Efficint Software Cost Estimation Using Partitionig Techniques

K. Subba Rao[*1] L.S.S Reddy[2] Nagaraju Devarakonda[3]
1. IT Department, LBRCE , INDIA.
2. CSE Department, KL University, INDIA.
3. CSE Department, Acharya Nagarjuna University, INDIA.

ABSTRACT

Software Cost estimation and effort allocation are the key challenges for successful project planning and management in software development Cost estimation process involves distinctive steps, tools, algorithms and assumptions.. Therefore, the research community has been working on various models and techniques to accurately predict the cost of projects. Recently, researchers have started debating whether the prediction performance depends on the structure of data rather than the models used. Industry competitiveness depends on the cost, performance, and timely delivery of the product. Thus, an accurate, rapid, and robust product cost estimation model for the entire product life cycle is essential. This research applies three machine learning methods decision tree, bootstrap tree, bootstrap forest – to solve product life cycle cost estimation problems. The performance of a number of cost estimation models, statistical regression analyses, three machine learning models, are compared in terms of their performance. The estimation results and performance reveal that these models provide more accurate performance than conventional models. Finally, a more accurate, available, and generalisable cost estimation model is presented. This research can serve as a useful reference for product cost planning and control in industries.

keywords: Software cost estimation, decision tree, bootstrap, JMP Pro, MV-dataset.

1 INTRODUCTION

Software cost and effort estimate is one of the most important activities in software project management. It is the accuracy of cost and effort calculation that enable quality growth of software at a later stage. With an effective estimate of software cost and effort, software developers can efficiently decide what recourses are to be used frequently and how efficiently these resources can be utilized [118]. For efficient software, accurate software development parameters are required, these include effort estimation, development time estimation, cost estimation, team size estimation, risk analysis, etc[118]. Since the effort and cost estimation is done at an early stage of software development; hence a good model is required to calculate these parameters precisely. In past few decades several researchers have worked in the field of software effort estimation, and many conventional models were designed to estimate software,

size and effort [118]. These models require inputs which are difficult to obtain at early stages of software development. Moreover these models take Values of software development factors based on experience and approximation, with zero reasoning Capability. Due to few such limitations of conventional algorithmic models, non-algorithmic models based on Soft Computing came into picture, which include Neural Network, Fuzzy logic and Genetic algorithms [118]. The non-algorithm based algorithm work with real life situations and a vast flexibility for software development factors was provided.

Software effort estimation is the process of predicting the most realistic use of effort required to develop or maintain software. Effort estimates are used to calculate effort in person-months (PM) for the Software Development work elements of the Work Breakdown Structure (WBS).Accurate Effort Estimation is important because:

- It can help to classify and compute development projects with respect to an overall business plan.
- It can be used to predispose what resources to commit.
- It can be used to assess the impingement of changes and support re-planning.
- Projects can be easier to manage and control when resources are better matched to real needs.
- Customers expect actual development costs to be in line with estimated costs.

Accurate software estimation such as size estimation, effort estimation, cost estimation, quality estimation and risk analysis is a major issue in software project management [1]. If the estimation is not properly done, it may result in the failure of software project. Accurate software estimation can provide powerful assistance for software management decisions. The principal challenges are 1) the relationships between software output metrics and contributing factors exhibit strong complex nonlinear characteristics; 2) measurements of software metrics are often imprecise and uncertain; 3) difficulty in utilizing both expert knowledge and numerical project data in one model. In this research proposal, a soft computing framework is presented to tackle this challenging problem [4]. Soft computing is a monopoly of methodologies that works synergistically and provides, in one form or another, formative information processing capability for handling real-life opaque situations [3]. Its aim is to exploit the tolerance for imprecision, uncertainty, approximate reasoning, and partial truth in order to achieve tractability, robustness, low-cost solutions, and close resemblance to human-like decision making [2]. The guiding principle is to devise methods of computation that lead to an acceptable solution at low cost by seeking for an approximate solution to an imprecisely/precisely formulated problem.

Several classes of software cost estimation models and techniques: parametric models, expertise-based techniques, learning-oriented techniques, dynamics based models, regression-based models, and composite-Bayesian techniques for integrating expertise-based and regression-based models [5] [6]. The following figure 1.1 shows different cost estimation techniques.

