Comparative Analysis of Selected Heterogeneous Classifiers for Software Defects Prediction Using Filter-Based Feature Selection Methods

Abimbola G. Akintola, *Abdullateef O. Balogun, Fatimah B. Lafenwa-Balogun and Hameed A. Mojeed
Department of Computer Science, University of Ilorin, Ilorin, Nigeria
{akintola.ag|balogun.ao1|raji.fb|mojeed.ha}@unilorin.edu.ng

Abstract— Classification techniques is a popular approach to predict software defects and it involves categorizing modules, which is represented by a set of metrics or code attributes into fault prone (FP) and non-fault prone (NFP) by means of a classification model. Nevertheless, there is existence of low quality, unreliable, redundant and noisy data which negatively affect the process of observing knowledge and useful pattern. Therefore, researchers need to retrieve relevant data from huge records using feature selection methods. Feature selection is the process of identifying the most relevant attributes and removing the redundant and irrelevant attributes. In this study, the researchers investigated the effect of filter feature selection on classification techniques in software defects prediction. Ten publicly available datasets of NASA and Metric Data Program software repository were used. The topmost discriminatory attributes of the dataset were evaluated using Principal Component Analysis (PCA), CFS and FilterSubsetEval. The datasets were classified by the selected classifiers which were carefully selected based on heterogeneity. Naïve Bayes was selected from Bayes category Classifier, KNN was selected from Instance Based Learner category, J48 Decision Tree from Trees Function classifier and Multilayer perceptron was selected from the neural network classifiers. The experimental results revealed that the application of feature selection to datasets before classification in software defects prediction is better and should be encouraged and Multilayer perceptron with FilterSubsetEval had the best accuracy. It can be concluded that feature selection methods are capable of improving the performance of learning algorithms in software defects prediction.

Keywords— Data Mining, Machine Learning, Software Defects.

1. INTRODUCTION

Defect in software module occurs when incorrect programming logic or code are used which further produces wrong output and lead to poor quality software product. Defects in software module may cause software to be rejected by a user or terminate the contracted agreement with the company (Fenton & Pfleeger, 1996; Koru & Liu, 2005). High development, maintenance cost and customer dissatisfaction are the end result of a defective software. Faults in software systems continue to be a major problem. Fault is a flaw that results in failure (Ishani, Vivek & Anju, 2014). Further, Software fault prediction facilitates software engineers to pay attention to development activities on defect less code which enhance the software quality and minimize the cost and time to develop software system in today's era. There are many prediction models which are used to filter the software defects (Khoshgoftaar, Bullard & Gao, 2009; Lessman, Baeseus, Mues & Pietsch, 2008; Shivaji, Whitehead, Akella & Kim, 2009; Okutan & Yildiz, 2012).

Software defect prediction as a classification problem; it can be viewed as a supervised binary classification problem. Software modules are represented with software metrics, and are labelled as either defective or non-defective. To train defect predictors, data tables of historical examples are formed where one column has a boolean value for "defects detected" (i.e. dependent variable) and the other columns describe software characteristics in terms of software metrics (i.e. independent variables).

Feature selection is a process that selects a subset of original features. It is also known as attribute selection or reduction. It is one of the most important techniques used in data pre-processing in data mining (Ameen, Balogun, Usman & Fashoto, 2016). Mining on a reduced set of attributes offers benefits as it reduces the number of attributes appearing in the extracted patterns. The goal of feature selection is to discover the smallest subset of features that best identifies categories from data according to important criteria (Balogun, Mabayoje, Salihu & Arinze, 2015). Feature selection in unsupervised learning is mostly difficult due to the absence of class labels but in supervised learning, feature selection attempts to maximize accuracy. It is easier to perform feature selection for supervised learning since they make use of class labels (Miao, Liu, & Zhang, 2012; Khoshgoftaar, Gao, Napolitano & Wald, 2013). It is essential to predict the defect of software metrics in order to plan a better maintenance strategy. Researchers have developed many classification models for software defect prediction (Lessman *et al*, 2008; Peng, Khoshgoftaar *et al*., 2013; Miao *et al*., 2012; Hall, Beecham, Bowes, Gray & Counsell, 2012).

