The impact of parameter optimization of ensemble learning on defect prediction

Muhammed Maruf Öztürk

Abstract

Machine learning algorithms have configurable parameters which are generally used with default settings by practitioners. Making modifications on the parameters of machine learning algorithm is called hyperparameter optimization (HO) performed to find out the most suitable parameter setting in classification experiments. Such studies propose either using default classification model or optimal parameter configuration. This work investigates the effects of applying HO on ensemble learning algorithms in terms of defect prediction performance. Further, this paper presents a new ensemble learning algorithm called novelEnsemble for defect prediction data sets. The method has been tested on 27 data sets. Proposed method is then compared with three alternatives. Welch’s Heteroscedastic F Test is used to examine the difference between performance parameters. To control the magnitude of the difference, Cliff’s Delta is applied on the results of comparison algorithms. According to the results of the experiment: 1) Ensemble methods featuring HO performs better than a single predictor; 2) Despite the error of triTraining decreases linearly, it produces errors at an unacceptable level; 3) novelEnsemble yields promising results especially in terms of area under the curve (AUC) and Matthews Correlation Coefficient (MCC); 4) HO is not stagnant depending on the scale of the data set; 5) Each ensemble learning approach may not create a favorable effect on HO. To demonstrate the prominence of hyperparameter selection process, the experiment is validated with suitable statistical analyzes. The study revealed that the success of HO which is, contrary to expectations, not depended on...
the type of the classifiers but rather on the design of ensemble learners.

**Keywords:** Defect prediction, parameter optimization, ensemble learning.

# 1 Introduction

An intensive effort is devoted to defect prediction which helps practitioners to plan the budget of a software engineering project [1]. Due to the advance in software systems, including limited budgets, much works are required in this domain. Defect prediction works are generally focused on software metrics [2], predictors [3], pre-processing methods [4], and prediction models [5]. However, the works that explore which type of classifiers should be used in data sets having a broad scale of range are intriguing for researchers [6]. In addition to this, in recent years, predictor configuration studies, which aim to find appropriate settings of the predictors, have emerged [7]–[10]. They open new horizons for the development of enhanced prediction models.

Defect prediction is a process which aims to predict future defects of a software project by using historical metrics of related data sets along with defectiveness information. It is generally considered as a binary classification problem. In defect prediction, high error rates are decreased by combining more than one classifier. Predictors could make different decisions about the label of an instance depending on its structural properties. Thus, each predictor yields the best results in different data sets. Regarding this case, various ensemble learning approaches have been developed. Combining more than one classifier to solve a computational problem is generally named ensemble learning [11]. It is employed for improving the performance of a model such as clustering, classification, and approximation. Ensemble learning methods generally adopt three techniques. These are stacking [11], bagging [12], and boosting [13]. Figure 1 demonstrates how an instance is labeled through ensemble learning techniques.

Generally, an ensemble learning method first selects the classifiers to be employed in the method. To this end, the number of the clas-
optimization of ensemble learning . . .

Figure 1. Three different approaches for ensemble learning. (a) Stacking: Uses a meta-learner to decide the label of instance, (b) Bagging: Gives equal weights to the classifiers, (c) Boosting: Utilizes the same mechanism with Bagging but it gives different weights to the classifiers.

Classifiers should be determined in the beginning. The performance of a classifier strongly depends on the type of the data sets. Enormous software systems give large-scale defect prediction data sets to practitioners. Therefore, practitioners should work with the classifiers that are suitable for large-scale data sets. Thereafter, a design is established depending on the error rates and performance parameters. In this design, the number of the classifiers and the settings of hyperparameters are determined in accordance with experimental knowledge.

One of the important points that have long intrigued experts in ensemble learning is how to select hyperparameter settings [14]. Default settings of hyperparameters may not produce consistent results depending on the used data sets. Rather, employing ensemble models generated with various hyperparameters which are determined according to the classifier and the data sets may be a better choice. In the meantime, tuning hyperparameters is relatively time-consuming and effort-intensive process comparing to the traditional defect prediction [7].
With respect to the ensemble learning approaches, the number of classifiers is not entirely standard in the existing methods [15], [16]. Further, it is also ambiguous that which types are to be selected from a great number of predictors. To make a good selection, each predictor should be evaluated for a general hyperparameter configuration.

Various challenges are encountered while using an ensemble learning method. The most notable of them is whether default settings of the classifiers are adopted or not [17]–[19]. Generally, default configuration of the classifiers is used in an ensemble learning method. Preferring such a way degrades the reliability of an experiment. Conversely, exploring hyperparameter settings by testing previous assumptions may yield better results with regard to the proposed method.

HO was examined in terms of defect prediction in a few works [7], [8], [10]. However, the effects of HO on ensemble methods have not been investigated yet. This case can be considered as an obstacle that should be removed to fully comprehend HO. Moreover, whether HO improves overall performance of advanced classifiers is an open issue.

Cross-project defect prediction (CPDP) is one of the main long-standing problems of defect prediction [20], [21]. In this type, training and testing data are taken from different software projects. On the other hand, within-project defect prediction (WPDP) aims to perform prediction by taking different data sets of the same project so that there is not any metric contradiction problem thanks to the use of same metric sets.

In our preceding work, the effects of HO were investigated in CPDP and WPDP [22]. It was intended to provide valuable findings to find out whether HO increases the success of the prediction when heterogeneous metrics are employed. The method was tested on 20 data sets with heterogeneous experimental setting. As a result, it was detected that performance values of WPDP tend to be in a wide range. With respect to AUC and F-measure parameters, if the classifiers working with tree-structured data are employed, CPDP is capable of yielding feasible performance values in which the depth of trees is high.

Main objectives of the paper are as follows: 1) Investigate the effects of applying HO on three ensemble learning methods; 2) Develop a
robust ensemble learning method that results in performance increase when HO is utilized; 3) Provide a new insight into defect prediction in terms of HO. In addition, this paper aims at developing a novel ensemble learning method called novelEnsemble which is expected to produce promising results when HO is employed.

