Software Metric and Fault Prediction Using Hybrid FISHER Filter- ANNIGMA Framework

Peddada Venkateswara Rao1, Dorababu Sudarsa 2, Ravi Kumar Tata 3, Kotakonda Madhubabu4
1,2,4 Asst. Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India.
3 Associate Professor, Department of C.S.E, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India.
1pvrao.pd@gmail.com, 2dorababu.sudarsa@gmail.com, 3rktata@kluniversity.in, 4madhubabu@kluniversity.in

ABSTRACT
Software Quality is one of the biggest assets in the Software development process. Any projects success or failure is dependent on the Software Quality. Software Quality can be assured through various phases of testing [1]. Testing techniques like alpha-beta testing, Black box testing and White box testing techniques can assure the software quality to a certain level. Faults can be predicted and rectified by these techniques. Manually testing can be very expensive and sometime impossible to perform testing. An automated hybrid software fault prediction model that is capable to measure the relative quality of software and identify their faulty components can significantly reduce software development effort and the threat of software project failure. In our paper we used a hybrid algorithm using ANN(Artificial Neural Networks) Wrapper and FISHER Filter techniques.

Key words : ANN Wrapper, FISHER Filter, MR Filter, Defect prediction.

1. INTRODUCTION
The considerable amount of money spending on software testing gives its crucial role being played in the software enterprise. However, the modern programming style in software systems are large and complex which makes it challenging in finding out the defects and selection of the best metrics to be used when predicting defects. Also, continuous upgradations of processors, operating systems make it hard to predict these defects so, overcoming these complexities is very challenging. Since for large and complex program structures there must be a significant number of test cases where they are too expensive and tedious. Manual prediction of software defects can lead to inaccuracies and further complexities, one of the following paper[16] showed this, in order to avoid those circumstances, it is better to use automated software fault detection strategies which provide accurate results.

Commonly, to understand the level of software reliability we use software quality measures mainly internal and external metrics. The internal metrics are used to measure the source code whereas external metrics to predict the behavior and functionalities of the software. Major changes in the key attributes of different entities can indicate problems in design, quality, and reliability.

Many of the automated defect-prone strategies are in high importance for dynamic software systems in the course of software development processes[21]. Many of the automated approaches use machine learning (ML) techniques such as classification algorithms, k-means clustering, Naïve Bayes (NB), support vector machines (SVM), and neural networks as an internal metric for software defect prediction. [15] [17] [18] these papers covered how efficiently defect detection can be handled.

Prior work to the researchers combines both the filter and wrapper approaches to produce a better metric selection. Filter models use heuristic functions on the datasets to reduce the overall search space. However, these filter models do not consider their performance evaluation [13] during metric ranking process is done. Therefore, using filters can give poor performance consuming a large amount of time and sometimes inefficient in fault prediction of a software system. On the other hand, wrappers use an induction algorithm to predict the performance of a selected subset.

The research of previous authors proposed a hybrid wrapper-filter approach for software defect prediction and metric selection. The two wrapper-based hybrid models include Support Vector Machines (SVM) and Artificial Neural Networks (ANN) heuristics, and then these models are used to find the defects in a software system and helps to choose the best metric for the respective framework.

The key contributions In the Hybrid Framework (Sequential) are as follows:
1. Defect prediction and key metric selection problem are combined as a single optimization problem.
2. Uses a hybrid model that combines SVM and ANN with maximum relevance (MR) filter for predicting high accuracies.
3. It checks the performance of the hybrid algorithm using the filter score. Since the above proposed cross breed structure sets aside an excessive amount of effort to complete their handling and is to be considered as a key issue in this advanced period of PC design, one approach to conquer this restriction is to utilize a parallel forecast framework for bridging the full intensity of the PC engineering. To improve the speed various frameworks are associated through a rapid system to execute them in a parallel forecast, so the determination of indicator must be planned carefully[22].

2. BACKGROUND AND RELATED WORK
A systematic review, primarily intended for software fault prediction (SFP) using ML techniques, of 64 primary studies with seven categories from January 1991 to October 2013, was presented in [5]. ML models are superior to the traditional statistical models to classify software modules as either defective or non-defective. It was identified in the study that; the most used metric selection technique is the alter approach of correlation-based feature selection and most frequently used metrics are the procedural metrics. The most useful OO metrics are CBO, RFC, and LOC and the most frequently used dataset is the NASA dataset.

A systematic literature review was carried out in [12] on 106 primary studies to determine which are the most widely used metrics for software defect prediction. It is observed from the study that 49% of the metrics used are OO (Chidamber and Kemerer (CK) metrics are among the highest), 27% are traditional source code metrics or 24% are process metrics. Other comparisons are shown in Table 1. In the proposed framework a combination of filter and wrapper is used.