This study aims at performing a comparative analysis on heterogeneous classifiers used for software defects prediction based on filter feature selection. In line with the set aim of this study, the scope of this research is to cover the experimental investigation of software defect prediction as a classification problem. The study will investigate the use of four classifier algorithms based on their nature and categories for software defects prediction.

2. FEATURE SELECTION

There are three methods used for feature selection – Filter method, Wrapper method and Embedded method. These methods are explained below.

(1) **Filter method:** This method makes use of statistical information in order to assign scores to the features It produces a ranking of the feature (attribute) from most important to the least important.

(2) **Wrapper method:** This is basically represented as a search problem which considers the selection of a set of features.

*Corresponding Author

(3) **Embedded method:** This method selects the features that best improve the model accuracy while the model is being created (Aruna, Dilsha, Radhika, & Swathi, 2016).

2.1 Principal Component Analysis (PCA)

The aim of PCA is to reduce the dimensionality of dataset that contains a large number of correlated attributes by transforming the original attributes space to a new space in which attributes are uncorrelated. The algorithm then ranks the variation between the original dataset and the new one. Transformed attributes with most variations are saved and discard the rest of attributes.

2.2 Correlation Based Feature Selection (CFS)

CFS is a simple filter algorithm that ranks feature subsets and discovers the merit of feature or subset of features according to a correlation based heuristic evaluation function. The purpose of CFS is to find subsets that contain features that are highly correlated with the class and uncorrelated with each other. The rest of features should be ignored. Redundant features should be excluded as they will be highly correlated with one or more of the remaining features. The acceptance of a feature will depend on the extent to which it predicts classes in areas of the instance space not already predicted by other features.

2.3 Filter Subset Evaluation

Class for running an arbitrary subset Evaluator on Data that has been passed through an arbitrary filter (note: filters that alter the order or number of attributes are not allowed) like the evaluator, the structure of the filter is based exclusively on the training data.

3. CLASSIFICATION TECHNIQUE

Classification Technique is one of the data mining techniques that are used to analyze a given dataset. Classification is a two-step process, during first step the model is created by applying class algorithm on training data set then in second step the extracted model is tested against a predefined test dataset to measure the model trained performance and accuracy**.** Classification technique is used to find out, in which group each data instance is related within a given data set. It is used for classifying data into different classes according to some constraints. Several major kinds of classification algorithms include Multilayer Perceptron, K-Nearest Neighbour, Naive Bayes, Support Vector Machine, Decision Tree etc.

3.1 Naïve Bayes

The Naive Bayes algorithm is a simple probabilistic classifier that calculates a set of probabilities by counting the frequency and combinations of values in a given dataset. It is also a statistical method for classification. The algorithm uses Bayes theorem and assumes all attributes to be independent given the value of the class variable (Tina & Sherekar, 2013). Naïve Bayes is particularly suited when the dimensionality of the inputs is high and it uses the maximum likelihood for its parameter estimation.

```
BEGIN
    Input:
    Build the training data set D = {(X1, C1),...,
(Xn, Cn)}
    X =(X1,...., Xn) new instance to be classified
    For each labeled instance (Xi, Ci) do
    If X has an unknown system call then
    X is abnormal.
    else then
        For each process Di in training
data do
        calculate Sim(X, Di);
            if Sim(X, Di) == 1.0 then
                X is normal and
            exit;
        Order Sim(X, Di) from Lowest to
highest, (i = 1,...,N);
        Find K biggest scores of Sim(X, D);
        Select the nb instances to X: Dnb;
        Assign to x the most frequent class
in Dnb;
        Calculate Sim_Avg for Naive
bayes;
        If Sim_Avg > threshold then
        X is normal;
        else then
        X is abnormal;
END
```

