International Journal of Management, IT & Engineering
Vol. 8 Issue 5, May 2018,
ISSN: 2249-0558
Impact Factor: 7.119Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gate as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Importance of Algorithm Analysis to Assess the Performance: With Special Reference to Empirical Metrics

Ekansh Upadhyay

National Institute of Technology, Kurukshetra

Abstract –The research paper was elaborated the importance of algorithm analysis to assess the performance especially considering empirical metrics their failure, growth, run-time etc. In this article we propose some modern prognostic-modified assessment metrics and explain that, relative to other standard metrics, they can accurately test different algorithms. Relevance Vector Machine, Gaussian Process Regression, Artificial Neural Network, and Polynomial Regression are the four prognostic algorithms being contrasted. The sophistication of these algorithms and their capacity to handle ambiguity about expected estimates differ. The findings demonstrate that these algorithms are ranked in a particular way by the current metrics; acceptable metrics may be selected based on the criteria and constraints. In addition to these observations, this paper provides insights about how metrics that are relevant for prognostics can be structured to standardize the assessment process. Overall, the paper is intended to strengthen the general perception of these measures so that they can be more optimized and adopted as normative metrics for the output evaluation of prognostic algorithms by the scientific community. During the multiple stages of the software creation life cycle, the importance and potential purpose of this analysis involved successful control of interacting attributes. In addition, the human component of software creation, quantified by individual indicators, which play a role in evaluating other software measures that have yet to be extensively analysed.

KEYWORD: Algorithm Analysis, Performance, Empirical Metrics

I. INTRODUCTION

Software production is a dynamic mechanism requiring the smooth and frictionless collaboration of different development departments, collaborating for product delivery separately and in tandem. Complexity at the process level exists due to information and connectivity differences between developers, while at the software level they occur separately interconnected due to the interaction of code fragments, resulting in low end product quality [1].

A significant number of quality models have been proposed over the last three decades, representing the understanding of 'quality' and 'model' in the software engineering sector. The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) have established the ISO/IEC 25010 software quality specifications in order to include a detailed design and validation framework[2] for the standardization of these quality models. A generic quality model is specified by ISO/IEC 25010, which can be adapted to the particular intent of any software product. This consistency model was mostly broken down into eight quality features (its predecessor ISO/IEC 9126-1 standard consisted of six features) and 40 software product quality sub-features (Fig. 1).

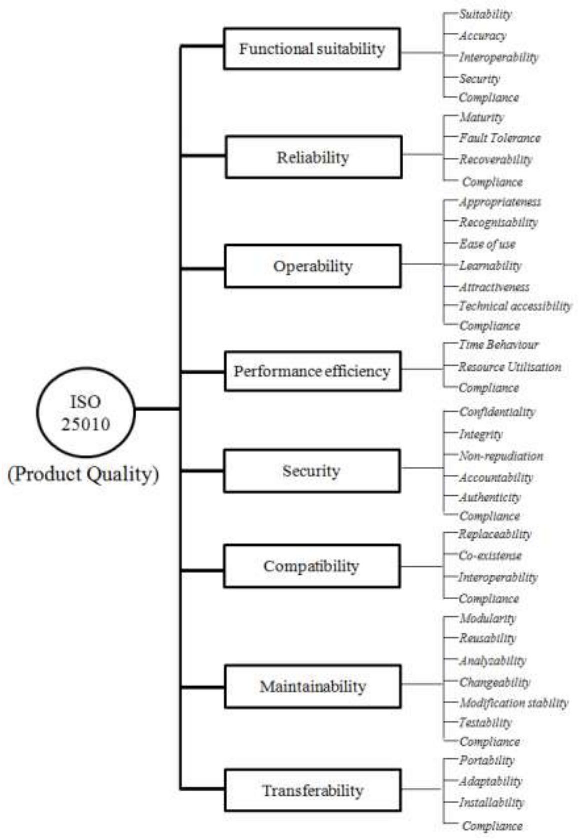
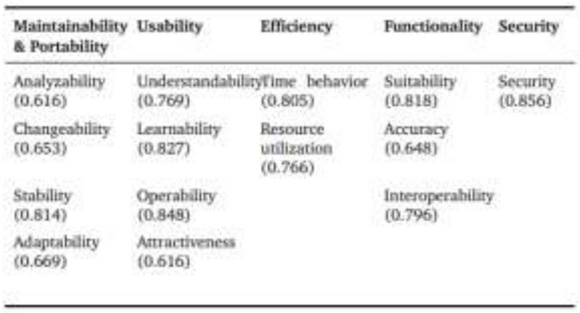


Figure 1: Value Norm for ISO/IEC 25010 Applications (Adapted from Wagner et al.[3])

Table 1: ISO 9126 Consistency Level Association between Characteristics and Sub-characteristics (Adapted from Jung et al. [4])



Maintainability & Portability	Usability	Efficiency	Functionality	Security
Analyzability (0.616)	Understandability (0.769)	Time behavior (0.805)	Suitability (0.818)	Security (0.856)
Changeability (0.653)	Learnability (0.827)	Resource utilization (0.766)	Accuracy (0.648)	
Stability (0.814)	Operability (0.848)		Interoperability (0.796)	
Adaptability (0.669)	Attractiveness (0.616)			

Jung et al.[4] performed an analytical analysis to examine the connection between the ISO/IEC 9126 standard (predecessor to ISO/IEC 25010) characteristics and sub-characteristics. The key factor review (PCA) on the data obtained for five features and 14 sub- features was performed in this report. The values of sub-characteristic correlation coefficients against a characteristic are reproduced in Table 1. The corresponding significance of a sub-characteristic is shown in brackets in Table 1. The results indicate the absence in the consistency model of a rigid grouping of sub-characteristics into a single function. The foregoing quality norm offers instructions on the calculation of the indicated quality factors (characteristics), but there is still no concrete formulation of the measurement [3].

A collection of metrics to test crucial elements of RUL forecasts was proposed by the previous attempt. This paper would explain how the efficiency of prognostic algorithms can be measured using certain parameters. In addition, it would determine if these measures capture the success parameters they were developed for. The paper would concentrate on measures expressly developed for prognostics that are already being utilised for diagnostics and other forecasting uses outside standard metrics. In general, these measures answer the dilemma of how much the forecasts of the RUL outlook change over time as more calculation evidence becomes usable. In addition to enhancing RUL calculation, a successful prognostic algorithm should also maintain a fair forecast horizon and levels of faith in the predictions.

II. OBJECTIVES

1. To survey algorithm analysis to assess the performance with special reference to various empirical metrics and its major applications of prediction models
2. To examine the importance of prognostic algorithm performance analysis to predict remaining useful life (RUL) and its need

III. MOTIVATION

It is a challenging challenge to define the right collection of metrics, as no particular set of metrics applies uniformly to all ventures. This is apparent from the overlap of numerous sub-characteristics with the consistency model characteristics outlined in Fig 1. However the choosing of suitable metrics becomes simpler when integrating multiple metrics and outcomes in a predictive model with enhanced capabilities. The advantages and comparative benefits of utilizing a mix of source code indicators to predict bugs have been explored in the new literature[5, 6]. In designing an effective statistical model, Wahyudin et al.[7] explored the cumulative influence of commodity and project metrics. Lee et al.[8] suggested 56 engagement measures capturing the behavioural pattern of the creator and evaluated their cumulative impact on product efficiency empirically. When CK and other Object-oriented (OO) metrics were used together, D'Ambros et al.[9] registered an improved model output, which was endorsed by research made with five separate software modules. The influence of one metric over another was not included in this method of merging metrics, however. The Depth of Inheritance Tree (DIT) and Number of Children (NOC) are interrelated in the CK metrics suite, for example, and their relationship impact must be taken into account when designing regression-based models.

