicated by the HEL for F1-score and MCC values.
- The results of the study illustrate that the change in base learners for each HEL does not significantly improve their performance in the SCP domain. BG, DC, ROF and RSS showed a decline in their performance with J48, CART, JRip and MLP as their base learners. On the other hand, AB, DG and MB exhibited enhanced performances with these base learners. Thus, changing the base learners might not always give promising results for HEL, while developing SCP models.

In future, we would like to evaluate the heterogeneous ensemble classifiers for predicting change-prone classes. For the continuous growth of software, the maintenance of large and complex software systems is important. It is comprehended that efficient SCP models reduce the effort and cost required for maintaining large OSS. The results of the study would aid software managers in choosing optimum classifiers for developing SCP models. Furthermore, effective planning and resource allocation can be implemented using the developed SCP models.

References

1. Malhotra, R., Khanna, M.: Software Change Prediction: A Systematic Review and Future Guidelines. eInformatica Software Engineering Journal, 13(1). 227-259(2019).
2. Malhotra, R., Khanna, M.: An empirical study for software change prediction using imbalanced data. Empirical Software Engineering, 22(6). 2806-2851(2017).
3. Zhou, Y., Leung, H., Xu, B.: Examining the potentially confounding effect of class size on the associations between object metrics and change-proneness. IEEE Transactions on Software Engineering, 35(5). 607-623(2009).
4. Catolino, G., Ferrucci, F.: Ensemble Techniques for Software Change Prediction: A Preliminary Investigation. In: IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), pp. 25 - 30. IEEE, Italy(2018).
5. Zhu, X., He, Y., Cheng, L., Jia, X., Zhu, L.: Software change-proneness prediction through combination of bagging and resampling methods. Journal of Software: Evolution and Process 30(12), 1-17(2018).
6. Rathore, S.S., Kumar, S.: An empirical study of ensemble techniques for software fault prediction. Applied Intelligence, 1-30(2020).
7. Aljamaan, H., Alazba, A.: Software Defect Prediction using Tree-Based Ensembles. In: 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering, pp. 1-10. ACM, USA(2020).

8. Yucular, F., Ozcift, A., Boranbag, E., Kilinc, D.: Multiple-classifiers in software quality engineering: Combining predictors to improve software fault prediction ability. Engineering Science and Technology, an International Journal, 23(4). 938-950(2020).
9. Kaur, A., Kaur, K.: Performance Analysis of Ensemble Learning for Predicting Defects in Open Source Software. In: 2014 international conference on advances in computing, communications and informatics (ICACCI), pp. 219-225. IEEE, India(2014).
10. Malhotra, R., Bansal, A.: Investigation of various data analysis techniques to identify change-prone parts of an open source software. International Journal of System Assurance Engineering and Management, 9(2). 401-426(2017).
11. Elish, M.O., Aljamaan, H., Ahmad, I.: Three empirical studies on predicting software maintainability using ensemble methods. Soft Computing, 19(9). 2511-2524(2015).
12. Kumar, L., Lal, S., Goyal, A., Murthy, N.L.: Change-Proneness of Object-Oriented Software Using Combination of Feature Selection Techniques and Ensemble Learning Techniques. In: Proceedings of the 12th Innovations on Software Engineering Conference (formerly known as India Software Engineering Conference), pp. 1-11. ACM(2019).
13. Chidamber, S., Kemerer, C.: A metric suite for object-oriented design. IEEE Transactions on Software Engineering, 20. 476-493(1994).
14. Lorenz, M., Kidd, J.: Object-oriented software metrics: a practical guide. Prentice-Hall, Inc..(1994).
15. Breiman, L.: Bagging predictors. Mach Learn 24(2). 123–140(1996).
16. Webb, G.I. : Multiboosting: A technique for combining boosting and wagging. Machine Learning. 40(2). 159-196(2000).
17. Brieman, L.: Random forests. Mach. Learn. 45 (1). 5–32(2001).
18. Ho, T. K.: The random subspace method for constructing decision forests. IEEE transactions on pattern analysis and machine intelligence, 20(8), 832-844(1998).
19. Chicco, D., Jurman, G.: The advantages of the Matthews correlation coefficient (MCC) over F1-score and accuracy in binary classification evaluation. BMC genomics, 21(1). 1-13(2020).
20. Shatnawi, R.: Improving software fault-prediction for imbalanced data. In: 2012 International Conference on Innovations in Information Technology (IIT), pp. 54-59. IEEE, UAE(2012).
21. Baskin, I. I., Marcou, G., Horvath, D., Varnek, A.: Random Subspaces and Random Forest. Tutorials in Chemoinformatics. 263-269(2017).
22. Bustamam, A., Musti, M.I.S., Hartomo, S., Aprilia, S., Tampubolon, P.P., Lestari, D.: Performance of rotation forest ensemble classifier and feature extractor in predicting protein interactions using amino acid sequences. BMC Genomics, 20(9). 950-963(2019).
23. Alpaydin, E.: Introduction to Machine Learning. 3rd edn. MIT Press, Massachusetts, USA(2014).
24. Malhotra, R., Khanna, M.: An explanatory study for software change prediction in object-oriented systems using hybridized techniques. Automated Software Engineering, 24(3), 673-717(2017).
25. Sohail, M.N., Jiadong, R., Uba, M.M., Irshad, M., Iqbal, W., Arshad, J. and John, A.V.: A hybrid Forecast Cost Benefit Classification of diabetes mellitus prevalence based on epidemiological study on Real-life patient's data. Scientific reports, 9(1), 1-10(2019).