An Exploratory Study of Search Based Training Data Selection For Cross Project Defect Prediction

Seyedrebvar Hosseini
M3S, Faculty of IT and Electrical Engineering
University of Oulu, 90014, Oulu, Finland
Email: rebvar@oulu.fi

Burak Turhan
Department of Computer Science
Brunel University, London UK
Email: burak.turhan@brunel.ac.uk

Abstract—**Context: Search based approaches are gaining attention in cross project defect prediction (CPDP). The complexity of such approaches and existence of various design decisions are important issues to consider. Objective: We aim at investigating factors that can affect the performance of search based selection (SBS) approaches. We study a genetic instance selection approach (GIS) and present an evaluation of design options for search based CPDP. Method: Using an exploratory approach, data from different options of models are gathered and analyzed through ANOVA tests and effect sizes. Results: Both feature sets and validation dataset selection options show small or insignificant impacts on F-measure and precision, unlike the more affected false positive and true negative rates. Size of training data does not seem to be related to significant changes in F-measure and precision and high variability in performance are discouraging evidence for using larger datasets. Fitness function is one of the major factors that impact performance with much larger effect than the choice of validation dataset. Finally, while showing slight impacts, data label changes do not seem to be the top contributor to performance. Conclusions: We conclude that exploratory approaches can be effective for making design decisions in constructing search based CPDP models. Effect of individual tuned learners and their interaction with other affecting parameters and more in depth study of quality affecting factors guided by label changes are directions to investigate.**

Keywords-**Cross Project Defect Prediction, Exploratory Search Based Optimization, Training Data Selection**

I. Introduction

Applying defect prediction in practice has been a challenge. The lack of and the difficulty of collecting and organising defect related data is one of the reasons why companies do not consider using defect prediction in practice [1] and it is usually limited to research studies [2]. Cross Project Defect Prediction (CPDP) comes to the rescue in such circumstances as it is an affordable solution for companies which look for minimal effort of data collection. Additionally, the use of CPDP is justified, since the change of practices in software development over time affects the relevance of collected local data, because the existing practices might not be representative anymore [3], [2]. On the other hand, considering the key premise of CPDP, i.e. learning and applying, from and to, different sets of projects [4], [1], in presence of relevant data from other projects including open source ecosystems, CPDP can result in practical applications as even a tiny decrease in the bug rates can lead to significant financial savings in terms of quality assurance costs [1], as opposed to exponential growth in repair costs and damages [1], [5] as a result of failure to discover bugs in a timely manner.

Despite having numerous models in within and cross project settings, there is little known about the confounding factors which impact the model construction processes especially when search based approaches are used which usually have a larger number of design options. We demonstrated the impact of such a model, i.e. search based data selection with and without feature selection in our previous studies [6], [7]. The results of these studies showed clear improvements over other cross project benchmarks and comparability to the within project benchmarks.

Through an exploratory approach, this study aims at investigating the impact of a set of identified factors, including validation data, training data, value/fitness function, search operators and feature selection. We aim at answering the following research question in this paper:
RQ: In what way do the identified model design factors and their interactions affect the performance of the search based approach?

While we use our previously proposed method, i.e. GIS, in this work, the model can be extended to any other search based method. Therefore, this study not only provides detailed investigation of contributing factors on performance of GIS, but also, provides a means for evaluation and efficient design selection for similar models with various options before making model related decisions.

II. Related Work

Studies in the area of cross project defect prediction, do not always agree in their conclusions. The systematic literature review by Hosseini et al [2], presents detailed discussions of the state of the art CPDP approaches with a specific focus on data approaches used in the literature. We describe the most relevant studies to our work next.

Instance and dataset selection methods have been explored in CPDP. These include relevancy filtering by Turhan et al.[1] based on the euclidean distance measure, data distributional characteristics and meta-learners by He et al. [8], clustering by Herbold [9] and selective learning by Ryu et al. [10]. These studies however, do not consider a search based approach.

Examples of search based studies in CPDP, include generating evolving mathematical equations to fit the data [11], multiobjective cost-effectiveness logistic regression parameter optimization using NSGA-II [12] and logistic regression parameter optimization using small proportions of within project data [13], all of which involve learning techniques and optimizing them, and not the instance/data selection problem. Further, none of the aforementioned studies target data quality.

Instance selection and data quality are investigated by our previous studies [6], [7], the details of which are presented in the next section and the current study delves into the details of the search process. The findings, in turn, can potentially provide insights on how to make better decisions regarding design selection for search based approaches.

III. Research Methodology

A. Motivation

Availability of many open source defect prediction datasets may lead to prediction models that may be able to discover sources of error in code and decrease the costs and amount of required resources. Attempts at building such prediction models have revealed critical challenges (data heterogeneity, class imbalance, representativeness, etc.) which threaten the value of the available data. While the available data is valuable, not all instances can help in the process of building effective prediction models, major reasons of which are the dataset shift problem [14] and changes in practices [3], [2]. The data filtering and instance selection approaches are proposed to help fulfill the desired outcome. GIS [6], [7], is one such approach which tries to produce evolving training datasets through a search based approach (genetic algorithm in this case). Such search based approaches however, usually involve different parameters, which can affect their performance and careful analysis can be crucial to optimize their performances.