This influence of the mixture[7-9] may be obtained and confirmed through a small number of experiments. This provided us an opportunity to explore the influence of the mixture of metrics on the creation of models. In addition, the studies described above suggested that the mixture of metrics was significant without explicitly delving into the relationships between metrics.

In certain instances, comparison databases or models are used on a standard basis to test certain techniques so that they can be evaluated equally. Prognostic systems have none of these solutions, in most situations. Different researchers have used multiple criteria to test their algorithms, rendering it very challenging to equate different algorithms even though, based on their respective tests, they have been considered efficient. It is known that prognostic approaches need to be adapted to particular applications, rendering it impossible to create a useful generalized algorithm for any scenario. Customized measures can be used in such situations, but there are features of prognostic applications that stay constant and a framework for comparisons may be defined for subsequent performance measurement. When it comes to evaluating and contrasting multiple algorithms, relatively little has been achieved so far to find shared ground. It can be shown that there is a shortage of systematic methodology for performance assessment in two studies of prognostic approaches (one of data-driven methods and one of artificial intelligence-based methods), and in certain situations, performance evaluation is not even formally discussed. A firm description of certain metrics is also absent in the ISO norm for prognostics in condition control and computer diagnostics. Therefore we are introducing some new metrics in this paper and demonstrating how they can be used efficiently to evaluate various algorithms. We aim to have some starting points for potential conversations with these reflections on importance of algorithm analysis to assess the performance with special reference to empirical metrics.

IV. RESEARCH METHOD

We are selected four data-driven algorithms in this attempt to illustrate the usefulness of different measures in assessing their efficiency. These algorithms vary from basic regression of polynomials to complex methods of Bayesian learning. The methods used here have been defined in [10, 11] previously, but for the sake of completeness and readability, they are replicated here. The method for how each of these algorithms is applied to the battery health management dataset is also briefly described.

Polynomial Regression (PR): Approach To create a benchmark for the success of battery health prediction and uncertainty evaluation; we used a basic data-driven procedure. The equivalent harm threshold in the RE+RCT (dth=0.033) is gleaned from the relationship between RE+RCT and capability C at baseline temperature (25oC) for this data-driven method as the first step. Next, RE+RCT was monitored at elevated temperatures (here 45^{o}C) through extracted features from the EIS measurements. A second degree polynomial was used at the estimation points to extrapolate out to the harm threshold, missing the first two data points (which act close to what is called 'wear-in' trend in other domains). In order to calculate the predicted RUL values, this linear extrapolation is then used.

*Relevance Vector Machines (RVM):*The Relevance Vector Machine (RVM) [12] is a Bayesian method describing the Support Vector Machine (SVM) generalised linear model of the same functional form [13]. Although SVM is a state-of-the-art classification and regression process, it suffers from a variety of drawbacks, one of which is the absence of more practical probabilistic outputs in health monitoring applications. In a Bayesian setting, the RVM tries to solve these very problems. In addition to the probabilistic analysis of its output, for comparable generalization efficiency, it usually uses far less kernel functions.

This form of supervised machine learning begins with the input vector set {xn}, n = 1,..., N and the corresponding {tn} goals. In order to allow correct estimates of t for unseen values of x, the goal is to develop a model of the reliance of objectives on the inputs. The predictions are usually focused on some F(x) function specified over the input space, and the process of inferring the parameters of this function is learning. Samples from the model with additive noise are assumed to be the targets:

$$t_n = F(x_n; w) + \varepsilon_n$$

where, ε_n are independent samples from some noise process (Gaussian with mean 0 and variance σ^2). Assuming the independence of t_n, the likelihood of the complete data set can be written as:

$$p(t|w,\sigma^2) = (2\pi\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}\|t-\Phi w\|^2\right\}$$

where, $w=(w_1, w_2, ..., w_M)^T$ is a weight vector and is the N x (N+1) design matrix $\Phi=[\varphi(t_1), \varphi(t_2), ..., \varphi(t_N),]^T$; in which $\varphi(t_N)=[1, K(X_n, X_1), K(X_n, X_2), ..., K(X_n, X_N)]^T$ $K(X, X_i)$ being a kernel function

A preference for simpler functions is encoded by the choice of a zero-mean Gaussian prior distribution over w parameterized by the hyperparameter variable to avoid over-fitting. η. To complete the specification of this hierarchical prior, the hyperpriors over η and the noise variance σ^2 are approximated as delta functions at their most probable values η_{MP} and $\sigma^2{}_{MP}$. Predictions for new data are then made according to:

$$p(t^*|t) = \int p(p(t^*|w, \sigma^2{}_{MP})p(w|t, \eta_{MP}, \sigma^2{}_{MP})dw,$$

Gaussian Process Regression (GPR): Gaussian Process Regression (GPR) is a nonlinear regression probabilistic technique that calculates estimates of posterior deterioration by restricting the prior distribution to match the training data available [14]. A Gaussian Method (GP) is a set of random variables with a mutual Gaussian distribution of some finite number. A real GP f(x) is fully defined by its mean function m(x) and covariance function k(x,x') defined as m(x) and covariance function k(x,x'):

$$m(x) = E[f(x)]$$
$$k(x,x') = E\{(f(x)-m(x)(f(x')-m(x'))]$$
$$f(x) \sim GP(m(x), k(x,x'))$$

The X ∈ R index collection is the set of potential inputs that do not actually need to be a time vector. In view of the prior GP knowledge and the set of training points {(xi,fi)|i = 1,...,n}, the corresponding distribution of functions is obtained by placing a constraint on the previous joint distribution to involve only those functions which are in harmony with the data points observed. These functions should be considered to be chaotic since we only have recourse to noisy observations in real-world conditions rather than precise function values.i.e. yi = f(x) + ε where is additive IID N(0, σ_n^2). It can be used to determine statistical values for the test data points until we have a posterior distribution. The predictive distribution for GPRR is described by the following equations. [15].

Prior

$$\begin{bmatrix} y \\ f_{test} \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(X,X)+\sigma_n^2 & K(X,X_{test}) \\ K(X_{test},X) & K(X_{test},X_{test}) \end{bmatrix}\right)$$

Posterior

$$f_{test} \mid X, y, X_{test} \sim N(\overline{f}_{test} \operatorname{cov}(f_{test})), where$$

$$\overline{f}_{test} \equiv E[f_{test} \mid X, y, X_{test}] = K(X, X_{test})[K(X,X)+\sigma_n^2 I]^{-1} y,$$

$$\operatorname{cov}(f_{test}) = K(X_{test}, X_{test}) - [K(X, X_{test}) + \sigma_n^2 I]^{-1} K(X, X_{test})$$

The covariance function (*K(X, X* ')) that encodes the conclusions about the functions to be learned by specifying the relationship between data points is a key component in a Gaussian process predictor. GPR needs prior awareness of the covariance function structure, which if necessary, must be obtained from the context. In comparison, the functions of covariance consist of separate hyper-parameters describing their properties. In studying the desired functions, setting the proper values of such hyper-parameters is yet another difficulty. Although the consumer must determine the option of the covariance function, the corresponding hyper-parameters may be learned from the training data using a gradient-based optimizer, such as maximising the marginal probability of hyper-parameter data observed[16].

