Choosing the Best Classification Performance Metric for Wrapper-based Software Metric Selection for Defect Prediction

Huanjing Wang
Western Kentucky University
huanjing.wang@wku.edu

Taghi M. Khoshgoftaar
Florida Atlantic University
khosgof@fau.edu

Amri Napolitano
Florida Atlantic University
amrifau@gmail.com

Abstract—Software metrics and fault data are collected during the software development cycle. A typical software defect prediction model is trained using this collected data. Therefore the quality and characteristics of the underlying software metrics play an important role in the efficacy of the prediction model. However, superfluous software metrics often exist. Identifying a small subset of metrics becomes an essential task before building defect prediction models. Wrapper-based feature (software metric) subset selection uses a classifier to discover which feature subsets are most useful. To the best of our knowledge, no previous work has examined how the choice of performance metric within wrapper-based feature selection will affect classification performance. In this paper, we used five wrapper-based feature selection methods to remove irrelevant and redundant features. These five wrappers vary based on the choice of performance metric (Overall Accuracy (OA), Area Under ROC (Receiver Operating Characteristic) Curve (AUC), Area Under the Precision-Recall Curve (PRC), Best Geometric Mean (BGM), and Best Arithmetic Mean (BAM)) used in the model evaluation process. The models are trained using the logistic regression learner both inside and outside wrappers. The case study is based on software metrics and defect data collected from a real world software project. The results demonstrate that BAM is the best performance metric used within the wrapper. Moreover, comparing to models built with full datasets, the performances of defect prediction models can be improved when metric subsets are selected through a wrapper subset selector.

I. INTRODUCTION

Software metrics collected are often associated with defects found during pre-release as well as post-release. Software defect prediction models are built based on these software metrics and fault data. Subsequently, the quality of currently under-development program modules is estimated, e.g., fault-prone (fp) or not-fault-prone (nfp). The models help to reduce software development effort and produce highly reliable software. Software defect prediction has been a high-focus area of research in the software engineering community [1], [2], [3], [4]. Defect predictors are often used by software quality practitioners in guiding them to intelligently allocate limited project resources toward program modules that are likely to have poor quality and reliability. A practical challenge faced by these practitioners is the selection of the right software metrics prior to building a defect predictor. Selecting the right static code metrics prior to building a defect prediction model has many benefits, including removing noisy attributes, reducing the set of software metrics to use, and perhaps even improving defect prediction performance [5]. Feature (software metric) selection algorithms (which select a subset of features from the original dataset) are often used to reduce the original feature set down to a subset containing only the most important features.

In this paper, we evaluate the wrapper-based feature subset selection method: classification models are built with logistic regression (LR) and feature subsets are selected according to their predictive capability, which are measured by a performance metric. We used five different performance metrics: Overall Accuracy (OA), Area Under ROC (Receiver Operating Characteristic) Curve (AUC), Area Under the Precision-Recall Curve (PRC), Best Geometric Mean (BGM), and Best Arithmetic Mean (BAM). When building a defect prediction classifier, we used ten runs of five-fold cross-validation, along with five-fold cross-validation within the wrapper process.

The experiments of this study were carried out on nine datasets from a real world project. These datasets are further divided into three groups with different levels of imbalance. There are three datasets in each group. We compared model performance on the smaller subsets selected by the wrapper-based feature subset selector. We also compared the classification performance on the smaller subsets of attributes with those on the original dataset. Results demonstrate that when BAM is used inside the wrapper, models built with the selected feature subset have the best performance, while PRC gives the worst performance.

The key contributions of this research are:
- Implementation and investigation of the wrapper-based feature subset selection technique. Although five performance metrics and one learner are used in the study, other performance metrics and learners can be used.
- The use of imbalanced data for software quality prediction. Such an extensive range of wrapper-based feature selection along with imbalanced data for software quality prediction is unique to this study.

The rest of the paper is organized as follows. We review relevant literature on feature selection in Section II. Section III provides detailed information about the wrapper-based feature selection, performance metrics, learner, and cross-validation methods used in our study. Section IV provides a description of the software measurement datasets used and presents empirical results of our study. Finally, in Section V, the conclusion is presented and the suggestions for future work are indicated.