To achieve those goals, open-source data sets are collected in the beginning of the experiment. In the second step, some ensemble learning and HO methods are selected. Third step encompasses developing novelEnsemble in accordance with constructive methods. In the last step, performance results obtained from both applying HO and without HO on ensemble learning methods are recorded. Meanwhile, the most known ensemble learning methods are tried to be evaluated with general hyperparameters. Main steps of the experiment are demonstrated in Figure 2. As known from the figure, data sets are initially divided into two parts. Subsequently, data sets are exposed to normalization and exploited by the algorithms in which default and optimized parameters are used. Last, some performance parameters are yielded and evaluated in the test process.

The contributions of the paper can be summarized as follows:
1) A novel ensemble learning method namely novelEnsemble, which uses mean classification error and total number of defects for instance labeling, is proposed.
2) This paper reveals which properties should an ensemble learning method have when HO is applied
3) Obtained findings of the experiment shows that the type of the parameter search is negligible in HO performance.

The remainder of the paper is organized as follows: In Section 2, the motivation and the shortcomings of previous works are elaborated. Section 3 refers to ensemble learning and HO along with formal definitions. Related works and ensemble learning methods associated with the paper are also in Section 3. Section 4 presents the proposed method. Section 5 describes the data sets and experimental environment. Experimental results are given in Section 6. Threats for the validity are given in Section 7. Last, the findings and the outcomes of the paper are presented in Section 8.
2 Motivation

This section provides four motivations for the experiment. Motivations of the paper are elaborated by giving some references to draw a clear outline.

HO studies in machine learning algorithms are not new in defect prediction [23]–[25]. Their main objective is to find optimal parameters of learning algorithms rather than using default configuration. They were able to produce promising results in terms of HO. However, due to a great number of hyperparameters and classifiers, practitioners who perform hyperparameter selection feel confused [10]. Therefore, bringing a new perspective to HO in terms of classifiers constructing ensemble learner is the first motivation.

Defect prediction is generally a binary classification problem. Therefore, it has configurable settings. HO can be considered as a new research field in terms of defect prediction. Preceding works have investigated hyperparameter search method [10], automated hyperparameter optimization [8], and the effects of HO on defect prediction [7]. However, with respect to the defectiveness indicators of defect prediction data sets, the effects of HO in the classification comprising more than one classifier have not been fully investigated up to this study. Second motivation is to find optimal design of an ensemble learner to yield high classification performance.

On the other hand, ensemble learning methods are preferable to
increase the success of defect prediction [26], [27]. Note that an ensemble learner cannot be devised arbitrarily. The number of classifiers and used approach including bagging, boosting, and stacking do affect the results of ensemble learning method. In addition, configuration of hyperparameters plays an important role in the performance of the prediction. Previous works have pointed out that each classifier may not be compatible with ensemble learning. Further, some classifiers such as C5.0 and neural network tend to produce high performance [8]. Comparing classifiers in contributing the performance of an ensemble learner is the third motivation.

Ensemble learning methods were developed on the basis of pioneer methods such as triTraining [28]. However, in such methods, instance labeling is focused on classification error and the number of classifiers is constant. Instead, practitioners need a novel ensemble learning method which does not depend on the number of classifiers and can be devised by considering the properties of defect prediction data sets. Constructing an ensemble learner by regarding common properties of defect prediction data sets is the fourth motivation.

3 Background and related works

3.1 Preliminaries

If a software is represented with various modules $s_1, s_2, s_3, ..., s_n$, a set of modules are labeled with $l_1, l_2, l_3, ..., l_n$. These labels are either "defective" or "not-defective". A classifier is trained on some parts of labeled instances to predict unlabeled testing instances. This operation is done with a classifier $c$. In some cases, the number of classifiers may be higher than one $c_1, c_2, c_3, ..., c_m$. In such experiments, classifiers are unified by ensemble techniques for deciding the labels of testing instances.

To perform ensemble learning, a classifier $c$ is weighted according to the type of learning. One could assume that $w_1, w_2, w_3, ..., w_m$ are the weights of $c_1, c_2, c_3, ..., c_m$. To decide a label $l$ of $s$, classifiers make a bias set such as 1, 0, 1, 1, 1, ..... That set is combined with the weights
For a binary classification problem, an ensemble learner produces an output $y$ as 1 or 0 in compliance with the type of learners. In this process, the classifiers used in constructing the ensemble learner and the type or the data sets are of great importance. An underlying deciding mechanism of an ensemble learner is given in Figure 3.

![Figure 3. An underlying deciding mechanism of an ensemble learner. The learners are combined by giving a specific set of weights to create a combination for producing an output.](image)

In testing, a set of instances to be predicted can be denoted with $st_1, st_2, st_3, ..., st_p$. $p$ denotes the number of testing instances. However, it is not compulsory to equalize $p$ and $n$. Instead, $p$ is generally determined as less than $n$. For this purpose, a specific percentage is used to perform testing. Otherwise, cross-validation methods are applied. 10-fold cross-validation is one of the most popular of them.

### 3.2 Related Works

Related works can be examined in twofold: these are ensemble learning and HO studies which are closely related to defect prediction. In this respect, existing works have been summarized and the need for performing the experiment has been stressed. Due to the fact that HO is not a new research field for defect prediction, the number of related studies is limited. However, prevalence of HO in search-based software engineering has helped to solve this issue.

The efficacy of ensemble learning approaches depends on the prop-
properties of defect prediction data sets in a specific ratio. In a study [26] where the prediction performance is examined by combining feature selection with ensemble learning, forward selection outperformed other feature selection techniques. It can be considered one of the basic studies because its findings revealed that feature selection is of great importance for ensemble learning. In another study [29], feature selection helped to solve class imbalance problem and increased the success of ensemble learning. Such hybrid models had been tested in some areas except defect [27]. For instance, a hybrid model was tested on banking system and yielded promising results [27].

Wang et al proposed a new classifier for defect prediction data sets by benefiting the advantages of kernel and ensemble learning approaches [6]. Although the method produced high results in F-measure, employing only industrial data sets without considering open-source ones creates a crucial threat for the validity. Further, a tuning operation was not conducted during the classification experiment.