B. Search based instance selection

This section presents a brief description of the search based method which is used in this study. Knowledge of the model helps to identify the design factors which are presented afterwards. We will refer to the Genetic Instance Selection (GIS) proposed in our previous studies [6], [7] as Search Based Selection (SBS) in the rest of the paper. To make predictions for a test set, SBS starts by generating random subsets of instances belonging to the pool of instances from other project. A validation dataset is generated, initial training datasets from the first population are evaluated on the validation dataset, and fitness values are calculated. Using the genetic operations, e.g. mutation and crossover, new generations are created and best datasets survive and are transferred to next generations. The original GIS approach used fixed length training datasets, NN-Filter as validation dataset generation method and Naive Bayes (NB) as learner. The mutation and cross over operations are designed to address the data quality and dataset shift problem to some extent through random label changes in training data, accounting for the potential noise in the data.

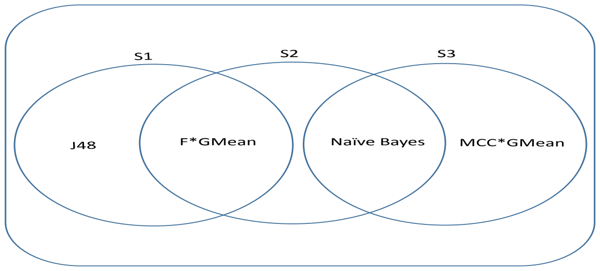

Fig. 1: Experiment Scenarios

C. Experiment Design

Two learners (NB, J48), where NB: Naive Bayes and J48: Decision Tree; three validation dataset selection methods (VNN, VAT, VRnd), where VNN: Validation set generated with NN-filter, VAT: All training data as validation set, and VRnd: Random validation set; two training dataset generation methods (FIXED, VAR) methods, where FIXED and VAR corresponds to fixed and variable size training datasets; and two fitness functions (F$\times$GMean and GMean$\times$MCC) are used in the experiments. These options make three scenarios, depicted in Figure 1. Scenario 1 (S1) uses J48 as learner and F$\times$GMean as fitness function. Scenarios 2 and 3 (S2 and S3) use NB as learner, but their fitness functions are different, namely F$\times$GMean and GMean$\times$MCC, respectively.

The presence of randomness is addressed by repeating the experiments (20 times for each dataset) and having multiple scenarios for different options. The population size is 30 and the maximum number of generations is 30. The reason for selecting these values is due to having small population sizes which, despite the small sizes, cover almost all of the original training instances in every iteration (95%) in the initial populations. Further, the process converges quickly justifying the selection of the maximum number of generations.

We use an exploratory approach during data gathering and analyses. We collect dataset information for validation datasets, final training datasets, survived and deleted population members information (class counts, label changes, validation and test performances), test datasets information, as well as the evolution of each instance in each dataset (count of mutation, label changes).

ANOVA tests are performed to determine the importance of the factors and their interactions. Normality assumption of the residuals from ANOVA tests were checked using statistical tests which rejected the normality in most cases. Therefore, we used two steps to address this problem. First, we used the Box-Cox, logarithm and square root transformations on the raw data, yet the normality assumption was still violated. Then, a visual inspection of QQ-plots revealed that the distribution of the residuals is deviating from normal distribution only in the tails. However, the number of data items collected for each scenario are reasonably high, exactly 3,120 data points, and the implications of the central limit theorem as well as the close approximation of normally distributed data except for the tails justify our use of ANOVA for analysis.

In terms of further notation, $Y - X$ refers to validation set selection method Y(VNN, VAT, VRnd) and feature set X(SBS_A, SBS_IG), where subscript A and IG correspond to using all features and feature selection with InfoGain, respectively. Further, $VAR.Y - X$ refers to variable size training dataset selection with validation set selection method Y (VNN, VAT, VRnd) and Method X(SBS_A, SBS_IG)

D. Discussion of search based design options

A set of design options for search based CPDP are identified and discussed in this section.

1) Validation dataset selection: Does the choice of validation dataset selection matter for SBS? To see the value of good validation datasets, consider the approximate **theoretical upper bound** based on selecting perfect validation data (VTST = test set as validation data) which leads to selecting instances with very high prediction power (for S1 the performances are VTST-SBS_A = {F: 0.643, rec: 0.775, prec: 0.577}, VTST-SBS_IG = {F: 0.659, rec: 0.782, prec: 0.585}, VAT.VTST-SBS_A = {F: 0.638, rec: 0.806, prec: 0.559}, VAR.VTST-SBS_IG = {F: 0.654, rec: 0.810, prec: 0.597}). The next important question is the representativeness of NN-Filter as validation dataset selection method which was used in GIS. It is not clear whether the choice of NN-Filter is one of the top contributing factors to better performances or not. We investigate this question by incorporating two alternative validation dataset selection methods in SBS, i.e. using the entire training dataset (VAT) and using random subset of all training instances (VRnd) as validation data.

2) Learning techniques: The NB learner, used in our previous studies, showed acceptable performances considering its simplicity and speed. While previous studies have demonstrated the potentials of NB learner, the correctness of its assumptions and its potentially lower performances in comparison with more sophisticated learners has raised concerns regarding its performance [2]. We address this concern, by repeating the experiments using J48 classifier in WEKA. Such an experiment, would shed light on the dependence on learning techniques, or lack-thereof.

