ISSN: 0976-3104
SPECIAL ISSUE: Emerging Technologies in Networking and Security (ETNS)
Rao and Kumar

ARTICLE OPEN ACCESS

GENERALIZED REGRESSION NEURAL NETWORK FOR SOFTWARE DEFECT ESTIMATION

Sankara Rao[1*] and ReddiKiran Kumar[2]
[*1]*JNTU, Kakinara, Andhra Pradesh, INDIA*
[2]*Krishna University, Machilipatnam, Andhra Pradesh, INDIA*

ABSTRACT

Software development estimation is important task in managing huge software projects. It is well known that software industry is unable to properly estimate effort, time and development cost. Many estimation models exist for effort prediction but there is a need for a new model to get more accurate estimates. This paper proposes a Generalized Regression Neural Network (GRNN) to use improved software estimation effort for COCOMO dataset. This paper uses Mean Magnitude Relative Error (MMRE) and Median Magnitude Relative Error (MdMRE) as evaluation criteria. The new GRNN is compared to varied techniques like linear regression, M5, RBF kernel and Sequential Minimal Optimization (SMO) Poly kernel.

Published on: 14th– August-2016

KEY WORDS

Effort estimation, Constructive Cost Model (COCOMO), Magnitude Relative Error (MMRE) and Median Magnitude Relative Error (MdMRE)

***Corresponding author: Email:** psankararao.cse@gmail.com

INTRODUCTION

A software defect is a condition which fails to meet software requirements or end user expectations in software products. A defect is an error/bug in coding/logic causing a program malfunction or produces incorrect or unanticipated results. Software defect prediction locates defective modules in software. To ensure high quality software, the final product should have very few defects. Early defects detection leads to reduced time, development cost and rework and reliable software. So, defect prediction is for good software quality. Software defect prediction metrics have a big role in constructing a statistical defect prediction model for by software organizations during early software development to identify defect modules [1].

Software cost estimation predicts effort to develop a software system. Many estimation models were used over three decades. Computing power is a subordinate resource for software developing companies as it doubles every 18 months, costing a fraction compared to late 60's. Personnel costs are an important expense in a software company's budget. In view of this, proper planning is a key aspect for companies. Software community developed tools/techniques like effort, size and cost estimation to offset challenges facing software development projects management. These tools/techniques are used for software development phases starting with software requirements specification. As demand for software applications goes up continually and software scope/complexity go higher software companies need accurate estimates of under development projects. Good software effort estimates are critical to companies/clients [2].

Software effort estimation methods are categorized as algorithmic/non-algorithmic methods. The former are mainly COCOMO, Function Points and Software Life-Cycle Model (SLIM). They are also called parametric methods as they predict software development effort using fixed mathematical formula parameterized from historical data. But, preliminary stage estimates of a project are difficult to get as primary estimate effort source is from a SRS document. They also face problems in modelling inherent complex relationships. Algorithmic methods limitations opt for non-algorithmic methods based on soft computing. These methods learn from previous data and model complex relationship between dependent (effort) and independent variables [3].

THE IIOAB JOURNAL
ISSN: 0976-3104

Costs and efforts are predicted by using mathematical formulae in algorithmic cost estimation. Formulae are historical data based. A known algorithmic cost model called COCOMO published by Barry Boehm in 1981 was developed from analyzing 63 software projects. Boehm proposed three model levels called Basic COCOMO, Intermediate COCOMO and Detailed COCOMO. Intermediate COCOMO is described as follows [4]:

Intermediate COCOMO: Basic COCOMO is based on relationship: Development Effort (DE),

$$DE = a*(SIZE)b \quad (1)$$

Where, SIZE is measured in 10000 delivered source instructions. Constants a, b are dependent upon 'mode' of projects development. DE is measured in man-months. Boehm proposed 3 project modes:

- Organic mode – simple projects engaging small teams working in known/stable environments.
- Semi-detached mode – projects engaging teams with a mixture of experience. It is between organic and embedded modes.
- Embedded mode – complex projects developed under tight constraints with changing requirements.

Basic COCOMO accuracy is limited as it does not consider hardware, personnel, modern tools use and attributes affecting project cost. Also, Boehm proposed an Intermediate COCOMO that adds accuracy to Basic COCOMO by multiplying 'Cost Drivers' into an equation with a new variable: Effort Adjustment Factor (EAF) seen in **[Table -1]** [5].

Table: 1. DE FOR THE INTERMEDIATE COCOMO

Organic	
Semi-detached	$DE = EAF*3.0*(\text{SIZE})^{1.12}$
Embedded	$DE = EAF*2.8*(\text{SIZE})^{1.2}$
Semi-detached	$DE = EAF*3.0*(\text{SIZE})^{1.12}$

An Artificial NN (ANN) is an information processing system resembling a biological neural network in characteristics. ANN's have many highly interconnected processing elements named neurons which are connected to others through a connection link. Each ink is associated with weights have input signal information. Neuron net uses the information to solve specific problems. A neuron has its own internal state called neuron activation level which is a function of inputs received by a neuron. There are many activation functions applied over net input like Gaussian, Sigmoid, Linear and Tanh. Neural nets frequently use sigmoid function [6].

This paper proposes to investigate MMRE/MdMRE using techniques like M5, SMOPolykernel, Linear regression, RBF kernel and the new GRNN. The COCOMO dataset is used for investigations. The paper is organized as follows: section 2 deals with related work, section 3 details materials and methods used. Section 4 provides experiments results and discussion of the same and section V concludes the paper.

RELATED WORKS

Kernel principal component analysis (KPCA) with Kernel Density Estimation (KDE) approach was applied to Tennessee Eastman process to detect faults by Samuel & Cao [7]. Results confirmed that associating KPCA with kernel density estimated control limits ensured better monitoring than using normal probability density function based control limits.

An innovative idea of the working of Principal Component Analysis (PCA) with ANN by keeping base of Constructive Cost Model II (COCOMO II) was presented by Patil et al., [8]. Feed forward ANN used delta rule learning method train a network. ANN training was PCA and COCOMO II sample dataset repository based. PCA was a classification method which filters multiple input values into certain values. It reduces the gap between actual/estimated effort. Test results from the hybrid model were compared to COCOMO II and ANN.

Use of KPCA as an approximation technique for nonlinear thermodynamics or kinetic functions parameterized using available plant archived data was explored by Mukhopadhyay et al., [9]. Simulation on a complex binary distillation column proved the new approach's applicability.