The success of ensemble learning was also evaluated for multi-class defect prediction. In one of them [30], initially, binary-labeled instances were converted to multi-class form. Thereafter, the classification was completed based on a sophisticated coding. Despite this method coped with imbalanced data sets, hyperparameter settings were not considered while developing ensemble learning method. The interaction between class imbalance and ensemble learning much intrigued researchers. In [31], two boosting methods were discussed on five performance measures. According to the obtained results, AdaBoost.NC showed best performance in general.

Laradji et al. proposed an ensemble learning module [26]. Their experiment using seven classifiers depicted that greedy selection is much preferable for feature selection. Although AUC results are close to 1 in three data sets, the method should be verified with different feature selection and ensemble methods. Likewise, Peng et al. pointed out that AdaBoost provides some advantages along with analytical hierarchy process.

Researchers have long strived to determine how a modification made on a software influences defect ratio. In [32], this impact was investi-
gated in terms of scale and proposed a two-layer ensemble method utilizing both bagging and stacking. The method was evaluated with three alternatives on six data sets. It revealed up to 70% of defect-prone modifications. Related paper stated that improving the parameters of proposed method was planned.

HO is one of the focuses of search-based software engineering [33]–[35]. Parameter tuning has long been a topic of interest in the study of search-based software engineering.

Fu et al claimed that tuning process of defect prediction requires too much time [10]. In their study, it was detected that HO is feasible and increase the accuracy of the prediction up to 60%, especially in defect data sets. While making tuning on selected parameters of the predictors, the effect of parameter search method is not negligible. Thus, a study [9] asserts that parameter search method strongly depends on the scale of the data sets. Moreover, random search method is much suitable with small-scale data sets.

One of the most comprehensive studies was conducted by Tantithamthavorn et al. [8] who examined almost all parameter search methods exhaustively. They concluded that some classifiers such as C5.0 and neural network are rather compatible for HO. It was also stressed that HO should be tested in terms of ensemble learning method.

Practitioners should employ HO without applying a machine learning model, asserted in a recent study [7]. K-nearest neighbor outperformed competing methods with the maximum enhancement in the accuracy of the prediction.

### 3.3 Ensemble learning algorithms

Ensemble learning approach is an interesting topic that emerged at the end of 90’s [36]–[38],[38]. As shortly stated in Section I, three different approaches are adopted in ensemble learning: boosting, bagging, and stacking. Therefore, related experiments are devised to encompass at least one of them. Uncommonly, a study could benefit more than one approach.
optimization of ensemble learning . . .

In the beginning, software defect prediction had relied on tests performed with one classifier [39], [40]. Ensemble learning studies were revealed after discussing the findings of preceding works [41]. Due to ensemble learning methods date back to the mid of the 1990s, basic methods were applied and their improved versions were produced. In this section, primary ensemble learning methods are elaborated.

Pioneer works preferred to use neural network in constructing ensemble learning methods [42], [43]. Experimental data sets are mainly associated with image processing.

Learn++ is an ensemble learning method which uses lazy classifiers and is fast to neural networks [44]. In Learn++, last decision mechanism is weighted voting. The method converges well as it does not need prior training data instances. Ensemble learning was also utilized in training process. In [45], a learning machine namely EN-ELM was developed and tested on some images. This study depicts the prominence of the selection of parameters for alleviating the computational cost of classification.

Consistent methods can be developed by combining various ensemble learning strategies. WAG is one of them [46] and it was constructed by unifying wagging and bagging. Unlike bagging, wagging assigns weights to training instances using Poisson distribution. The method was able to reduce test error remarkably.

Ensemble learning was also used in the classification of noisy data sets [47]. Thus, a model having high error tolerance and accuracy can only be obtained by that way.

While developing an ensemble learning method, having high number of main classifiers increases computational cost quadratically. Therefore, classifier count is tried to be optimal magnitude. For instance, triTraining is an ensemble method that consists of three classifiers in which classification errors are compared to decide a label of an instance [28]. Test operation is performed by comparing error ratios of three classifiers to decide instance label. triTraining showing better performance than three competing methods is regarded as a reference model to various ensemble learning methods.

In [48], a fuzzy cluster-based ensemble learning approach namely
IFCESR was proposed. It employs soft clustering techniques to create ensemble clusters. The effectiveness of the method was tested on UCI data sets. According to the obtained results, IFCESR surpassed state of the art alternatives in terms of clustering accuracy.

Customer scoring is an interesting field in which ensemble learning was utilized [27]. The method using hybrid methods simultaneously has better performance results with AdaBoost than other methods. Moreover, PCA is much feasible for feature selection rather than information gain and GA. Fuzzy cognitive map was improved with ensemble approach [49]. In doing so, it was observed that the performance of fuzzy cognitive map decreases remarkably when it is employed with Hebbian learning.

Pratama et al. presented a new ensemble learning method namely pEnsemble [50]. It consists of three components: drift detection, ensemble pruning, online feature selection. The main advantage of pEnsemble is that it features less complexity than its alternatives.

3.4 Limitations of existing approaches

Existing works have investigated various aspects of applying HO on defect prediction data sets so far [7]–[10]. They rely on specific assumptions. First, default configuration of classifiers does not yield high success according to the HO models. Thus, a special configuration is searched for each classifier. Second, it is asserted that parameter search method should be selected in accordance with the experimental data sets. In this respect, parameter search methods are compared in varying type of data sets. The common aspect of preceding works is that they do not include any evaluation of HO in terms of ensemble learning approaches. In particular, preceding works lack investigation of ensemble learning approaches in which HO is applied on constructor classifiers. Rather than establishing such an experimental environment, detail information is presented by extending the evaluation interval of hyperparameters. In this case, the experiment is enriched according to the main objective but the scope of the evaluation is restricted.

Ensemble learning approaches date back to older times then HO [6], [26], [47], [51]. Therefore, with respect to the defect prediction, the
number of ensemble studies is more than those of HO. Ensemble learning methods are generally proposed to address class imbalance and noisy data problems. However, mathematical models related to the defectiveness metrics are rarely employed in the construction of an ensemble learning method.

To alleviate the limitations mentioned above, this paper investigates the effects of HO in ensemble learning approaches. To this end, some promising classifiers such as C5.0 and neural network have been used in constructing experimental ensemble methods. In addition to this, novelEnsemble having no restriction on the number of classifiers is proposed. The method considers the number of defects depending on the number of line of codes while determining an instance label.

