Defence Science Journal, Vol. 61, No. 4, July 2011, pp. 306-316
© 2011, DESIDOC

Predicting Software Faults in Large Space Systems using Machine Learning Techniques

Bhekisipho Twala

University of Johannesburg, Johannesburg, South Africa
E-mail: btwala@uj.ac.za

ABSTRACT

Recently, the use of machine learning (ML) algorithms has proven to be of great practical value in solving a variety of engineering problems including the prediction of failure, fault, and defect-proneness as the space system software becomes complex. One of the most active areas of recent research in ML has been the use of ensemble classifiers. How ML techniques (or classifiers) could be used to predict software faults in space systems, including many aerospace systems is shown, and further use ensemble individual classifiers by having them vote for the most popular class to improve system software fault-proneness prediction. Benchmarking results on four NASA public datasets show the Naive Bayes classifier as more robust software fault prediction while most ensembles with a decision tree classifier as one of its components achieve higher accuracy rates.

Keywords: Software metrics, machine learning, classifiers, ensemble, fault-proneness prediction

1. INTRODUCTION

The growing demand for higher operational efficiency and safety in industrial processes has resulted in a huge interest in fault-detection techniques. Engineering researchers and practitioners remain concerned with accurate prediction when building systems. However, software fault or defect prediction remains the most popular research area. Software fault prediction has both safety and economic benefits in technical systems by preventing future failures and further improves process maintenance schedules. Lack of adequate tools to estimate and evaluate the cost for a software system failure is one of the main challenges in systems engineering. They used datasets of past projects to build and validate estimation or prediction systems of software development efforts, for example, which allows them to make management decisions, such as resource allocation. Or they may use datasets of measurements describing software systems to validate metrics predicting quality attributes. The techniques they used to build models to measure or predict, on the other hand, often requires good quality of data.

When using machine learning (ML) techniques to build such prediction systems, poor data quality in either training or test (unknown) set or in both sets, can affect prediction accuracy. Various ML techniques (e.g., supervised learning) have been used in systems engineering to predict faults or defects[1], software (project) development effort[2,3], software quality[4], and software defects[5,6]. Reviews of the use of ML in software engineering[7,8] report that ML in software engineering is a mature technique based on widely-available tools using well understood algorithms. The decision tree (DT) classifier is an example of a ML algorithm that can be used for predicting continuous attributes (regression) or categorical attributes (classification). Thus, software prediction can be cast as a supervised learning problem, i.e., the process of learning to separate samples from different classes by finding common features between samples of known classes. An important advantage of ML over statistically-based approaches as a modelling technique lies in the fact that the interpretation of, say, decision rules, is more straightforward and intelligible to human beings than, say, principal component analysis (a statistical tool for finding patterns in data of high dimension). In recent years, there has been an explosion of papers in the ML and statistics communities discussing how to combine models or model predictions.

One of the major problems for applying ML algorithms in fault or failure prediction is the (sometimes) unavailability and scarcity of software data, i.e., data for training the model. Most of the companies that own (or control) do not share their failure data from space systems due to corporate propriety interests and national security concerns so that a useful database with a great amount of data cannot be developed.

Most techniques for predicting attributes of a large space system require past data from which models will be constructed and validated. Often data is collected either with no specific purpose in mind (i.e., it is collected because it might be useful in future) or the analysis being carried out has a different goal than that for which the data was originally collected. The relevance of this issue is strictly proportional to the dimensionality of the collected data. Thus, accurate prediction of software faults remains a priority among empirical engineering researchers.

Many works in both the ML and statistical pattern recognition communities have shown that combining

Received 23 March 2011, published online 25 July 2011

(ensemble) individual classifiers is an effective technique for improving classification accuracy. An ensemble is generated by training multiple learners for the same task and then combining their predictions. There are different ways in which ensembles can be generated, and the resulting output combined to classify new instances. The popular approaches to creating ensembles include changing the instances used for training through techniques such as bagging[9], boosting[10], changing the features used in training[11], and introducing randomness in the classifier itself[12]. The interpretability of classifiers can produce useful information for experts responsible for making reliable classification, making decision trees an attractive scheme.

Few studies have been carried out on the effect of the top classifiers in data mining. Wu[13], *et al*. studies that the effect of ensemble classifiers on software faults predicting accuracy in space systems are rare in engineering.

2. MACHINE LEARNING ALGORITHMS

In supervised learning, especially for multivariate data, a classification function $y = f(x)$ from training instances of the form $\{(x_1, y_1),..,(x_m, y_m)\}$, predicts one (or more) output attribute(s) or dependent variable(s) given the values of the input attributes of the form $\{x, f(x)\}$. The x_i values are vectors of the form $\{x_{i1},..x_{im}\}$ whose components can be numerically ordered, nominal or categorical, or ordinal. The y values are drawn from a discrete set of classes $\{1,\ldots K\}$in the case of classification. Depending on the usage, the prediction can be definite or probabilistic over possible values.

Given a set of training examples and any given prior probabilities and misclassification costs, a learning algorithm outputs a classifier. The classifier is a hypothesis about the true classification function that is learned from or fitted to training data. The classifier is then tested on test data. A wide range of algorithms, in both classical statistics and from various ML paradigms, have been developed for this task of supervised learning classification.

2.1 Apriori

The Apriori is a classical data mining algorithm used for learning association rules[14]. It calculates rules that express probabilistic relationships between items in frequent item-sets. For example, a rule derived from frequent item-sets containing A, B, and C might state that if A and B are included in a transaction, then C is likely to be included.

An association rule states that an item or group of items implies the presence of another item with some probability. For an example, a rule like: If a customer buys wine and bread, he/she often buys cheese, too. It expresses an association between (sets of) items, which may be products of a supermarket or a mail-order company; special equipment options of a car; optional services offered by telecommunication companies, etc. An association rule states that if we pick a customer at random and find out that he/she selected certain items (bought certain products, chose certain options, etc.), we can be confident, quantified by a percentage, that he/she also selected certain other items (bought certain other products, chose certain other options, etc.). Of course, they do not want just any association rules; they want good rules, rules that are expressive and reliable. The standard measures to assess association rules are the support and the confidence of a rule, both of which are computed from the support of certain item-sets. Unlike decision tree rules (described in Section 4.2), which predict a target, association rules simply express a correlation.

2.2 Decision Trees

Decision tree (DT) induction is one of the simplest and yet one of the most successful forms of supervised learning algorithm. It has been extensively pursued and studied in many areas such as statistics[15], and ML[16] for the purposes of classification and prediction.

Decision tree classifiers have four major objectives, these are:

(i) To classify correctly as much of the training sample as possible.
(ii) Generalise beyond the training sample so that unseen samples could be classified with as high accuracy as possible.
(iii) Be easy to update as more training samples become available (i.e., be incremental), and
(iv) Have as simple a structure as possible.