THE IIOAB JOURNAL ISSN: 0976-3104

An adaptation mechanism for a mixture of Gaussian process regression models based soft sensor model was proposed by Grbic et al., [10]. Also presented was a procedure for input variable selection based on mutual information. This selects most important input variables for output variable prediction, simplifying the model for development/adaptation. This soft sensor is used for adaptive process monitoring in addition to online prediction of difficult-to-measure variables. The proposed method's efficiency was benchmarked with commonly applied recursive PLS and recursive PCA method on Tennessee Eastman process and on two real industrial examples.

A popular component analysis methodology, i.e., PCA, in complex reproducing kernel Hilbert spaces (CRKHS) formulated by Papaioannou&Zafeiriou [11] defined a widely linear complex kernel PCA framework. Also it shows how to efficiently perform linear PCA in small sample sized problems. Finally, it shows the new framework's usefulness in robust reconstruction using Euler data representation.

Performance of the aforementioned feature selection methods on LR and ℓ1-regularized logistic regression using different statistical measures was assessed by Musa [12]. Varied performance metrics like sensitivity, precision, specificity, accuracy, area under receiver operating characteristic curve and receiver operating characteristic analysis was used. This study included a comprehensive statistical analysis.

A new method to assign weights to features by considering their specific importance on cost was proposed by Tosun et al., [13]. Two weight assignment heuristics inspired by a popular statistical technique called PCA was used.

Potential/accuracy of MART as a new software effort estimation model compared to recently published models like neural networks, Radial Basis Function (RBF) linear regression, and Support Vector regression models with linear and RBF kernels was evaluated by Elish [14]. Comparison was based on a NASA software project dataset.

A new model using COCOMO II, 5 Scale factors and 17 Effort multipliers used as input was proposed by Attarzadeh&Ow [15]. A sigmoid activation function created a network to accomplish post architecture COCOMOII model. Results regarding MMRE, and Pred (0.25) were compared to traditional COCOMO.

Kalichanin-Balich&& Lopez-Martin [16] used Feed forward NN to estimate software development effort on short-scale projects. Totally 132 projects verified the new mechanism. Accuracy was measured regarding MER, i.e., MMER was 0.26, LRM 0.26 and NN 0.25.

A Modified MMRE proposed Dave &Dutta [17] used NASA dataset of 60 projects. They undertook experiments with three differing evaluation methods, i.e., MMRE, Modified MMRE, and Relative Standard Deviation (RSD). Three estimation modes were used, i.e., FFNN, Regression analysis and RBFNN. RBFNN was found to be better for effort estimation based on RSD and Modified MMRE according to the authors.

To ensure good results for problems with noise inputs, complex relationships between inputs and outputs and where inputs had high noise levels RBFN was proposed by Srichandan [18]. COCOMO 81 and Tukutuku were datasets. Clustering algorithm configured RBFN hidden layer. After using widths for models, it was found RBF accuracy using minimum width was better than using maximum width.

Various parameters affecting software development effort studied by Park &Baek [19] identified six variables other than software size for accurate effort estimation by using NN. Authors compared NN model with the two current regression models and human expert judgments. It was revealed that NN model was more accurate than other estimation procedures.

A NN based model and stepwise regression model for software development effort was implemented by de BarcelosTronto et al., [20]. Author reported results restate that NN model estimated software development effort more accurately. Authors compared results with multiple regression, COCOMO and SLIM models, showing that NN model suited effort estimation.

A multilayer feedforwardNN to accommodate the COCOMO model was proposed by Reddy &Raju [21]. COCOMO database consisting 63 projects was the dataset. Data set was divided into training and validation sets in a 80 %: 20 % ratio. Training set had 50 randomly chosen projects while the validation set had 13 projects.

THE IIOAB JOURNAL
ISSN: 0976-3104

MATERIALS AND METHODS

This section describes COCOMO dataset, RBFN, KPCA, MMRE, Linear regression, GRNN methods.

PROMISE EFFORT ESTIMATION DATASET COCOMO

COCOMO dataset has details of 63 software projects each described by 16 cost derivers or effort multipliers. Of 16 attributes, 15 are measured on a scale of six categories: *very low, low, nominal high, very high, and extra high*. A numeric values represents categories. Kilo Delivered Source Instructions (KDSI) is a numeric attribute. COCOMO dataset assesses new techniques comparative accuracy. **[Figure -1]** shows a COCOMO dataset's effort histogram [22].

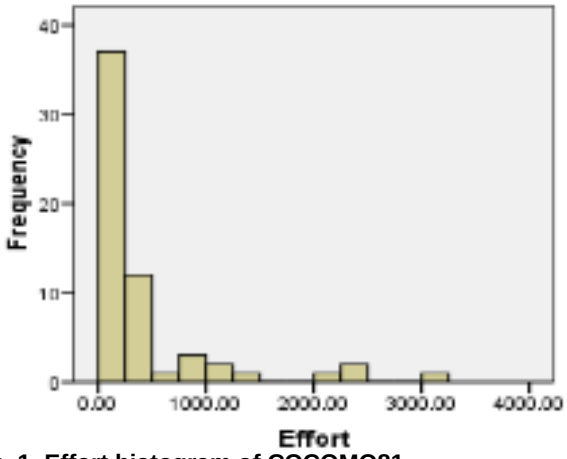

Fig: 1. Effort histogram of COCOMO81

REGRESSION ANALYSIS WITH MEAN MAGNITUDE RELATIVE ERROR (MMRE) AND MEDIAN MAGNITUDE RELATIVE ERROR (MDMRE) FORMULA

An effort predictor's value is reported in many ways including MMRE and probability of a project having relative error less than or equal to L (PRED (L)). MMRE and PRED (L) are the accepted evaluation criteria to evaluate different software effort estimations [23].

MMRE and PRED are computed from relative error, or RE, the relative size of difference between actual/estimated value of individual effort i

$$RE_i = (predicated\ effort_i - actual\ effort_i) / (actual\ effort_i) \quad (2)$$

Magnitude of Relative Error (MRE) was calculated by taking absolute value of relative error that is,

$$MRE_i = abs(\mathrm{RE}_i) \quad (3)$$

MRE value is calculated for every observation i of actual/predicted effort. MRE aggregation over multiple observations (N) is achieved through Mean MRE (MMRE) as follows:

$$MMRE = \frac{1}{N}\sum_{i}^{N} MRE_i \quad (4)$$

A complementary criterion is a prediction at level L, Pred (L) = k/N, where k is number of observations where MRE is less than/equal to L and N is total observations. Thus, Pred (25) gives projects percentage predicted with a MRE less than or equal to 0.25.