3. FILTERS AND WRAPPERS
Filter [4] and wrapper [3] methods are useful for the selecting the key metrics. Filter analyses the intrinsic properties of the metrics, then the evaluation by the prediction model, to score the selected subset of metrics and to determine the best subset of metrics. Filters are computationally fast as there is no need to run an algorithm for scoring the subset of metrics. But in cases filters may predict redundant metrics which results in less performance.

Wrapper applies prediction algorithm 2ⁿ times to select the best metric subsets from the n-dimensional metrics. This method is computationally expensive. Wrapper method provides good prediction performance due to having interactions with the subset elements. It has a potential risk of overfitting. There are other techniques such as Greedy forward selection, threshold moving, ANFIS, Naïve Bayes, Random forest, Defect Proneness which has its own significance and Limitation. The significance of Greedy Forward Selection is Uses Swarm optimization - Key feature selection by quantum particle swarm optimization (QPSO) technique and applying these features to an ANN based classifier to detect software fault-proneness.

Threshold moving technique is applied at the end of a boosting algorithm to reduce the effect of misclassification. The limitation is No feature selection

4. DRAWBACKS IN SEQUENTIAL MR-FILTER AND ANN.WRAPPER
4.1 MR-FILTER:
MR heuristic can select salient features in data mining area. It uses mutual information which statistically summarizes the degree of relevance. Significant metrics provide more relevant information about the class variable than the insignificant metrics. So, MR-filter could be applied to select the significant metrics which are more relevant to the class variable, by means of using statistical heuristics.

In Sequential Hybrid MR-Filter and ANN-Wrapper, there are back ward elimination methods and MR-Filter does not give hundred percent performance. No filter can give complete performance but using fisher we can avoid back ward elimination steps in the algorithm that reduces algorithm complexity and improves performance. This can be done in both serial and parallel methods. In previous works[10] they used SVM-Wrapper and a BE process is engaged, but in SVM(Support Vector Machines)[2], the main drawback is to identify a kernel for a given data set which is very difficult and it does not give accurate results for large data.

4.2 FISHER-FILTER
In FISHER filter [12], a subset of features are clustered in a way that the distance between different data points in

<table>
<thead>
<tr>
<th>Approach name</th>
<th>Significant contribution</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td>Computationally faster</td>
<td>Does not evaluate the subset</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Good prediction performance</td>
<td>A potential risk of over-fitting</td>
</tr>
<tr>
<td>Greedy Forward selection</td>
<td>Uses Swarm Optimization</td>
<td>Makes Search space wider compared to filter</td>
</tr>
<tr>
<td>ANFIS</td>
<td>Fuzzy Membership</td>
<td>Only classification</td>
</tr>
<tr>
<td>Random forest</td>
<td>Uses method level metric</td>
<td>No Significant improvement in approach and no feature selection</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>Uses method level Metrics</td>
<td>No Significant improvement in approach and no feature selection</td>
</tr>
<tr>
<td>Defect-proneness</td>
<td>A feature selection framework uses existing classifier</td>
<td>Two separate steps for feature selection and classification and similar search strategy of filter approach</td>
</tr>
</tbody>
</table>
different classes becomes as far as possible and in the data points in the same class becomes as close as possible for the data domain of the selected features.

Let us consider that there are distinct classes \( c_k \in C \) with sample size \( s_k, k=1,2,...d \). Here \( \mu_k \) is the mean and \( \sigma_k \) is the standard deviation, of \( k \)th class.

Calculating FISHER score:
\[
FISHERR(p) = \sum_{k=1}^{d} s_k (\mu_k - \mu)^2 / (\sigma_k)^2
\]

Where \((\sigma)^2 = \sum_{k=1}^{d} s_k (\sigma_k)^2\)

5. PROBLEM STATEMENT

Here there are a set of \( X \) training software modules \( Y, \{Y: y_{ij}\}_{j=1,2,3,...,x} \) which have the corresponding set \( C \) of fault/no-fault classes \( \{C : c_l | l=1,2,3,...,X\} \). We have a set of \( x \) metrics \( \{P : p_i | i = 1,2,3,...,x\} \) based on that, the faults are identified in \( Y \). The problem of determining the set of significant software metric mathematically can be defined as

\[
G(P,Y) = \text{argmax} \ Perf(S_k | Y)
\]

Here \( Perf(S_k | Y) \) is a performance function that determines the capability of a set of metrics \( S_k \) to identify the software fault which can be defined as

\[
Perf(S_k | Y) = \sum y^*i / X
\]

where \( y^*i = 1 \) if the predicted fault type \( c^*i \) by the decision function \( D \) of the wrapper is equal to the actual type \( c_i \); otherwise \( y^*i = 0 \).