To minimize the evolution of internal parameters (RE+RCT) of the battery over time, we used GPR. From experimental data[11], the association between these parameters and battery power has been learned. The internal parameters for the data collected were therefore regressed and the resulting calculations were converted using the relationship acquired at 25oC to an approximate battery ability of 25^{o}C.

Neural Network (NN) Approach: An alternate data-driven strategy for prognostics was known as a neural network dependent approach. A simple feed forward neural network with back propagation training was used; information can be found in[17, 18] regarding this algorithm. Data at 25^{o}C was used as stated earlier with the other methods, to learn the relationship between the internal parameter RE+RCT and capability C using the NN1 neural network. Furthermore the data from 45 oC was used as a test case. Here the internal parameter RE+RCT measurements are only usable up to tPP time (time at which RUL prediction is made). In order to forecast potential values, the usable RE+RCT measurements are extrapolated to period tP. This extrapolation is carried out using the NN2 neural network that learns the connection between time and RE+RCT. Once potential values are determined using NN2 for RE+RCT, these RE+RCT values are used as an input to NN1 to obtain C. The NN1 layout comprises of two opaque layers of one and three nodes. Logsigmoid transfer functions were selected for the secret layers of tansigmoid transfer functions and for the output layers. As an output metric, preparation considers arbitrary initial weights, a reduced memory LevenbergMarquardt algorithm, 200 training epochs, and meansquared error.

V. LITERATURE REVIEW

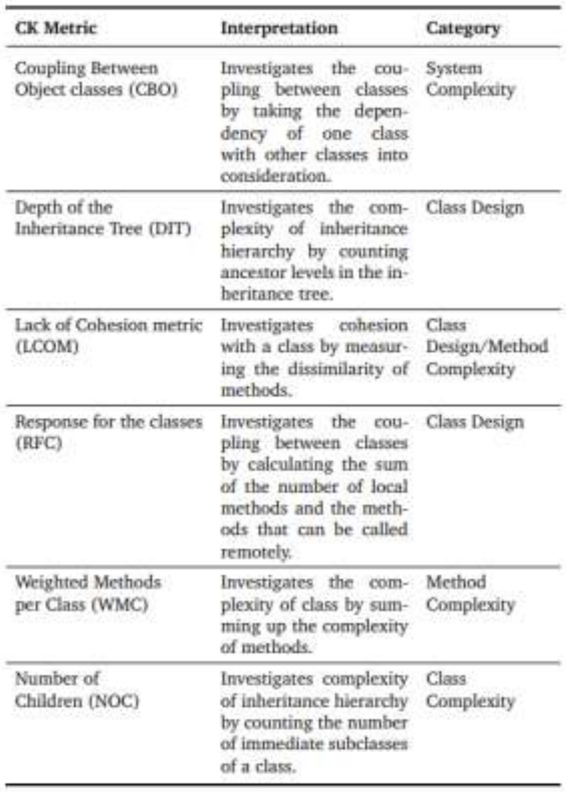
This section surveys different software fault predictions progressed through different data analytic techniques reported in the software engineering literature. Although developments of models make use of different kinds of metrics, the review of literature presented here essentially focuses on the Object oriented metrics. In comparison to, procedural language based system; Object Oriented (OO) technology based systems introduce new abstractions and building blocks. Therefore, development of the new set of metrics and fault prediction models will foster quality in the developed software.

Major Software Metrics and its Applications

Software metrics can be categorized as product metrics, process metrics or resource metrics. Product metrics measure different features of developed programs like Methods and Class level metrics in an object oriented systems. Process metrics are related to the measurement and quantification of activities like design, implementation, testing, and maintenance. Resource metrics focus on all other resources involved in development such as programmers, cost of the product and processes, etc. [19–20]. These metrics have shown a corresponding relationship with a variety of external quality characteristics of software, such as reliability, testability and maintainability.

Chidamber. A generally applied and validated metrics package, usually known as the Chidamber&Kemerer (CK) metrics suite of six metrics, was introduced by Kemerer and Kemerer[21] (Table 2)

Table 2: Metrics proposed by Chidamber&Kemerer [21]



CK Metric	Interpretation	Category
Coupling Between Object classes (CBO)	Investigates the coupling between classes by taking the dependency of one class with other classes into consideration.	System Complexity
Depth of the Inheritance Tree (DIT)	Investigates the complexity of inheritance hierarchy by counting ancestor levels in the inheritance tree.	Class Design
Lack of Cohesion metric (LCOM)	Investigates cohesion with a class by measuring the dissimilarity of methods.	Class Design/Method Complexity
Response for the classes (RFC)	Investigates the coupling between classes by calculating the sum of the number of local methods and the methods that can be called remotely.	Class Design
Weighted Methods per Class (WMC)	Investigates the complexity of class by summing up the complexity of methods.	Method Complexity
Number of Children (NOC)	Investigates complexity of inheritance hierarchy by counting the number of immediate subclasses of a class.	Class Complexity

Importance of Algorithm in Modelling Techniques

Empirical information engineering has seen an increased use of diverse data analytical methods in recent years, stemming from the public accessibility of a multitude of software repositories and progressive analysis demonstrated by the community in machine learning and data mining. There have been detailed studies of nonparametric approaches such as Regression Tree, Random Forest, Support Vector Machine, Neural Network, etc.. The following is a study of the development of the fault prediction paradigm focused on the basis of applied data analysis routines. Fig. 2 describes the categorization and role of the empirical approaches studied in this segment in the hierarchy of the wide continuum of data science.

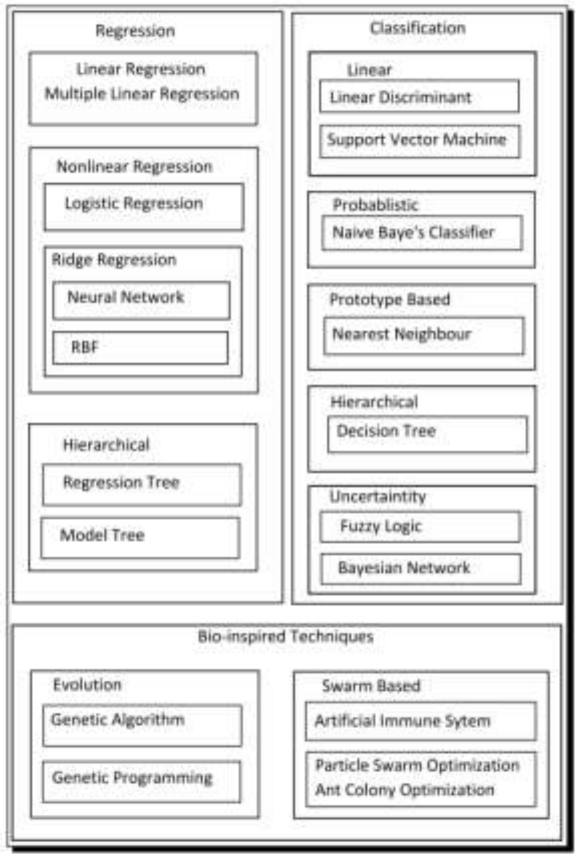


Figure 2: Taxonomy of Data Analysis Techniques

- Linear and Logistic Regression

In component collection, logistic regression was used and the combination of parameters was consequently statistically diagnosed. A measured model prepared with a J coefficient[22] as a statistical measure was responsible for the effect of this metric mix. Marcus et al.[23] suggested a new class cohesion formula called the Conceptual Cohesion of Classes (C3) and further argued that the C3 metric was superior to other linear combinations resulting from empirically defined results by means of PCA (Principal Component Analysis).