MMRE computes average of MREs over reference projects. As MMRE is susceptible to an individual outlying prediction, MdMRE is adopted when many observations are available. MREs median for n projects is MdMRE less sensitive to extreme values of MRE adopted. Despite this, MMRE is used in estimation accuracy. MMRE was criticized as it is unbalanced for validation procedures and often results in overestimation [24].

$$MdMRE = \underset{i}{median}\left(MRE_i\right) \quad (5)$$

THE IIOAB JOURNAL
ISSN: 0976-3104

KERNEL PRINCIPAL COMPONENT ANALYSIS (KPCA)

In KPCA, this is crucial at two levels. From a practical point of view, this connection allows reduction of Eigen decomposition of (infinite dimensional) empirical kernel covariance operator to Eigen decomposition of kernel Gram matrix, which makes an algorithm feasible. From theory's view, it is a bridge between kernel covariance's spectral properties and those of the kernel integral operator [25].

So, KPCA's properties theoretical insight goes beyond this algorithm with direct consequences for understanding the kernel matrix/kernel operator's spectral properties. This makes a study of KPCA interesting: kernel Gram matrix is a central object in kernel methods and its spectrum has a major role in kernel algorithms; this was shown in Support Vector Machines (SVM). Understanding the kernel matrices eigenvalues behaviour, their stability and how they relate to eigenvalues of corresponding kernel integral operator is crucial to understand kernel-based algorithms statistical properties.

KPCA is a PCA's functional generalization similar to Locally Linear Embedding, Isomap or spectral clustering methods. It allows as many principal components as data samples in a training set, with directions being nonlinear [26].

The approach's rationale is considering straight lines to recover from n-dimension space are data's principal components. Though linear, PCA cannot be used as:

- directions looked for are not orthogonal;
- There are more directions to find than space dimension (under determined context).

Though Kernel PCA was defined in RKHSs, same framework applies to reproducing kernel space, provided it has a representer theorem.

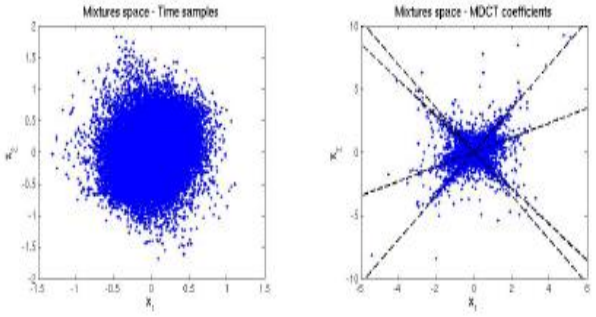

Fig: 2(a). Scatter plot of x1 w.r.t x2 in the time-domain. (b) Scatter plot of x1 w.r.t x2 in the MDCT domain - the dashed lines represent the directions of the mixing matrix

……

It refers to Algorithm 1 for precise descriptions of algorithm involving KPCA in underdetermined contexts. For a given (reproducing) kernel $k : X \times X \to \square$ and data set $\{x_t\}_{t=1,.....,N}$, one performs Eigen decomposition of a kernel matrix defined as:

$$K = [k_{tt'}]_{t,t'=1,.....,N}$$

$$with: \; k(x_t, x_{t'}) \tag{6}$$

(in PCA, this is performed over correlation matrix E[xxT]). Directions corresponding to n largest eigenvalues are scatter data line estimates.

It has to assure that K's eigenvectors are recoverable, i.e. that are expressed as a finite linear kernels combination evaluated on training data. This is got through a representer theorem that extends Wahba and Kimeldorf's to Krein reproducing spaces, thereby ensuring that one expresses the eigenvectors as:

$$v_i(X) = \sum_{t=1}^{N} \alpha_t^i k(x, x_t) \tag{7}$$

With $\{\alpha_1^i, ..., \alpha_N^i\} \text{ in } \square^2$, for any x in R^2, and v_i eigenvector corresponding to ith largest eigenvalue. This Equation proves that KPCA yields linear directions if the kernel k (·, ·) is linear. Next paragraph is dedicated to presenting kernel in Algorithm 1.

Algorithm 1: KPCA for mixing matrix estimation.

THE IIOAB JOURNAL ISSN: 0976-3104

Step 0: *Initialization*

Set $\eta, \theta_0 (m, N$ *and* n *are known*);

Define reproducing kernel $k(.,.)$ *as in equation* (8);

Define $x_t = [x_{1,t} x_{2,t}]^T$; *perform shrinkage with threshold*

η *on the* $\| x_t \|$ *x's; map* (*by symmetry*) *all remaining*

x_t *'s to the positive half - plane.*

Step 1: *Krein* kernel *principal component analysis*

Define matrix $K = [k_{tt'}]_{t,t'=1,.....,N}$ *as in equation* (6);

Find the eigenvalues and eigenvectors for K.

Step 2: *Estimation of the mixing matrix A*

Keep the eigenvectors corresponding to the

n *largest eigen - values*;

Express the corresponding n directions

in X *using equation* (7).

Fig: 3. Reproducing kernel k (•, •) plotted as a function (left); eigenvalues for k (•, •) (right)

The KPCA kernel needs to be:

- Linear, as it looks for lines in R^m;
- Parameterized so that KPCA can be tuned according to problem difficulty (sparsity of the sources? noisy data?);
- Designed in a Krein framework (obvious linear kernel $< .,. > X$ which is a Hilbert rk that cannot be parameterized regarding angles).

Denoting $(\hat{x_t, x_{t'}})$ angle between x_t and $x_{t'}$, kernel

$$k_{\theta_0}(x_t, x_{t'}) = \begin{cases} \dfrac{\cos(\hat{x_t, x_{t'}}) - \cos\theta_0}{1 - \cos\theta_0} & if \ (\hat{x_t, x_{t'}}) \le \theta_0 \\ 0 & else \end{cases} \quad (8)$$

$$(x, x') \to \frac{k(x, x')}{\sqrt{k(x,x)k(x',x')}}$$

Satisfies all the required conditions (it recall that if k (·, ·) is a RKKS kernel, then so is and that $\cos(\hat{x_t, x_{t'}}) = < x_t, x_{t'} > X$ as the xt's have unit norm). This kernel is plotted in **[Figure -3]**, with eigenvalues.

Computing dot products in feature space:From $\tilde{K} : \tilde{K}_{ij} = (\tilde{\phi}(y_i).\tilde{\phi}(y_k))$ Equation, it is seen that to compute kernel matrix, only vectors dot products in feature space F are required, while explicit calculation of map U(y) need not be known. A dot product is computed through use of a kernel function. This is a ''kernel trick''. The Mercer kernels alone are used as a kernel function [27].

Kernel function $k(y_i, y_j)$ calculates dot product in space F directly from input space vectors RM:

$$k(y_i, y_j) = (f(y_i).f(y_j)) \quad (9)$$

Common kernel functions are Gaussian kernel and polynomial kernel.