3) Fitness function: The fitness function can affect the path which the search process takes for optimization. Selecting a good fitness function is crucial for converging to the ideal goals. We used F$\times$GMean in our previous studies as fitness function. While the reason for selecting such a fitness function was the desire for a balanced performance of precision and recall, we observed more recall at the cost of losing more precision. While neglected in many studies, True Negatives can be important as making mistakes in detecting them increases the probability of false alarm which could lead to less confidence in the applicability of the models [15]. The MCC compound measure seemed like a good candidate as it includes all four components of the confusion matrix. However, MCC has its drawbacks since for it, TPs and TNs have the same importance which might not be desirable for researchers/industry.

4) Training data and class imbalance: The investigation into the imbalance space is interesting, since it is believed that the imbalance problem is one of the root causes for the weak performance of CPDP [2], [16], [17], [18], [19], [9], [20]. In our previous studies [6], [7], we used a fixed length chromosome structure containing a constant number of instances, selected randomly from other projects. This method of dataset generation however, limits the proportion of the space covered by the generated datasets. Our previous search operations did not change the size of the datasets (one point cross over), therefore, new offsprings remained on the same point/line. Please note that when varying size datasets are generated, it is possible that they become very large which could possibly contain the entire training dataset (pool) or even more (instance repetition). While we use variable dataset sizes, we made some limitations on them for practical considerations by modifying the cross over operation to use middle point, if their size exceeds a constant higher than half the size of all available training instances (4,000 in this case).

5) Mutation and cross over patterns: The majority of the datasets for CPDP and defect prediction in general are generated through automated heuristic approaches such as SZZ [21]. However, SZZ and its extensions are not entirely accurate [2]. This implies that the labels for the defective and non defective instances could be noisy or change over time while new bugs are introduced or discovered. We aim at finding the effect of using the search operations that change data labels and their relation to the changes in performance.

E. Datasets and Metrics

We used 13 projects from the PROMISE repository which were used in [6]. These datasets are Ant-1.7, Camel-1.6, Ivy-2.0,JEdit-4.3, Log4j-1,2, Lucene-2.4, Poi-3.0, Prop-6.0, Synapse-1.2, Tomcat-6.0, Velocity-1.6, Xalan-2.7 and Xerces-1.4. Each dataset contains a number of instances corresponding to the classes in the release. Each instance has 20 static code metrics including object oriented, size and complexity metrics. The details of the datasets are available in [22]. We use the full set of metrics in SBS_A and the subset selected by iterative infogain subsetting in SBS_IG.

F. Performance Measures and Tools

To assess the performance of the models, a diverse set of performance indicators are used: Precision, Recall, F-Measure, GMean, False Negative Rate (FNR), True Negative Rate (TNR), Probability of False Alarms (PF) and Matthews Correlation Coefficient (MCC). Selection of a broad range of performance measures not only demonstrate different aspects of the approach, but also is useful for comparison purposes for future studies [2], [23]. All the experiments are conducted using **WEKA** machine Learning tool version 3.8.1. The related WEKA classes for instance manipulation are modified for the data collection process. Statistical tests are carried out using **scipy**, **statsmodels**, and **Python** version 3.6.3. The plots are generated using **matplotlib** and **seaborn** libraries. The scripts as well as the modified WEKA version are available online in our replication package [24].

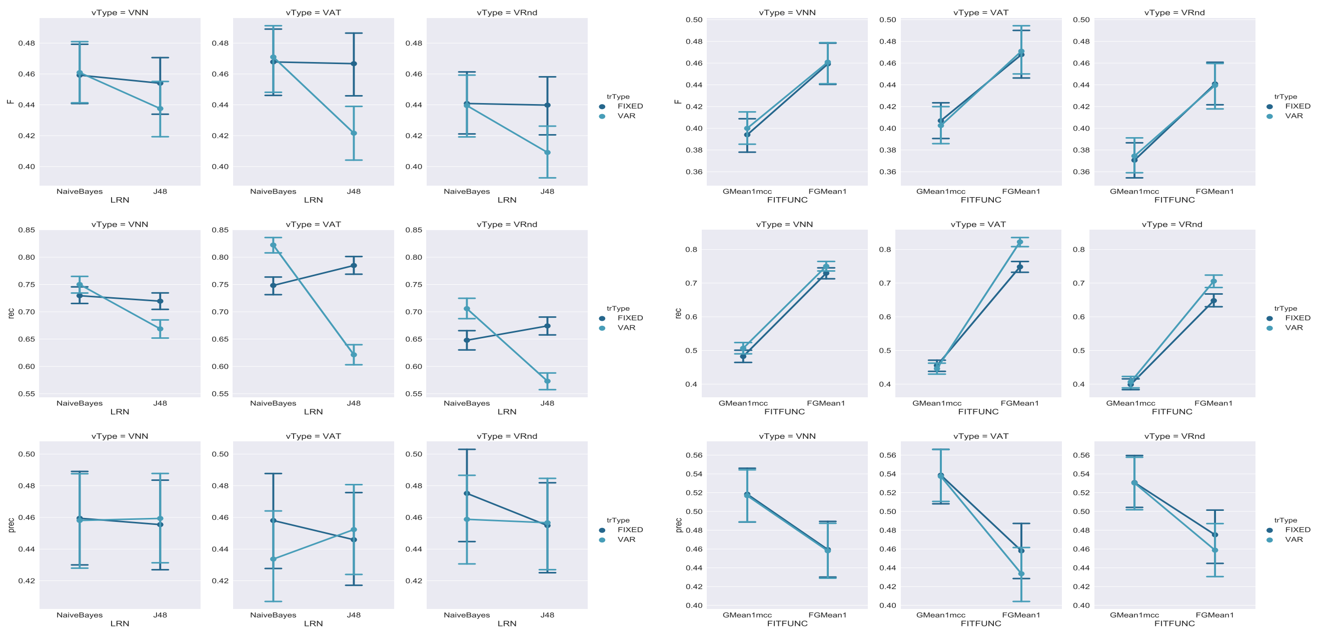

Fig. 2: Factor Plots for the relationship of learning technique (LRN, left side) or Fitness Function (FITFUNC, right side), with validation dataset type (vType), and training dataset type (trType) based on F-measure, recall and precision