Objective (i) is actually highly debatable and not all tree classifiers are concerned with objective (iii).

Decision trees are non parametric (i.e., no assumptions about the data are made) and a useful means of representing the logic embodied in software routines. A decision tree takes as input a case or example described by a set of attribute values, and outputs a Boolean or multi-valued decision. For the purpose of this paper, we shall stick to the Boolean case.

A classification tree (which is what will be covered in this paper) as opposed to a regression tree means that the response variable is qualitative rather than quantitative. In the classification case, when the response variable takes value in a set of previously-defined classes, the node is assigned to the class which represents the highest proportion of observations. Whereas, in the regression case, the value assigned to cases in a given terminal node is the mean of the response variable values associated with the observations belonging to a given node. Note that in both cases, this assignment is probabilistic, in the sense that a measure of error is associated with it.

2.3 *k*-Nearest Neighbour

One of the most venerable algorithms in ML is the nearest neighbour (NN). Nearest-neighbour methods are sometimes referred to as memory-based reasoning or instance-based learning (IBL) or case-based learning (CBL) techniques and have been used for classification tasks. They essentially work by assigning to an unclassified sample point the classification of the nearest of a set of previously-classified points.

The entire training set is stored in the memory. To classify a new instance, the Euclidean distance (possibly weighted) is computed between the instance and each stored training instance and the new instance is assigned the class of the nearest neighbouring instance. More generally, the *k*-nearest neighbours are computed, and the new instance is assigned the class that is most frequent among the *k* neighbours (which from now onwards shall be denoted as *k*-NN). IBL's have three defining general characteristics; a similarity function (how close together the two instances are), a typical instance selection function (which instances to keep as examples), and a classification function (deciding how a new case relates to the learned cases).

A further non-parametric procedure of this form is the *k*-NN approach. To classify an unknown pattern, the *k*-NN approach looks at a collection of the *k* nearest points and uses a voting mechanism to select between them, instead of looking at the single nearest point and classifying according to that with ties broken at random. If there are ties for the k^{th} nearest observations, all candidates are included in the vote.

2.4 Naïve Bayes Classifier

There are two roles for Bayesian methods: (i) to provide practical learning algorithms such as Naïve Bayes learning and Bayesian belief network learning by combining prior knowledge with observed data and (ii) to provide a useful conceptual framework that could provide a gold standard for evaluating other learning algorithms.

Bayesian learning algorithms use probability theory as an approach to concept classification. Bayesian classifiers produce probabilities for (possibly multiple) class assignments, rather than a single definite classification. Bayesian learning should not be confused with the Bayes optimal classifier. Also, Bayesian learning should not be confused with the Naïve Bayes or idiot's Bayes classifier, which assumes that the inputs are conditionally independent given the target class.

The naïve Bayes classifier is usually applied with categorical inputs, and the distribution of each input is estimated by the proportions in the training set; hence the naïve Bayes classifier (NBC) is a frequentist method.

The NBC is perhaps the simplest and most widely studied probabilistic learning method. It learns from the training data, the conditional probability of each attribute A_i, given the class label *C*. The strong major assumption is that all attributes A_i are independent given the value of the class *C*. Classification is therefore done applying Bayes rule to compute the probability of *C* given $A_1,...,A_n$ and then predicting the class with the highest posterior probability. The assumption of conditional independence of a collection of random attributes is critical. Otherwise, it would be impossible to estimate all the parameters without such an assumption.

2.5 Support Vector Machines

Support vector machines (SVMs) are pattern classifiers that can be expressed in the form of hyperplanes to discriminate positive instances from negative instances pioneered by Vapnik[39]. Motivated by statistical learning theory, SVMs have successfully been applied to numerical tasks, including classification. These can perform both binary classification (pattern recognition) and real-valued function approximation (regression estimation) tasks. These perform structural risk minimisation (also used in other systems such as neuro-fuzzy) and identify key support vectors (the points closest to the boundary). Risk minimisation measures the expected error on an arbitrarily large test set with the given training set and supports vector machines non-linearly map their *n*-dimensional input space into a high dimensional feature space. In this high dimensional feature space, a nonlinear classifier is constructed.

Support vector machines have been recently proposed as a new technique for pattern recognition. Intuitively, given a set of points which belong to either of the two classes, a linear SVM finds the hyperplane leaving the largest possible fraction of points of the same class on the same side, while maximising the distance of either class from the hyperplane. The hyperplane is determined by a subset of the points of the two classes, named support vectors, and has a number of interesting theoretical properties.

2.6 Ensemble Procedure

A motivation for ensemble is that a combination of outputs of many weak classifiers produces powerful ensembles with higher accuracy than a single classifier obtained from the same sample. The ensemble then makes use of all data available and utilises a systematic pattern of classification results. The generalised ensemble algorithm is summarised in Fig. 1, with the overall six-stage scheme of the technique shown in Fig. 2 and described in more detail in the following sub-sections.

1. Partition original dataset into n training datasets, TR1, TR2,…,TRn.
2. Construct n classifiers (CF1, CF2, …, CFn) with the different training datasets TR1, TR2, …, TRn to obtain n individual classifiers (ensemble members) generated by different machine learning algorithms, thus different.
3. Select m de-correlated classifiers using de-correlation maximisation algorithm.
4. Using Eqn. (3), obtain m classifier output values (misclassification error rates) of unknown instance.
5. Transform the output value to reliability degrees of positive class and negative class, given the imbalance of some datasets
6. Fuse the multiple CFs into aggregate output in terms of majority voting. When there is a tie in the predicted probabilities, choose the class with the highest probability, or else, use a random choice when the probabilities between the two methods are equal.

Figure 1. The ensemble algorithm.

2.6.1 Partitioning Original Dataset

Due to the shortage in some data analysis problems, such approaches, such as bagging[9,12] have been used for creating samples varying the data subsets selected. The bagging algorithm is very efficient in constructing a reasonable size of training set due to the feature of its random sampling with replacement. Therefore such a strategy (which we also use in this study) is a useful data preparation method for ML.