Kernel for linear PCA: If the kernel function is chosen as polynomial kernel of order one

$$k(y_i, y_j) = (y_i, y_j) \quad (10)$$

Then linear PCA is performed on sample realizations. Using a kernel matrix to perform linear PCA is the same as ''method of snapshots'' known in reduced-order modeling. This method is computationally efficient than K–L expansion's standard implementation. Using a kernel matrix, an eigenvalue problem of size N*N is needed, whereas in size of eigenvalue problem is M*M. In most cases, available experimental data is smaller than data dimensionality.

THE IIOAB JOURNAL ISSN: 0976-3104

Kernel for nonlinear PCA: Choosing a nonlinear kernel function results in nonlinear PCA performance. A common kernel function is Gaussian kernel:

$$k(y_i, y_j) = \exp\left(-\frac{\| y_i - y_j \|^2}{2\sigma^2}\right) \quad (11)$$

Where $\| y_i - y_j \|$ is squared L2-distance between two realizations. A kernel width parameter σ controls kernel flexibility. A larger value of σ allows more ‘‘mixing’’ between realizations elements, whereas smaller value of σ uses few significant realizations. A choice for σ is average minimum distance between two realizations in input space:

$$\sigma^2 = c\frac{1}{N}\sum_{i=1}^{N}\min_{j\neq i} \| y_i - y_j \|^2, \quad j = 1,\ldots\ldots,N, \quad (12)$$

Where c is a user-controlled parameter.

Why KPCA is better than PCA: 1) PCA does not support mean for multilayer NN. 2) Large dataset like 0.000009 assume non-linear value PCA does not take nonlinear space value [28]. KPCA identifies the kernel’s principal directions where data varies largely. 3) PCA supports mapping and so PCA works with single layer NN. A huge data set in practice leads to huge K, storing which is an issue. A way to handle this is to perform clustering on the huge dataset, populating the kernel with the clusters means. 4) KPCA supports implicit mapping and so it works with multilayer NN.

LINEAR REGRESSION

Regressions techniques predict software evaluate accuracy in evaluation/validation. A Regression Analysis views effect of independent variables on a dependent variable. It aims to see how much dependent variable are independent variables based. Linear regression is a statistical technique used for prediction or evaluating a linear interrelationship between two numerical variables.

A linear regression model having exponential transformation predicts relations between variables involving software size and effort to raise reliability in software effort estimation. It changes the independent variable value and sees resulting dependent variable change. The aim is to locate to what extent a dependent variable is described using an independent variable. Simple Linear Regressions have one dependent and independent variable each [29].

$$Y - a + bX + C \quad (13)$$

Where
Y: Dependent variable X: Independent variable
b: Coefficient of variable X
a: Y intercept
C: Constant
More than one independent variable describes dependent variable change in Multiple Linear Regression Analysis [30].
C: Constant
Consider the problem of approximating set of data,

$$D = \left\{\left(x^1, y^1\right),\ldots.,\left(x^l, y^l\right)\right\}, \quad x \in \Re^n, y \in \Re \quad (14)$$

With a linear function,

$$f\left(x\right) = \left\langle w, x\right\rangle + b \quad (15)$$

The optimal regression function is given by the minimum of the functional,

$$\Phi\left(w, \xi\right) = \frac{1}{2}\left\|w\right\|^2 + C\sum_i\left(\xi_i^- + \xi_i^+\right) \quad (16)$$

Where C - a pre-specified value, and ξ- , ξ+ - slack variables that represent system outputs upper and lower constraints.

SMOPolykernel and Radial Basis Neural Networks (RBFN) kernel

The RBFN NN involves three layers. An input layer of sources nodes (cost drivers); a hidden layer where neurons compute output using RBF, which is a Gaussian function, and an output layer that constructs a linear hidden neuron outputs weighted sum and supplies the network’s response (effort). A RBF NN configured for software effort estimation has an output neuron. Hence, it implements an output-input relation in the Equation which is composition of nonlinear mapping realized by hidden layer realized by the output layer’s linear mapping [31]

$$F(\mathrm{x}) = \sum_{j=1}^{M}\beta_j\phi_j(\mathrm{x}) \quad (17)$$

THE IIOAB JOURNAL ISSN: 0976-3104

Where M is number of hidden neurons, $x \in \square^{p}$ is input, β_j are RBFN networks output layer weights and $\phi(x)$ is Gaussian RBF given by:

$$\phi_j(x) = e^{\left(\frac{\|x-c_j\|^2}{\sigma_j^2}\right)} \quad (18)$$

Where $c_j \in \square^{p}$ and σ_j are center and width of jth hidden neuron and ||.|| denotes Euclidean distance.
RBFN are powerful alternatives approximating classifying a pattern set some times better than Multi-Layer Perceptron (MLP) NN [32].

RBFs differ from MLPs as an overall input-output map is constructed from local Gaussian axons contributions and need fewer training samples training faster than a MLP. A popular method to estimate centers and widths includes using an unsupervised technique called k-nearest neighbour rule. The clusters centers give RBF's centers and distance between clusters is the Gaussians width. Centers computation, used in RBF NN kernels function is main focus to achieve efficient algorithms in the pattern set's learning process. Adequate centers choice implies high performance, concerning convergence, learning times and generalization.

SVMs based methods are for classification [33]. For a training data $(x_i, y_i),\ i=1,..,n$, where $x_i \in \Re^d$ is a feature vector and $y_i \in \{+1,-1\}$ indicates class value of x_i solves the optimization problem:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \quad (19)$$

Subject to

$$y_i\left(w^T\Phi(x_i)+b\right) \geq 1-\xi_i \text{ for i=1...n}$$

$$\xi_i \geq 0 \quad (20)$$

Where $\Phi : \Re \to H$, H being a high dimensional space $w \in H,$ and $b \in \Re$. $C \geq 0$ Is a parameter controlling margin errors minimization and margins maximization. Φ is chosen that an efficient kernel function K exists. In practice, this optimization problem is solved using Lagrange Multiplier. SMO [34] is an to solve SVM QP problem. Its advantage is its ability to solve Lagrange multipliers without numerical QP optimization. The Lagrangian form is as follows :

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K\left(x_i,x_j\right) - \sum_{i=1}^{n}\alpha_i \quad (21)$$

Subject to

$$0 \leq \alpha_i \leq C \quad \text{for i=1...n}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \quad (22)$$

On solving optimization problem, w is computed as [35]:

$$w = \sum_{i=1}^{n}\alpha_i y_i \Phi\left(x_i\right) \quad (23)$$

x_i Is a support vector if $\alpha_i \neq 0$. New instance x is computed by:

$$f(x) = \sum_{i=1}^{n_S}\alpha_i y_i K\left(s_i,x\right) + b \quad (24)$$

Where si are support vectors and nS is number of vectors.
The polynomial kernel function is given by:

$$K\left(x_i,x_j\right) = \left(\gamma x_i^T x_j + r\right)^d, \text{ where } \gamma>0 \quad (25)$$

And the Radial basis function (RBF) kernel:

$$K\left(x_i,x_j\right) = \exp\left(-\gamma\left\|x_i - x_j\right\|^2\right), \text{ where } \gamma>0 \quad (26)$$

M5 algorithm

Tree-building algorithms like C4.5 determine which attributes best classify remaining data, followed by iterative tree construction. Decision trees immediate conversion to rules easily interpreted by decision-makers is their advantage. For numeric data mining

THE IIOAB JOURNAL
ISSN: 0976-3104

prediction, it is common to use regression or model trees [36]. Both build decision tree structures where leaves are responsible for a specific input space local regression. The difference is that while regression trees generate constant output values for input data subsets (zero-order models), model trees generate linear (first-order) models for every subset.