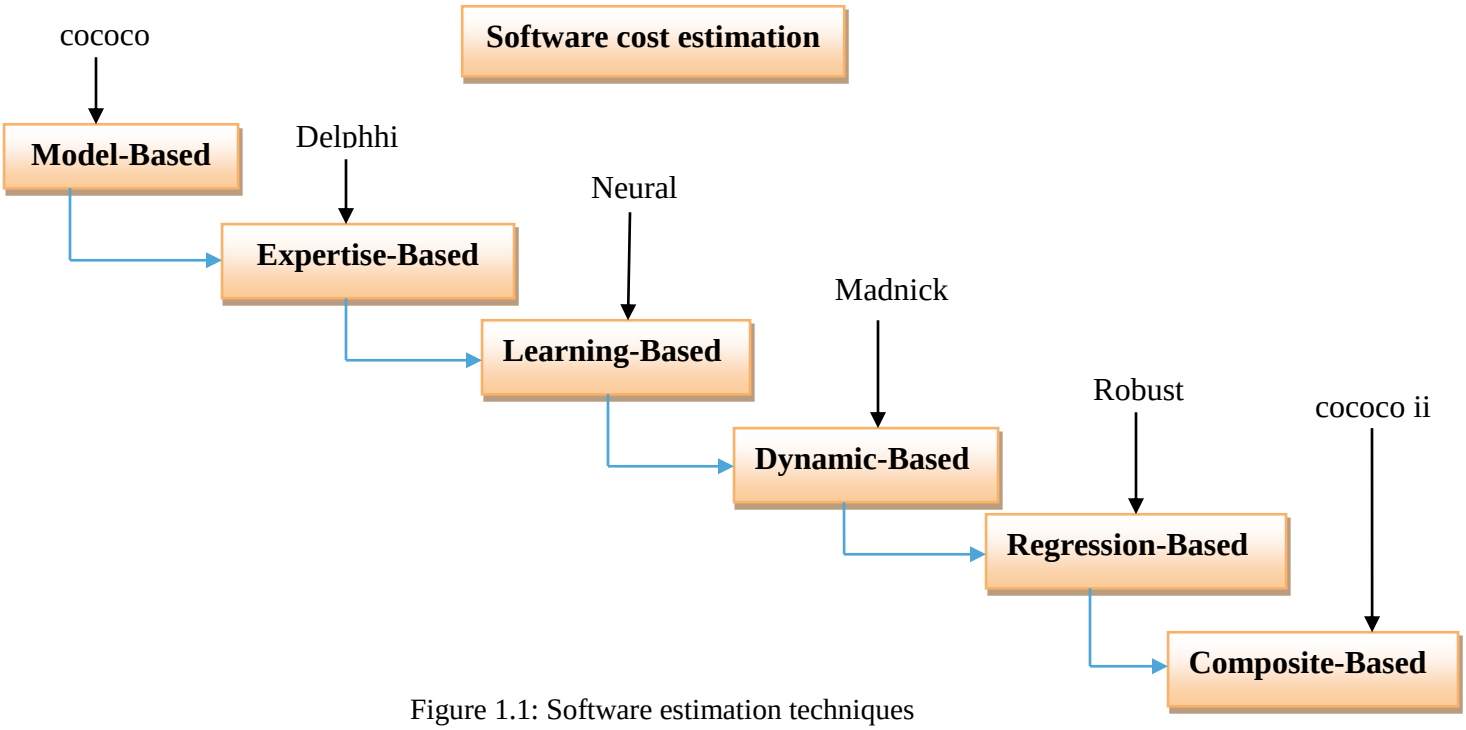


Figure 1.1: Software estimation techniques

The paper is organized in following sections: section 1 describes introduction, sections 2 and 3 describes existing work and software cost estimation tools. Section 4 discusses the software cost estimation process. Proposed work and Experimental results is described in section. Section 6 ends the paper with a conclusion.

2 LITERATURE REVIEW

Decision trees (DT) are hierarchical data structures that are based on a divide-and-conquer strategy. They can be used for both classification and regression and require no assumptions concerning the data. In the case of classification, they are called classification trees. The nodes of a classification tree correspond to the attributes that best split data into disjoint groups, while the leaves correspond to the average effort of that split. The quality of the split is determined by an impurity measure. The tree is constructed by partitioning the data recursively until no further partitioning is possible while choosing the split that minimizes the impurity at every occasion. Concerning the estimation of software effort, the effort of the new project can be determined by traversing the tree from top to bottom along the appropriate paths.

As software development has become an essential investment for many organizations [2], software estimation is gaining an ever-increasing importance in effective software project management. In practice, software estimation includes cost estimation, quality estimation, risk analysis, etc. Accurate software estimation can provide powerful assistance for software management decisions. The principal challenges are 1) the relationships between software output

metrics and contributing factors exhibit strong complex nonlinear characteristics; 2) measurements of software metrics are often imprecise and uncertain; 3) difficulty in utilizing both expert knowledge and numerical project data in on model[8]. To solve software estimation problems, soft computing framework is based on the "divide and conquer" approach.

Software engineering cost (and schedule) models and estimation techniques are used for a number of purposes. These include:

- **Budgeting:** the primary but not the only important use. Accuracy of the overall estimate is the most desired capability.
- **Tradeoff and risk analysis:** an important additional capability is to illuminate the cost and schedule sensitivities of software project decisions (scoping, staffing, tools, reuse, etc.).
- **Project planning and control:** an important additional capability is to provide cost and schedule breakdowns by component, stage and activity.
- **Software improvement investment analysis:** an important additional capability is to estimate the costs as well as the benefits of such strategies as tools, reuse, and process maturity [6].

In the early days of software, computer programs were typically less than 1000 machine instructions in size (or less than 30 function points), required only one programmer to write, and seldom took more than a month to complete[10]. The entire development costs were often less than $5000. Although cost estimating was difficult, the economic consequences of cost-estimating errors were not very serious [7]. Today some large software systems exceed 25 million source code statements (or more than 300,000 function points), may require technical staffs of 1000 personnel or more, and may take more than five calendar years to complete. The development costs for such large software systems can exceed $500 million; therefore, errors in cost. The following 2.1 shows classical view of the algorithmic cost estimation process.

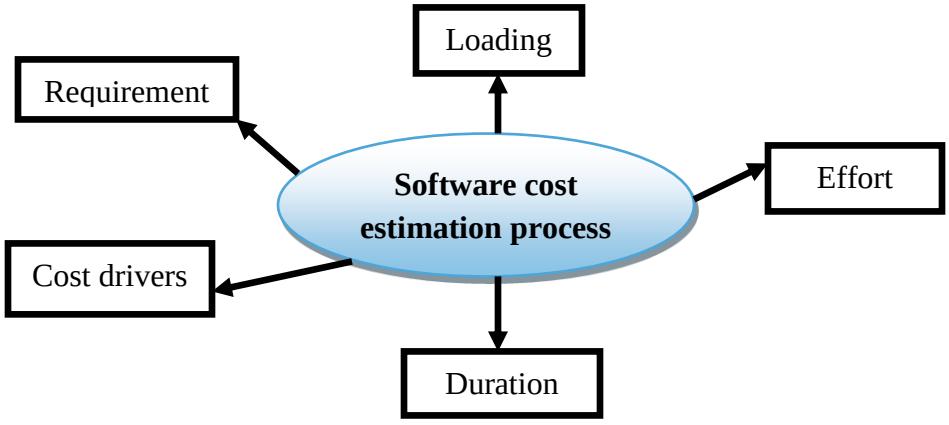


Figure2.1: Classical view of the algorithmic cost estimation process

3 SOFTWARE COST ESTIMATION TOOLS

There are 21 commercial-off-the-shelf (COTS) cost estimation tools. This set reflects a wide range of methodologies, levels of aptness or mellowness, features, and cost. Most of the tools are parametric models [9]. Some tools address hardware as well as software, but most do not. A few tools offer a stochastic model.