Performing the experiment for a data set could fill the gaps as follows: 1) Observing the success change of ensemble learning approaches in case of applying HO; 2) Encouraging the development of ensemble methods that are designed for defect prediction rather than general data mining operations; 3) Determining whether each HO operation has a profound impact on defect prediction; 4) Revealing the compatibility level of ensemble learning approaches on HO in which static code metrics are exploited.

4 Method

In the concept of ensemble learning, the main purpose is to combine various classifiers to increase prediction performance. The reason is that using only one classifier in a prediction experiment may not perform well in every experimental condition.

The method presented in Algorithm 1 is a defect prediction algorithm. It can be consisted of various classifiers depending on the experimental design. In this study, Algorithm 1 comprises three classifiers RandomForest, C5.0, and neural network. The error mentioned in the following paragraph is the percentage of incorrectly classified instances in the prediction. The algorithm utilizes defectiveness rate while comparing the errors of the classifiers. The classifier producing the minimum error in the iteration determines the label of an instance in testing.
Algorithm 1 takes three parameters including $O$, $U$, and $C$. This method includes an error additive model. First loop of the algorithm encompasses $S_i = \text{BootStrap}(O)$ and $E_i = 0.5$ which is an arbitrary error value that is used if related classifier does not change in terms of error rate during the learning.

In the second loop, $N$ denotes the number of the instances to be exposed to ensemble learning. $E_1 = C_1(U_i), E_2 = C_2(U_i), E_3 = C_3(U_i)$ calculates the error rates of three classifiers. $\text{total} \leftarrow \text{total} + E_j$ calculates the total error up to the reached instances. $\text{total}$ changes incrementally until all the instances are exposed to training. In the latter step, mean error is calculated and assigned to $av$. Mean error is associated with the number of instances and the number of classifiers. $i$ represents the index of an instance. In the third step, the classifier having minimum error is detected with $z \leftarrow \min(e_1, e_2, e_3)$. $y \leftarrow \int av + x^2 + c$ is a function that relies on total defect, mean error, and the number line of codes $av, x, c$. Since a defective region can be detected with at least two curves as in Figure 4, an integral function is utilized in Algorithm 1.

![Figure 4](image-url). An example illustration of area between curves determining defective region of a defect prediction data sets.
The ambiguousness of the boundary of defective region reveals the need for integral calculation. \((z <= y)\) compares the minimum error obtained in the related instance with the average error of defective region. If the error of the classifier is less than average error, the classifier is used for labeling \(U_i\). Otherwise, \(C_1, C_2, C_3\) calculates the number of labeling biases along with a straight bagging. According to this calculation, testing is completed.

Calculating \(S_i\) in Algorithm 1 is similar to that of triTraining. However, the criteria used while deciding the label of an instance are different. For instance, during examining the individual errors of the instances, novelEnsemble is interested in produced errors until mean error and related training instance are in use.

**Algorithm 1** novelEnsemble Algorithm

1: **Input** \(O\):Original labeled instance set,  
\(U\):Unlabeled instance set,  
\(C\):RandomForest, C5.0, NeuralNet
2: for each item \(i\) in \(C_i\) do  
3: \(S_i = BootStrap(O)\)  
4: \(C_i = Learn(S_i)\)  
5: \(E_i = 0.7\)
6: end for
7: for each item \(i\) in \(N\) do  
8: \(E_1 = C_1(U_i), E_2 = C_2(U_i), E_3 = C_3(U_i)\)
9: total ← total+\(E_i\)
10: av ← total/i
11: \(z ← \min(E_1, E_2, E_3)\)
12: \(y ← \int av + (total)^2 + c \ // av=mean error, x=i. total error, c=total defects/line of code\)
13: if \((z <= y)\)
14: \(U_i ← C_z\)
15: else \(U_i ← compute(C_1, C_2, C_3)\)
16: end for
5 Experiment Design

This section presents experimental data sets and research questions (RQ). Further, experimental settings and performance parameters are elaborated. Last, research questions are discussed.

5.1 Data sets

27 data sets have been used to evaluate novelEnsemble. They include the versions of following projects: ant [52], berek, camel, ckjm [53], e-learning, ivy, jedit, kalkulator, log4j, lucene, nieruchomosci, tomcat, and xalan [54]. These data sets have been obtained from tera-promise repository.

Some properties of experimental data sets are given in Table 1. These projects are open-source and coded with java programming language. An instance denotes either a software module or a class. The number of the instances in the projects is rather different. For instance, while tomcat has 858 instances, berek has 70 instances. Employing various instances in comparison projects alleviates the burden in evaluating the results of the experiments. Further, it ensures a clear bias about novelEnsemble. Table 2 gives used metrics with their definitions and types. There are 24 software metrics. 4 out of 24 metrics are of process metrics.

5.2 Performance Parameters

Performance parameters used in a defect prediction experiment change depending on the type of the problem. For example, if a method is developed for class imbalance, PF, g-mean, and AUC are frequently involved in the evaluation [55], [56]. It is well known that the works including HO have a great number of parameters. However, in recent works, MCC has become popular among researchers handling with defect prediction [57], [58].

In this study, AUC, MCC, and classification error have been used for evaluating novelEnsemble. Due to the main focus of the paper is ensemble learning methods, the scale of the performance parameters
optimization of ensemble learning . . .

Table 1. Details of the projects used in the experiment.