Fig. 1: Naive Bayes Pseudocode
(Source: Tina & Sherekar, 2013)

3.2 Multilayer Perceptron

Multi-layer Perceptron Networks (MLP) are feed-forward artificial neural networks which is a famous model for machine learning (Ahmad & Nashat, 2012). MLP was developed to replicate learning and generalization abilities of humans with an attempt to model the functions of biological neural networks and they have many potential applications in the areas of Artificial Intelligence (AI) and Pattern Recognition (PR).

```
1: choose an initial weight vector ~w
2: initialize minimization approach
3: while error did not converge do
4: for all (~x, ~d) ∈ D do
5: apply ~x to network and calculate the
network output
6: calculate ∂e(~x)
7: end for
8: calculate ∂E(D)
9 for all weights summing over all
training patterns
10 perform one update step of the
minimization approach
11: end while
```

Fig. 2: Multilayer Perceptron Pseudocode
(Source: Ahmad & Nashat, 2012)

3.3 *k*- Nearest Neighbour

k-Nearest Neighbour (*k*NN) is an instance based learning method for classifying objects based on the closest training examples in the feature space. It is a type of lazy learning where the function is only approximated locally and all computations are deferred until classification. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (Ameen, Balogun, Usman & Fashoto, 2016).

```
BEGIN
    Input:
    Build the training data set Dt = {(X1, C1),...,
(XN, CN)}
    X =(X1,...,XN) new instance to be classified
    For each labeled instance (Xi, Ci) do
    If X has an unknown system call then
    X is abnormal;
    else then
        For each process Di in training
    data do
        calculate Sim(X, Di);
            if Sim(X, Di) == 1.0 then
                X is normal and
            exit;
        Order Sim(X, Di) from Lowest to
    highest, (i = 1,...,N);
        Find K biggest scores of Sim(X, D);
        Select the K nearest instances to X:
    DkX;
        Assign to x the most frequent class
    in DkX;
        Calculate Sim_Avg for k-nearest
    neighbours;
        If Sim_Avg > threshold then
        X is normal;
        else then
        X is abnormal;
    END
```

Fig. 3: k-Nearest Neighbour Pseudocode
(Source: Ameen, Balogun, Usman & Fashoto, 2016)

3.4 J48 Decision Tree

A Decision Tree Classifier consists of a decision tree generated on the basis of instances. Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). The root and the internal nodes are associated with attributes; leaf nodes are associated with classes. To determine the class for a new instance using a decision tree, beginning with the root, successive internal nodes are visited until a leaf node is reached. At the root node and at each internal node, a test is applied. The outcome of the test determines the branch traversed, and the next node visited. The class for the instance is the class of the final leaf node (Beniwal & Arora, 2012).

```
BEGIN
    Input:
Build the training data set Dt = {(X1, C1),..., (XN, CN)}
    X =(X1,...,XN) new instance to be classified
    For each labeled instance (Xi, Ci) do
    If X has an unknown system call then
    X is abnormal;
    else then
    For each process Di in training data do
        calculate Sim(X, Di);
            if Sim(X, Di) == 1.0 then
                X is normal and
            exit;
    Order Sim(X, Di) from Lowest to highest, (i
    = 1,...,N);
        Find K biggest scores of Sim(X, D);
    Select the Decision instances to X: DkX;
    Assign to x the most frequent class in DkX;
    Calculate Sim_Avg for Decision tree;
        If Sim_Avg > threshold then
        X is normal;
        else then
        X is abnormal;
    END
```

Fig. 4: J48 Decision Tree Pseudocode
(Source: Beniwal & Arora, 2012)

4. RELATED LITERATURE

Although there are many defect predictors in the literature, there are not so many extensive benchmarking studies. Comparing the accuracy of the defect predictors is very important since most of the time the results of one method is not consistent across different datasets (Lessmann, Baesens, Mues, & Pietsch, 2008).