II. RELATED WORK

Feature (attribute) selection is an important preprocessing step in the area of data mining and machine learning. Researchers and practitioners are interested in which feature subset is important in the model building process when high-dimensional datasets are considered. Specifically, feature selection seeks to reduce the number of features by targeting an optimum subset of features and removing
the rest of the features from further consideration during subsequent analysis. The idea is that the features that are not being considered are either irrelevant to the problem at hand or redundant when compared to the features within the optimum subset. Guyon and Elisseeff [6] outlined key approaches used for attribute selection, including feature construction, feature ranking, multivariate feature selection, efficient search methods, and feature validity assessment methods. Hall and Holmes [7] investigated six attribute selection techniques that produce ranked lists of attributes and applied them to several datasets from the UCI machine learning repository. Liu and Yu [8] provided a comprehensive survey of feature selection algorithms and presented an integrated approach to intelligent feature selection.

Feature selection techniques can be separated into two categories: filter and wrapper. Filters are algorithms in which a feature subset is selected without involving any learning algorithm. Wrappers are algorithms that use feedback from a learning algorithm to determine which feature(s) to include in building a classification model. Wrapper methods differ from filter-based feature selection in that they use a learner when evaluating the features, either separately or as subsets. In this study we focused on wrapper-based feature selection. Typically, wrapper-based feature selection uses subset evaluation, which looks at the possible subsets of features and tests them for their ability to differentiate between the classes. It should be noted that in order to test each possible subset, one would have to perform \(2^n - 1\) tests (there is no point in testing an empty set) where \(n\) is the number of features. Search algorithms can reduce this space significantly, however, they do not resolve the problem of necessitating many evaluations, however.

Very limited research exists on wrapper-based feature subset selection in the software quality and reliability engineering domain. Chen et al. [9] have studied the applications of wrapper-based feature selection in the context of software cost/effort estimation. They conclude that the reduced dataset improved the estimation. Rodríguez et al. [5] applied attribute selection with three filter models and two wrapper models to five software engineering datasets. It was stated that the wrapper model was better than the filter model; however, that came at a very high computational cost. Their conclusions were based on evaluating models using cross-validation. Their work is very limited since the same performance metric was used both inside and outside the wrapper. Although they performed ten runs of ten-fold cross validation for final classification, when performing wrapper feature selection (within the cross-validation process), the goodness of the wrapper learners was evaluated against the exact same data used to train them. In our work, we evaluate five wrapper-based subset evaluators, with these wrappers differing based on the choice of performance metric used in the model evaluation process inside the wrapper. We used five-fold cross-validation to build the learners within the wrapper, in addition to ten runs of five-fold cross-validation used to evaluate our overall classification models: within every run and fold of external (classification) cross-validation, we perform separate folds of internal cross-validation to evaluate subsets of features inside the wrapper. We also evaluate our final classification models using the five different performance metrics, and our choice of datasets enables us to study three different levels of class imbalance. An extensive comparative study of feature subset selection techniques is very unique to this paper, especially within the software engineering community.

III. METHODOLOGY

In this study, we focus on wrapper-based feature subset selection techniques and apply these techniques to software engineering datasets. Five performance metrics are used to evaluate the models built during the wrapper selection process giving a total of five different wrappers. The final classification models are also evaluated using these same five performance metrics. Logistic Regression is used to build classification models both inside and outside the wrapper.

A. Classifier

Classification is used to build a model that can accurately classify instances. The goal is to build a model which minimizes the number of classification errors. The first step of classification is to build a classification model that can describe the predetermined set of data classes, and in the second step, the classification model is evaluated using an independent test dataset. In this study, the software quality prediction models are built with logistic regression [10] both inside and outside the wrapper. Logistic Regression (LR) [10] is a statistical technique that can be used to solve binary classification problems. Based on training data, a logistic regression model is created which is used to decide the class membership of future instances. LR was selected because of its common use in software engineering and other application domains, and also because it does not have a built-in feature selection capability. We use default parameter settings for the learner as specified in WEKA [11].