<table>
<thead>
<tr>
<th>Project</th>
<th>Version</th>
<th>Number of instances</th>
<th>Number of defects</th>
<th>Defectiveness (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ant</td>
<td>1.7</td>
<td>745</td>
<td>338</td>
<td>22</td>
</tr>
<tr>
<td>arc</td>
<td>1</td>
<td>234</td>
<td>33</td>
<td>11</td>
</tr>
<tr>
<td>berek</td>
<td>1</td>
<td>70</td>
<td>33</td>
<td>43</td>
</tr>
<tr>
<td>camel</td>
<td>1.0</td>
<td>339</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>camel</td>
<td>1.2</td>
<td>608</td>
<td>522</td>
<td>41</td>
</tr>
<tr>
<td>camel</td>
<td>1.4</td>
<td>872</td>
<td>335</td>
<td>16</td>
</tr>
<tr>
<td>camel</td>
<td>1.6</td>
<td>965</td>
<td>188</td>
<td>19</td>
</tr>
<tr>
<td>ckjm</td>
<td>1.8</td>
<td>10</td>
<td>23</td>
<td>50</td>
</tr>
<tr>
<td>e-learning</td>
<td>1</td>
<td>64</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>ivy</td>
<td>1.1</td>
<td>111</td>
<td>233</td>
<td>56</td>
</tr>
<tr>
<td>ivy</td>
<td>1.4</td>
<td>241</td>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>ivy</td>
<td>2.0</td>
<td>352</td>
<td>56</td>
<td>11</td>
</tr>
<tr>
<td>jedi</td>
<td>3.2</td>
<td>272</td>
<td>382</td>
<td>90</td>
</tr>
<tr>
<td>jedi</td>
<td>4.0</td>
<td>306</td>
<td>226</td>
<td>24</td>
</tr>
<tr>
<td>jedi</td>
<td>4.1</td>
<td>312</td>
<td>217</td>
<td>25</td>
</tr>
<tr>
<td>jedi</td>
<td>4.2</td>
<td>367</td>
<td>106</td>
<td>13</td>
</tr>
<tr>
<td>jedi</td>
<td>4.3</td>
<td>492</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>kalkulator</td>
<td>1</td>
<td>27</td>
<td>7</td>
<td>25</td>
</tr>
<tr>
<td>log4j</td>
<td>1.0</td>
<td>135</td>
<td>61</td>
<td>25</td>
</tr>
<tr>
<td>log4j</td>
<td>1.1</td>
<td>109</td>
<td>82</td>
<td>33</td>
</tr>
<tr>
<td>log4j</td>
<td>1.2</td>
<td>205</td>
<td>498</td>
<td>92</td>
</tr>
<tr>
<td>lucene</td>
<td>2.0</td>
<td>195</td>
<td>268</td>
<td>46</td>
</tr>
<tr>
<td>lucene</td>
<td>2.2</td>
<td>247</td>
<td>413</td>
<td>57</td>
</tr>
<tr>
<td>lucene</td>
<td>2.4</td>
<td>340</td>
<td>630</td>
<td>59</td>
</tr>
<tr>
<td>nieruchomosci</td>
<td>1</td>
<td>27</td>
<td>13</td>
<td>37</td>
</tr>
<tr>
<td>tomcat</td>
<td>6</td>
<td>858</td>
<td>114</td>
<td>8</td>
</tr>
<tr>
<td>xalan</td>
<td>2.4</td>
<td>723</td>
<td>155</td>
<td>15</td>
</tr>
</tbody>
</table>
Table 2. Metrics of experimental data sets.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>wmc</td>
<td>Weighted Methods per Class</td>
<td>Static code</td>
</tr>
<tr>
<td>dit</td>
<td>Depth of inheritance</td>
<td>Static code</td>
</tr>
<tr>
<td>noc</td>
<td>Number of children</td>
<td>Static code</td>
</tr>
<tr>
<td>cbo</td>
<td>Coupling between objects</td>
<td>Static code</td>
</tr>
<tr>
<td>rfc</td>
<td>Response for a class</td>
<td>Static code</td>
</tr>
<tr>
<td>lcom</td>
<td>Lack of cohesion</td>
<td>Static code</td>
</tr>
<tr>
<td>ca</td>
<td>Afferent coupling</td>
<td>Static code</td>
</tr>
<tr>
<td>ce</td>
<td>Efferent couplings</td>
<td>Static code</td>
</tr>
<tr>
<td>npm</td>
<td>Number of Public Methods</td>
<td>Static code</td>
</tr>
<tr>
<td>lcom3</td>
<td>A variant of lcom</td>
<td>Static code</td>
</tr>
<tr>
<td>loc</td>
<td>Line of codes</td>
<td>Static code</td>
</tr>
<tr>
<td>dam</td>
<td>Data Access Metric</td>
<td>Static code</td>
</tr>
<tr>
<td>moa</td>
<td>Measure of Aggregation</td>
<td>Static code</td>
</tr>
<tr>
<td>mfa</td>
<td>Measure of functionality abstraction</td>
<td>Static code</td>
</tr>
<tr>
<td>cam</td>
<td>Cohesion Among Methods of class</td>
<td>Static code</td>
</tr>
<tr>
<td>ic</td>
<td>Inheritance Coupling</td>
<td>Static code</td>
</tr>
<tr>
<td>cbm</td>
<td>Coupling between Methods</td>
<td>Static code</td>
</tr>
<tr>
<td>amc</td>
<td>Average Method Complexity</td>
<td>Static code</td>
</tr>
<tr>
<td>nr</td>
<td>Number of revisions</td>
<td>Process</td>
</tr>
<tr>
<td>ndc</td>
<td>Number of distinct commiters</td>
<td>Process</td>
</tr>
<tr>
<td>nml</td>
<td>Number of modified lines</td>
<td>Process</td>
</tr>
<tr>
<td>ndpv</td>
<td>Number of defects fixed in previous version</td>
<td>Process</td>
</tr>
<tr>
<td>max_ce</td>
<td>Maximum Class Coupling</td>
<td>Static code</td>
</tr>
<tr>
<td>avg_ce</td>
<td>Average Class Coupling</td>
<td>Static code</td>
</tr>
</tbody>
</table>

is not large. Moreover, to the best of our knowledge, this study is first to discuss ensemble learning in defect prediction along with HO. Therefore, the initiative is made rather than focusing on the detail of the results.

In Table 3, performance parameters used in the experiment are presented. MCC consists of confusion matrix members and AUC is the area under the ROC. In the formula indicating AUC, \( m \) are the data points and \( i \) refers to the execution number on \( m \) data points that denotes true label [59]. On the other hand, \( j \) refers to the execution time of \( n \) data points. The formula of AUC in each iteration produces 1 if \( p_i > p_j \). \( p \) denotes the probability value assigned by the classifier to
optimization of ensemble learning . . .