Developers of parametric models derive Cost Estimating Relationships (CERs) by regression analysis on historical data on project attributes (cost drivers) and cost. Cost estimation models use these relationships as scale factors in an exponential equation to calculate the effort and schedule required for the software development effort[11][12][13]. The following 3.1 shows software cost estimation tools.

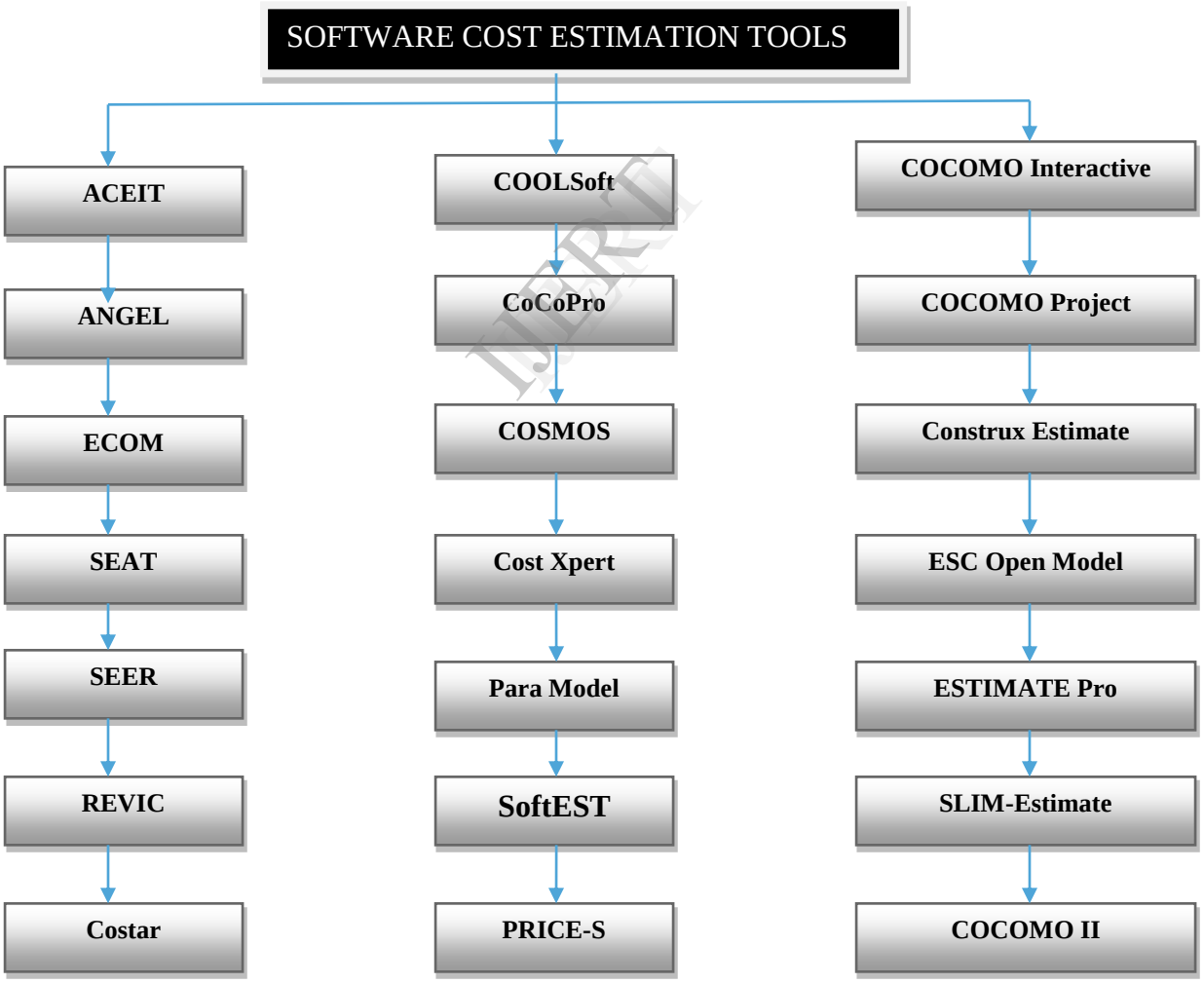


Figure 3.1: shows software cost estimation tools.

ACEIT

Automated Cost Estimating Integrated Tools (ACEIT) from Tecolote Research is an automated architecture and framework for cost estimating and many other analysis tasks[18]. ACEIT is a government-developed tool that has been used for over a decade to standardize the Life Cycle Cost estimating process in the government environment. ACEIT is installed at over 400 sites within government and industry[16][19].

ANGEL

Empirical Software Engineering Research Group (ESERG) at Bournemouth University has a research project focused on estimating software development costs using case-based reasoning (analogy). A brief bibliography and the downloadable ANGEL tool are provided. The tool is not well supported[25].

COCOMO Interactive

COCOMO Interactive (Texas A&M University) is an on-line, interactive software package that assists in budgetary and schedule estimation of a software project [110]. This was a class project that is not being supported.

COCOMO Project

The COCOMO Project is a program of research conducted by Barry Boehm. COCOMO II is an update of COCOMO 1981, which addresses 1990s and 2000s software development practices [114][105]. A public version of COCOMO II, including a COTS version (COCOTS), is available. USC-CSE, UC Irvine, and 29 affiliate organizations are developing it[115].