Okutan and Yildiz (2012), proposed a novel method using Bayesian networks to explore the relationships among software metrics and defect proneness. In addition to the metrics used in Promise data repository, two more metrics, i.e. NOD for the number of developers and LOCQ for the source code quality has been proposed. The usefulness of the marginal defect proneness probability of the whole software system, the set of most effective metrics, and the influential relationships among metrics and defectiveness has been derived.

Vipul, Manohar and Sureshchandar (2016), provided a framework which is expected to be more user-friendly, effective and acceptable for predicting the defects in multiple phases across software enhancement projects. A series of empirical experiments were carried out based on input and output measures extracted from 50 'real world' project subsystems. In order to increase the adoption and make the prediction framework easily accessible to project managers, a graphical user interface (GUI) based tool was designed and implemented that allowed input data to be fed easily.

Preetika and Sameer, (2017) showed that the impact of the classification technique is minimum using NASA datasets. Further, they applied this to two new datasets i.e. the cleaned version of the NASA dataset and PROMISE dataset, which contains open source software developed in a variety of settings. The results in these new datasets showed a clear, statistically distinct separation of groups of techniques, i.e., the choice of classification technique has an impact on the performance of defect prediction models.

The main objective of this study is to evaluate the effect of filter feature selection techniques on the performance of heterogeneous classifiers in software defect prediction.

5. METHODOLOGY

5.1 Description of Dataset

The datasets used in this study are Ten public-domain software defect datasets provided by the National Aeronautics Space Administration (NASA) Facility Metrics Data Program (MDP) repository.

5.2 Experimental Architecture

Based on the experimental architecture (Figure 5), each of the dataset was analyzed based on 10 fold cross validation; where the datasets were divided into 10 subsets with 9 subsets used for training the classifier and the remainder one subset for testing the model generated by the classifier. Before the training of the

classifiers, the datasets were pre-processed by applying selected filter feature selection technique (PCA, CFS, FilterSubsetEval) as depicted in Figure 5. Thereafter, the pre-processed datasets were passed into each classifiers based on 10-fold cross validation and the generated models were tested accordingly. The results from the experiments were examined to determine the effect of filter feature selection technique on classifiers in software defect prediction. WEKA data mining tool was used as the environment for the experiment. The datasets were classified by the selected classifiers which were carefully selected based on their characteristics; this study also used popular classifier performance evaluation metrics such as: Accuracy, Precision, Recall and F-measure in its analysis.

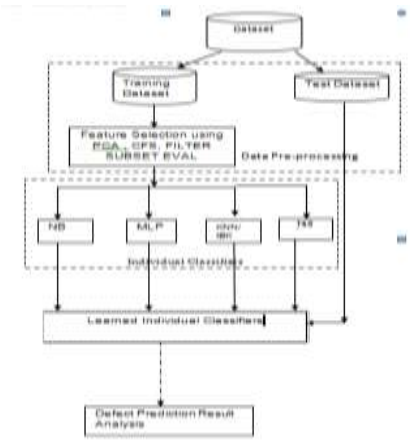

Fig. 5: Experimental Architecture

6. Results and Discussion

From Tables 1, 2, 3 and 4 above, it can be observed that the results of Naïve Bayes, k-Nearest Neighbor (kNN), Multilayer Perceptron (MLP) and Decision Tree (J48) with the preprocessed data, that is, the dataset that have been reduced using each of the selected filter feature selection methods (PCA, CFS and FilterSubsetEval, gave a better result compared to when the datasets were not pre-processed.