Table 3. Performance parameters used in the experiment.

<table>
<thead>
<tr>
<th>name</th>
<th>formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCC</td>
<td>( \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP) \times (TP+FN) \times (TN+FP) \times (TN+FN)}} )</td>
</tr>
<tr>
<td>AUC</td>
<td>( \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} 1_{p_i &gt; p_j} )</td>
</tr>
<tr>
<td>Classification Error ( E_i )</td>
<td>( \frac{f}{n} \times 100 )</td>
</tr>
</tbody>
</table>

the related data point. Classification Error is denoted with \( E_i \), where \( f \) is the number of incorrectly classified instances and \( n \) denotes the total number of instances in prediction.

5.3 Experimental conditions

Besides novelEnsemble, the results produced from three ensemble learning methods are discussed throughout the paper. This section explains how do four methods exploit experimental data and which configurations are optimal.

Figure 5. The schema of parameter search.
Parameter search is applied on three methods: novelEnsemble, ensembleHyper, and triTraining. Figure 5 presents a five-level schema. First level selects the data set to be involved in HO. In the second level, parameter bounds are determined. Parameters of second level are similar for three methods. Notwithstanding this similarity, type and the number of parameter change in accordance with the type of the classifier. In order to find suitable parameters, parameter search methods of third level are exposed to a process including 30 iterations. The values obtained from 10-fold cross-validation are returned in the fifth level.

Firstly, ensembleNoHyper is an ensemble learning method and it does not include any HO process. ensembleNoHyper is run by combining five classifiers with bagging approach. Five classifiers are: RegressionAdaBoostLearner, RegressionRandomForestLearner, RegressionSquareLossGradientBoostLearner, C5.0, and neural net. Configurations of RegressionAdaBoostLearner are Iterations=50, learning rate=1, minSplitSize=1, seed=42, maxTreeDepth=15. minSplitSize represents the value in a node of tree model that must exist before a split is performed. Learning rate affects convergent ratio to local minimal. maxTreeDepth limits depth of a tree model with the predefined value. subSampleRatio is used to overcome class imbalance problem when there is a good model to be trained. featuresPrSplit is the number of features to split in each node. Parameter settings of RegressionAdaBoostLearner are the same that of C5.0. Neural net uses only learningrate=0.1. The parameter settings of RandomForestLearner are trees=100, minSplit=1, maxTreeDepth=2000, featuresPrSplit=0, subSampleRatio=1, seed=42. RegressionSquareLossGradientBoostLearner employs iterations: 80, learningRate: 0.028, maximumTreeDepth: 12, subSampleRatio: 0.559, featuresPrSplit: 10. In ensembleNoHyper, any parameter search method is not used. Instead, it is exposed to a process including 200 iterations with bagging. In the meantime, a data set is divided into 70% training and 30% testing parts. Mean values of testing results of the prediction and error rates are recorded afterwards.

Specific parameter settings are used for each data set in ensemble-
Hyper. In order to find the optimal settings, firstly parameter search methods are determined. These are RandomSearch and GridSearch. RandomSearch [60] examines random combinations of a range of values of parameters. In the beginning, the number of iterations must be defined. Despite RandomSearch does not guarantee to find the best parameter settings, it is very fast in testing. On the other hand, GridSearch [61] aims to configure optimal parameters for a given model. The main advantage of GridSearch is that it is not limited to be applied to one model. Rather, GridSearch can be utilized for multi-model machine learning parameter search. Average values obtained from the search methods are applied for each data set in accordance with detected settings. Parameter search methods of RegressionAdaBoostLearner are harnessed in the range presented in Table 4. After determining suitable classification settings with 10 fold cross-validation, a process including 200 iterations is performed as in ensembleNoHyper. Average results are obtained with the same training and testing ratios.

In the experiment, triTraining includes RegressionRandomForestLearner, neural net, and C5.0. Some methods such as C5.0 and neural net are involved in the experiment due to their advantages in constructing ensemble methods [8]. To search a parameter, GridSearch has been used. As in preceding methods, triTraining is harnessed to produce average results obtained with 200 iterations.

novelEnsemble is an ensemble learning method that has not restriction on the number of classifiers. To construct novelEnsemble, three classifiers have been used. However, the number of the classifiers constructing novelEnsemble can be augmented. Initially, novelEnsemble selects training data via BootStrap to assign them to the related classifier. The ranges presented in Table 4 are of parameters of classifiers given in Algorithm 1. RandomSearch and GridSearch are utilized in searching parameters. To this end, test process including 200 iterations is performed after completing 10 fold cross-validation. Thereafter, performance values are recorded. The distinctive property of novelEnsemble is that it considers classification error along with the number of line of codes and the number of defects.
Table 4. Parameter bounds of experimental classifiers.

<table>
<thead>
<tr>
<th>RegressionAdaBoostLearner and C5.0</th>
<th>Min Max Transform Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterations</td>
<td>1 1000</td>
</tr>
<tr>
<td>learningrate</td>
<td>0 0.9</td>
</tr>
<tr>
<td>maxTreeDepth</td>
<td>1 15</td>
</tr>
<tr>
<td>minSplitSize</td>
<td>1 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RegressionRandomForestLearner</th>
<th>Min Max Transform Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterations</td>
<td>1 1000</td>
</tr>
<tr>
<td>splitsize</td>
<td>1 4</td>
</tr>
<tr>
<td>maxTreeDepth</td>
<td>1 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RegressionSquareLossGradientBoostLearner</th>
<th>Min Max Transform Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterations</td>
<td>1 1000</td>
</tr>
<tr>
<td>learningrate</td>
<td>0 1</td>
</tr>
<tr>
<td>maxTreeDepth</td>
<td>1 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NeuralNet</th>
<th>Min Max Transform Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterations</td>
<td>1 1000</td>
</tr>
<tr>
<td>learningrate</td>
<td>0 1</td>
</tr>
</tbody>
</table>

5.4 Research Questions

**RQ1:** How does novelEnsemble perform comparing with the alternatives in terms of classification error? Iteration number in ensemble learning approaches remarkably affects the success of the prediction. However, error rates are expected to be low. The main purpose of RQ1 is to compare error rates of novelEnsemble with other ensemble approaches. The comparison is done with using novelEnsemble in which parameters are optimized before executing classification process. In doing so, the reliability of ensemble learning approaches can be discussed depending on the testing iteration.