2.6.2 Creating Diverse Classifiers

Diversity in ensemble of learnt models constitute one

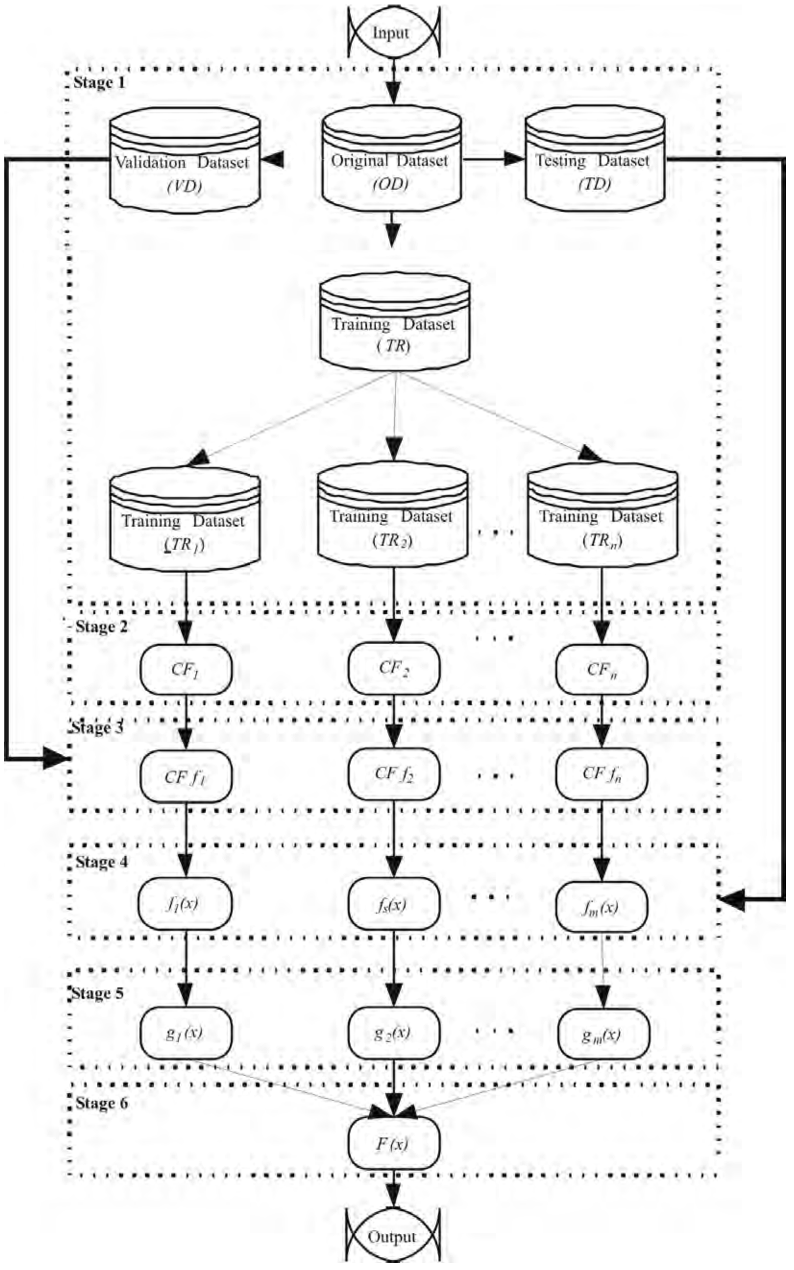

Figure 2. General formation process of missing data ensemble learning model.

of the main current directions in ML and data mining. It has been shown theoretically and experimentally that in order for an ensemble to be effective, it should consist of high-accuracy base classifiers that should have high diversity in their predictions. One technique, which has been proved to be effective for constructing an ensemble of accurate and diverse base classifiers, is to use different training data, or so-called ensemble training data selection. Many ensemble training data selection strategies generate multiple classifiers by applying a single learning algorithm, to different versions of a given dataset, Two different methods for manipulating the dataset are normally used: (i) random sampling with replacement (also called bootstrap sampling) in bagging, and (ii) re-weighting of the misclassified training instances in boosting. The authors have used bagging in this study.

2.6.3 Selecting Appropriate Ensemble Members

After training, each individual classifier grown has generated its own result. However, if there is a large number of individual members (i.e., classifiers), one needs to select a subset of representatives in order to improve ensemble efficiency. Furthermore, it does have to follow the rule 'the more the better' rule as mentioned by some researchers. In this study, a de-correlation maximisation method[17,18] is used to select the appropriate number of ensemble members. The idea is that the correlations between the selected classifiers should be as small as possible. The de-correlation matrix can be summarised in the following steps:

1. Compute the variance-covariance matrix and the correlation matrix;
2. For the ith classifier ($i = 1,2,\ldots, p$), calculate the plural-correlation coefficient η_i;
3. For a pre-specified threshold θ, if $\eta_i < \theta$, then the ith classifier should be deleted from the p classifiers. Conversely, if $\eta_i > \theta$, then the ith classifier should be retained;
4. For the retained classifiers, perform Eqns (1)-(3) procedures iteratively until satisfactory results are obtained.

2.6.4 Performance Measure Evaluation

In the previous phase, the classifier outputs are used as performance evaluation measures (in terms of misclassification error rates). It has often been argued that selecting and evaluating a classification model based solely on its error rates is inappropriate. The argument is based on the issue of using both the false positive (rejecting a null hypothesis when it is actually true) and false negative (failing to reject a null hypothesis when it is in fact false) errors as performance measures whenever classification models are used and compared. Furthermore, in the business world, decisions (of the classification type) involve costs and expected profits. The classifier is then expected to help making the decisions that will maximise profits.

For example, predicting faults or failures or defects of software systems involves two types of errors: (i) predicting software faults as likely to be high when in fact these are low, and (ii) predicting software faults as likely to be low when in fact these are high. Now, mere misclassification rate is simply not good enough to predict software effort. To overcome this problem and further make allowances for the inequality of mislabelled classes, variable misclassification costs are incorporated in our attribute selection criterion via prior specification for all our experiments. This also solves the imbalanced data problem. Details about how misclassification costs are used for both splitting and pruning rules are presented by Breiman[19], *et al.*

2.6.5 Integrating Multiple Classifiers into Ensemble Output

Depending upon the work in the previous stages, a set of appropriate number of ensemble members can be identified. The subsequent task is to combine these selected members into an aggregated classifier in an appropriate ensemble strategy. Common strategies to combine these single DT results and then produce the final output are simple averaging; weighted averaging, ranking and majority voting*.

- *Simple averaging* is one of the most frequently used combination methods. After training the members of the ensemble, the final output can be obtained by averaging the sum of each output of the ensemble members. Some experiments have shown that simple averaging is an effective approach[20].
- *Weighted averaging* is where the final ensemble result is calculated based on individual ensemble members' performances and a weight attached to each individual member's output. The gross weight is 1 and each member of an ensemble is entitled to a portion of this gross weight according to its performance or diversity.
- *Ranking* is where members of the ensemble are called low-level classifiers and they produce not only a single result but a list of choices ranked according to their likelihood. Then the high-level classifier chooses from these set of classes using additional information that is not usually available to or well represented in a single low-level classifier.
- *Majority voting* is the most popular combination method for classification problems because of its easy implementation. Members of trees voting decide the value of each output dimension. It takes over half the ensemble to agree a result for it to be accepted as the final output of the ensemble (regardless of the diversity and accuracy of each tree generalisation). Majority voting ignores the fact that some trees that lie in a minority sometimes do produce correct results. However, this is the combination strategy approach which has been followed in this study.

