# Improving Machine Learning-based Code Smell Detection via Hyper-parameter Optimization

Lei Shen[†], Wangshu Liu[†§*] , Xiang Chen[‡§], Qing Gu[§], Xuejun Liu[†]
[†]*School of Computer Science and Technology*, *Nanjing Tech University*, Nanjing, China
[‡]*School of Information Science and Technology*, *Nantong University*, Nantong, China
[§]*State Key Laboratory for Novel Software Technology*, *Nanjing University*, Nanjing, China

***RQ1: Compared to the default values, whether using hyperparameter optimization can significantly improve code smell detection performance when considering different classifiers?***

To answer RQ1, we first show the code smell detection performance of four optimizers and Default baseline on six classifiers in Table III and V for two code smells.

TABLE III
THE PERFORMANCE OF USING FOUR OPTIMIZERS AND DEFAULT VALUE IN TERMS OF AUC FOR CODE SMELL *Data Class*

| Classifier | Average AUC Value on Eight Projects | | | | |
|---|---|---|---|---|---|
| | Bayes | DE | Random | Grid | Default |
| CT | .911272 | .892349 | .910450 | **.915507** | .836185 |
| KNN | .913078 | .898707 | .910832 | **.915049** | .859658 |
| NB | .832012 | .832224 | .832216 | **.832229** | .828786 |
| RF | .947471 | **.951487** | .945920 | .945818 | .921153 |
| RULE | .931293 | **.936406** | .927868 | .930622 | .920735 |
| SVM | .913641 | .906120 | .913548 | **.916264** | .891811 |

TABLE V
THE PERFORMANCE OF USING FOUR OPTIMIZERS AND DEFAULT VALUE IN TERMS OF AUC FOR CODE SMELL *Feature Envy*

| Classifier | Average AUC Value on Eight Projects | | | | |
|---|---|---|---|---|---|
| | Bayes | DE | Random | Grid | Default |
| CT | .922133 | .872815 | .922595 | **.923075** | .866148 |
| KNN | .878776 | .868533 | .878765 | **.878964** | .827597 |
| NB | .808290 | .808293 | **.808294** | **.808294** | .807845 |
| RF | .946995 | **.947967** | .946763 | .946685 | .935155 |
| RULE | .930075 | .924472 | .918116 | .929318 | **.930118** |
| SVM | .911073 | .909786 | .910976 | **.911288** | .898747 |

Then we conduct a significance test (i.e., Cohen's *d* effect size) between hyperparameter optimization and default setting in Table IV and VI.

TABLE IV
THE COHEN'S *d* EFFECT SIZE FOR CODE SMELL *Data Class* BETWEEN DEFAULT VALUE AND FOUR OPTIMIZERS FOR EACH CLASSIFIER

| Classifier | Cohen's *d* Effect Size | | | |
|---|---|---|---|---|
| | Bayes | DE | Random | Grid |
| CT | **1.8295** | **1.5587** | **1.8073** | **2.0380** |
| KNN | **1.8207** | **1.3964** | **1.7319** | **1.8958** |
| NB | 0.0644 | 0.0688 | 0.0687 | 0.0689 |
| RF | **1.1726** | **1.4051** | **1.1124** | **1.1147** |
| RULE | *0.4844* | 0.6950 | *0.3178* | *0.4494* |
| SVM | **0.8327** | 0.5204 | **0.8325** | **0.9491** |

TABLE VI
THE COHEN'S *d* EFFECT SIZE FOR *Feature Envy* BETWEEN DEFAULT VALUE AND FOUR OPTIMIZERS FOR EACH CLASSIFIER

| Classifier | Cohen's *d* Effect Size | | | |
|---|---|---|---|---|
| | Bayes | DE | Random | Grid |
| CT | **2.2277** | *0.2381* | **2.2511** | **2.2700** |
| KNN | **1.3995** | **1.0810** | **1.4009** | **1.4103** |
| NB | 0.0134 | 0.0135 | 0.0135 | 0.0135 |
| RF | 0.6277 | 0.6790 | 0.6153 | 0.6164 |
| RULE | 0.0027 | *0.3524* | 0.7639 | 0.0501 |
| SVM | 0.6003 | 0.5399 | 0.5934 | 0.6070 |

All the results are the mean values among eight projects, and the experiments are conducted in the finer-grained code smell sequence of *Data Class* and *Feature Envy*.