IV. Results

A. Impacting factors on performance

We start the discussion of each impacting factor with presenting the results of the ANOVA test for that factor. The interaction details are presented if a multi-way ANOVA is performed. These results (p-values and effect sizes) are used to measure the effectiveness and impact of each factor/option. Figure 2 will act as a demonstration for the following sections.

1) Validation dataset selection and features: We perform two way ANOVA test which considers the validation dataset and feature set parameters and their interaction. These two parameters are considered for a two way ANOVA as they both are responsible for changing the shape/size of the data (smaller or larger with validation data and smaller with feature selection). In S1, the results of the two way test for F-measure shows a non significant interaction of the two variables ($p-val$ =0.920), but significant with tiny effect sizes for the individual parameters (vType = [$p-val$ =0.045, ω^2=0.001], features=[$p-val = \ll 0.001$, ω^2=0.005]). The effect size for feature sets is slightly higher in S1. Significant differences in terms of precision, are not observed for the variables and their interaction.

Unlike S1, the tests for F-measure in S2 show a significant difference with validation datasets (ω^2 =0.020), but insignificant for features ($p-val$ =0.836) and the interaction of the two ($p-val$ =0.712). Similar to S1, the tests for precision show insignificant results for the variables and their interaction ($p-val = \{vType = 0.353, features = 0.192, interaction = 0.816\}$). In S3, significant differences are observed for the interaction. For the variables however, significant differences are present with effect size for validation dataset selection (0.005) being slightly larger than that of feature sets(0.001). Precision is not affected significantly in this case as well.

For feature sets, while a significant p-value can be observed in many cases, the effect sizes are very small. Further, the choice of feature set does not have a significant impact on precision. In terms of median F-measure, in S1, VNN-SBS$_{\text{IG}}$ and VAT-SBS$_{\text{IG}}$ are the first and second best performing approaches among the benchmarks. The performance, of the VAT-SBS$_{\text{IG}}$ however, is based on very high recall values. This comes at the cost of having the lowest precision values. The *-IG* methods, have higher recall values in comparison with their *-A* counterparts. Also, the performance of the approaches that have larger validation dataset sizes (VAT, VNN, VRnd in decreasing size order), in their feature groups (*-A* and *-IG*) are more recall based.

2) Training datasets: Most of the variable size training datasets have lower performances in comparison with their fixed size counterparts as depicted in Figure 2. This behaviour seems to be more recall related than precision. The case of VAR.VNN-SBS$_{\text{IG}}$ is an exception as it is the third best performing approach based on median F-measure and has the highest median precision value. While the variable size training datasets, have the highest FNRs, they have the highest median TNR rates as well and less false alarms consequently. ANOVA tests are performed to assess the extent of impact of the training dataset generation strategies. For F-measure, S1 shows significant impact with tiny effect size (ω^2 =0.005), but significant differences can not be seen in S2 ($p-val$ =0.891) and S3 ($p-val$ =0.792). In all three scenarios, precision results do not show significant changes for using one training dataset generation method over the other.

3) Fitness Function: The data from S1 and S2 are combined and used in ANOVA test to investigate the existence of significant differences or lack thereof with regard to the fitness function. The one-way ANOVA test show significant increase or decrease based on different measures. Regarding F-measure, a significant difference toward the options of S2 can be observed (ω^2 =0.023) as is the case also for GMean (ω^2 =0.041). A group of important differences are observed for other measures. In terms of FNR, S2 is better than S3 with observed effect size ω^2 =0.339. Conversely, S3 results in higher rate of true negatives with ω^2 =0.407. S3 makes significantly less false positive mistakes, according to the observed effect size ω^2 =0.407. This comes at the cost of losing recall which is better in S2 with ω^2 =0.339. This drop in recall in not compensated by precision, despite being higher in S3 (ω^2 =0.011), but rather in terms of higher TNR and lower PF. This was expected to some extent for S3, as the inclusion of MCC in its fitness function, gives high weight to true negatives, while F-measure and GMean, ignore it. These differences are depicted in the right side of Figure 2 for F-measure, precision and recall.

Earlier, we considered the impact of validation dataset selection in isolation for each scenario. We can now, consider the interaction between validation dataset selection and fitness function and assess their order of impact. We perform a two way ANOVA for this purpose. The results of the tests for different measures show that the effect of fitness function in comparison with validation datasets selection is comparably very larger (effect sizes 15-100+ times larger). Also, in terms of F-measure, the effects of validation dataset itself and the interaction between it and fitness function are not significant. This is true for GMean and precision as well. This is not to say that the validation dataset is not important, but rather shows that better validation selection methods are required.