3. RELATED WORK

Significant advances have been made in the past few decades regarding methodologies which handle the prediction of software faults, failures, and defects. Unfortunately, these methodologies are often not available to many researchers for a variety of reasons (for example, lack of familiarity, computational challenges). As a result, researchers often resort to adhoc approaches in dealing with the software faults. Nonetheless, several researchers have still examined various statistically-based and machine learning approaches to solve the problem in engineering. Specific results are discussed.

Marwala and Hunt[21] use vibration data to identify faults in structures. The idea is to use a committee of neural networks which employ both frequency response functions and modal data simultaneously in order to identify faults in structures. Their proposed approach is tested on simulated data and it shows very promising results. Marwala[22] follows up this approach by proposing a Bayesian formulated neural network approach using hybrid Monte Carlo to scaled conjugate gradient method for indentifying faults in structures using modal properties and the coordinate models assurance criterion and vibration data. Marwala argues that such an approach gives identities of damage and their respective standard deviations. The proposed approach gives more accurate fault identification results than the results given using the existing approaches although the proposed approach was found to be computationally expensive.

An anomaly detection method for spacecraft system based on association rules is proposed by Yairi[23], *et al*. The method constructs a system behaviour model in the form of a set of rules by applying pattern clustering and association rule mining using past housekeeping data of engineering test satellite and time series data. The proposed approach has the advantage of not requiring prior knowledge on the system. Thus, it can be applied to various kinds of spacecrafts at small costs. The association-rules-based approach compares favourably with existing data mining methods but has the slight advantage of detecting some anomalies which (otherwise) could have been overlooked by conventional approaches.

Tree-based models were used by Koru and Tian[24] to investigate the relationship between high defect and high complexity software modules in six large-scale products of IBM and Nortel Networks. The study was conducted on 15 method-level metrics for IBM products and 45 method-level products for Nortel Networks. They provided evidence of high defect-prone modules as generally complex modules and also locate below the most complex modules.

The pioneering work by Guo[25], *et al*. performed a comprehensive simulation study to evaluate three techniques in the context of software faults modelling using NASA's KC2 project dataset. These techniques are Bayesian networks (BN), logistic regression (LR) and discriminant analysis (DA). Their results show BN as not only being more robust to software faults prediction but also as a more cost-effective strategy. Their results further show LR as a better method for choosing the best software metrics that are more prone to faults.

Menzies[26], *et al*. performed a simulation study comparing two machine learning methods for software detection prediction with the probability of detection and the probability of false alarm as their performance evaluation metrics. Based on the National Aeronautics and Space Administration (NASA) dataset, their results showed the NBC performing better than the J48 algorithm. Menzie[27] followed up this work by carrying out a comparative evaluation (in terms of software fault prediction) between linear regression, trees, ROCKY and Delphi detectors. NASA datasets were used for this task. Their results showed ROCKY performing better than the other methods.

Fujimaki[28], *et al*. propose novel anomaly detection method for a spacecraft system that is based on kernel feature (attribute) space and directional distribution which constructs a system behaviour model using telemetry data obtained from a simulator of an orbital transfer vehicle designed to make a rendezvous manoeuvre with the internal space station. The effectiveness of the method shows promising results

* For more information on these strategies, the reader is referred to Dietterich[12], which are otherwise briefly discussed below.

Another comparative study of J48, KStar, artificial neural networks (ANN), Bayesian networks, and SVM in the context of software fault estimation was carried out by Koru and Liu[29] who suggested using fault predictors on large components. The simulation study was carried out using public NASA datasets for both method and class-levels, with the F-ratio as the performance measure. Their results show J48 exhibiting higher accuracy rates while methods such as BN, ANN and SVM struggled. In addition, most of the methods did not perform well whenever there were small components in the datasets. In their next study, Koru and Liu[29], built prediction models using J48, KStar and random forests on NASA datasets. Both method-level and class-level metrics were used. They showed how class-level metrics improved model performance. Detection of faults was also shown to be at class-level instead of model level.

The use of ML for the purposes of predicting or estimating a software module's fault-proneness is proposed by Gondra[30] who views fault-proneness as both a continuous measure and a binary classification task. An ANN is used to predict the continuous measure with SVM used for the classification task. A comparative study of the effectiveness of both methods was then performed on a NASA public dataset. The experimental results confirmed the superior performance of SVMs over ANNs.

The performance of LR, DA, DT, boosting, kernel density, NBC, J48, IBk, voted perceptron, VF1, hyper-pipes and random forest techniques was analysed by Ma[31], *et al.* using NASA datasets with g-mean1, g-mean2 and F-ratio as performance evaluation metrics. Their results shows balanced random forests yielding good results (especially on large datasets) with boosting, rule-set and DTs being less robust methods for software fault prediction.

Challagulla[32], *et al.* evaluated memory-based reasoning (MBR) as a strategy for predicting software faults using NASA datasets and 21 method-level metrics. The probability of detection, the probability of false alarm and accuracy were used as performance evaluation metrics. Their results show promise, especially when a framework based on MBR configuration is used.

LR, NBC, random forests, and *k*-NN with generalisation techniques were evaluated using NASA's KC1 datasets in the software fault context by Zhou and Leung[33]. Their results showed better performances by methods for low-severity faults compared to high-severity faults. They concluded that other Chidamber-Kemerer metrics could also be useful for fault prediction.

The use of the NBC and the J48 algorithm for predicting software faults on a NASA dataset using method-level metrics was investigated by Menzies[34] *et al*. The performance evaluation metrics used were the probability of faults and the probability of defects. Their results showed NBC as more efficient with the dataset characteristics playing a major role in terms of performance for both algorithms.

Catal and Diri[34] evaluate the impact of random forests and algorithms based on artificial immune systems with the area under the receiver operating characteristics (ROC) curve used as a performance evaluation measure. NASA datasets were utilised for this task. Their results show random forests achieving the highest accuracy rates with other notably good performances by methods such as NBC (especially for small datasets). Mendes and Koschke[35] evaluated several data mining algorithms based on fault prediction using 13 NASA datasets. Using the area under the ROC curve as the performance measure, they could not find any statistically significant difference in terms of performance among the algorithms.

According to the above studies, it appears that there is currently reasonable data to model software fault prediction although the use of public datasets by researchers. Secondly, method-level metrics appear less to be dominant in software fault prediction with class-level metrics hardly utilised. Also, ML algorithms are still the most popular methods compared to either the statistical methods or the expert opinion-based approaches. However, among some of the ML algorithms, the results are not so clear, especially for large amounts of data. It also appears that the poor and good performances of each algorithm are highly dependent on each respective dataset characteristics. In other words, the nature of the attributes determines the applicability of fault-detection techniques.