**Summary for RQ1**: Applied machine learning techniques based on hyper-parameter optimization for code smell detection, we find that results are superior to default values, and RF can achieve better performance. Thus, we suggest that hyper parameter optimization is conducive to code smell detection.

***RQ2: When the specified classifier Random Forest (RF) is given, is there a hyper-parameter optimization method that has superior performance?***

To answer RQ2, we present with the performance of four optimizers on eight projects by drawing the bar chart in Figure 4 and 5, where *y*-axis represents the detection performance (AUC).

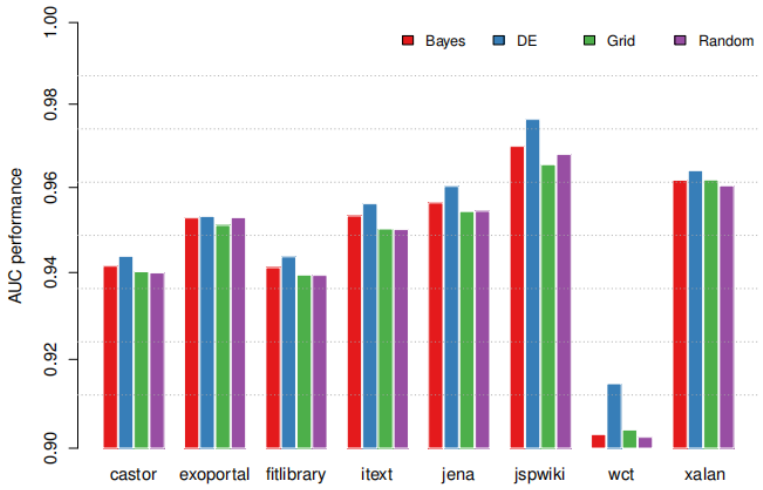



Fig. 4. The AUC performance for *Data Class* detection of different optimizers when the classifier is RF.

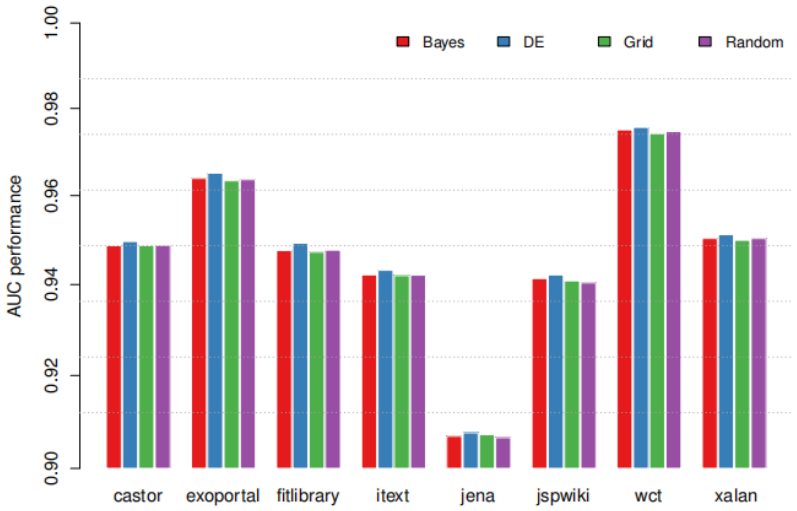



Fig. 5. The AUC performance for *Feature Envy* detection of different optimizers when the classifier is RF.