Table 1: Experimental result of Naïve Bayes on the datasets

		CFS	FILTER	PCA
Metrics	**Full Data sets**	**Reduced Datasets**	**Reduced Datasets**	**Reduced Datasets**
Accuracy	82.51%	87.61%	87.60%	86.95%
Precision	0.8768	0.8787	0.8787	0.8659
Recall	0.8252	0.8761	0.876	0.8659
F-Measure	0.8357	0.8727	0.8727	0.8647

Table 2: Experimental result of KNN on the datasets

		CFS	FILTER	PCA
Metrics	**Full Data sets**	**Reduced Datasets**	**Reduced Datasets**	**Reduced Datasets**
Accuracy	87.32%	87.27%	87.40%	86.79%
Precision	0.8686	0.8692	0.8697	0.8617
Recall	0.8733	0.8729	0.8742	0.8678
F-Measure	0.8703	0.8706	0.8715	0.8643

Table 3: Experimental result of Multilayer Perceptron on the datasets

		CFS	FILTER	PCA
Metrics	**Full Datasets**	**Reduced Datasets**	**Reduced Datasets**	**Reduced Datasets**
Accuracy	88.96%	89.90%	89.94%	89.13%
Precision	0.8758	0.8776	0.8763	0.8754
Recall	0.8896	0.8991	0.8995	0.8913
F-Measure	0.8777	0.8809	0.8815	0.878

Table 4: Experimental result of Decision Tree on the datasets

		CFS	FILTER	PCA
Metrics	**Full Datasets**	**Reduced Datasets**	**Reduced Datasets**	**Reduced Datasets**
Accuracy	88.42%	89.28%	89.33%	89.58%
Precision	0.8703	0.8697	0.8703	0.8779
Recall	0.8841	0.8929	0.8934	0.8959
F-Measure	0.8751	0.8737	0.8741	0.8759

Naïve Bayes classifier for full dataset had the average accuracy of 82.51% while for the preprocessed datasets; Naïve Bayes with CFS had 87.61%, Naïve Bayes with FilterSubEval had 87.60% and Naïve Bayes with PCA had 86.95% respectively. KNN classifier had the average accuracy of 87.32% for full dataset and for the preprocessed datasets: KNN with CFS had 87.28%, Filter had 87.40% and PCA had 86.79% . When the dataset were preprocessed with MLP classifier had the average accuracy of 88.96%, MLP with CFS had 89.90%, MLP with FilterSubsetEval had 89.94% while MLP had PCA had 89.13226%, J48 classifier full dataset had 88.4178% of average accuracy,J48 with CFS had 89.28%, J48 with filter subset eval had 89.33% and J48 with PCA had 89.58% respectively. From the results, MLP classifier with FilterSubsetEval has the highest average accuracy of 89.94%, this also showed that feature selection is a good technique to be applied on datasets in software defects prediction.

7. Conclusion and Future Work

This study looked into the effect of filter feature selection on classifiers in software defect prediction. Filter feature selection algorithm such Principal Component Analysis (PCA), Filter Subset Evaluation (FSE) and Correlation Feature Selection Subset Evaluation (CFS) on Classifiers such as Naïve Bayes (NB), Decision Tree(J48), MLP and K-Nearest Neighbour were applied on ten NASA datasets (KC1, KC2, KC3, MC1, MC2, MW1, PC1, PC2,

PC3, PC4). The datasets were classified by the selected classifiers which were carefully selected based on their characteristics. Naïve Bayes was selected from Bayes category Classifier, KNN was selected from Instance Based Learner category, J48 Decision Tree from Trees Function classifier and Multilayer perceptron was selected from the neural network classifiers. From the results, it can be seen that the application of feature selection to datasets before classification in software defects prediction is better and should be encouraged and Multilayer perceptron with FilterSubsetEval had the best accuracy. It can be concluded that feature selection methods are capable of improving the performance of learning algorithms in software defects prediction by removing the irrelevant or redundant attributes from the data before the classification process. The researchers intend to extend this work by considering Multi Criteria Decision Making Method to evaluate the performance metric in order to get the best model or classifier for software defects prediction.