CoCoPro

CoCoPro from ICONIX Software Engineering is one an integrated suite of 10 analysis and design tools supporting the major phases of the system development lifecycle[15][14], including analysis, design, coding and the management of complex systems. It based upon Barry Boehm's constructive cost modeling methods. CoCoPro supports the Intermediate level of the COCOMO methodology[101].

Construx Estimate

Construx Software Builders provide Construx Estimate, a free estimation tool that includes the functionality of both COCOMO II and SLIM (the QSM product below)[17]. Construx Estimate

uses Monte Carlo simulations to model complex interactions in the face of uncertain estimating assumptions, making the company one of the few who offer a stochastic method[24].

COOLSoft

COOLSoft from Wright Williams & Kelly utilizes a hybrid approach of intermediate and detailed versions of COCOMO[20]. This allows for the reuse of existing code, development of new code, and integration of both hardware and third party code[107]. The model comes in a standard Microsoft Excel spreadsheet format.

COSMOS

COSMOS (Design Studio Group/Oak Ridge National Laboratory) is unique in that it combines the well-known Function Point and COCOMO models as well as a Rayleigh model of manpower buildup proposed by Larry Putnam[97][95]. These models can be used independently or work together[57].

Costar

Costar from Softstar Systems is a cost estimation tool that supports COCOMO II, COCOMO 81, REVIC, Ada COCOMO, and Incremental COCOMO[111]. Costar is an interactive tool that permits managers to make trade-offs and what-if analyses to arrive at the optimal project plan. Costar 6.0 has 13 built-in models[83].

Cost Xpert

Cost Xpert (The Cost Xpert Group) is a software cost estimating tool that integrates multiple estimating models into one tool to provide accurate and comprehensive estimates[109][110]. It claims to be the only tool offering support for sophisticated modeling techniques such as system dynamic modeling, knowledge based modeling, both stochastic and deterministic modeling, and a variety of cost models including the latest release of COCOMO II[111][89].

ECOM

ECOM (ESA's COst Model) is not a modeling tool. It is a software tool for collection, retrieval and processing of cost data from past ESA programs and projects[79]. ECOM is linked to a hybrid cost estimation tool combining items from the ECOM database, Cost Estimating Relationships (CERs) and estimates using the PRICE-H parametric tool. The combining tool is ACEIT (from Tecolote Research, see above), which has been customized to a European and ESA like environment[80].

ESC Open Model

The ESC Open Model (Tecolote Research) is a suite of Cost Estimating Relationships and metrics used to estimate the effort required for commercial off-the-shelf (COTS) and non-developmental item (NDI)-intensive software efforts[83][84].

ESTIMATE Pro

ESTIMATE Professional (Software Productivity Center) makes use of three mature estimation approaches: Putnam methodology, COCOMO II, and Monte Carlo simulation [88]. Putnam methodology is based on the insight that efficiently run software projects follow well-defined patterns that can be modeled with a set of exponential equations [111]. COCOMO II is a continuation of work begun by Barry Boehm [66] [28]. Monte Carlo simulation models complex interactions in the face of uncertain estimating assumptions.

ParaModel

ParaModel from Mainstay is not database driven, it is a parametric estimating tool. It integrates hardware and software components to create a complete program estimate[22][21]. ParaModel combines hardware and software estimates, supports any level of the WBS, and provides presentations meaningful to management[23].

PRICE-S

PRICE-S (Parametric Review of Information for Cost & Evaluation—Software) and PRICE-H from PRICE Systems at Lockheed Martin[66] are well-known cost estimating models for software and hardware[86]. PRICE-H, useful for estimating the cost of hardware development, has limited usefulness in estimating the cost of hardware purchased for use in data systems development. PRICE-S claims to deliver estimates to within 5% of actual cost after calibration by supplying industry-average values for actual input data that has not yet been specified[75].

REVIC

REvised Intermediate COCOMO (REVIC) is available for downloading from the US Air Force Cost Analysis Agency (AFCAA). (Note: REVIC is not Y2K compliant and was replaced by SoftEST.)[26]

SEAT

Software Estimation and Analysis Tool (SEAT) was a student project of Design Studio Group, East Tennessee State University, and appears to be unsupported[27]. Available for download over the Internet, it is a Windows-based tool integrating Function Point Analysis (FPA) with the COCOMO model[30].

SEER

Galorath Incorporated (also known as GA SEER™ Technologies) provides a comprehensive set of decision-support and production optimization tools to help manage product design and manufacturing operations[50]. SEER-SEM is the parametric cost model and SEER-SSM estimates the expected size of software. They derive cost, schedule, labor and materials estimates by assessing the interaction and impact of product, organizational and even operational variables[55][53].

SLIM-Estimate

Quantitative Software Management (QSM) offers their clients Software Life-cycle Management (SLIM) tools for software cost estimating (SLIM-Estimate), reliability modeling, schedule estimating, planning, tracking, and benchmarking[29].

Soft EST

SPAWAR Systems Engineering Process Office (SEPO) makes available the Software Estimation Model (SoftEST) developed by MCR Federal Inc[70]. for the Air Force Cost Analysis Agency. SoftEST is the follow-on to the REVIC software estimation model. It is capable of varying development environment at CSCI level and supports complex projects developed with rapid prototyping, incremental, evolutionary, and spiral development methods[65].

Building a model for estimating the cost of distributed scientific data systems (and centers) is highly dependent on the software development environment, including methods and standards. The selected tool must not only have this capability, but must also fit the cost estimation process[35].

4 SOFTWARE COST ESTIMATION PROCESS

Throughout the software life cycle, there are many decision situations involving limited resources in which software engineering techniques provide useful assistance. To provide a feel for the nature of these economic decision issues, an example is given below for each of the major phases in the software life cycle[32].