4) Learning techniques: Similar to the discussion for fitness function, We have explored the effects of two learning techniques by performing the experiments of S1 and S2 scenarios. The data from both of these scenarios are used to perform ANOVA test to detect possible important sources of impact. ANOVA test result reveals that while the selection of learning technique can potentially impact the performance, the impact might be very small, in terms of some of the measures. Particularly, with F-measure, despite seeing a significant difference ($p-val$ =0.001) a very small effect size is observed (ω^2 =0.001). With recall, the observed effect size is very small as well (ω^2 =0.022), while no significant difference is observed for precision ($p-val$ =0.713). As with fitness function, two way ANOVA tests for validation dataset selection and learning technique are performed. In this case, there are tiny differences in terms of effect size between the two for F-measure ($\omega^2 = \{learn{-}tech : 0.001, vset{-}type : 0.002\}$ and insignificant interaction effect) and precision (all insignificant).

B. Class imbalance space coverage

Figures 3 and 4 show the F-measure values corresponding to all the surviving datasets generated through the search process for scenario 1. All of the generated datasets are evaluated on the test set and their performance is recorded. The plots shown in Figure 3 show the performances for fixed size generated datasets and the plots in Figure 4 represent the performance for the variable size counterparts. Each plot is divided into six parts according to the validation methods (VNN, VAT, VRnd) and feature sets (-A,-IG). The two lines in each plot, represent median performance for each class label. The red line illustrates the results based on increase in number of instances belonging to class 1=defective and the green line is for 0=non-defective class. We report average ± standard deviation for fixed size datasets and medians for variable size datasets.

For the fixed size datasets shown in Figure 3, increase in the number of instances in each class corresponds to decrease in the number of instances in the other class. Hence, the curves must be interpreted in the following way. Starting with VNN-SBS$_{\mathrm{A}}$ and VNN-SBS$_{\mathrm{IG}}$, an increase in the number of items belonging to class 1, seems to be related to decrease in performance. At the same time, an increase in the rate of class 0 instances, shows better average F-measure performances. While this is the case for *VNN-*, the pattern illustrates lower performances for small percentages of class 1 and very high percentages of class 0 instances as well which is similar to the overall number of class 1 and class 0 instances that are present in the pool of all training data. The only exception is VAT-SBS$_{\mathrm{IG}}$, which shows steady average performance with increases in class 1. VNN-SBS and VRnd-SBS show similar behaviours. One can see very similar standard deviations with increases in each class except for extreme cases like the very good performance seen for high class 1 in VRnd-SBS$_{\mathrm{IG}}$ (std = 0) or the very low performance for the same class and same case, for low number of class 1 instances.

Variable size datasets are different and an increase in one class does not mean decrease in the number of instances in the other class. Since, identical instances are counted as many times as they appear, the increase in size of datasets could increase the number of items belonging to both classes. While this is expected, this increase would happen for one class at the very least. Unlike the case of fixed size datasets, decreases in one class also could decrease the number of items in both classes. Due to the enforced limitations on size, one can not observe higher number of instances than approximately 5000 in one class as illustrated in Figure 4. VAR.VNN-SBS$_{\mathrm{A}}$ and VAR.VRnd-SBS$_{\mathrm{A}}$ show similar patterns at the beginning but one can see more clear drops in performance in VAR.VRnd-SBS$_{\mathrm{A}}$ with increases in class 1. Both of these cases, as well as VAR.VNN-SBS$_{\mathrm{IG}}$ show comparatively higher variations in performance. The relation between increase in size and more variations in performance is clear in almost all cases. Except at the beginning, VAR.VAT-SBS$_{\mathrm{IG}}$ shows a steady pattern, like its fixed counterpart, VAT-SBS$_{\mathrm{IG}}$.

C. Mutation and cross over patterns

As described earlier, these operations could modify the label of training data in order to address the data quality issue which is present in many of the available defect datasets. Before presenting the performance details, we first look at the rate of label changes that are introduced by these operations.

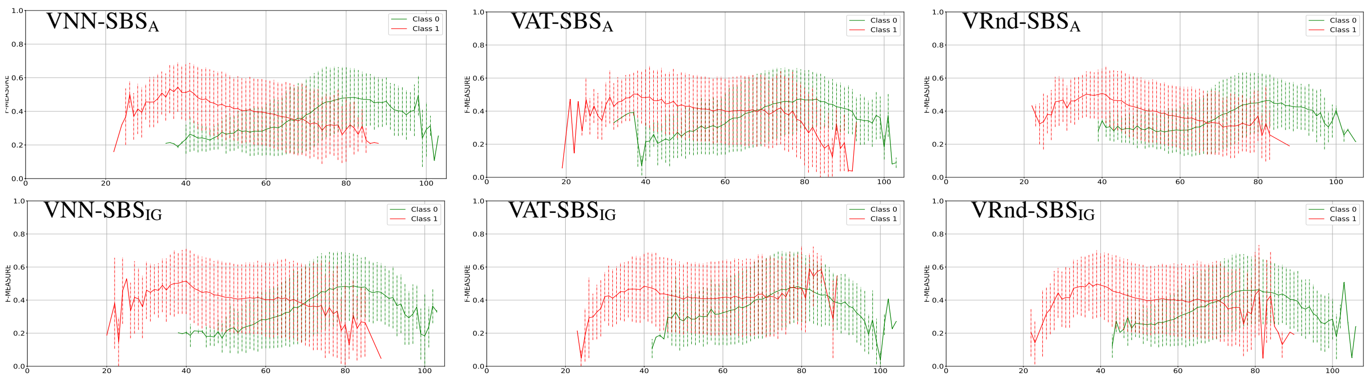

Fig. 3: Distribution of instances and their related F-measure performances for fixed size training datasets