B. Wrapper-based Feature Selection

Filter-based feature selection techniques have been studied extensively in the past [12]. By comparison, very little research has been focused on wrapper-based feature selection. The wrapper-based feature selection methods employ some predetermined learning algorithms (classifiers or learners) to evaluate the goodness of the subset of features being selected. The performance of this approach relies on three factors: (1) the strategy to search the feature space for possible optimal feature subsets; (2) the learner; and (3) the criterion to evaluate the classification model built with the selected subset of features.

Suppose a large set of \(n\) features is given, we need to find a small subset of features for future model building. Inspecting all candidate subsets \(2^n\) is impractical. There are some strategies that can solve the problem. One way is to use a search algorithm to generate the possible feature subsets. Based on preliminary experimentation, we chose the Greedy Stepwise approach, which uses forward selection to build the full feature subset starting from the empty set. At each point in the process, the algorithm creates a new family of potential feature subsets by adding every feature (one at a time) to the current best-known set. The merit of all these sets are evaluated, and whichever performs best is the new known-best set. The wrapper procedure terminates when none of the new sets outperform the previous known-best set.

During the search process, classification models are built using a potential feature subset and using the performance of this model as a score for the merit of that subset [13]. For our experiments the wrapper process uses five-fold cross-validation: the training set is divided into five equal folds (partitions), a classifier is trained on four folds, then tested on the last (fifth) fold. This process is repeated five times, and the results are averaged to give the merit of the potential feature subset. In this study, logistic regression is used within the wrapper-based feature subset selector. The classification model is assessed on the performance of the model based on five different performance metrics (Overall Accuracy (OA), Area Under ROC (Receiver Operating Characteristic) Curve (AUC), Area Under the Precision-Recall Curve (PRC), Best Geometric Mean (BGM),
and Best Arithmetic Mean (BAM)). All five performance metrics associated with each potential feature subset are obtained.

C. Performance Metrics

In a two-group classification problem, such as fault-prone and not fault-prone, there can be four possible prediction outcomes: true positive (TP), false positive (FP), true negative (TN) and false negative (FN), where positive represents fault-prone and negative represents not-fault-prone. The numbers of cases from the four sets (outcomes) form the basis for several other performance measures that are well known and commonly used for classifier evaluation. We used five of them for the wrapper-based feature subset selection in this study. They are:

1) Overall Accuracy (OA): It provides a single value range from 0 to 1. It can be obtained by \( \frac{TP + TN}{N} \), where N is the total number of instances in the dataset.

2) Area Under ROC (Receiver Operating Characteristic) Curve (AUC): It has been widely used to measure classification model performance \[14\]. AUC is a single-value measurement that ranges from 0 to 1. The ROC curve is used to characterize the trade-off between true positive rate (defined as \( \frac{TP}{TP + FN} \)) and false positive rate (defined as \( \frac{FP}{FP + TN} \)). A classifier that provides a large area under the curve is preferable over a classifier with a smaller area under the curve. A perfect classifier provides an AUC that equals 1. It has been shown that AUC has lower variance and is more reliable than other performance metrics (such as precision, recall, or F-measure) \[15\].

3) Area Under the Precision-Recall Curve (PRC): It is a single-value measure that originated from the area of information retrieval. Precision and Recall are two widely used performance metrics in the context of classification tasks. The closer the area is to one, the stronger the predictive power of the attribute \[16\]. The PRC graph depicts the trade off between recall and precision. A classifier that is near optimal in AUC space may not be optimal in precision/recall space.

4) Best Geometric Mean (BGM): The Geometric Mean (GM) is a single-value performance measure that ranges from 0 to 1, and a perfect classifier provides a value of 1. GM is defined as the square root of the product of true positive rate and true negative rate (defined as \( \sqrt{TP/(TP + FN)} \)). It is a useful performance measure since it is inclined to maximize the true positive rate and the true negative rate while keeping them relatively balanced. Such error rates are often preferred, depending on the misclassification costs and the application domain. The Best Geometric Mean (BGM) is the maximum Geometric Mean value that is obtained when varying the threshold between 0 and 1 \[17\].