**RQ2:** Does HO create a favorable affect on ensemble learning meth-
ods? Despite the fact that HO has been investigated on various predictors, RQ2 emerges because any HO technique has not been tried on an ensemble learning method yet. RQ2 compares ensemble learning methods having optimized hyperparameters with the others which have no tuning process. In order to perform this, some performance parameters such as AUC and MCC are utilized.

RQ3: Has the experiment produced different results in terms of statistical values? RQ3 investigates whether the difference of experimental findings are remarkable. To this end, initially Welch’s Heteroscedastic F Test is applied on AUC and MCC results. Thereafter, to validate the difference of p-value, Cliff’s delta is performed on the performance results.

6 Results

6.1 Classification errors

Answer to RQ1: In order to analyze the relationship between competitive methods, four methods are considered: ensembleHyper, ensembleNoHyper, triTraining and novelEnsemble. Error performance values of three data sets including ant-1.7, camel-1.4 [62], and jedit-4.0 are illustrated in Figure 6-8. Mean error values change according to the iteration of testing process. Three data sets have different project structure and scales. For this reason, they are involved in the error observation so that the generality of the empirical analysis is reinforced. Favorable effects of HO have been detected, especially in novelEnsemble. The case is so different in the others. Unlike novelEnsemble, in which ensemble methods and hyperparameters are different, error values of the others are dramatically high. ensembleHyper has also a HO process that its error rates are noticeably low comparing with ensembleNoHyper. However these two methods are stagnant in three data sets. Thus, fluctuations of error rates are not much depending on the iteration. On the other hand, in such experiments, a linear improvement in error values is inherent. Although triTraining shows a linear improvement, it has the highest error rates.
Figure 6. Error rates of ensemble classifiers on ant-1.7.

Figure 7. Error rates of ensemble classifiers on camel-1.4.
It is clear that novelEnsemble behaves not like commonly reported classifiers in low iterations. For instance, in Figure 8, the success of novelEnsemble falls especially between 0-50 iterations. However, novelEnsemble seems to be the most preferable method. It is worthwhile to note that optimization is an extremely depended on constraints. Having the number of instances which is not greater than 1000 and limiting the number of iterations with 180 are shortcomings which seriously affect the reliability of the findings.

6.2 Effects of HO on ensemble learners

**Answer to RQ2:** novelEnsemble is used along with four methods to answer RQ2. In order to investigate advantages and disadvantages of HO, some methods having HO or not should be added to the experiment. Therefore, HO has not been applied on ensembleNoHyper. Mean values of AUC and MCC results are presented in Figure 9-10. The most scatted values are of ensembleNoHyper that has not any HO process.
Figure 9. Mean AUC results of four algorithms on all data sets.
Figure 10. Mean MCC results of four algorithms on all data sets.
It is clearly known from these figures that HO enforces the values to be in a specific interval. In the meantime, triTraining has the lowest mean of AUC values. It has been devised as compatible as working with error values of three classifiers. However, triTraining has not a special evaluation for some classifiers such as C5.0 and neural network which have produced promising results in recent years. Despite the fact that triTraining may produce unfavorable predictions, it is notable that triTraining is a pioneer learner in its field to encourage practitioners to improve future versions of it. Making modifications on the experimental design could achieve a general bias. The figures of the results show that novelEnsemble outperformed its alternatives in terms of the consistency of performance measures. On the other hand, triTraining has even yielded worse results than ensembleNoHyper that does not include any HO process. Although the performance of an ensemble learner seems to be depended on the selected classifiers and data sets, it is also of great importance to parameter search methods. In conclusion, RandomSearch and GridSearch are available in ensembleNoHyper but triTraining does not need such an operation. Further, it has been detected that ensemble learning methods developed based on classification errors such as triTraining cannot remedy the challenges encountered in constructing an ensemble learning method.

Detail AUC results given in Figure 11-14 help to examine the success of data sets individually. In these figures, data sets having high values vary in accordance with their properties. The values recorded in 0.5 – 0.6 are of triTaining and ensembleNoHyper. ensembleNovelHyper has an AUC value greater than 0.8. Compared to ensembleHyper, triTraining, and ensembleNoHyper, it is clearly seen that HO has increased AUC values at 0.2.

6.3 Statistical analyzes

**Answer to RQ3**: The results produced by ensemble learning methods differ according to iteration counts. Therefore, evaluating comparison methods for one aspect is difficult and misleading [63]–[65]. To solve this problem, the difference of AUC and MCC results is investigated.
optimization of ensemble learning . . .

Figure 11. AUC results of ensembleHyper on all data sets.
Figure 12. AUC results of novelEnsemble on all data sets.
optimization of ensemble learning ...

Figure 13. AUC results of ensembleNoHyper on all data sets.
Figure 14. AUC results of triTraining on all data sets.
The data sets exposed to prediction have completely normal distribution thanks to a normalization step of the experiment. However, unequal variances exist in the values of data sets. Welch’s Heteroscedastic F Test is suitable in case of unequal variances [66]. It has been applied on AUC and MCC values produced by comparison algorithms. As seen from Table 5, statistical findings are so different in two performance parameters.

Cliff’s Delta is selected during the examination of the difference magnitude of performance parameters. The reason is that the results have not any normal distribution. Thus, Cohen’s d, which is an alternative method to measure the magnitude of statistical difference, is vulnerable to normality violation. Table 6 presents Cliff’s Delta results recorded by benefiting "effsize" library available in R package. The magnitude of the difference is almost remarkable in overall matched methods. In particular, contrary to their structural similarity, delta estimate of novelEnsemble-triTraining is very large. However, the difference is negligible in ensembleHyper-ensembleNoHyper. The types and counts of hyperparameters may have led to the difference. Further, the compatibility level of default configurations could be incidentally high.

Table 5. Welch’s Heteroscedastic F Test of performance parameters. Differences are statistically significant for two parameters.