4. EXPERIMENTAL SET-UP

4.1 Experiment I

To empirically evaluate the performance of one of the top five classifiers in data mining (AR, DT, *k*-NN, NBC and SVM), an experiment was conducted on four datasets in terms of misclassification error rate. For each dataset, different types of metrics are used to predict modules that were likely to harbour faults. In other words, to carry out the experiments, they relate individual requirements with software modules.

Three of the four datasets were collected by the NASA metrics data program (MDP) data repository (http://mdp.ivv.nasa.gov). They comprised three projects (CM1, JM1 and PC1) of which only partial requirement metrics are available. There are 10 attributes that described the requirements. By definition, CM1 described software artifacts of a NASA spacecraft instrument; JM1 represented a real-time ground system that uses simulations to generate flight predictions; PC1 refers to an Earth orbiting satellite software system. A combination of the requirement metric and static code metrics were used in their experiments with CM1 eventually having 266 instances; JM1 having 97 instances; and PC1 having 477 instances.

The other dataset was drawn from an institutional database of anomaly reports for multiple missions managed for NASA by the NASA's Jet Propulsion Laboratory (JPL/NASA)[16]. The reporting mechanism for the anomalies is a report form called incident/surprise/anomaly (ISA). Incident/surprise/anomaly is written whenever the behaviour of the system differs from the expected behaviour as judged by the operator. The dataset consisted of 199 critical ISAs from seven NASA spacecraft that occurred between the launch date of each spacecraft and 21 August 2001.

To perform the experiment, each complete dataset was split randomly into 5 parts (Part I, Part II, Part III, Part IV, Part V) of equal (or approximately equal) sizes. Five-fold cross validation was used for the experiment. For each fold, four of the parts of the instances in each category were placed in the training set, and the remaining one was placed in the corresponding test as shown in Table 1.

Table 1. Partitioning of dataset to training and test sets

	Training set	Test set
Fold 1	Part II + Part III + Part IV + Part V	Part I
Fold 2	Part I + Part III + Part IV + Part V	Part II
Fold 3	Part I + Part II + Part IV + Part V	Part III
Fold 4	Part I + Part II + Part III + Part V	Part IV
Fold 5	Part I + Part II + Part III + Part IV	Part V

They construct the predictive models using five classifiers from the WEKA toolkit[36]. The WEKA is an ensemble of tools for data classification, regression, clustering, association rules, and visualisation. The toolkit is developed in Java and it is a open source software. All the five classifiers are used with their default settings (and in some cases, control parameters) as implemented in WEKA. Furthermore, both the WEKA library and the NASA datasets are publicly available. Thus, their results can be easily checked and reproduced.

To measure the performance of these classifiers, the training set-test methodology is employed, i.e., each classifier is built on the training data and the predicted accuracy is measured by the smoothed error rate estimated on the test data. Instead of summing terms that are either 0 or 1 as in the error-count estimator, the smoothed error rate uses a continuum of values between 0 and 1 in terms that are summed. The resulting estimator has a smaller variance than the error-count estimate. Also, the smoothed error rate can be very helpful when there is a tie between two competing classes.

Another point to note is the reason for using difference in error rates when making comparisons between the classifiers instead of, say, division or ratios of error rates. First, differences are natural and on understandable scale in this context, that is, people would understand a *p* per cent point worsening in error rate to mean a simple addition of *p* per cent. Secondly, ratios of error rates would lead to statements like *A* increases error rate by *p* per cent which would be misinterpreted as meaning a *p* per cent difference in error rate. Finally, the analysis of variance assumes the error rates to be on an additive rather than multiplicative scale.

All statistical tests were conducted using the MINITAB statistical software program[37]. Analyses of variance, using the general linear model (GLM) procedure[38] were used to examine the main effects and their respective interactions. This was done using a repeated measures design (where the effect was tested against its interaction with datasets). The fixed effect factor is the five classifiers. The four datasets were used to estimate the smoothed error rate. The results were averaged across five folds of the cross-validation process before carrying out the statistical analysis. The averaging was done as a reduction in error variance benefit.

4.1.1 Results

Experimental results on the effects of five classifiers on software faults predictive accuracy have been described. The error rates for each classifier as a method for handling software faults were initially averaged over the four datasets. Then results on the effects of five classifiers for individual datasets were presented. Also, all the error rates increase over the complete data case are formed by taking differences.

From Fig. 3, NBC is the overall winner as a fault prediction technique with an excess error rate of 22.4 per cent, followed by DT, SVM and *k*-NN, with excess error rates of 27.3 per cent, 29.3 per cent and 30.0 per cent, respectively. The worst technique is AR, which exhibits an error rate of 32.1 per cent.

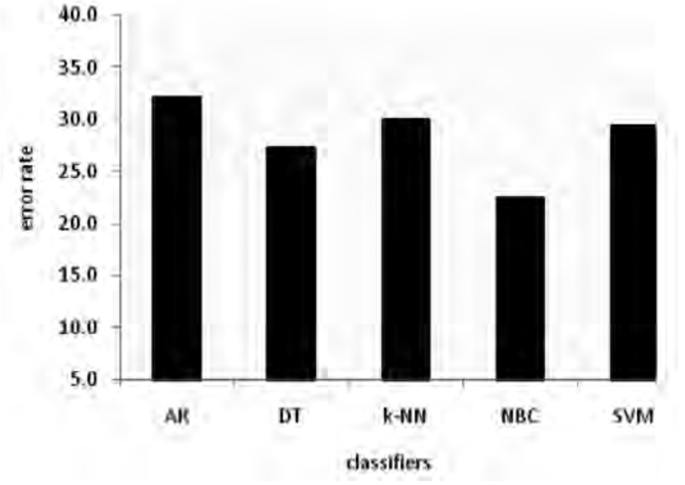

Figure 3. Performance of classifiers.

Tukey's multiple comparison tests showed significant differences between most of the classifiers (with the exception of SVM and *k*-NN). The significance level for all the comparison tests was 0.05.

The results for the performances of the five classifiers for individual datasets has been given in Fig. 4. From Fig. 4, the following results are observed.