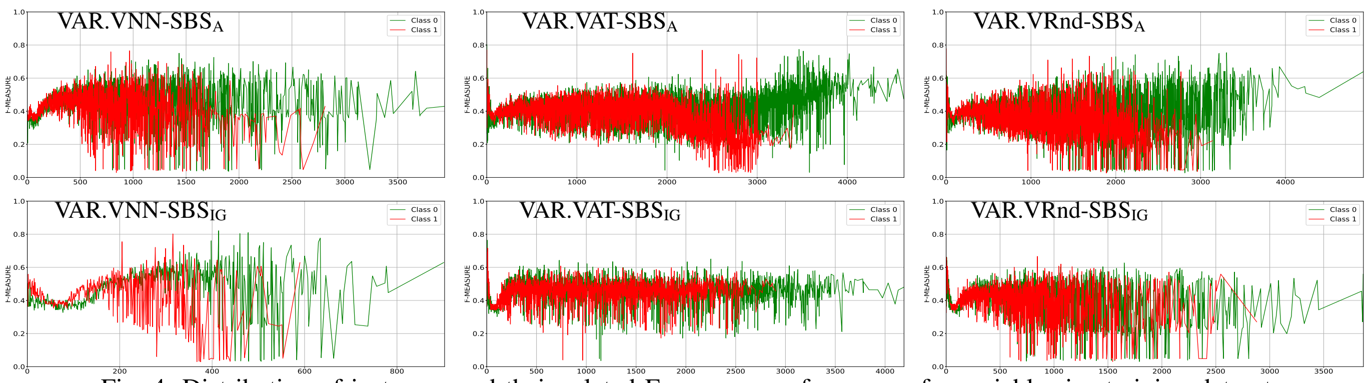

Fig. 4: Distribution of instances and their related F-measure performances for variable size training datasets

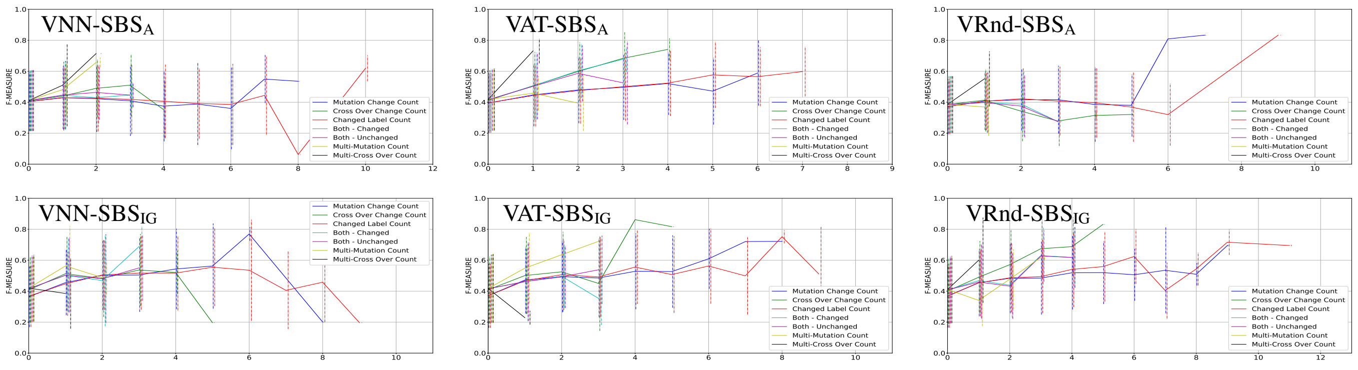

Fig. 5: Rate of changes based on related label changes for F-measure and fixed size datasets

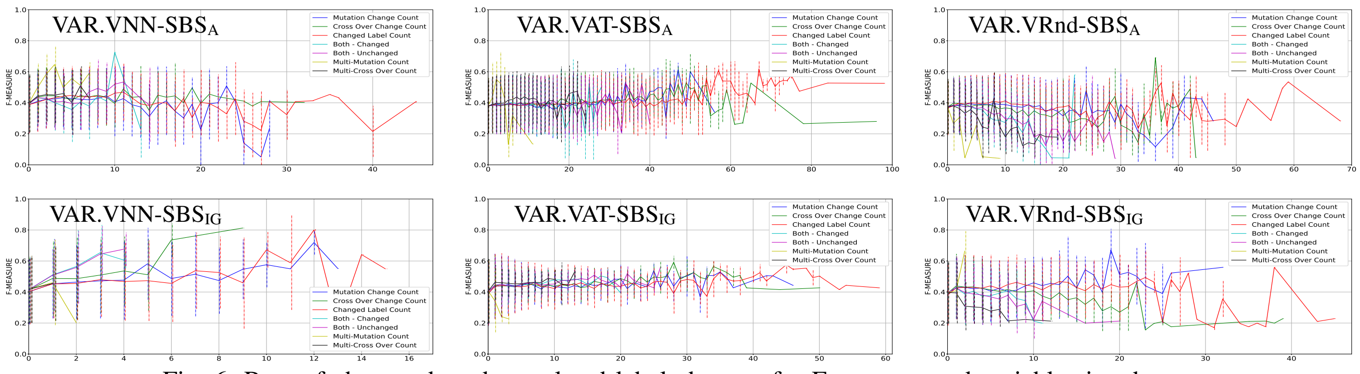

Fig. 6: Rate of changes based on related label changes for F-measure and variable size datasets