One of the most important challenge is to find which wrapper will maximize the performance function \( F \).

Here, eq 1 is computationally expensive task. Using different filter, we can reduce the complexity. But only to a certain level complexity can be reduced.

6. PROPOSED FRAMEWORK

Hybrid Framework using FISHER-Filter and ANN Wrapper: The Wrapper proposed in our framework is Artificial Neural Network Based Gain Measurement Approximation (ANNIGMA) [6]. Maximum performance is guaranteed using this wrapper because evaluation is done for each subset \( S_k \). This may improve the performance but increases the search space complexity as there are no BE steps involved.

Algorithm1 uses ANNIGMA Wrapper and Fisher-Filter. The input of this Algorithm will be \( N \) which will initially be True.

Decision function:
\[
Ds = \sum F(\sum pi * Nir) * Nrs
\]

Here, \( i, r, s \) are input layer, hidden layer and output layer. \( N_s \) and \( N_r \) are weights is the Neural network. From Fig.2

ANNIGMA Score is calculated. The calculation is as follows:

\[
\text{ANNIGMA}(p_a) = LG_{sa} / \text{max} LG_{sa}
\]

where \( LG \) is Local Gain

\[
LG = \sum N_s * N_r
\]

Average ANNIGMA score of all the folds should be calculated.

\[
\text{ANNIGMA}(p_a)_{\text{average}} = 1/x \sum \text{ANNIGMA}(p_a)_{\text{fold}}
\]

\[
\text{ANNIGMA}(p_a) = 1/k \sum \text{ANNIGMA}(p_a)_{\text{average}}
\]

\( K \) is the total no of output nodes.

FISHER-ANNIGMA= FISHER\(p_i\) + ANNIGMA\(p_o\)

The fisher filter score \( p_i \) is calculated from equation (3).

Algorithm1: Hybrid FW using FISHER-FILTER

1. Input \( \{Y: y_{ij}\}_{j=1,2,3...X} \} \) data from software modules with \( n \) metric
2. Input \( N \) True for ANN
3. Output \( S \) BEST significant metrics
4. \( S \leftarrow y_{ij} \) whole set of \( x \) metrics
5. \( S0 \leftarrow \) initial set of metrics
6. Select the FISHER-FILTER
7. For \( i=0 \) to \( n-1 \) do
8. Compute Fisher score using (3)
9. For fold=1 to \( x \) do
10. Train the wrapper ANN with software metric
11. set Surrent
12. Compute ANNIGMA scores of all metrics in the set Surrent using equation (4)
13. Compute accuracy
14. End for
15. Compute average ANNIGMA of Surrent using equation (5)
16. Compute average ANNIGMA of one-fold using equation (6)
17. Rank the metrics in descending order
18. Combine the current metrics with the initial metrics
\[ S_0 = S_0 \cup \text{current} \]
19. End for
20. Identify the best metrics with highest accuracy
21. Return best metric $S_{best}$

7. EXPERIMENT RESULT
In this paper we can evaluate our algorithm by taking different NASA data sets available, but here we have taken PC3 data set for performance analysis. Here we compare our Hybrid Model with the Maximum relevance and ANNIGMA algorithms. A MEWMA(Multivariate exponentially weighted moving average)[20] is used to continuously monitor and measure the performance of two or more characteristics of any output control situation. The following Table 2 shows our Hybrid framework achieved 94.5% in control vs. 4% as out of control situation and 93% vs. 5.6% for in control vs. 4% as out of control situation, respectively.

<table>
<thead>
<tr>
<th>Table 1: MEWMA results for software fault prediction dataset PC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characteristics</td>
</tr>
<tr>
<td>Total Points</td>
</tr>
<tr>
<td>Total OC Points</td>
</tr>
<tr>
<td>Total IC Points</td>
</tr>
<tr>
<td>Percentage(IC)</td>
</tr>
<tr>
<td>Percentage(OC)</td>
</tr>
</tbody>
</table>

Therefore, we can come to a conclusion that our Hybrid model is best compare to other models.

8. CONCLUSION
In this paper we discussed about the most important part of Fault prediction and rectification process. Most of the Legacy models are not reached up to expected level. To overcome the problems mentioned in this paper, we proposed hybrid software fault prediction model that is capable to measure the relative quality of software and identify their faulty components can significantly reduce software development effort and the threat of software project failure. In our paper we used a hybrid algorithm using ANN(Artificial Neural Networks) Wrapper and FISHER Filter techniques. Using Machine learning techniques for software defect detection is the feature scope of this field.

REFERENCES