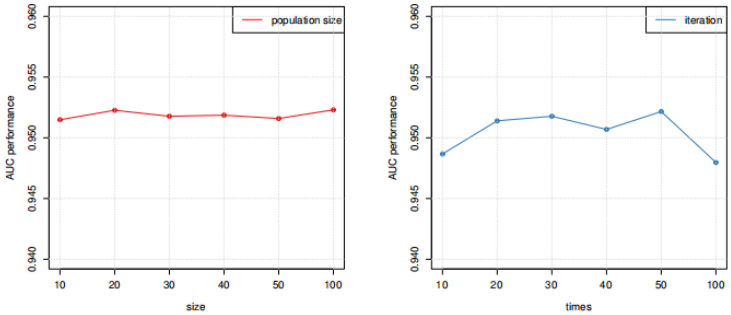
Notice that due to promising code smell detection performance of Random Forest (RF), the results of RQ2 are based on the classifier RF. For all the optimizers, we set the iteration number to 30 for a fair comparison.

From Figure 4 and 5, we find DE is always the best in a total of eight projects followed by Bayes, while the performance of Grid looks similar to Random, sometimes even worse than Random.

**Summary for RQ2**: After using the heuristic search algorithm, DE-based RF outperforms other models. We also perceived that the data with a lower proportion of code smells is prone to perform better. This finding suggests that using Random Forest based on differential evolution to detect code smell and prioritizing projects with a low code smell ratio are good choices.

***RQ3: Applying the parameter optimization to differential evolution (DE) optimizer, could our machine learning-based method achieve a better performance?***

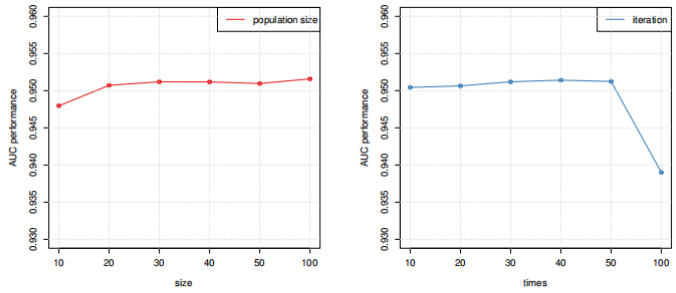
To answer RQ3, we mainly inspect two impact factors, the number of populations (*pop*) and the number of iterations(*iter*), of differential evolution, which has been confirmed as the most effective method in RQ2. Then we adopt the variable controlling approach to analyze the influence of these factors. To be consistent with the previous results, we repeatedly fix one variable as 30 and study the other variable's variance at different values. Figure 6 and 7 depict the line chart to present the empirical observations, where *y*-axis provides the detection performance (AUC).




(a) the size of populations (*pop*) (b) the number of iterations (*iter*)

Fig. 6. The effect of impact factors on DE optimizer under Random Forest for detecting *Data Class*. The left subfigure indicates that the iteration is set to 30 while the population size varies from 10 to 100. The right subfigure indicates that the population size is fixed with 30 while the iteration varies from 10 to 100.

For Figure 6, we first consider the impact of factor *pop*. As the factor *pop* increases, the change of detection performance is quite limited, and the AUC values are mainly distributed between 0.951 and 0.952.




(a) the size of populations (*pop*) (b) the number of iterations (*iter*)

Fig. 7. The effect of impact factors on DE optimizer under Random Forest for detecting *Feature Envy*. The left subfigure indicates that the iteration is set to 30 while the population size varies from 10 to 100. The right subfigure indicates that the population size is fixed with 30 while the iteration varies from 10 to 100.

For Figure 7, we first also consider the impact of factor *pop*. Similar to *Data Class*, when the factor *pop* increases, the AUC values are slowly varied from 0.948 to 0.952.Except at 10 of *pop*, the performance fluctuation of other *pop* settings can be negligible.

**Summary for RQ3**: After considering impact factors *pop* and *iter* of DE, we find that parameter optimization on DE optimizer can further improve the performance of Random Forest by increasing population size. However, without a limited scope, tuning the number of factor *iter* may reduce the performance. The conclusion of studies that encourage other researchers to narrow their *iter* and *pop* of DE when applying hyper-parameter optimization for code smell detection.