M5 algorithm builds trees with leaves being linked to multivariate linear models and tree nodes are chosen over an attribute maximizing expected error reduction as a standard deviation function of output parameter. M5 algorithm builds decision trees that divide attribute space in orthohedric clusters, with border paralleling the axis. Their advantage is their being converted o rules easily; each tree branch has the following condition: attribute ≤ value or attribute > value.

GENERALIZED REGRESSION NEURAL NETWORKS (GRNN)

ANN is used for cost estimation as it learns from earlier data. Three factors defining ANN are: i) interconnection pattern between different neuron layers, ii) learning process to update interconnections weights and iii. activation function converting a neuron's weighted input to output activation.

Network nodes are split into input layer which is linked to weights having information on input signals and output layer which goes through network nodes in hidden layer [37].

A basic NN has inputs applied by weights combined to give an output. Different NN learning algorithms like Delta rule leaning, Perceptron learning and back propagation learning are used. It uses Delta rule learning algorithm to train a NN and solve different problems. Delta rule learning algorithm uses sigmoid activation function in which every neuron has continuous activation function rather than threshold activation function.

GRNN is a RBF based, supervised learning model used for classification, regression and time series predictions. GRNN architecture is seen in **[Figure -4]** [38].

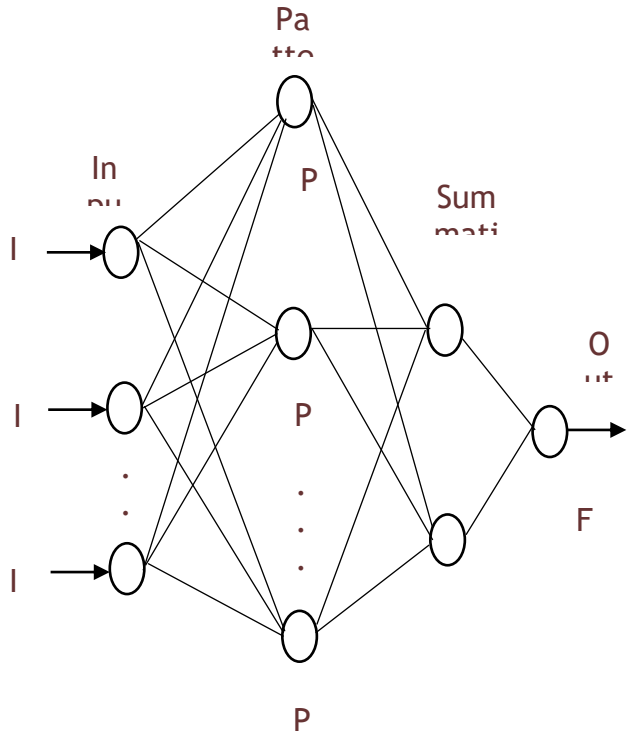

Fig: 4. GRNN Architecture

GRNN comprises four layers named input layer, pattern layer, summation layer and output layer. Input units numbers depend on total observation parameters i.e. input vector 'I' (feature matrix Fi). Input layer connected to pattern layer has neurons to ensure training patterns and output to summation layer for performing normalization of resultant output set. Each pattern layer is connected to summation neurons which calculate weight vector using these equations [39].

THE IIOAB JOURNAL ISSN: 0976-3104

$$W_i = e^{\left[\frac{\|I - I_t\|^2}{2h^2}\right]}$$

$$F(I) = \frac{\sum_{i=1}^{n} T_i W_i}{\sum_{i=1}^{n} W_i} \quad (27)$$

Where output F (I) is weighted average of target values Ti of training cases Ii close to input case I.
GRNN is one-pass learning algorithm based having a highly parallel structure. GRNN is a powerful memory based network that estimates continuous variables and converges to an underlying regression surface. GRNN's strength is its ability to deal with sparse data effectively. GRNNs feature fast training times, model non-linear functions and are known to do well in noisy environments if provided enough data. The GRNN algorithm can ensure smooth transition from one observed value to another, even with sparse data in a multidimensional measurement space. GRNN applications produce continuous valued outputs.

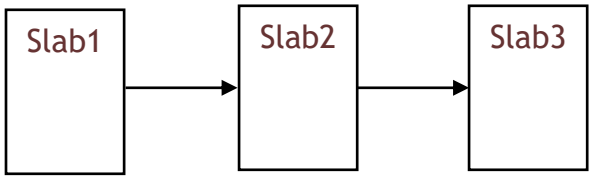

Fig: 5.GRNN Network

For GRNN networks, the number of hidden layer (Slab2) neurons is the number of patterns in a training set as a neuron represents each pattern. Input layer neurons (Slab1) are the inputs, and output layer neurons (Slab3) correspond to number of outputs.

GRNN's advantage is the speed with which a network is trained. There are no training parameters like learning rate and backpropagation network momentum, but a smoothing factor is applied after training a network. Smoothing factor is an GRNN adjustable parameter, so overtraining is not likely in GRNN. Smoothing allows GRNN interpolate between patterns/spectra in training sets. Smoothing determines how tightly a network matches predictions to training patterns data. Smoothing factor for GRNN networks must be greater than 0 ranging from 0.01 to 1 with good results.
The proposed GRNN parameters are tabulated in **[Table -2]**.

Table: 2.PARAMETERS FOR THE PROPOSED GRNN

Number of input	17
Number of output	1
Number of hidden layer	2
Number of neurons in hidden layer	6
Number of cluster centers	85
Competitive rule	conscience full metric – Euclidean
Activation function	tanh
Momentum	0.5
Learning rate	0.1

RESULTS AND DISCUSSION

The attributes of the COCOMO dataset is used as it is and feature transformation of the attributes using KPCA). In the first set of experiment without data transformation using PCA, the MMRE and MdMRE is evaluated using various techniques such as M5, Linear regression, SMOPolykernel and RBF kernel and the proposed Generalized Regression Neural Network. In the second set of experiments, the same is evaluated with feature transformation of the attributes using KPCA.