5) Best Arithmetic Mean (BAM): The Arithmetic Mean is just like geometric mean but using the arithmetic mean of the true positive rate and true negative rate instead of the geometric mean. It is also a single-value performance measure that ranges from 0 to 1. The Best Arithmetic Mean (BAM) is just like the BGM, but using the maximum arithmetic mean that is obtained when varying the threshold between 0 and 1, instead.

D. Cross-Validation

In the experiments, we use ten runs of five-fold cross-validation to build and test the final classification models. That is, the datasets are partitioned into five folds, where four folds are used to find the appropriate feature subset and train the classification model, and the model (using the chosen feature subset) is evaluated on the fifth. This is repeated five times so that each fold is used as hold out data once. In addition, we perform ten independent repetitions (runs) of each experiment to remove any bias that may occur during the random selection process. In total, 50 training subsamples are generated for each dataset. Note, feature selection is performed inside of this cross-validation procedure. That means, wrapper feature selection is applied to each of the training subsamples. Furthermore, the one run of five-fold cross-validation discussed in Section III-B is in addition to these ten runs of five-fold cross-validation: within every fold of external (classification) cross-validation, we perform separate folds of internal (wrapper) cross-validation, to evaluate the quality of our feature subsets. In addition to the models built during the wrapper process, a total of 2700 (9 datasets × 5 wrappers × 10 runs × 5 folds cross-validation + 9 datasets × 10 runs × 5 folds cross-validation) final classification models are trained and evaluated during the course of our experiments. The classification results reported in the next section represent the average across these ten runs of five-fold cross-validation.

IV. EXPERIMENTS

A. Experimental Datasets

The software metrics and fault data for this case study were collected from a real-world software project, the Eclipse project \[18\]. We consider three releases of the Eclipse system, where the releases are denoted as 2.0, 2.1, and 3.0. In particular, we use the metrics and defects data at the software package level. We transform the original data by: (1) removing all nonnumeric attributes, including the package names, and (2) converting the post-release defects attribute to a binary class attribute: fault-prone (fp) and not fault-prone (nfp). Membership in each class is determined by a post-release defects threshold \( t \), which separates \( fp \) from \( nfp \) packages by classifying packages with \( t \) or more post-release defects as \( fp \) and the remaining as \( nfp \). In our study, we use \( t \in \{10, 5, 3\} \) for release 2.0 and 3.0, while we use \( t \in \{5, 4, 2\} \) for release 2.1. These values are selected in order to have datasets with different levels of class imbalance. All nine derived datasets contain 208 independent attributes (software metrics). Releases 2.0, 2.1, and 3.0 contain 377, 434, and 661 instances respectively. A different set of thresholds is chosen for release 2.1 because we wanted to maintain relatively similar class distributions for the three datasets in a given group. Table I presents key details about the different datasets used in our study. The three groups of datasets exhibit different class distributions with respect to the \( fp \) and \( nfp \) modules, where Eclipse-I is relatively the most imbalanced and Eclipse-III is the least imbalanced.

B. Experimental Design

We first used five wrapper-based feature subset selectors (LR + one of five performance metrics) to select the subsets of attributes.
Subsequently, the reduced dataset with the selected feature subset was used to build the respective final classification model. Note that the same learner (LR) is used within and outside wrapper. The experiments were conducted to discover the impact of (1) wrappers with different performance metrics; (2) different combinations of performance metrics used inside and outside the wrappers; and (3) three different groups of software metrics datasets with different class-imbalance levels from the software quality prediction domain.

We implemented wrapper-based feature subset selection in WEKA and used it for the defect prediction model building and testing process. In the experiments, ten runs of five-fold cross-validation were performed. The five results from the five folds were then combined to produce a single estimation.