Figures 5 and 6 depict performances based on multiple factors. These factors include number of mutated instances (blue), number of instances having cross over (green), total number of changed labels by mutation or cross over or both (red), count of **changed** labels having both cross over and mutation (cyan), count of **unchanged** labels having both cross over and mutation (magenta), multiple mutation count (yellow) and multiple cross over count (black), respectively. As expected, in fixed size datasets, the total number of mutated items are not very high (matching the #generations and mutation prob%). The counting process does not exclude identical instances (simple weighting mechanism). Therefore, counting is not based on unique items. Having said these, one can see that variable size datasets can have near 100 changed labels, a portion of which is due to the mentioned simple weighting mechanism. The rate of changed labels show performance changes in some cases. These operations are not however, the only contributing factors and the performance increase might not be associated with them **only**.

V. Discussions

The results of the statistical tests and observed effect sizes, revealed a set of findings regarding the impact of the factors. These are discussed below:

Validation data and feature set: Both feature sets and validation dataset selection methods show significant, but tiny effects on F-measure. Further, precision is not affected significantly by these parameters in different scenarios in isolation while making false positives and true negatives are clearly influenced by the choice of validation dataset. While validation dataset selection is important and NN-Filter outperforms random selection, there clearly is room for improvements. This brings into question, the value of the similarity measures, used in NN-Filter for CPDP. The size of the validation dataset as well as the number of available features seem to be important when tuning toward a particular measure. The larger the validation dataset is and the less the number of available features are, the more recall based performances in expense of losing precision are observed. The results of the random validation dataset method demonstrate the positive impact (albeit rather small) of validation dataset selection as depicted in Figure 2. The performance of random methods, is mostly lower in terms of both recall and precision and consequently, F-measure. However, search for alternatives to NN-Filter is a valid direction to investigate considering the small improvements.

Training data: While the size of the training dataset (fixed, variable) can in theory impact the performance and preciseness of prediction models, the statistical tests do not favour one over the other based on F-measure except in S1, for which a tiny effect size is observed. In all three scenarios, no significant impact can be observed for precision. Based on the medians, variable size datasets show higher precision and lower recall performances. The direction of performance suggests that the test and validation performances are similar to some extent and hence, similarity in the number of instances belonging to each class, to that of the large pool of training data could increase performance. This, however, is the overall result and is subject to change for certain datasets with skewed distribution of instances. In the same line as previous studies such as [25], while increases in training dataset size could possibly increase performance, the instability of performance, is a discouraging evidence for using larger datasets which lead to more variations in performance.

Fitness function: Careful selection of fitness function, has major impact on performance. This is a step toward increasing precision and lowering false alarms which are very common is CPDP research [2]. Depending on the target group and preferences the fitness function can be very effective toward particular goals. In current settings, the effect of fitness function is demonstrably larger than the choice of validation selection methods.

Learning technique: Learning techniques without hyper-parameter tuning, might not be effective means of performance improvement considering their differences in complexity and power. Learners show similar effect sizes to those observed for validation dataset selection and are generally very small. Effects of tuned learners and their interaction with other affecting parameters is a direction to investigate.

Search operators: While showing slight changes in performances, the label changes do not seem to be the top contributors to performance direction. Having said these, they are an essential part of the model and steps toward better performances and addressing data quality issues.

VI. Threats to Validity

During an empirical study, one should be aware of the potential threats to the validity of the obtained results and derived conclusions [26].

Construct validity: We used our proposed approach GIS, as the search based method in this experiments. GIS is based on genetic algorithm and considers data quality in defect datasets, which can be different from other search based methods. The experimental datasets used in this study belong to a set of datasets collected from Java projects and the datasets can be subject to quality issues. This and our previous papers, have proposed steps to address these issues. To address different aspects of the model different strategies were used. Validation dataset selection methods, training dataset types, learning techniques and fitness functions are some of the model parameters that were investigated in more details to shed light on specific properties of such search based approaches.

External validity: It is difficult to draw general conclusions from empirical studies of software engineering and our results are limited to the analyzed data and context. Even though many researchers have used the same datasets as the basis of their conclusions, there is no assurance about the generalization of conclusions drawn from these projects. Particularly the applicability of the conclusions for commercial, proprietary and closed source software might be different.

Conclusion validity: Our experiments are repeated 20 times to address the randomness and the results are compared using one and two way ANOVA tests. Further, to calculate the magnitude of the difference, appropriate effect sizes are reported. Another threat is the choice of the evaluation measure. We have reported a diverse set of performance measures to address this problem.

VII. Conclusions

Through an exploratory approach, we investigated variants of a search based method by analyzing collected data from a series of experiments using different options. We used the latest releases of 13 projects from PROMISE repository in our experiments. The results revealed the value of the identified design options in the context of relevant performance measures. The experiments and analyses in this study acted, not only as detailed investigation of our previously proposed search based approach (GIS), but also, provided insights into the careful selection of design choices for such models with relatively higher number of various options. Through this study, we analyzed the value and validity of using NN-Filter for validation dataset generation, fixed vs variable size training datasets, and the effects of tuned fitness functions beside the investigation into learning techniques, class imbalance and the genetic operators. We considered potential interactions among different parameters based on their nature of impact.