- Learning Dependent on SVM and Instance

Using a kernel function, a Support Vector Machine (SVM) optimally divides data points into two groups. The model thus generated using a suitable kernel function is strongly similar to the neural network which while beginning with a limited training sample, generalises well. This provides an appropriate strategy for the creation of fault prediction models for SVM, where knowledge on difficulty metrics is very minimal in the early phase of SDLC.Elish et al.[24] assessed Support vector machine (SVM) output in the classification of fault-prone programme modules utilising four publicly accessible NASA data sets. These data sets were extracted from software projects produced in the various languages of computing (C, C++ and Java). Compared to eight other statistical and machine learning techniques (LR, KNN, RBF, MLP, NB, BBN, RF, and DT), the predictive accuracy of the models built by SVM with 21 static module level metrics with 10 fold cross validation 1 SVM showed superior recall measurement efficiency while retaining substantial high F-measurement values.

- Additive Models and Trees

Gokhale and Lyu [25] were the first to report the regression tree model for fault prediction. Since then, these trees-based regression approaches have been used in a significant range of experiments, including the following: Khoshgoftaar et al.[26] demonstrated the basic efficacy of a regression tree algorithm by classifying fault-prone modules for a very large telecommunication device. Via classification, Bibiet al.27] conducted regression by discretizing target variables for classification model preparation, and then reversed the method to turn the produced output back into a numerical prediction. The enhanced performance of random forests over logistic regression and discriminant analysis was concluded by Guo et al.[28] utilising five NASA data set case studies.

- Models Focused on Perceptron

Neural networks are a universal nonlinear regression system approximation class based on the behaviour of biological neurons. The concepts' Neural Network '(NN) and' Artificial Neural Network '(ANN) are usually part of a Multilayer Perceptron Network. Probabilistic Neural Networks (PNN), General Regression Neural Networks (GRNN), Ward neural network (WNN), Radial Base Feature (RBF), Recurrent Networks and Hybrid Networks are additional prototypes of the neural network[29]. In order to build neural network dependent predictive models, Zheng, et al.[30] took the magnitude of type II error into account. The Type II error deals with the misclassification of defect-prone modules, while the misclassification of non-defect-prone modules is related to the Type I error. The cost-sensitive Adaboost (boosting technique) neural network demonstrated a reduced number of such type II failures. The usage of the neural network was first demonstrated by Khoshgoftaar et al.[31] for EMERALD (Enhanced calculation for early risk detection of latent defects), a collaborative project between Nortel and Bell Canada to increase software reliability. Their findings showed that according to discriminant studies, neural handles Type II classification error effectively.

- Approaches founded upon Fuzzy Logic

Verma et al.[32] suggested a fuzzy logic-based development effort appraisal system and documented improved success on both artificial and live project results. Their findings statistically evaluate the usefulness of the method relying on fuzzy reasoning to handle the input data inaccuracy. Aljahdali et al.[33] published promising findings in modelling cumulative faults in software modules by utilising fuzzy nonlinear regression.

- Techniques of Bio-inspired

Bio-inspired meta-heuristic methods are evolutionary strategies and exhibit typical characteristics [34]. The application of these strategies starts with the quest space being built up by a community of candidate solutions. Based on the issue formulation, a selection algorithm selects better alternatives by a derived fitness criterion. By replication and recombination, new solutions evolve.

Azar et al. [35] improved current models of software quality estimation utilising the methodology of ant colony optimization (ACO). ACO was optimised for model development using previously built predictive models to use popular domain and context-specific data. This makes statistical models constructed with one dataset for new knowledge to be inferred. Compared to C4.5 and random guessing strategies, the outcome of this study ended with better ACO results. The effect of genetic programming (GP) in the creation of decision trees was explored by Khoshgoftaar et al.[36] to address the issue of software consistency classification while also growing the expense of misclassification and tree scale. In this analysis, two initial releases of broad windows-based embedded systems containing more than 27 million lines of data set produced by codes were used. The findings concluded that GP-based simulation of decision tree accounts for greater versatility in constructing optimal models of classification.

It has been suggested that suitable collections of these parameters or their derivatives must be selected to better measure the efficiency of the algorithms based on the end user requirements.

VI. PERFORMANCEANALYSIS OF EMPIRICAL MATRICS

Nine separate performance indicators are defined in this portion. Four of them are the community's most often employed measures, i.e. precision, accuracy, Mean Squared Error (MSE), and Mean Absolute Percentage Error (MAPE). These measures have been used to show how valuable these metrics are however differing facets of prognostic predictions cannot catch time. In addition, five additional metrics that provide certain fascinating characteristics have been added. These metrics were first briefly specified and then analyzed on the basis of the battery health management results as stated in the following section.

Average Bias (Accuracy): One of the conventional indicators that has been seen as a test of precision in several respects is average bias. After prediction begins for the *i*th UUT, it combines the errors in predictions produced at all subsequent periods. To define general prejudices, this metric can be generalised to average biases for all UUTs.

$$B_i = \frac{1}{l}\sum_{i=1}^{l}\Delta^l(i)$$

*Sample Standard Deviation (Precision):*The dispersion/spread of the error with respect to the sample mean of the error is determined by sample standard deviation. This metric is limited to the expectation of an error being transmitted normally. Therefore it is proposed that a visual analysis of the error plots should be carried out in order to evaluate the distribution characteristics prior to interpreting this metric.

$$S = \sqrt{\frac{\sum_{i=1}^{l}(\Delta(i) - m)^2}{l - 1}}$$

*Mean Squared Error (MSE):*The basic average bias metric suffers from the assumption that negative and positive errors are cancelled and the metric does not represent high variance. For both forecasts, thus, MSE averages the squared forecast error and encapsulates both accuracy and precision. Root Mean Squared Error is a derivative of MSE and is sometimes used (RMSE).

$$MSE = \frac{1}{l}\sum_{i=1}^{l}\Delta(i)^2$$

*Mean Absolute Percentage Error (MAPE):*It is necessary to distinguish between errors observed well away from the EOL and those observed near to the EOL in prediction applications. As EOL approaches, fewer mistakes are preferable. Therefore in the different projections, MAPE weighs errors with RULs and combines total percentage errors. Median may be used to compute Median absolute percentage error (MdAPE) in a similar manner instead of the mean.

$$MAPE = \frac{1}{l}\sum_{i=1}^{l}\left|\frac{100\Delta(i)}{r_*(i)}\right|$$

It must be remembered that in situations where only a distribution of RUL predictions is accessible as the performance of the algorithm or several predictions are available from many UUTs to calculate the statistics, the above metrics may be used more accurately. While in these situations, these metrics can express useful details; these metrics are not meant for systems where RULs are regularly modified as more information is available. It is desirable to provide indicators that can characterize a prognostic algorithm's success improvement when time approaches close to the end of existence. In this article, one such application is explored in which algorithms monitor the health of the battery and explain how newer indicators can encapsulate knowledge that is useful for the efficient application of prognostics. Next, we explore modern prognostic-tailored metrics and explain how they are more insightful than those commonly used.