REFERENCES

Ahmad, A. K., & Nashat, M. (2012). Metaheuristic Optimization Algorithms for Training Artificial Neural Networks. *International Journal of Computer and Information Technology, 1*(2). 1-6.

Ameen, A. O., Balogun, A. O., Usman, G. & Fashoto, S. G. (2016): Heterogenous Ensemble Methods Based On Filter Feature Selection. *Computing, Information System Development Informatics & Allied Research Journals.* Vol 7 No 4. Pp 63-78.

Aruna, S., Dilsha, D., Radhika, R., & Swathi, J.N. (2016). Cost Sensitive Classification and Feature Selection for Software Defect Prediction. *International Journal of Advanced Research in Computer Science and Software Engineering. 6*(4), 1-2.

Balogun, A. O., Mabayoje M. A., Salihu, S. & Arinze, S.A. (2015): Enhanced Classification Via Clustering Using Decision Tree for Feature Selection. *International Journal of Applied Information Systems (IJAIS).* 9(6):11-16.

Beniwal, S., & Arora, J., (2012). Classification and Feature Selection Techniques in Data Mining. *International Journal of Engineering Research & Technology (IJERT)*, Vol. 1 Issue 6, August–2012.

Fenton, N. E., & Pfleeger, S.L., (1996). Software Metrics: A Rigorous and Practical Approach, *2nd ed. International Thomson Computer* Press.

Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). A Systematic Literature Review on Fault Prediction Performance in Software Engineering. *IEEE Transactions on Software Engineering*, *38*(6), 1276–1304.

Ishani, A., Vivek, T., & Anju, S. (2015). Open Issues in Software Defect Prediction. *International Conference on Information and Communication Technologies, 46*. 1-2, doi:10.1016/j.procs.2015.02.161

Khoshgoftaar, T. M., Bullard, L. A., & Gao, K.,(2009). Attribute selection Using Rough Set in Software Quality Classification. *International Journal of Reliability, Quality and Safety Engineering* 16(1). 73 – 89.

Khoshgoftaar, T. M., Gao, K., Napolitano, A., & Wald, R. (2013). A Comparative Study of Iterative and Non-Iterative Feature Selection Techniques For Software Defects Prediction. *Information system Frontiers* 16(5). 73 – 89. DOI: 10.1007/s10796-013-9430-0

Koru, A. G. & Liu, H. (2005). An investigation of the effect of module size on defect prediction using static measures, *SIGSOFT Software Engineering Notes,*Vol.30, No.4,pp.1-5

Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings. *IEEE Transactions on Software Engineering*, *34*(4), 485–496.

Miao, L., Liu, M., & Zhang, D., (2012). Cost Sensitive Feature Selection with application in software defect prediction. *International Conference on Pattern Recognition,* Tsukuba, Japan, p.967-970.

Okutan, A. & Yıldız, O. T. (2012). Software defect prediction using Bayesian networks. *Empirical Software Engineering, (2012), pp. 1-28.*

Preetika, V. & Sameer, A. (2017). Analysing Software Defect Prediction Techniques – A Review of Literature. *International Journal of Research and Development in Applied Science and Engineering* (IJRDASE). ISSN: 2454-6844

Shivaji, S., Whitehead, E., Akella, R., & Kim, S. (2009). Reducing Features to Improve Bug Prediction, Automated Software Engineering, 2009. ASE'09.24th IEEE/ACM International Conference on, pp.600-604..

Tina, R. & Sherekar, S. (2013). Performance Analysis of Naive Bayes and J48 Classification Algorithm for Data Classification. *International Journal of Computer Science and Applications Vol. 6, p 2.*

Vipul, V., Manohar, L., & Sureshchandar, G. S. (2016), Defect Prediction Framework Using Neural Networks for Software Enhancement Projects. *British Journal of Mathematics & Computer Science* .16(5): 1-12.