Based on the results, we observed small effects for validation data, non-tuned learning techniques, and size of training data; and larger effects for fitness functions. We also observed insignificant, and in some cases tiny effects on precision, by changing the parameters. Some of the measures however, were heavily affected based on the fitness functions, e.g. MCC for TNR.

Some of the observations in this study, revealed potential directions for future investigations. Perhaps, better validation dataset generation strategies, tuning learners, and search operators are the most notable among them. At the same time, the class imbalance problem investigations and the effect of training dataset types demonstrate the need for additional research on the topic.

References

[1] Burak Turhan, Tim Menzies, Ayşe B Bener, and Justin Di Stefano. On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, 14(5):540–578, 2009.

[2] Seyedrebvar Hosseini, Burak Turhan, and Dimuthu Gunarathna. A systematic literature review and meta-analysis on cross project defect prediction. *Software Engineering, IEEE Transactions on*, 2017.

[3] Barbara A Kitchenham, Emilia Mendes, and Guilherme H Travassos. Cross versus within-company cost estimation studies: A systematic review. *IEEE Transactions on Software Engineering*, 33(5), 2007.

[4] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 91–100. ACM, 2009.

[5] Barry W Boehm. Understanding and controlling software costs. *Journal of Parametrics*, 8(1):32–68, 1988.

[6] S. Hosseini, B. Turhan, and M. Mäntylä. Search based training data selection for cross project defect prediction. In *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, PROMISE 2016, pages 3:1–3:10. ACM, 2016.

[7] S. Hosseini, B. Turhan, and M. Mäntylä. A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction. *Information and Software Technology*, 95:296–312, Jun 2017.

[8] Zhimin He, Fengdi Shu, Ye Yang, Mingshu Li, and Qing Wang. An investigation on the feasibility of cross-project defect prediction. *Automated Software Engineering*, 19(2):167–199, 2012.

[9] Steffen Herbold. Training data selection for cross-project defect prediction. In *Proceedings of the 9th International Conference on Predictive Models in Software Engineering*, page 6. ACM, 2013.

[10] Duksan Ryu, Jong-In Jang, and Jongmoon Baik. A hybrid instance selection using nearest-neighbor for cross-project defect prediction. *Journal of Computer Science and Technology*, 30(5):969–980, 2015.

[11] Yi Liu, Taghi M Khoshgoftaar, and Naeem Seliya. Evolutionary optimization of software quality modeling with multiple repositories. *IEEE Transactions on Software Engineering*, 36(6):852–864, 2010.

[12] Gerardo Canfora, Andrea De Lucia, Massimiliano Di Penta, Rocco Oliveto, Annibale Panichella, and Sebastiano Panichella. Defect prediction as a multiobjective optimization problem. *Software Testing, Verification and Reliability*, 25(4):426–459, 2015.

[13] Xin Xia, David Lo, Sinno Jialin Pan, Nachiappan Nagappan, and Xinyu Wang. Hydra: Massively compositional model for cross-project defect prediction. *IEEE Transactions on software Engineering*, 42(10):977–998, 2016.

[14] Burak Turhan. On the dataset shift problem in software engineering prediction models. *Empirical Software Engineering*, 17(1-2):62–74, 2012.

[15] Kim Herzig and Nachiappan Nagappan. Empirically detecting false test alarms using association rules. In *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ICSE '15, pages 39–48, Piscataway, NJ, USA, 2015. IEEE Press.

[16] Lin Chen, Bin Fang, Zhaowei Shang, and Yuanyan Tang. Negative samples reduction in cross-company software defects prediction. *Information and Software Technology*, 62:67–77, 2015.

[17] Duksan Ryu, Okjoo Choi, and Jongmoon Baik. Value-cognitive boosting with a support vector machine for cross-project defect prediction. *Empirical Software Engineering*, 21(1):43–71, 2016.

[18] Yasutaka Kamei, Takafumi Fukushima, Shane McIntosh, Kazuhiro Yamashita, Naoyasu Ubayashi, and Ahmed E Hassan. Studying just-in-time defect prediction using cross-project models. *Empirical Software Engineering*, pages 1–35, 2015.

[19] Duksan Ryu, Jong-In Jang, and Jongmoon Baik. A transfer cost-sensitive boosting approach for cross-project defect prediction. *Software Quality Journal*, 25(1):235–272, Mar 2017.

[20] Ying Ma, Guangchun Luo, Xue Zeng, and Aiguo Chen. Transfer learning for cross-company software defect prediction. *Information and Software Technology*, 54(3):248–256, 2012.

[21] Jacek Śliwerski, Thomas Zimmermann, and Andreas Zeller. When do changes induce fixes? In *ACM sigsoft software engineering notes*, volume 30, pages 1–5. ACM, 2005.

[22] Marian Jureczko and Lech Madeyski. Towards identifying software project clusters with regard to defect prediction. In *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, page 9. ACM, 2010.

[23] Tracy Hall, Sarah Beecham, David Bowes, David Gray, and Steve Counsell. A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering*, 38(6):1276–1304, 2012.

[24] Seyedrebvar Hosseini and Burak Turhan. Replication package, 2018. Available at https://doi.org/10.5281/zenodo.1200526.

[25] Tim Menzies, Burak Turhan, Ayse Bener, Gregory Gay, Bojan Cukic, and Yue Jiang. Implications of ceiling effects in defect predictors. In *Proceedings of the 4th international workshop on Predictor models in software engineering*, pages 47–54. ACM, 2008.

[26] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.