- **Feasibility Phase:** How much should one invest in information system analyses (user questionnaires and interviews, current-system analysis, workload characterizations, simulations, scenarios, prototypes) in order to obtain convergence on an appropriate definition and concept of operation for the system to be implemented?
- **Plans and Requirements Phase:** How rigorously should requirements be specified? How much should be invested in requirements validation activities (automated completeness, consistency, traceability checks, analytic models, simulations, prototypes) before proceeding to design and develop a software system?

- **Product Design Phase:** Should developers organize software to make it possible to use a complex piece of existing software which generally but not completely meets requirements?
- **Programming Phase:** Given a choice between three data storage and retrieval schemes which are primarily execution time-efficient, storage-efficient, and easy-to-modify, respectively; which of these should be implemented?
- **Integration and Test Phase:** How much testing and formal verification should be performed on a product before releasing it to users?
- Maintenance Phase: Given an extensive list of suggested product improvements, which ones should be implemented first?
- **Phaseout:** Given an aging, hard-to-modify software product, should it be replaced with a new product, should it be restructured, or should it be left alone?

Software cost engineering estimation typically involves a top-down planning approach in which the cost estimate is used to derive a project plan. Typical steps in a planning process include:

- The project manager develops a characterization of the overall functionality, size, process, environment, people, and quality required for the project.
- A macro-level estimate of the total effort and schedule is developed using a software cost estimation model[35].
- The project manager partitions the effort estimate into a top-level work breakdown structure. In addition, the schedule is partitioned into major milestone dates and a staffing profile is configured[46].

The actual cost estimation process involves seven steps:

- Establish cost-estimating objectives.
- Generate a project plan for required data and resources.
- Pin down software requirements.
- Work out as much detail about the software system as feasible.
- Use several independent cost estimation techniques to capitalize on their combined strengths
- Compare different estimates and iterate the estimation process; and
- Once the project has started, monitor its actual cost and progress, and feedback results to project management.

Regardless of which estimation model is selected, consumers of the model must pay attention to the following to get the best results:

- Since some models generate effort estimates for the full software life-cycle and others do not include effort for the requirements stage, coverage of the estimate is essential[36].
- Model calibration and assumptions should be decided beforehand.

- Sensitivity analysis of the estimates to different model parameters should be calculated.

The microeconomics field provides a number of techniques for dealing with software life-cycle decision issues such as the ones mentioned early in this section[66]. Standard optimization techniques can be used when one can find a single quantity such as rupees or dollars to serve as a "universal solvent" into which all decision variables can be converted. Or, if nonmonetary objectives can be expressed as constraints (system availability must be 98%, throughput must be 150 transactions per second),[70] then standard constrained optimization techniques can be used. If cash flows occur at different times, then present-value techniques can be used to normalize them to a common point in time.

Inherent in the process of software engineering estimation is the utilization of software engineering economics analysis techniques[80]. One such technique compares cost and benefits. An example involves the provisioning of a cell phone service in which there are two options.

- Option A: Accept an available operating system that requires $80K in software costs, but will achieve a peak performance of 120 transactions per second using five $10K minicomputer processors, because of high multiprocessor overhead factors[31].
- Option B: Build a new operating system that would be more efficient and would support a higher peak throughput, but would require $180 in software costs[32].

In general, software engineering decision problems are even more complex as Options A and B and will have several important criteria on which they differ such as robustness, ease of tuning, ease of change, functional capability, and so on[35]. If these criteria are quantifiable, then some type of figure of merit can be defined to support a comparative analysis of the preference of one option over another[90]. If some of the criteria are unquantifiable (user goodwill, programmer morale, etc.), then some techniques for comparing unquantifiable criteria need to be used.

In software engineering, decision issues are generally complex and involve analyzing risk, uncertainty, and the value of information[99]. The main economic analysis techniques available to resolve complex decisions include the following:

- Techniques for decision making under complete uncertainty, such as the maximax rule, the maximin rule and the Laplace rule [19]. These techniques are generally inadequate for practical software engineering decisions.
- Expected-value techniques, in which one estimates the probabilities of occurrence of each outcome; i.e., successful development of a new operating system, and complete the expected payoff of each option: EV = Prob (success)*Payoff (successful OS) + Prob (failure) *Payoff (unsuccessful OS)[55][57]. These techniques are better than decision making under complete uncertainty, but they still involve a great deal of risk if the Prob (failure) is considerably higher than the estimate of it.

- Techniques in which one reduces uncertainty by buying information [40]. For example, prototyping is a way of buying information to reduce uncertainty about the likely success or failure of a multiprocessor operating system; [39] by developing a rapid prototype of its high-risk elements; one can get a clearer picture of the likelihood of successfully developing the full operating system [37].

Information-buying often tends to be the most valuable aid for software engineering decisions. The question of how much information-buying is enough can be answered via statistical decision theoretic techniques using Bayes' Law, which provides calculations for the expected payoff from a software project as a function of the level of investment in a prototype[42][44]. In practice, the use of Bayes' Law involves the estimation of a number of conditional probabilities which are not easy to estimate accurately. However, the Bayes' Law approach can be translated into a number of value-of-information guidelines, or conditions under which it makes good sense to decide on investing in more information before committing to a particular course of action[70].

- **Condition 1:** There exist attractive alternatives which payoff varies greatly, depending on some critical states of nature[78]. If not, engineers can commit themselves to one of the attractive alternatives with no risk of significant loss.
- **Condition 2:** The critical states of nature have an appreciable probability of occurring. If not, engineers can again commit without major risk[92]. For situations with extremely high variations in payoff, the appreciable probability level is lower than in situations with smaller variations in payoff.
- **Condition 3:** The investigations have a high probability of accurately identifying the occurrence of the critical states of nature. If not, the investigations will not do much to reduce the risk of loss incurred by making the wrong decision.
- **Condition 4:** The required cost and schedule of the investigations do not overly curtail their net value. It does one little good to obtain results which cost more than those results can save for us, or which arrive too late to help make a decision.
- **Condition5:** There exists significant side benefits derived from performing the investigations. Again, one may be able to justify an investigation solely on the basis of its value in training, team-building, customer relations, or design validation[98][99].