*Prognostic Horizon (PH):*Prediction Horizon has been accessible in the literature for quite some time, but there is no systematic description. The definition implies that more time is required for the predicted horizon to function on the basis of a projection that has some legitimacy. We described Prognostic Horizon as the gap between the current time index I and EOP, given the forecast follows the desired criteria, using data accumulated up to the time index i. This definition must be defined in terms of the permissible binding error (□) about the actual EOL. This metric means that the forecast predictions are around the real EOL within defined limits and thus the forecasts can be deemed trustworthy. During the validation process, PHs are supposed to be calculated offline for an algorithm-application pair and then these numbers are used as instructions when the algorithm is applied in the test application where exact EOL is not specified in advance. An algorithm with a longer predicted horizon will be favoured when evaluating algorithms.

$$H = EOP - i$$
$$i = \min\{j \mid (j \in l) \wedge (r_*(1-\alpha) \le r'(j) \le r_*(1+\alpha)\}$$

For e.g., when a given algorithm begins forecasting estimates that are within 5 percent of the actual EOL, a PH with error bound of = 5 percent defines. Other parameters can be used as needed to derive PH.

□ □ □ *Accuracy*

International Journal of Management, IT & Engineering
Vol. 8 Issue 5, May 2018,
ISSN: 2249-0558
Impact Factor: 7.119Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gate as well as in Cabell’s Directories of Publishing Opportunities, U.S.A

Another approach to measure the accuracy of the forecast could be by a metric that calculates whether at specific periods the prediction falls below defined precision thresholds. These time instances can be defined from the moment the first forecast is produced or a given absolute time period until EOL is reached as a percentage of the overall remaining life. In our deployment process, we define □ □ □ accuracy as at particular time instance t, the prediction precision to be within *100 percent of the actual RUL represented as a fraction of time between the stage that an algorithm begins to forecast and the actual loss. This statistic, for example, measures if a forecast is 20 percent correct. (i.e., □ □ □ =0.2) halfway to failure from the time the first prediction is made (i.e., □ =0.5).

$$[1-\alpha].r_*(t) \le r'(t_\lambda) \le [1-\alpha].r_*(t)$$

where α : accuracy modifier;□ : time window modifier t = P + λ(EOL − P)

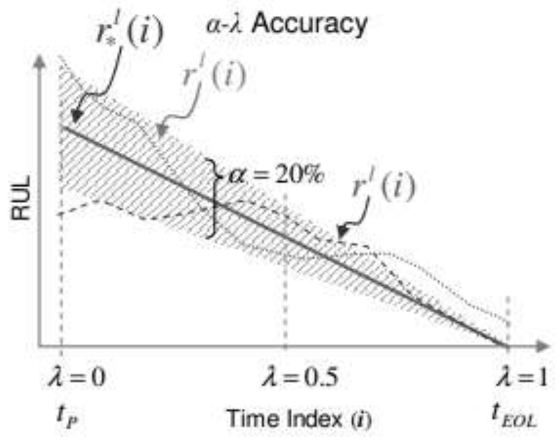


Figure 3Schematic depicting □ □ □ Accuracy.

Relative Accuracy (RA): Relative prediction accuracy is a notion similar to □ □ □ accuracy where, we calculate the accuracy level instead of figuring out whether the forecasts fell beyond a specified level of accuracy at a given moment in time. From the point where the first forecast is made, the period moment is again described as a fraction of the actual remaining useful life. It is beneficial to provide an algorithm of greater relative precision.

$$RA_\lambda = 1 - \frac{|r_*(t_\lambda) - r'(t_\lambda)|}{r_*(t_\lambda)}$$

wheret_λ = P + λ(EOL − P) .

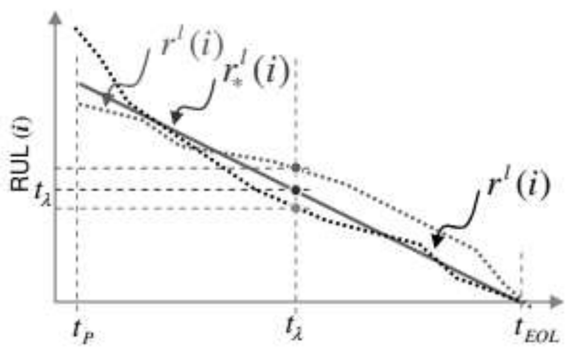


Figure 4Schematic showing Relative Accuracy concepts.

*Cumulative Relative Accuracy (CRA):*Relative precision may be measured in different instances of time. We describe Accumulated Relative Accuracy as a standardized weighted total of relative prediction accuracies at particular time instances to aggregate these precision levels.

$$CRA_{\lambda} = \frac{1}{l}\sum_{i=1}^{l} w(r^{l})RA_{\lambda}$$

where w is a weight factor as a function of RUL at all-time indices. In certain situations, weighing the relative accuracy closer to the EOL is preferable.

Convergence: Convergence is specified to measure the way in which every variable increases over time to achieve its perfect score, such as precision or accuracy. Three cases overlap at varying speeds, as shown below. It can be seen to measure convergence by the difference between the root and the centroid of the field under the curve for a metric. Faster indicates lower reach. Convergence is a valuable measure since if more evidence accumulates over time, we anticipate a prognostic algorithm to converge to the true value. Furthermore to achieve a high trust in holding the forecast horizon as broad as feasible, a faster convergence is required.

Let (xc, yc) be the region's center of mass under the curve M (i). The CM convergence will then be represented by the Euclidean distance between the center of mass and (tp, 0), where the distance between the center of mass is Euclidean:

$$C_M = \sqrt{(x_c - t_p)^2 + y_c^2}\ ,$$

$$x_c = \frac{\frac{1}{2}\sum_{i=P}^{EOP}(t_{i+1}{}^2 - t_i^2)M(i)}{\sum_{i=P}^{EOP}(t_{i+1}{}^2 - t_i^2)M(i)}\ ,$$

$$y_c = \frac{\frac{1}{2}\sum_{i=P}^{EOP}(t_{i+1} - t_i)M(i)^2}{\sum_{i=P}^{EOP}(t_{i+1} - t_i)M(i)}$$

M(i) is a non-negative prediction error accuracy or precision metric.

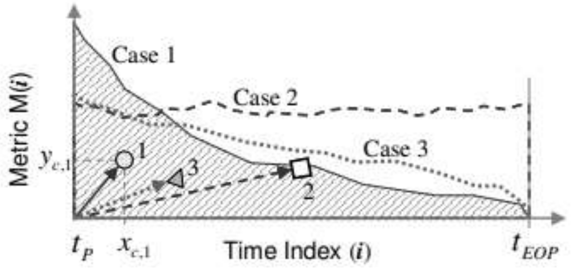
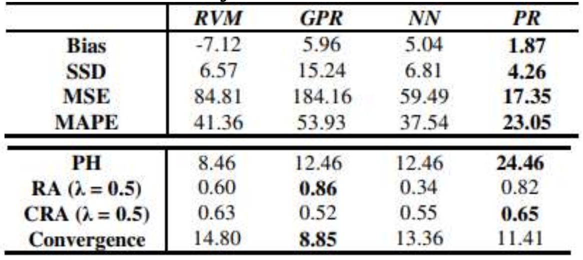


Figure 5Schematic for a metric's convergence.