Table: 3. PARAMETERS FOR THE PROPOSED GRNN

M5 Actual	M5 - Kernel PCA predicted	MMRE
387	251.58	0.35
18	-61.69	4.43
15	-103.45	7.9
237	259.65	0.1
958	628.78	0.34

THE IIOAB JOURNAL
ISSN: 0976-3104

14	49.01	2.5
57	107.34	0.88
33	31.3	0.05
98	463.89	3.73
605	1021.06	0.69
423	655.96	0.55
702	2917.59	3.16
724	1126.56	0.56
70	255.04	2.64
20	52.31	1.62
523	242.14	0.54
7.3	8.69	0.19
1272	589.25	0.54
88	171.16	0.95
55	135.16	1.46
8	53.42	5.68
45	376.78	7.37
1075	874.58	0.19
243	2134.84	7.79
38	82.67	1.18
106	239.03	1.26
321	1213.44	2.78
1063	1093.85	0.03
201	133.11	0.34
126	576.56	3.58
240	1259.74	4.25
6600	11253.02	0.71
87	502.26	4.77
61	324.14	4.31
122	316.15	1.59
8	-227.09	29.39
40	643.72	15.09
1600	1664.47	0.04
11400	2364.59	0.79
6400	2361.72	0.63
79	-206.75	3.62
2455	1586.89	0.35
156	63.17	0.6
41	424.56	9.36
8	-77.01	10.63
130	1267.25	8.75
6	-361.3	61.22
82	468.67	4.72
12	-218.62	19.22
73	952.22	12.04
36	65.26	0.81
176	445.23	1.53
83	250.66	2.02
218	1048.4	3.81
453	320.4	0.29
539	523.01	0.03
5.9	-338.05	58.3
9	220.04	23.45
43	13.26	0.69
230	142.14	0.38
47	173.46	2.69
2040	270.99	0.87
50	-4014.22	81.28

THE IIOAB JOURNAL
ISSN: 0976-3104

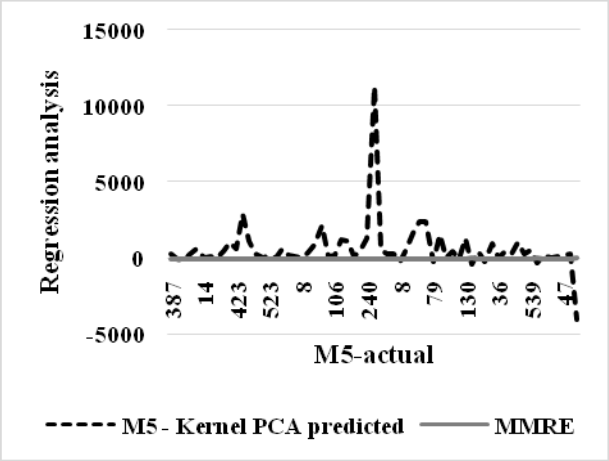

Fig: 6. M5-actual

From the **[Table -3]** and **[Figure -6]**, it can be observed that the MMRE method averagely decreased by 195.62% when compared with M5 - Kernel PCA predicted method with M5-actual.

Table: 4.LINEAR REGRESSION-ACTUAL

Linear regression - actual	linear regression Kernel PCA-Predicted	MMRE
387	469.65	0.21
18	182.52	9.14
15	205.73	12.72
237	255.78	0.08
958	327.05	0.66
14	175.42	11.53
57	187.25	2.29
33	270.92	7.21
98	251.77	1.57
605	463.41	0.23
423	380.06	0.1
702	3096.17	3.41
724	1050.5	0.45
70	329.5	3.71
20	176.36	7.82
523	427.19	0.18
7.3	159.62	20.87
1272	3698.57	1.91
88	219.79	1.5
55	261.59	3.76
8	169.59	20.2
45	466.34	9.36
1075	306.92	0.71
243	1099.36	3.52
38	211.47	4.56
106	395.59	2.73
321	349.34	0.09
1063	621.22	0.42
201	362.85	0.81
126	734.3	4.83
240	598.59	1.49
6600	11830.69	0.79
87	299.23	2.44
61	113.51	0.86
122	149.44	0.22

THE IIOAB JOURNAL
ISSN: 0976-3104

8	127.66	14.96
40	126.16	2.15
1600	1775.59	0.11
11400	1963.6	0.83
6400	2037.19	0.68
79	284.55	2.6
2455	1724.29	0.3
156	704.66	3.52
41	173.58	3.23
8	163.35	19.42
130	334.35	1.57
6	196.49	31.75
82	244.74	1.98
12	259.87	20.66
73	177.68	1.43
36	320.99	7.92
176	323.42	0.84
83	352.96	3.25
218	379.3	0.74
453	799.12	0.76
539	696.17	0.29
5.9	170.66	27.93
9	165.67	17.41
43	165.9	2.86
230	298.88	0.3
47	563.01	10.98
2040	940	0.54
50	338.73	5.77

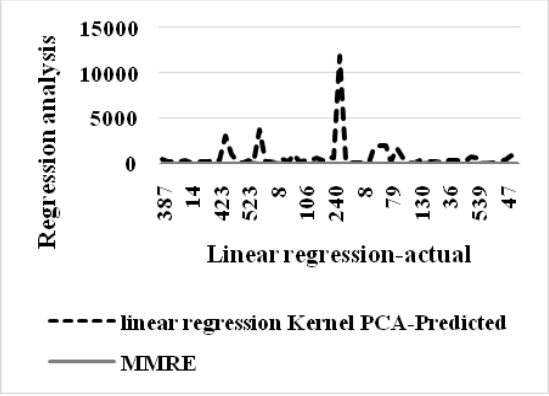

Fig:7. Linear regression-actual

From the **[Table -4]** and **[Figure -7]**, it can be observed that the MMRE method averagely reduced by 197.18% when compared with linear regression Kernel PCA-Predicted method with Linear regression-actual.