- For the PC1 data problem, it appears that NBC predicts software faults better than the other methods with an accuracy rate of 78.7 per cent followed by DT (75.3 per cent), SVM (73.3 per cent), *k*-NN (71.7 per cent) and AR (67.1 per cent). Multiple comparison tests showed significant differences in performances amongst all the methods at the 5 per cent level.
- The results for the JM1 data problem shows NBC (once again) exhibiting the highest accuracy rate (76.6 per cent), followed by SVM (72.9 per cent), *k*-NN (70.7 per cent) and DT (68.0 per cent). The worst performance was by AR with accuracy rate of 65.7 per cent. Once again, multiple comparison tests showed the five methods as significantly different from each other at the 5 per cent level.
- The results presented for the CM1 data problem shows the NBC achieving an accuracy rate of 74.9 per cent, followed by DT (70.9 per cent), SVM (69.9 per cent), *k*-NN (66.8 per cent) and AR (64.4 per cent). Four of the five classifiers achieve a bigger error rates for this

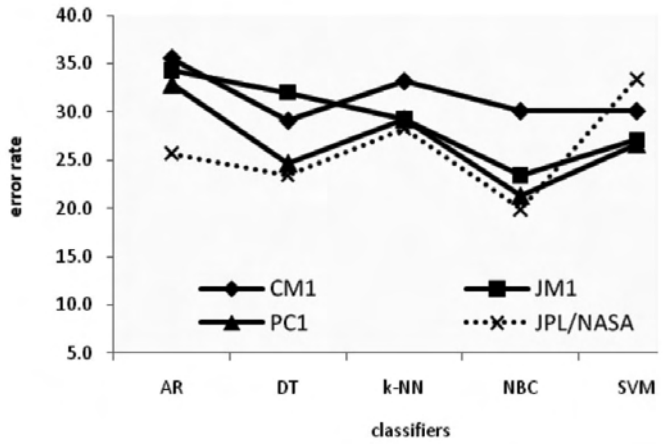

Figure 4. Performance of classifiers (individual datasets).

kind of dataset compared with rates achieved for the other datasets. No significant difference in performance between DT and SVM was observed at the 5 per cent level.

- For the JP/NASA data problem, four of the classifiers achieved the smallest error rates compared to rates obtained from across the other three datasets. In fact, the average error rate for all the classifiers for this kind of dataset is 26.1 per cent. SVM is the only classifier that achieved the highest error rate (33.4 per cent) for this kind of dataset across the other three datasets (with error rates of 26.7 per cent for CM1, 27.1 per cent for JM1, and 30.1 per cent for PC1).

4.2 Experiment II

Experiment II is similar to that described in the previous experimental section. Hence, detailed experimental methods are not included but only a subset of the experiment is given. The main objective of this experiment is to compare the performance of different ensembles for software fault prediction. The results for each classifier are used as a baseline. The correlation maximisation method was used to select the appropriate number of ensemble classifier members, of which three classifiers per ensemble were chosen. For each ensemble, four sampling procedures (bagging, boosting, feature selection, and randomization) were considered. This is the case for each individual dataset.

The ensembles (ES1 to ES10) that consists of three individual classifiers are given as: ES1 (AR, DT, k-NN); ES2 (AR, DT, NBC); ES3 (AR, DT, SVM); ES4 (AR, k-NN, NBC); ES5(AR, k-NN, SVM); ES6(DT, k-NN, NBC); ES7(DT, k-NN, SVM); ES8(DT, NBC, SVM); ES9(k-NN, NBC, SVM); ES10 (k-NN, NBC, SVM).

4.2.1 Results Main Effects

All the main effects were found to be significant at the 5 per cent level of significance (F = 87.3, df = 9 for ensembles methods; F = 17.4, df = 3 for sampling procedures; p < 0.05 for each).

Figure 5 summarises the error rates for 10 classifier ensembles on software fault-proneness prediction. The behaviour of these ensemble methods was explored under varying sampling procedures. The error rates of each ensemble were averaged over the four datasets. From the results it follows that the ensemble of AR, DT, and k-NN achieved the highest accuracy rates (especially when bagging was used as a sampling procedure). The worst performance was by ES9

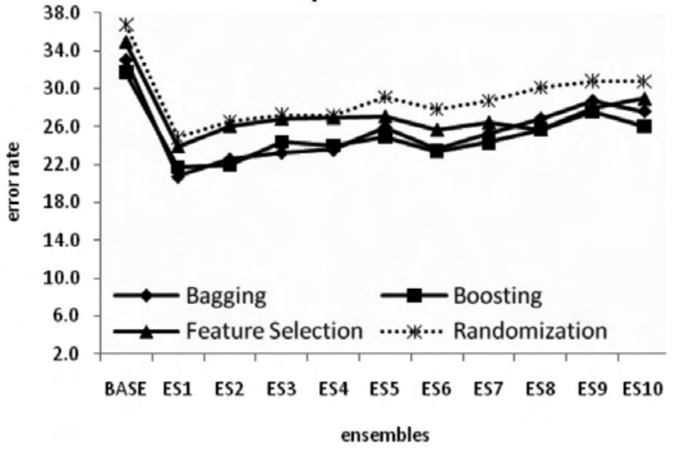

Figure 5. Performance of baseline and ensemble classifiers (all datasets).

(with feature selection) whose components are k-NN, NBC and SVM. All the ensembles performed badly when randomization was used as a sampling procedure.

For the CM1 data problem, Fig. 6 shows that the ensembles have on an average the best accuracy throughout the sampling procedure spectrum compared to individual classifiers. Also, it appears that most of the ensembles, with DT

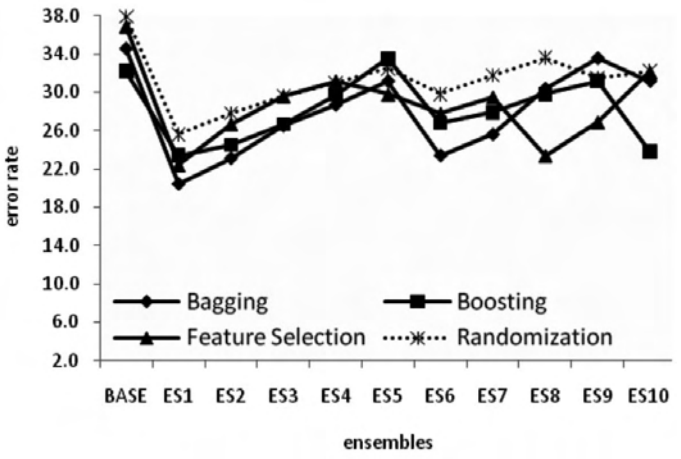

Figure 6. Performance of baseline and ensemble classifiers (CM1).

as one of its components achieve higher accuracy rates. Most of the ensembles achieved higher accuracy when boosting or bagging was used as a sampling procedure.

Figure 7 shows a comparison of results for the JM1 data problem which show the error rates of the 10 ensembles as a function of predictive accuracy. Randomisation systems that combine outputs from models constructed using AR and DT (on the one hand) and either k-NN or SVM or NBC (on

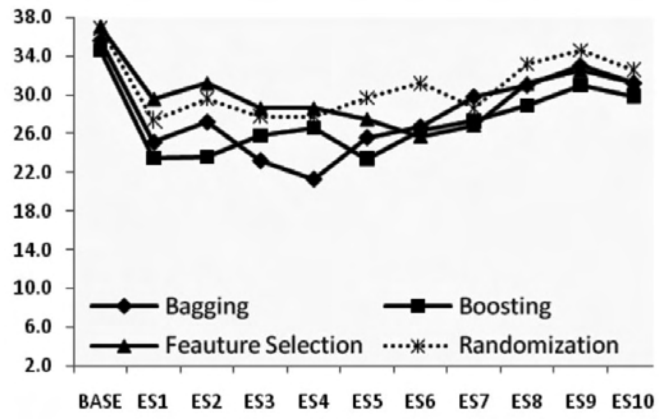

Figure 7. Performance of baseline and ensemble classifiers (JM1).

the other hand) performed well. However, the performance of ensembles using randomisation systems and with SVM and NBC as components (on the one hand) and either NBC or SVM (on the other hand) performed poorly. Once again, both bagging and boosting have come out to be the strongest sampling procedures.