Battery health measurements were taken every four weeks, as reported earlier. Each algorithm was then charged with forecasting every four weeks after week 32, which provides eight data points to understand the pattern of deterioration. Algorithms forecast RULs before the end-of-prediction is reached, i.e. figures indicate that 70 percent of the maximum potential of one ampere hour has already been reached through battery capacity. Using all nine metrics, subsequent projections are then tested. For all weeks beginning from week 32 through week 64, estimates were available. Algorithms such as RVM were always conservatively expected, i.e. a quicker deterioration was projected than actually observed. Other algorithms such as NN and PR began forecasting at week 32, but were unable to forecast past week 60, as their predictions had already passed the threshold of failure before that. However before it could have any forecasts, GPR needed further training data. Therefore, GPR forecasts begin at week 48 and go until week 60.

Table 3 Efficiency assessment for all four Error Bound test algorithms = 5%



	RVM	*GPR*	*NN*	*PR*
Bias	-7.12	5.96	5.04	**1.87**
SSD	6.57	15.24	6.81	**4.26**
MSE	84.81	184.16	59.49	**17.35**
MAPE	41.36	53.93	37.54	**23.05**
PH	8.46	12.46	12.46	**24.46**
RA ($\lambda = 0.5$)	0.60	**0.86**	0.34	0.82
CRA ($\lambda = 0.5$)	0.63	0.52	0.55	**0.65**
Convergence	14.80	**8.85**	13.36	11.41

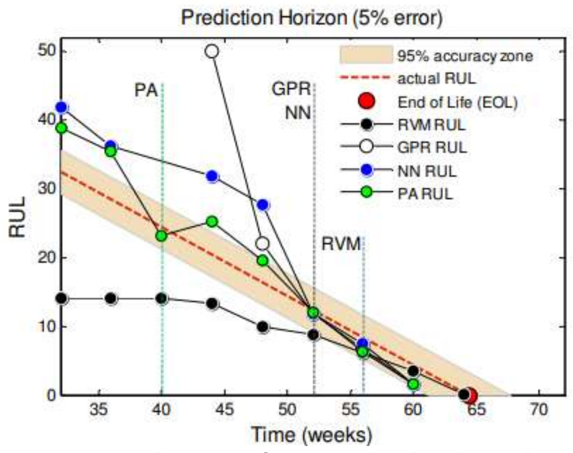
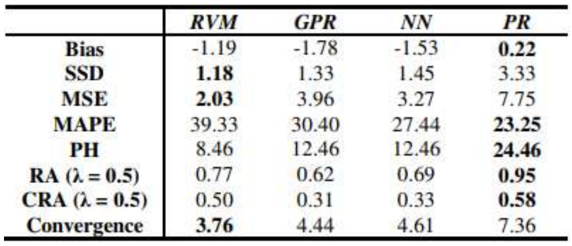


Figure 6Predictions from multiple algorithms at different periods fell beyond the bound mistake.

Table 3 aggregates the findings on the basis of all available forecasts. These observations explicitly demonstrate that in nearly all situations, the polynomial fit solution outperforms all other algorithms. Even if the properties of convergence are not the greatest, they are equal to the highest figures. Using all predictions to measure these metrics, though, results in a broad variety of values, making it impossible to compare how other methods operate, even though they might not actually be the greatest. Many indicators define how near or far the forecasts are to the true value, yet the prediction horizon shows that certain predictions are constrained by the stated error and may thus be deserving of confidence (see Figure 8). Early on, PR reaches the error limit where as all other algorithms steadily converge as periods fly by. This property is encapsulated by the convergence metric which demonstrates that algorithms such as GPR converge quicker for better predictions which can be useful later on. We have also discovered that the new convergence metric does not take into consideration instances when algorithms begin to forecast instances at various times. Algorithms that begin forecasting early on may have a drawback in certain situations. While this metric works well in most situations, to render it stable in severe cases, a few changes might be required.

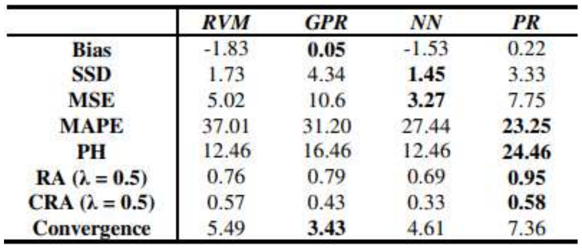
Table 4 Output assessment with Error Bound for all four test algorithms for predictions produced inside the forecast horizon = 5%.



	RVM	*GPR*	*NN*	*PR*
Bias	-1.19	-1.78	-1.53	**0.22**
SSD	**1.18**	1.33	1.45	3.33
MSE	**2.03**	3.96	3.27	7.75
MAPE	39.33	30.40	27.44	**23.25**
PH	8.46	12.46	12.46	**24.46**
RA (λ = 0.5)	0.77	0.62	0.69	**0.95**
CRA (λ = 0.5)	0.50	0.31	0.33	**0.58**
Convergence	**3.76**	4.44	4.61	7.36

The criteria requirements are another component of success measurement. The success appraisal requirements often alter as parameters change. To highlight this argument, the forecast horizon has now been established with a 10% relaxed error bound. For most algorithms, prediction horizons become longer, as predicted, and hence more forecasts are taken into consideration while computing the metrics. With the latest prediction horizons, Table 5 displays the outcomes and now the NN-based method still appears to do well on some metrics. This suggests that if possible, simplified methods which be preferred for those applications where more relaxed criteria are appropriate.

Table 5 Output assessment with Error Bound for all four test algorithms for predictions produced inside the forecast horizon = 10%.



	RVM	*GPR*	*NN*	*PR*
Bias	-1.83	**0.05**	-1.53	0.22
SSD	1.73	4.34	**1.45**	3.33
MSE	5.02	10.6	**3.27**	7.75
MAPE	37.01	31.20	27.44	**23.25**
PH	12.46	16.46	12.46	**24.46**
RA (λ = 0.5)	0.76	0.79	0.69	**0.95**
CRA (λ = 0.5)	0.57	0.43	0.33	**0.58**
Convergence	5.49	**3.43**	4.61	7.36

Figure 8 shows the □ □ □ Metric of consistency withthese four algorithms. Since from week 32 onwards all algorithms except GPR begin estimation, tis calculated to be about 48.3 weeks. Only PR lies within 80 percent precision levels at that stage. GPR continues to forecast week 44 forward, i.e. its tis calculated to be about week 54.3 where the criteria appear to be fulfilled. This metric implies when a specific algorithm approaches halfway to the EOL from the point it begins to estimate within the desired precision level. Another factor that could be of concern is when an algorithm crosses a specified time period ahead of the EOL to the required precision stage. For e.g., in that scenario, if tis selected as 48 weeks then GPR would not fulfill the requirement. This metric can then be adjusted to include situations in which not all algorithms can start forecasting at the same moment.