The following figure 4.1 shows Cost Estimation Process Model (Preliminary):

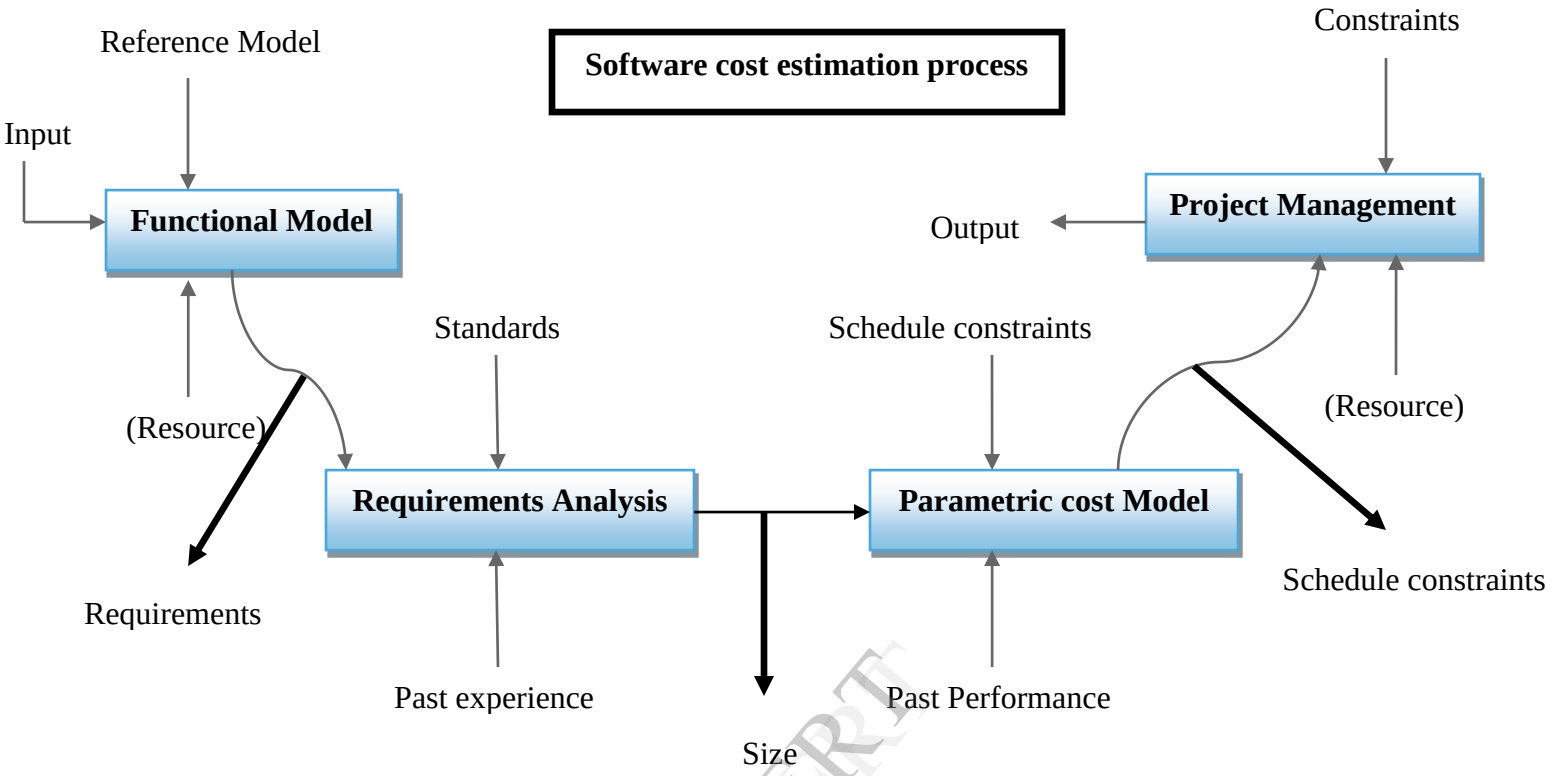


Figure 4.1: Cost Estimation Process Model (Preliminary)

5 PRPOSED WORK

The partition platform provides three methods for producing a final tree: 1). Decision Tree method 2).Bootstrap Forest method 3).Boosted Tree method. In paper, we used MV-dataset that provides 11 physical characteristics of MV-dataset (x_1 to x_{10}, y). There are a total of 40698 samples.

DATASET			**Performance Criteria**	**Model Used**		
NAME	FEATURES	INSTANCES		DDECISION TREE	BOOSTRAP FOREST	BOOTSRAP TREE
MV	11	40698	RSQUARE	0.410	0.926	0.990
			RMSE	8.0023787	2.8378287	1.0619691

Table5.1: Computed RSQUARE and RMSE Criterion for All Models of sample dataset MV

The following table5.2 shows samples of MV-dataset with attribute values

S.NO	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	Y

1	-4.80915	-10.5062	0	-5.25312	0.665394	-23.6227	0	1	445.012	1082	-2.40458
2	-0.92871	-10.8176	1	-5.40878	0.430141	-25.329	0	0	250.038	1115	-26.2577
3	1.30456	-11.9609	0	-5.98045	0.815231	-7.46146	0	1	261.694	1163	0.65228
4	-4.06876	-11.7433	0	-5.87165	0.285156	-2.89871	1	0	175.301	1191	-6.96747
5	4.75025	-12.9444	2	-6.4722	0.701619	-12.8418	1	1	341.723	1010	2.37512
6	0.075015	-14.4788	1	-7.23941	-0.11676	-14.3854	1	0	288.747	1063	-14.3104
7	2.25798	-14.2765	0	-7.13827	0.286783	-9.16169	1	0	136.917	1148	-6.90371
8	4.61535	-14.8076	1	-7.4038	-0.54672	-26.3731	0	0	469.521	1059	-21.7577
9	1.62166	-14.5325	1	-7.26626	0.668163	-12.0381	1	1	132.771	1159	0.810828
10	1.71477	-12.9115	2	0.857385	0.366435	2.19516	0	0	286.766	1177	0.857385

Table5.2: Sample dataset MV with attribute values.