Table: 5: M5-ACTUAL

M5 Actual	M5 predicted	MMRE
387	270.89	0.3
18	-134.35	8.46
15	-27.83	2.86
237	32.02	0.86
958	528.87	0.45
14	-77.98	6.57
57	80.88	0.42
33	-413.87	13.54
98	892.3	8.11
605	2070.09	2.42
423	479.76	0.13
702	3028.23	3.31
724	1408.59	0.95
70	-208.06	3.97

THE IIOAB JOURNAL
ISSN: 0976-3104

20	-82.82	5.14
523	406.23	0.22
7.3	-26.33	4.61
1272	4881.49	2.84
88	224.52	1.55
55	119.59	1.17
8	-88.36	12.04
45	538.44	10.97
1075	1114.52	0.04
243	1954.62	7.04
38	-181.48	5.78
106	474.68	3.48
321	-60.63	1.19
1063	1749	0.65
201	237.37	0.18
126	276.43	1.19
240	130.22	0.46
6600	12754.64	0.93
87	-0.89	1.01
61	147.99	1.43
122	176.05	0.44
8	-58.91	8.36
40	113.44	1.84
1600	1388.09	0.13
11400	6500	0.43
6400	2907.02	0.55
79	-530.49	7.72
2455	2046.46	0.17
156	-110.05	1.71
41	214.05	4.22
8	-359.67	45.96
130	1130.62	7.7
6	-118.21	20.7
82	-100.9	2.23
12	171.41	13.28
73	-123.38	2.69
36	-30.83	1.86
176	788.17	3.48
83	355.92	3.29
218	-3.75	1.02
453	620.92	0.37
539	588.57	0.09
5.9	-65.98	12.18
9	250.13	26.79
43	27.17	0.37
230	897.51	2.9
47	-326.15	7.94
2040	901.71	0.56
50	-12.54	1.25

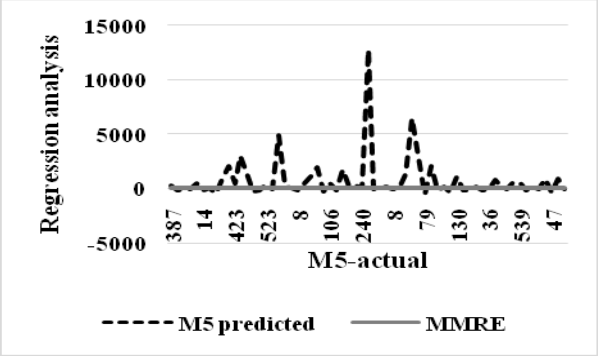

Fig: 8. M5-actual

THE IIOAB JOURNAL
ISSN: 0976-3104

From the **[Table -5]** and **[Figure -8]**, it can be observed that the MMRE method averagely decreased by 197.64% when compared with M5 Predicted method with M5-actual.

Table: 6. LINEAR REGRESSION-ACTUAL

Linear regression - actual	linear regression - Predicted	MMRE
387	478.42	0.24
18	182.61	9.14
15	208.85	12.92
237	258.51	0.09
958	329.45	0.66
14	175.51	11.54
57	185.44	2.25
33	273.74	7.3
98	251.19	1.56
605	469.18	0.22
423	378.98	0.1
702	3085.02	3.39
724	1040.45	0.44
70	326.36	3.66
20	179.67	7.98
523	435.07	0.17
7.3	162.49	21.26
1272	3673.21	1.89
88	217.67	1.47
55	260.24	3.73
8	169.03	20.13
45	473.4	9.52
1075	305.79	0.72
243	1090.32	3.49
38	213.79	4.63
106	398.51	2.76
321	355.72	0.11
1063	624.37	0.41
201	368.3	0.83
126	737.04	4.85
240	603.88	1.52
6600	11768.41	0.78
87	302.75	2.48
61	115.12	0.89
122	150.58	0.23
8	128.49	15.06
40	127.91	2.2
1600	1795.77	0.12
11400	1952.73	0.83
6400	2055.82	0.68
79	282	2.57
2455	1751.55	0.29
156	709.27	3.55
41	173.25	3.23
8	164.83	19.6
130	333.87	1.57
6	194.74	31.46
82	249.12	2.04
12	263.25	20.94
73	178.5	1.45
36	319.74	7.88
176	326.66	0.86
83	359.24	3.33
218	383.32	0.76
453	794.14	0.75
539	702.06	0.3
5.9	170.69	27.93
9	165.08	17.34

THE IIOAB JOURNAL
ISSN: 0976-3104

43	167.9	2.9
230	301.94	0.31
47	562.96	10.98
2040	936.85	0.54
50	337.21	5.74

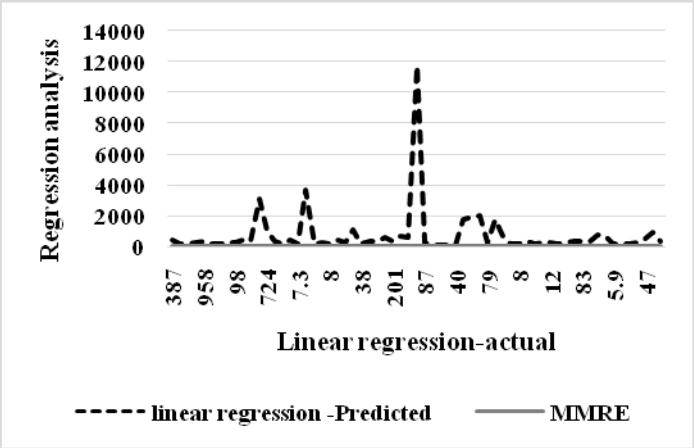

Fig:9. Linear regression-actual

From the **[Table -6]** and **[Figure -9]**, it can be observed that the MMRE method averagely reduced by 197.17% when compared with linear regression Predicted method with Linear regression-actual.

CONCLUSION

This paper proposes a Generalized Regression NN to use improved software effort estimation for COCOMO dataset. This paper uses Mean Magnitude Relative Error (MMRE) and Median Magnitude Relative Error (MdMRE) as evaluation criteria. The new method performance was compared to that of three regression algorithms including M5, linear regression and modified SVM to avoid quadratic problem. Two kernels were used for SVM with the first being a polykernel and second using RBF. It is found that the new method outperformed classical regression algorithms in experiments.

CONFLICT OF INTEREST

The authors declare no conflict of interests.

ACKNOWLEDGEMENT

None

FINANCIAL DISCLOSURE

The authors report no financial interests or potential conflicts of interest.

REFERENCES

[1] Paramshetti P, Phalke DA. [2015] Software Defect Prediction for Quality Improvement Using Hybrid Approach. *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, 4(6)
[2] Shubitsha P, Rajan JK.[2014] Artificial Neural Network Models For Software Effort Estimation.
[3] Santani D, Bundele M, Rijwani, P. Artificial Neural Networks for Software Effort Estimation: A Review.
[4] Prasad Reddy PVGD, Sudha KR, Rama Sree P. [2011] Prediction of Software Development Effort Using RBNN And GRNN. *International Journal of Computer Science Engineering and Technology*, 1(4):185-190.
[5] Reddy PVGD, Sudha KR, Sree PR, Ramesh SNSVSC. [2010] Software effort estimation using radial basis and generalized regression neural networks. arXiv preprint arXiv:1005.4021.
[6] Kaushik A, Soni AK, Soni R. [1969] A Simple Neural Network Approach to Software Cost Estimation. *Global Journal of Computer Science and Technology*, 13(1).
[7] Samuel RT, Cao Y. [2014] Fault detection in a multivariate process based on kernel PCA and kernel density estimation. In Automation and Computing (ICAC), 2014 20th International Conference on (pp. 146-151). *IEEE*.