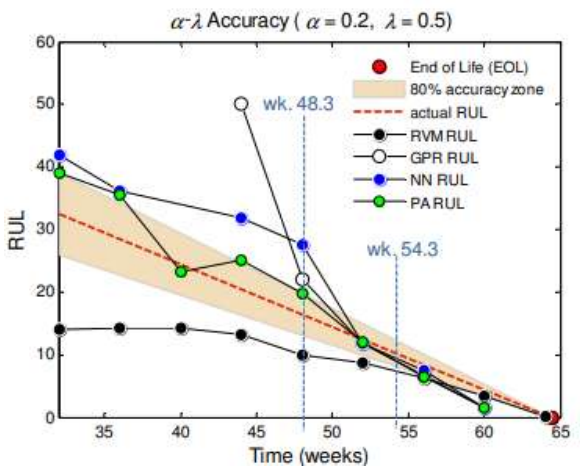


Figure 7The□ □ □ accuracy metric defines when at a given moment, forecasts are beyond the cone of the optimal accuracy ranges (t).

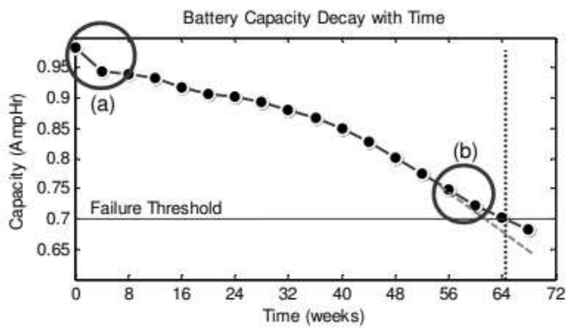


Figure 8The decay profile of battery power reveals many properties that are challenging to learn using basic regression approaches.

From the findings (Figure 8), it can be found that most algorithms struggle to obey the pattern at the end. The physical phenomenon from which batteries decay considers it challenging to learn these data-driven regression-based techniques. As seen in Figure 9, the battery power initially degrades very rapidly. The decay slows at the end of the race. Ses algorithms are not capable of studying this attribute and forecasting an earlier EOL.

Finally, we would like to discuss a few main points that are relevant for measuring success and should be considered before the indicators are selected. In multiple implementations, time scales observed in separate prognostic algorithms are also quite different. For e.g., time scales are on the order of weeks of battery health control, whereas it could be a matter of hours or seconds in other situations like electronics. The chosen metrics should therefore consider the value of the horizon of estimation and weigh errors with higher penalties similar to EOL. First it is necessary to change these metrics to fix asymmetric RUL error preferences. An early prediction is preferable over late predictions in most applications in which a failure can lead to disastrous effects. Finally, RUL figures were collected as a single value in the illustration described in this article, as opposed to an RUL distribution for each projection. For systems where RUL distributions are possible with minor adjustments, the metrics described in this paper may be added. Similarly, small changes are appropriate for cases where several UUTs are required to provide details.

VII. NEED AND FUTURE SCOPE

The technology of Prognostics is approaching a stage that it requires to be tested in a fully integrated manner in real life environments. However, before they can be accredited for sensitive systems, this includes thorough monitoring and appraisal on a number of performance metrics. It becomes necessary to develop a reasonable amount of confidence in the prognostic systems for end-of-life forecasts of vital systems before integrating their predictions into the decision-making method. In addition, success indicators help set specification criteria that have to be fulfilled. It has been challenging to quantify appropriate production limits and specify crisp and unambiguous criteria for designers in the absence of consistent indicators. Performance testing enables multiple algorithms to be compared and often provides positive reviews to further develop these algorithms.

In the sense of linear regression, the focus of this essay is limited to two-term association results. Non-linear regression has other well-developed feature selection heuristic methods, which can be discussed as the future career. In addition, important possible indicators cannot be instantly observable when evaluating software. During the multiple stages of the software creation life cycle, the importance and potential purpose of this analysis involved successful control of interacting attributes. In addition, the human component of software creation, quantified by individual indicators, which play a role in evaluating other software measures that have yet to be extensivelyanalyzed.

VIII. CONCLUSION

We have concluded as the life cycle of software production is transformations continuously that turn an initial artefact into a code artefact, i.e. an end product, in the form of a requirement specification, various metrics used to build models of fault prediction are typically interrelated. In published research, efforts to combine metrics while still considering contact have been practically non-existent. Subramanyam et al. [37] is the most prominent thesis. They concluded that in case studies utilizing Java and C++, the interaction word (CBO*DIT) is strongly correlated with error counts.

The metrics identified in the object-oriented metrics suite of Chidamber and Kemerer (CK)[21] are most widely used in the majority of studies aimed at identifying the fault-proneness of modules. As far as the modelling approach is concerned, Linear and Logistics Regressions are mainly used and account for 68 percent of reported study studies. In information engineering, the effect of contact in designing regression-based fault prediction models is uncommon; however in social and behavioural sciences, two terms and three-term relationships are studied in depth. Interactions above three words are scarce, so it is challenging to perceive interaction results at such a large level. The following study refers to the following:

Empirical study of the influence of interaction in the combined approach to fault prediction metrics: In this thesis, the importance of considering the interaction between metrics is statistically defined, culminating in a substantial increase in the explanatory capacity of the subsequent predictive model. We have used a publicly accessible data set[9] and considered Chidamber and Kemerer(CK) metrics[21], Object Oriented(OO) metrics, as well as a hybrid of the two, with and without considering interaction between the two, to statistically verify the effect of two-term interaction between metrics in the design of prediction models. In isolation as well as in combination, predictive models were developed and experimented with 17 separate metrics (6 CK and 11 OO metrics). Our studies have concluded that the mixture of metrics correctly represents fault-proneness (with interaction). In comparison, the observations obtained from the tests replicated with other programme modules utilizing the same data set were strikingly stable and our results were further confirmed.

Identifying the influential collection of interaction metrics: a parallel rise in interaction is often demonstrated by the amount of variables involved in fault prediction. In comparison, the interacting variables do not equally lead to the model's prediction capabilities. This research contributes to the creation of successful predictive models involving interaction with a reduced range of influential words between predictive variables, obtained by applying stepwise regression.

The predictive models were developed using selected parametric (M5 0 Model Tree) and Non-parametric (K- Nearest Neighbour) regression techniques to evaluate the suitability of the piece-wise linear model and non-parametric regression techniques in the management of broad interacting metrics. Reflecting this the prevalence of non-parametric regression approaches over conventional parametric regression was empirically founded in the successful management of increased numbers of predictors and nonlinearity produced due to complex metric interaction.The degrees of connections between the metrics were defined linguistically, accompanied by the cataloguing of the most prominent laws and influential metrics.

This paper is intended to serve as a start to establish certain metrics that can help summarize the success of prognostic algorithms. Additional knowledge that can be helpful in comparing prognostic algorithms is given by the latest metrics. Specifically, these measures track the progression through time of prediction performance and help assess whether it is appropriate to consider these predictions as trustworthy. Notions such as the horizon of convergence and estimation that have persisted for a long time in literature have been quantified so that they can be utilized in an automatic manner. Furthermore, utilizing metrics such as relative precision and efficiency, new notions of success assessments were instantiated at particular time instances. These metrics embody the idea that a forecast is only valuable if it is accessible sufficiently far in advance that it causes the expected contingency to be mitigated for some period.