<table>
<thead>
<tr>
<th></th>
<th>MCC</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>statistic</td>
<td>111.308</td>
<td>341.4222</td>
</tr>
<tr>
<td>num df</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>denom df</td>
<td>56.25705</td>
<td>53.12816</td>
</tr>
<tr>
<td>p.value</td>
<td>1.23E-23</td>
<td>1.10E-34</td>
</tr>
</tbody>
</table>

7 Threats to Validity

This section discusses threats to validity for the experiment. Threats are related to data sets, theoretical background of the method, and the
Table 6. Cliff’s Delta results of comparison algorithms.

<table>
<thead>
<tr>
<th>MCC</th>
<th>delta estimate confidence interval</th>
<th>inf</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>ensembleHyper-novelEnsemble</td>
<td>-0.8024691 (large)</td>
<td>95 percent</td>
<td>-1.4786337</td>
</tr>
<tr>
<td>ensembleHyper-ensembleNoHyper</td>
<td>0.1234568 (negligible)</td>
<td>95 percent</td>
<td>-0.1945679, 0.4178945</td>
</tr>
<tr>
<td>novelEnsemble-triTraining</td>
<td>1 (large)</td>
<td>95 percent</td>
<td>0.9976115, 1.0000000</td>
</tr>
<tr>
<td>ensembleNoHyper-triTraining</td>
<td>0.877915 (large)</td>
<td>95 percent</td>
<td>0.6848872, 0.9558020</td>
</tr>
<tr>
<td>ensembleNoHyper-novelEnsemble</td>
<td>-0.8125685 (large)</td>
<td>95 percent</td>
<td>-1.4786337</td>
</tr>
<tr>
<td>ensembleNoHyper-triTraining</td>
<td>0.88294 (large)</td>
<td>95 percent</td>
<td>0.5885892, 0.8544020</td>
</tr>
</tbody>
</table>

Threats to internal validity refers to the features used in the experiment. Static code attributes of 27 data sets have been used in the experiment. Therefore, with respect to the ensemble learning, novelEnsemble has a mathematical model based on static code metrics. Having homogeneous metrics has alleviated the burden of experimental process. Similar number of the instances exists in different versions of the projects. It has been tried to select the versions employed in preceding works. The heterogeneity of metrics is one of the main challenges in defect prediction. Since the experiment uses the data sets having same metrics, there is not any need for performing metric matching. However, a replication of this study on heterogeneous software projects would be desirable.

Threats to external validity relates to the data sets which entirely consist of open-source ones. Thus, any industrial data set has not been considered during the experiment. To remove this threat, various versions of data sets of a software project are analyzed.

Threats to construct validity is related to the classifiers utilized in constructing ensemble learners and performance measures. Choosing the classifiers used in the construction of an ensemble learning method is of great importance. Therefore, establishing an ensemble learner via unsuccessful prediction models is not reliable so that superior predictors
such as C5.0 and neural network have been preferred. If classification error is considered, examining error trends in small-scale could create an evaluation threat. To solve this problem, 200 iterations have been performed in testing process. To evaluate the results, AUC, classification error, and MCC have been used. Besides them, g-mean and f-measure are well known among practitioners who work with imbalanced data sets. For this reason, a small proportion of those is not involved in recording performance values.

**Threats to conclusion validity** refers to the treatment and the outcome. To evaluate the consistency of the results, Welch’s Heteroscedastic F Test and Cliff’s Delta have been conducted. While choosing statistical methods, it has been regarded whether the data have normal distribution.

# 8 Conclusion and Future Remarks

HO is a common performance improvement method in machine learning. In this work, the effects of applying HO on ensemble learners have been observed. Further, a novel ensemble learner namely novelEnsemble has been proposed. In preceding works, it has not been investigated how HO is applied on ensemble learning methods. Further, the success of an ensemble learner employing default configurations in defect prediction was not comprehensively examined. In this context, the findings of this study have been obtained from an experimental design solving some issues of defect prediction when ensemble learners are used. novelEnsemble outperformed the others in terms of AUC and MCC. The main findings of the paper can be summarized as follows:

1. novelEnsemble demonstrated a clear superiority to its closest alternative in AUC with 86%. The difference is almost 9%. It rises to 30% when novelEnsemble is compared with triTraining.

2. Some classifiers such as C5.0 and neural network are much more variable than previously thought. Thus, ensembleHyper, which includes some predictors except C5.0 and neural network, shows worse performance than novelEnsemble. Therefore, predictors used in constructing an ensemble learner should be changed according to data sets
and experimental settings.

3. The success of an ensemble learner may not change depending on parameter search method, used ensemble approach, and selected hyperparameters. For instance, ensembleHyper and ensembleNoHyper produced similar results in AUC and MCC values. Moreover, the difference of these values is negligible in performed statistical analyzes. In such cases, the experiment could be replicated by changing hyperparameters and the predictors constructing ensemble learners. The most ineffective configurations can be revealed by that way.

4. In an ensemble learner, besides classification error, other metrics indicating defectiveness should be considered. Thus, novelEnsemble decides the label of an instance by comparing average number of defects with classification error. triTraining has been found to be the most unsuitable ensemble learner among all methods. Because, it has an experimental design based solely on classification error and restricts the number of predictors with three.

The future agenda of the paper encompasses the following: 1) novelEnsemble used `BootStrap` to select training instances as in triTraining. Instead, it is planned to develop a method utilizing some supervised techniques such as clustering by taking data scale into account; 2) Although novelEnsemble is not dependent on the number of classifiers, C5.0 and neural network have been involved in the experiment because they yielded promising results in preceding works. However, it is still unclear to what extent parameter search can contribute to HO in ensemble learners. It would be interesting to investigate that topic; 3) In the experiment, 27 data sets have been used. Yet another direction is to validate the findings of the paper with industrial data sets.

References


optimization of ensemble learning . . .


[34] A. S. Sayyad, K. Goseva-Popstojanova, T. Menzies, and H. Ammar, “On parameter tuning in search based software engineering:


M. Maruf Öztürk


M. Maruf Öztürk

Received December 7, 2018

Department of
Computer Engineering
Faculty of Engineering
Isparta, TURKEY

Phone: +90 246 211 15 63
E-mail: muhammedozturk@sdu.edu.tr, maruf215@gmail.com