THE IIOAB JOURNAL ISSN: 0976-3104

[8] Patil LV, Waghmode RM, Joshi SD, Khanna V. [2014] Generic model of software cost estimation: A hybrid approach. In Advance Computing Conference (IACC), 2014 IEEE International (pp. 1379-1384). *IEEE*.

[9] Mukhopadhyay S, Gundappa M, Srinivasan R, Narasimhan S. [2013] A Novel attempt to reduce engineering effort in modeling non-linear chemical systems for Operator Training Simulators. *In American Control Conference (ACC)*, 2013 (pp. 1902-1907). *IEEE*.

[10] Grbic R, Sliskovic D, Kadlec, P. [2013]Adaptive soft sensor for online prediction and process monitoring based on a mixture of Gaussian process models. *Computers & chemical engineering*, 58:84-97.

[11] Papaioannou A, Zafeiriou S. [2014]Principal component analysis with complex kernel: The widely linear model. Neural Networks and Learning Systems, *IEEE Transactions on*, 25(9): 1719-1726.

[12] Musa AB. [2014] A comparison of ℓ1-regularizion, PCA, KPCA and ICA for dimensionality reduction in logistic regression. *International Journal of Machine Learning and Cybernetics*, 5(6):861-873.

[13] Tosun A, Turhan B, Bener AB. [2009] Feature weighting heuristics for analogy-based effort estimation models. *Expert Systems with Applications*, 36(7): 10325-10333.

[14] Elish MO. [2009]Improved estimation of software project effort using multiple additive regression trees. *Expert Systems with Applications*, 36(7):10774-10778.

[15] Attarzadeh I, Ow SH. [2010] Proposing a new software cost estimation model based on artificial neural networks. In Computer Engineering and Technology (ICCET), 2010 2nd International Conference on, *IEEE* 3: V3-487).

[16] Kalichanin-Balich I, Lopez-Martin C. [2010] Applying a feedforward neural network for predicting software development effort of short-scale projects. In Software Engineering Research, Management and Applications (SERA), 2010 Eighth ACIS International Conference on ,*IEEE*, (pp.269-275).

[17] Dave VS, Dutta K. [2011] Neural network based software effort estimation & evaluation criterion MMRE. In Computer and Communication Technology (ICCCT), 2011 2nd International Conference on (pp. 347-351). IEEE.

[18] Srichandan S. [2012]A new approach of Software Effort Estimation Using Radial Basis Function Neural Networks. *International Journal on Advanced Computer Theory and Engineering (IJACTE)*, 1(1):113-120.

[19] Park H, Baek S. [2008] An empirical validation of a neural network model for software effort estimation. *Expert Systems with Applications*, 35(3):929-937.

[20] deBarcelosTronto IF, da Silva, JDS, Sant'Anna, N. [2008]An investigation of artificial neural networks based prediction systems in software project management. *Journal of Systems and Software*, 81(3):356-367.

[21] Reddy CS, Raju KVSVN. [2009] A concise neural network model for estimating software effort. *International Journal of Recent Trends in Engineering*, 1(1):188-193.

[22] Boetticher G, Menzies T, Ostrand T. [2007] PROMISE Repository of empirical software engineering data http://promisedata.org/ repository, West Virginia University, Department of Computer Science.

[23] Singh BK, Misra AK. [2012] An Alternate Soft Computing Approach for Efforts Estimation by Enhancing Constructive Cost Model in Evaluation Method. iji, 10(1).

[24] Shepperd MJ, Schofield C. [1997] Estimating Software Project Effort Using Analogies, *IEEE Transaction on Software Engineering* 23:736-743.

[25] Blanchard G, Bousquet O, Zwald L. [2007]Statistical properties of kernel principal component analysis. *Machine Learning*, 66(2-3): 259-294.

[26] Desobry F, Févotte C. [2006] Kernel PCA based estimation of the mixing matrix in linear instantaneous mixtures of sparse sources. In Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on, *IEEE*, 5: V-V)..

[27] Ma X, Zabaras N. [2011] Kernel principal component analysis for stochastic input model generation. *Journal of Computational Physics*, 230(19):7311-7331.

[28] Waghmode S, Kolhe K. [2014] A Novel Way of Cost Estimation in Software Project Development Based on Clustering Techniques. *International Journal of Innovative Research in Computer and Communication Engineering*, 2(4)

[29] Bhatia S, Attri VK. [2015]MACHINE LEARNING TECHNIQUES IN SOFTWARE EFFORT ESTIMATION USING COCOMO DATASET. *International Joural of Research and Development organization (IJRD)*, 2 (6)

[30] V Vapnik, S Golowich, A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems* 9, pages 281–287, Cambridge, MA, 1997. MIT Press.

[31] Idri A, Zakrani A, Zahi, A. [2010] Design of radial basis function neural networks for software effort estimation. *International Journal of Computer Science*, 7(4).

[32] Bautista AM, Castellanos A, San Feliu T. [1993] SOFTWARE EFFORT ESTIMATION USING RADIAL BASIS FUNCTION NEURAL NETWORKS. INFORMATION THEORIES & APPLICATIONS, 319.

[33] C.-C. Chang and C.-J. Lin, LIBSVM: a library for support vector machines, 2001.

[34] Platt JC.[1999] Fast training of support vector machines using sequential minimal optimization". Advances in kernel methods: Support vector machines, B. Schokopf et al. (ed.), MIT Press,

[35] Breiman L, Friedman J, Olshen R, Stone CJ.[1984]Classification and Regression Trees, Chapman and Hall, New York, 1984.

[36] Waghmode RM, Patil LV, Joshi SD. [2013] A Collective Study of PCA and Neural Network based on COCOMO for Software Cost Estimation. *International Journal of Computer Applications*, 74(16):25-30.

[37] Ajay Prakash, BV Ashoka, DV Manjunath, Aradhya VN. [2015]. Estimating Software Development Effort using Neural Network Models. *International Journal of Advanced Research in Computer Science and Software Engineering*, 5(7)

[38] Thwin MMT, Quah TS. [2005] Application of neural networks for software quality prediction using object-oriented metrics. *Journal of systems and software*, 76(2):147-156.