For the PC1 data problem, ES2 has on an average the best accuracy throughout the spectrum of ensembles and this is the case when boosting was used as a sampling procedure (Fig. 8). Once again, the impact of AR and DT as key components of the ensemble was found to be prominent.

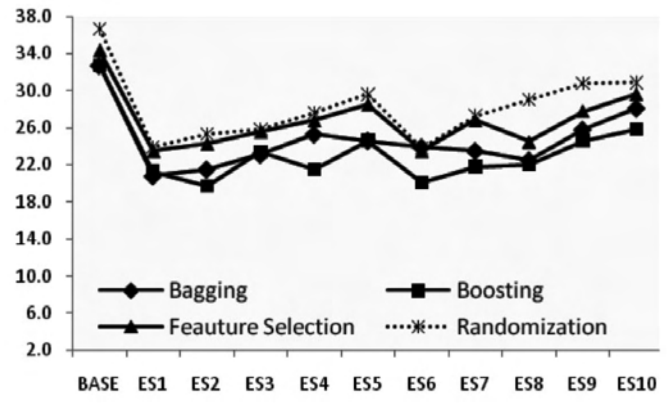

Figure 8. Performance of baseline and ensemble classifiers (PC1).

The results using ensemble methods for the JPL/NASA data problem (Fig. 9) are nearly identical to those observed for PC1 data. However, the performances of all the ensemble methods improve for this kind of dataset with error rates as small as 16.7 per cent (for ES1 when bagging was used) observed. ES10 using randomisation exhibits the worst performance with an error rate of 27.3 per cent.

One surprising result is the poor performance of ES10, especially when boosting was used. This was not the case for the other datasets where boosting always yielded good results.

5. DISCUSSION AND CONCLUSIONS

Accurate prediction of software faults in space systems can be very valuable to engineers, especially those dealing with software development processes. This is important for minimising cost and improving effectiveness of the software testing process. The major contributions have been the application of one of the top five machine learning algorithms to predict software faults in space systems and further use multiple classifier learning to improve software faults predictive accuracy. Four NASA public datasets were utilised for this task. The results suggest that the ML algorithms can be successfully applied in software faults prediction with multiple classifier learning providing overall significant increases in classification performance.

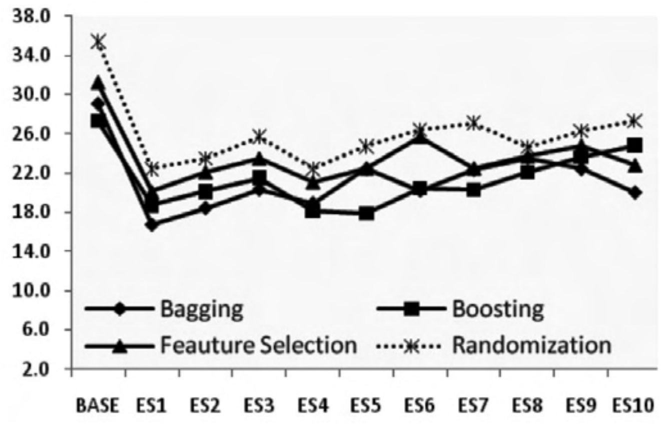

Figure 9. Performance of baseline and ensemble classifiers (JPL/NASA).

Based on evidence, it has been found that most of the ensembles improve the prediction accuracy of the baseline classifiers (AR, DT, *k*-NN, NBC and SVM) with the ensembles that have AR and DT as their components performing well. This improvement is achieved mainly in all the datasets for both bagging and boosting. Surprisingly, most of the ensembles with NBC as one of its components did not perform as good as when NBC was just a single classifier. Also, the overall performance of feature selection for all the ensembles was very poor. This was the case for all the datasets. Individually, NBC is the most effective classifier for all the datasets. The performance of DT is equally good, especially for the bigger datasets. SVM is more effective for small datasets while AR is a poor performer for any type of dataset. An important question is why does NBC outperforms the other classifiers by such significant margins? One reason could be the level of inertia displayed by each classifier.

From both experiments, there exists threats to the validity of the results. All the four datasets were obtained from NASA, hence, their conclusions could be biased. One such threat was duplicates in some of the instances. These were deleted from the analysis. The second threat was the amount of missing values in the dataset, of which multiple imputation[3] was used to deal with them. This was the case for all the four datasets, hence, in part, a time-consuming exercise. Help was also sought from the domain experts to give reaons why some attributes values could be missing? In addition, clarification on unclear descriptions on some of the software modules was also sought from the project personnel.

The performances of ensemble imputation methods in terms of smoothed misclassification error rate were observed. A natural extension would be to consider the impact of such ensemble methods on other measures of performance, and in particular, measures of group separation such as GINI or the magnitude of relative error that is also commonly used to assess classifier performances in the SE industry.

The ensembles require further investigation on a number of fronts, for example, in terms of training parameters and the combination rules that can be employed. Also, empirical studies of the application of the ensembles to datasets from other areas of data mining should be undertaken to assess their performance across a more general field.

ACKNOWLEDGEMENTS

This work was funded by Department of Electrical Engineering and Electronic Engineering Science at the University of Johannesburg, South Africa. The comments and suggestions from my colleagues and the anonymous reviewers greatly improved this paper. The author would also like to thank NASA for making the datasets available.