5.1 DECISION TREE

5.1.1 INTRODUCTION

The Decision Tree method makes a single pass through the data and produces a single tree. You can interactively grow the tree one split at a time, or grow the tree automatically if validation is used[109].

5.1.2 DATASET DESCRIPTION

The following Table5.3 shows calculation of different parameter with different datasets values using Decision tree method.

S.NO	DATASETS	FEATURES	INSTANCES	RSquare	RMSE	N	MEAN	S.D
1	AILERONS	35	13750	0.353	0.0003279	13750	-0.000872	0.0004078
2	CALIFORNIA	9	20640	0.310	95853.999	20640	173487.4	91509.837
3	ELEVATORS	19	16599	0.282	0.0056896	16599	0.21625	0.0067117
4	HOUSE	17	22786	0.127	4937.253	22784	50074.44	52843.476
5	TIC	86	9822	0.026	0.2337471	9822	0.059662	0.2368716

Table5.3 Different datasets with different parameter values using Decision tree method

5.1.3 ANALYSIS OF RESULTS

The Split button is used to partition the data, creating a tree of partitions. Repeatedly splitting the data results in branches and leaves of the tree. This can be thought of as growing the tree. The Prune button is used to combine the most recent split back into one group. The following 5.1 shows decision tree initial report.

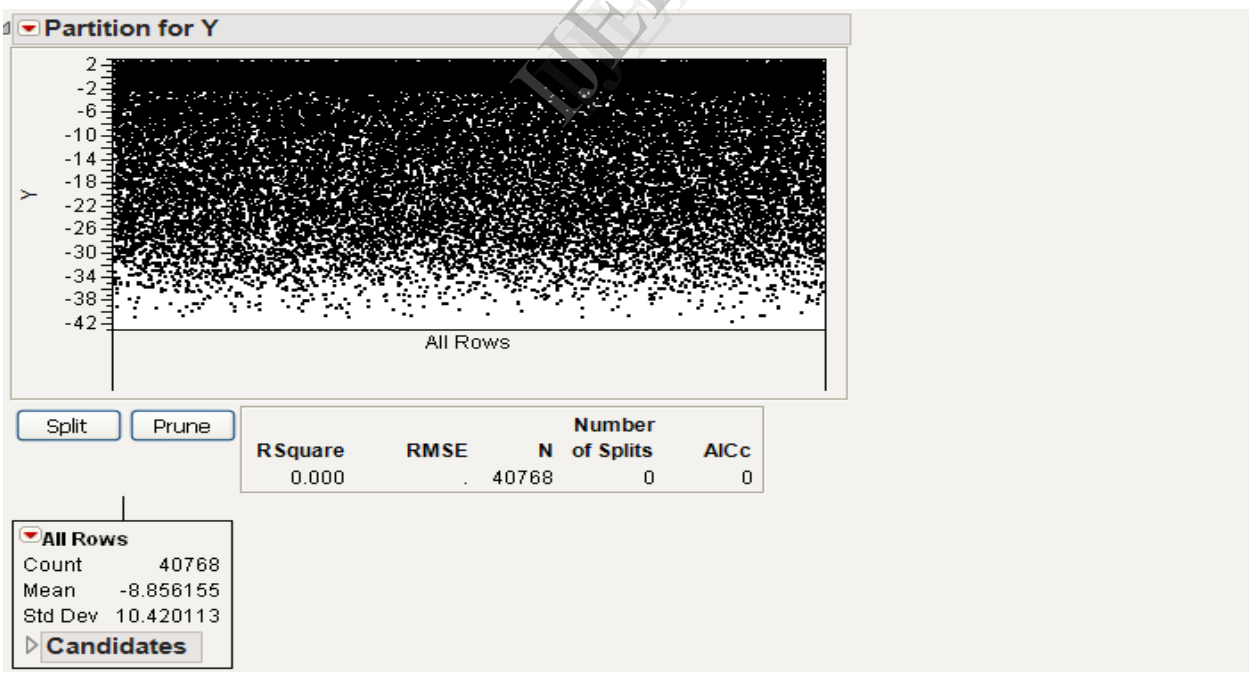
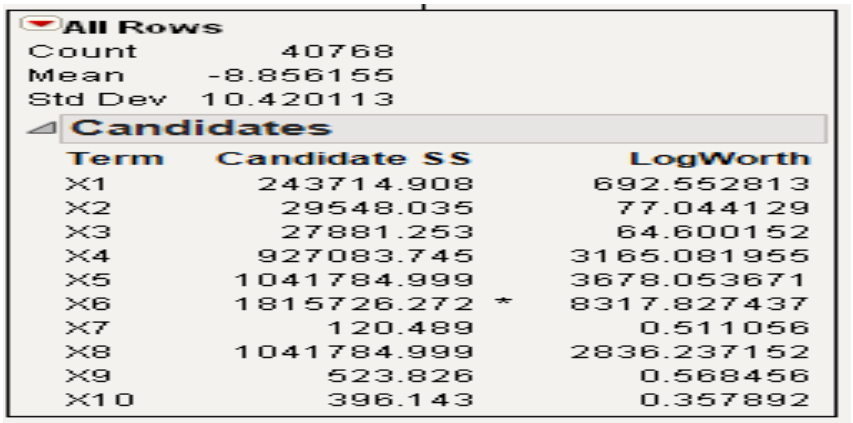


Figure5.1: decision tree initial report of MV dataset

Initially, all rows are in one branch. In order to determine the first split, each X column must be considered. The candidate columns are shown in the Candidates report. As shown in Candidates

Columns, the room's column has the largest Log Worth, and is therefore the optimum split to make. The following 5.2 shows Candidates Columns.