Terms and Notations

- UUT is the unit under test

- $\Delta^l(i)$ is the error between the predicted and the true RUL at time index i for UUT l.
- EOP (End-of-Prediction) is the earliest time index, i, after prediction crosses the failure threshold.
- EOL represents End-of-Life, the time index for actual end of life defined by the failure threshold.
- P is the time index at which the first prediction is made by the prognostic system.
- r^l (i) is the RUL estimate for the *l*th UUT at time ti as determined from measurement and analysis.
- r_*^l (i) is the true RUL at time ti given that data is available up to time ti for the *l*th UUT.
- *l* is the cardinality of the set of all time indices at which the predictions are made, i.e. *l*= (i | P ≤ i ≤ EOP)

REFERENCES

1. BurakTurhan, cetin Mericcli, and TekinMericli. Better, faster, and cheaper: what is better software? In Proceedings of the 6th International Conference on Predictive Models in Software Engineering, page 11. ACM, 2010.
2. BeataCzarnacka-Chrobot. The iso/iec standards for the software processes and products measurement. In SoMeT, pages 187–200, 2009.
3. Stefan Wagner. Quality planning. In Software Product Quality Control, pages 91–110. Springer, 2013.
4. Ho-Won Jung, Seung-Gweon Kim, and Chang-Shin Chung. Measuring software product quality: A survey of iso/iec 9126. IEEE software, 21(5): 88–92, 2004.
5. AhmetOkutan and OlcayTanerYildiz. Software defect prediction using bayesian networks. Empirical Software Engineering, pages 1–28, 2012.
6. Marco DAmbros, Michele Lanza, and RomainRobbes. Evaluating defect prediction approaches: a benchmark and an extensive comparison. Empirical Software Engineering, 17(4-5):531–577, 2012.
7. DindinWahyudin, Alexander Schatten, Dietmar Winkler, A Min Tjoa, and Stefan Biffl. Defect prediction using combined product and project metrics-a case study from the open source. In Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro Conference, pages 207–215. IEEE, 2008.
8. Taek Lee, Jaechang Nam, DongGyun Han, Sunghun Kim, and Hoh Peter In. Micro interaction metrics for defect prediction. In SIGSOFT FSE, pages 311–321, 2011.
9. Marco DAmbros, Michele Lanza, and RomainRobbes. An extensive comparison of bug prediction approaches. In Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on, pages 31–41. IEEE, 2010.

International Journal of Management, IT & Engineering
Vol. 8 Issue 5, May 2018,
ISSN: 2249-0558
Impact Factor: 7.119Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gate as well as in Cabell's Directories of Publishing Opportunities, U.S.A

10. GoebelK., B. Saha, A. Saxena, J. Celaya, and J. P. Christopherson, 'Prognostics in Battery Health Management,' in IEEE Instrumentation and Measurement Magazine. vol. 11, pp. 33 - 40, 2008.
11. GoebelK., B. Saha, and A. Saxena, 'A Comparison of Three Data-Driven Techniques for Prognostics,' in 62nd Meeting of the Society For Machinery Failure Prevention Technology (MFPT) Virginia Beach, VA, pp. 119-131, 2008.
12. Tipping, M. E., 'The Relevance Vector Machine,' in Advances in Neural Information Processing Systems. vol. 12 Cambridge MIT Press, pp. 652-658, 2000.
13. VapnikV. N., The Nature of Statistical Learning. Berlin: Springer, 1995
14. Rasmussen C. E. and C. K. I. Williams, Gaussian Processes for Machine Learning: The MIT Press, 2006
15. Williams C. K. I. and C. E. Rasmussen, 'Gaussian Processes for Regression,' in Advances in Neural Information Processing Systems. vol. 8, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. Cambridge, MA: The MIT Press, pp. 514-520, 1996.
16. MardiaK. V. and R. J. Marshall, 'Maximum Likelihood Estimation for Models of Residual Covariance in Spatial Regression,'Biometrika, vol. 71, pp. 135-146, 1984
17. DudaR. O., P. E. Hart, and D. G. Stork, Pattern classification, Second ed. New York: John Wiley & Sons, Inc., 2000.
18. HastieT., R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second ed., 2003.
19. Christof Ebert and Reiner Dumke. Measurement foundations. Software Measurement: Establish Extract Evaluate and Execute, pages 41–72, 2007.
20. A Gunes Koru and Hongfang Liu. Building effective defect-prediction models in practice. Software, IEEE, 22(6):23–29, 2005.
21. Shyam R Chidamber and Chris F Kemerer. A metrics suite for object oriented design. Software Engineering, IEEE Transactions on, 20(6):476–493, 1994.
22. Joseph Lee Rodgers and W Alan Nicewander. Thirteen ways to look at the correlation coefficient. The American Statistician, 42(1):59–66, 1988.
23. Andrian Marcus, Denys Poshyvanyk, and Rudolf Ferenc. Using the conceptual cohesion of classes for fault prediction in object-oriented systems. Software Engineering, IEEE Transactions on, 34(2):287–300, 2008.
24. Karim O Elish and Mahmoud O Elish. Predicting defect-prone software modules using support vector machines. Journal of Systems and Software, 81(5):649–660, 2008.
25. Swapna S Gokhale and Michael R Lyu. Regression tree modeling for the prediction of software quality. In proceedings of the Third ISSAT International Conference on Reliability and Quality in Design, pages 31–36. Citeseer, 1997.

26. Taghi M Khoshgoftaar, Edward B Allen, and Jianyu Deng. Using regression trees to classify fault-prone software modules. Reliability, IEEE Transactions on, 51(4):455–462, 2002.
27. BibiS, G Tsoumakas, I Stamelos, and I Vlahavas. Regression via classification applied on software defect estimation. Expert Systems with Applications, 34(3):2091–2101, 2008.
28. LanGuo, Yan Ma, BojanCukic, and Harshinder Singh. Robust prediction of fault-proneness by random forests. In Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on, pages 417–428. IEEE, 2004.
29. Ben Yuhas and Nirwan Ansari. Neural networks in telecommunications. Springer Publishing Company, Incorporated, 2012.
30. Jun Zheng. Cost-sensitive boosting neural networks for software defect prediction. Expert Systems with Applications, 37(6):4537–4543, 2010.
31. Taghi M Khoshgoftaar, Edward B Allen, John P Hudepohl, and Stephen J Aud. Application of neural networks to software quality modeling of a very large telecommunications system. Neural Networks, IEEE Transactions on, 8(4):902–909, 1997.
32. Harsh Kumar Verma and Vishal Sharma. Handling imprecision in inputs using fuzzy logic to predict effort in software development. In Advance Computing Conference (IACC), 2010 IEEE 2nd International, pages 436– 442. IEEE, 2010.
33. Sultan Aljahdali and Alaa F Sheta. Predicting the reliability of software systems using fuzzy logic. In Information Technology: New Generations (ITNG), 2011 Eighth International Conference on, pages 36–40. IEEE, 2011.
34. Thomas Back, David B Fogel, and ZbigniewMichalewicz. Handbook of evolutionary computation. IOP Publishing Ltd., 1997.
35. Danielle Azar and J Vybihal. An ant colony optimization algorithm to improve software quality prediction models: Case of class stability. Information and Software Technology, 53(4):388–393, 2011.
36. Taghi M Khoshgoftaar and NaeemSeliya. Fault prediction modeling for software quality estimation: Comparing commonly used techniques. Empirical Software Engineering, 8(3):255–283, 2003.
37. RamanathSubramanyam and Mayuram S Krishnan. Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects. Software Engineering, IEEE Transactions on, 29(4):297–310, 2003.