REFERENCES

1. Briand, L.; Basili, V. & Thomas, W. A pattern recognition approach to software engineering data analysis. *IEEE Trans. Soft. Engg.*, 1992, **18** (11), 931-42.
2. Shepperd, M.J. & Schofield, C. Estimating software project using analogies. *IEEE Trans. Soft. Engg.,* 1997, **23** (12), 736-43.
3. Twala, B., & Cartwright, M. Ensemble imputation methods for software effort prediction. *Intelli. Data Anal.*, 2010, **14** (1), 299-31.
4. Evett, M.; Khoshgoftaar, T.; Chien, P. & Allen, E. GP-based software quality prediction. *In* Proceedings of the 3rd Annual Genetic Programming Conference, 1998. pp. 60-65.
5. Fenton, N. & Neil, M. A critique of software defect prediction models. *IEEE Trans. Soft. Engg.,*1999, **25** (5), 675-89.
6. Neumann, D.E., An enhanced neural network technique for software risk analysis. *IEEE Trans. Soft. Engg.,* 2002, 904-12.
7. Mendonca, M. & Sunderhaft, N.L. Mining software engineering data: A survey. DACS-SOAR-99-3. A DACS state-of-the-art Report. DoD Data and Analysis Center for Software. 1999.
8. Menzies, T. Practical machine learning for software engineering and knowledge engineering. *In* Handbook of software engineering and knowledge engineering. 2001. http://tim.menzies.com/pdf/00ml.pdf (Accessed on 11 November 2010).
9. Breiman, L. Bagging predictors. *Machine Learning*, 1996a, **26**(2), 123-40.
10. Freund, Y. & Schapire, R. Experiments with a new boosting algorithm. *In* Machine Learning: Proceedings of the 13th International Conference, 1996. pp. 148-56.
11. Ho, T.K. Random decision forests. *In* Proceedings of the 3rd International Conference on Document Analysis and Recognition, 1995. pp. 278-82.
12. Dietterich, T. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 2000, **40**(2), 139-58.
13. Wu, X.; Kumar,V; Quinlan, J.R.; Ghosh, J.; Yang, Q.; Motoda, H.; Mclachlan, G.J.; Ng, A.; Liu, B.; Yu, P.S.; Zhou, Z-H.; Steinbach, M.; Hand, D.J. & Steinberg, D. Top 10 algorithms in data mining. *Knowl. Infor. Syst.*, 2008, **14**(1), 1-37.
14. Agrawal, R. & Srikant, R. Fast algorithms for mining association rules. *In* Proceedings of 20th International Conference on Very Large Databases (VLDB 1994, Santiago de Chile), 1994. pp. 487-99.
15. Breiman, L.; Friedman, J, Olshen, R., & Stone, C. Classification and regression trees, Wadsworth, 1984.
16. Lutz, R.R. & Mikulski, I.C. Empirical analysis of safety critical anomalies during operation. *IEEE Trans. Soft. Engg.,* 2004, **30**(3), 172-80.
17. Jolliffe, I. Principal component analysis. Springer Verlag, 1986.
18. Lai, K.K.; Yu, L.; Wang, S.Y. & Zhou, Lg.G Credit risk analysis using a reliability-based neural network ensemble model. *In* Lecture Notes in Computer Science, 2006, **4132**, 682–90.
19. Quinlan, JR. C.4.5: Programs for machine learning. Los Altos, California, Morgan Kauffman Publishers, Inc., 1993.
20. Breiman, L. Bias, variance, and arcing classifiers. Statistics Department, University of California at Berkeley, Technical Report No. 460, 1996b.
21. Marwala, T. & Hunt, H.E.M. Fault identification using finite element models and neural networks. *Mech. Syst. Signal Process.*, 1999, **13**(3), 475-90.
22. Marwala, T. Probabilistic fault identification using vibration data and neural networks. *Mech. Syst. Signal Process.,* 2001, **15**(4), 1109-128.
23. Yairi, T.; Ishihama, N.; Kato, Y.; Hori, K & Nakasuka, S. Anomaly detection method for spacecrafts based on association rule mining. *J. Space Technol. Sci.*, 2001, **17**(1), 1-10.
24. Koru, A.G. & Tian, J. An empirical comparison and characterization of high defect and high complexity modules. *J. Syst. Software*, 2003, **67**(3), 153-63.
25. Guo, L.; Cukic, B. & Singh, H. Predicting fault prone modules by Dempster-Shafer belief networks. *In* 18th IEEE International Conference on Automated Software Engineering Montreal, Canada, IEEE Computer Society, 2003. pp. 249-52.
26. Menzies, T.; Distefano, J.S.; Orrego, A. & Chapman, R. Assessing predictors of software defects. *In* Predictive Software Model Workshop, 2004.
27. Menzies, T. & Distefano, J.S. How good is your blind spot sampling policy? *In* 8th IEEE International Symposium on High Assurance Systems Engineering, Tampa, FL, USA. IEEE Computer Society, 2004. pp. 129-38.

28. Fujimaki, R.; Yairi, T. & Machida, K. An approach to spacecraft anomaly detection problem using kernel feature space. *In* Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. 2005. pp. 401-10.
29. Koru, G. & Liu, H. An investigation of the effect of module size on defect prediction using static measures. *In* Workshop on Predictor Models in Software Engineering, St Louis Missouri, 2005. pp. 1-5.
30. Gondra, I. Applying machine learning to software fault-proneness prediction. *J. Syst. Soft.*, 2005, **81**, 86-195.
31. Ma, Y.; Guo, L. & Cukic, B. A statistical framework for the prediction of fault-proneness. *In* Advances in machine learning application in software engineering. 2006. pp. 237-65.
32. Challagulla, V.U.; Bastani, F.B.; Yen, I. & Paul, R.A. Empirical assessment of machine learning-based software defect prediction techniques. *In* 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, Sedona, Arizona, IEEE Computer Society, 2005. pp. 263-70.
33. Zhou, Y. & Leaung, H. Empirical analysis of object-oriented design metrics for predicting high and low severity faults. *IEEE Trans. Soft. Engg.*, 2006, **32** (10), 771-89.
34. Menzies, T.; Dekhtyar, A.; Distefano, J.S. & Greenwald, J. Problems and precision: A response to comments on data mining static code attributes to learn defect predictors. *IEEE Trans. Soft. Engg.*, 2007, **33** (0), 637-40.
35. Mende, T. & Koschke, R. Revisiting the evaluation of defect prediction models. *In* Proceedings of the 5th International Conference on Predictor Models in Software Engineering, PROMISE '09, Vancouver, British Columbia, Canada, 18-19 May 2009. pp. 1-10.
36. Witten, I.H. & Frank, E. Data mining: Practical machine learning tools and techniques with java implementations. Morgan Kauffman. 1999.
37. MINITAB. MINITAB statistical software for Windows 9.0. MINITAB, Inc., PA, USA. 2002.
38. Kirk, R.E. Experimental design. Ed. 2. Brooks, Cole Publishing Co. Monterey, CA. 1982.
39. Vapnik, V. The nature of statistical learning theory, Springer, New York. 1995.

Contributor

Prof (Dr) Bhekisipho Twala received his BA (Economics and Statistics) from University of Swaziland, in 1993; MSc(Computational Statistics) from Southampton University, in 1995, and PhD (Machine Learning and Statistics) from the Open University, in 2005. Currently working as a Professor for Artificial Intelligence and Statistical Science in the Department of Electrical and Electronic Engineering Science, University of Johannesbug, South Africa. Currently, he is involved in developing novel and innovative solutions (using AI technologies) to key research problems in the field of electrical and electronic engineering science. His broad research interests include multivariate statistics, classification methods, knowledge discovery and reasoning with uncertainty, sensor data fusion and inference, and the interface between statistics and computing.