### C. Results and Analysis

Tables II to IV list the mean value and standard deviation for every classification model constructed over ten runs of five-fold cross-validation for each dataset group with different level of imbalance. Each column represents one choice of performance metric used to evaluate the final classification model, while each row is one choice of performance metric used within the wrapper. Within each column, the row which shows the best classification performance is indicated in **boldface** print, while the worst performance is *italic*. The first notable observation from these experimental results is that BGM is the best wrapper metric for all of the classification metrics. This was true across all different levels of imbalanced datasets for 13 out of 15 cases. Typically, BGM was the second-place wrapper metric, except for two cases when it was the best metric. Nonetheless, based on these results, we recommend BAM as the wrapper metric which will produce the best classification performance.

In addition to finding the best wrapper metrics, we also observed that PRC is the worst wrapper metric for the datasets with a moderate amount of class imbalance (that is, the Eclipse-II and Eclipse-III collections) regardless of which classification performance metric is used and was true for nine out of ten choices. For the most imbalanced datasets (the Eclipse-I collection), accuracy is the least favorable classification metric, it shows the worst performance.

We also performed experiments on the original datasets and results were shown in Table V. From these tables, we can conclude that the reduced feature subsets can have better prediction performance compared to the complete set of attributes (original dataset). This implies that software quality classification and feature selection were successfully applied in this study.

It is noted that Accuracy does not take class imbalance into account, while all four other metrics do so. So from the tables we can observe that accuracy suffers a much larger drop between the Balanced/Slightly Imbalanced datasets and the Imbalanced datasets than other performance metrics, such as AUC, have.
An n-way ANOVA can be used to determine if the means in a set of data differ when grouped by multiple factors. If they do differ, one can determine which factors or combinations of factors are associated with the difference. We built our one-way ANOVA model for final classification model performance metrics OA and AUC (due to space consideration, we only represent results for these two performance metrics) on three different group of datasets separately. The factor A represents five wrappers. The ANOVA model can be used to test the hypothesis that the OA (or AUC) for the main factor A are equal against the alternative hypothesis that at least one mean is different. If the alternative hypothesis (i.e., that at least one mean is different) is accepted, multiple comparisons can be used to determine which of the means are significantly different from the others. In this study, we performed the multiple comparison tests using Tukey’s honestly significant difference criterion. All tests of statistical significance utilize a significance level $\alpha = 0.05$.

The ANOVA results for the OA and AUC metrics are presented in Tables VI and VII, respectively. From the tables, we can see that for Eclipse-I-OA, Eclipse-I-AUC, and Eclipse-III-OA, the $p$-values (last column of the tables) for the main factor is greater than a typical cutoff value of 0.05, which implies no significant difference exists between the five wrappers across all 3 datasets in each group. The $p$-values for Eclipse-II-OA, Eclipse-II-AUC, and Eclipse-III-AUC are less than the typical cutoff value of 0.05, indicating that for the classification performance, the alternate hypothesis is accepted, namely, at least two group means are significantly different from each other for at least one pair of groups in the corresponding factors or terms.

Additionally Tukey’s HSD test based multiple comparisons for the main factor were performed to investigate the difference among the respective groups (levels). The test results are shown in Figure 1 (for OA) and Figure 2 (for AUC), where each sub-figure displays graphs with each group mean represented by a symbol ($\circ$) and 95% confidence interval. The results show the following facts: In general, wrapper with PRC performed worst among the 5 wrappers and BAM and BGM performed best.

V. Conclusion

Identifying a small set of software metrics for building high accuracy software defect predictors can help reduce software development costs and produce a more reliable system. In this study, wrapper-based feature subset selection techniques have been used to find small subsets of attributes to build software defect prediction models. We used logistic regression as our learner both inside and outside
the wrapper. Five performance metrics are used to evaluate models within the wrapper and also were used to evaluate the respective final classification models.

Our findings show that BAM is very reliable as a performance metric to evaluate models built within the wrapper. This is true for datasets with different degrees of imbalance.

Future work may include experiments using more classifiers, other feature subset search methods, and additional software metrics datasets from the software engineering domain. Given the imbalanced nature of the datasets studied in this paper, the use of sampling can also be considered to determine whether it will have any effects on model performance with feature selection.

REFERENCES