All Rows

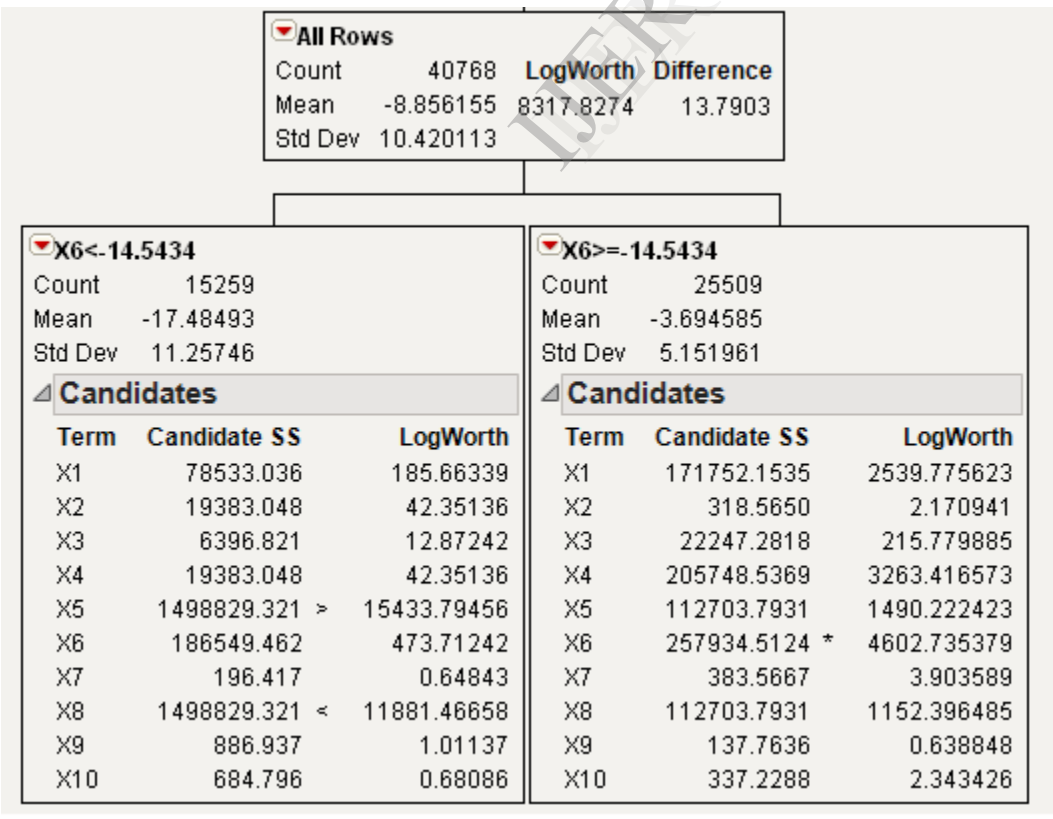
Count	40768
Mean	-8.856155
Std Dev	10.420113

Candidates

Term	Candidate SS		LogWorth
X1	243714.908		692.552813
X2	29548.035		77.044129
X3	27881.253		64.600152
X4	927083.745		3165.081955
X5	1041784.999		3678.053671
X6	1815726.272	*	8317.827437
X7	120.489		0.511056
X8	1041784.999		2836.237152
X9	523.826		0.568456
X10	396.143		0.357892

Figure 5.2: Candidates Columns of MV dataset

The optimum split is noted by an asterisk. However, there are cases where the SS is higher for one variable, but the Log worth is higher for a different variable. In this case > and < are used to point in the best direction for each variable. The asterisk corresponds to the condition where they agree. The following figure 5.3 shows first split of decision tree.



All Rows

Count	40768	LogWorth	Difference
Mean	-8.856155	8317.8274	13.7903
Std Dev	10.420113		

X6<-14.5434

Count	15259
Mean	-17.48493
Std Dev	11.25746

Candidates

Term	Candidate SS		LogWorth
X1	78533.036		185.66339
X2	19383.048		42.35136
X3	6396.821		12.87242
X4	19383.048		42.35136
X5	1498829.321	>	15433.79456
X6	186549.462		473.71242
X7	196.417		0.64843
X8	1498829.321	<	11881.46658
X9	886.937		1.01137
X10	684.796		0.68086

X6>=-14.5434

Count	25509
Mean	-3.694585
Std Dev	5.151961

Candidates

Term	Candidate SS		LogWorth
X1	171752.1535		2539.775623
X2	318.5650		2.170941
X3	22247.2818		215.779885
X4	205748.5369		3263.416573
X5	112703.7931		1490.222423
X6	257934.5124	*	4602.735379
X7	383.5667		3.903589
X8	112703.7931		1152.396485
X9	137.7636		0.638848
X10	337.2288		2.343426

Figure 5.3: first split of decision tree.

The original 40768 observations are now split into two parts:

- A left leaf, corresponding to x6 < 14.5434, has 15259 observations.
- A right leaf, corresponding to x6 ≥ 614.5434, has 25509 observations
-

For the left leaf, the next split would happen on the column x5, which has a SS of 1498829.321. For the right leaf, the next split would happen on the column x6, which has a SS of 257934.5124. Since the SS for the left leaf is lower, we can stop the splitting procedure. The following figure 5.4 shows second split of decision tree.

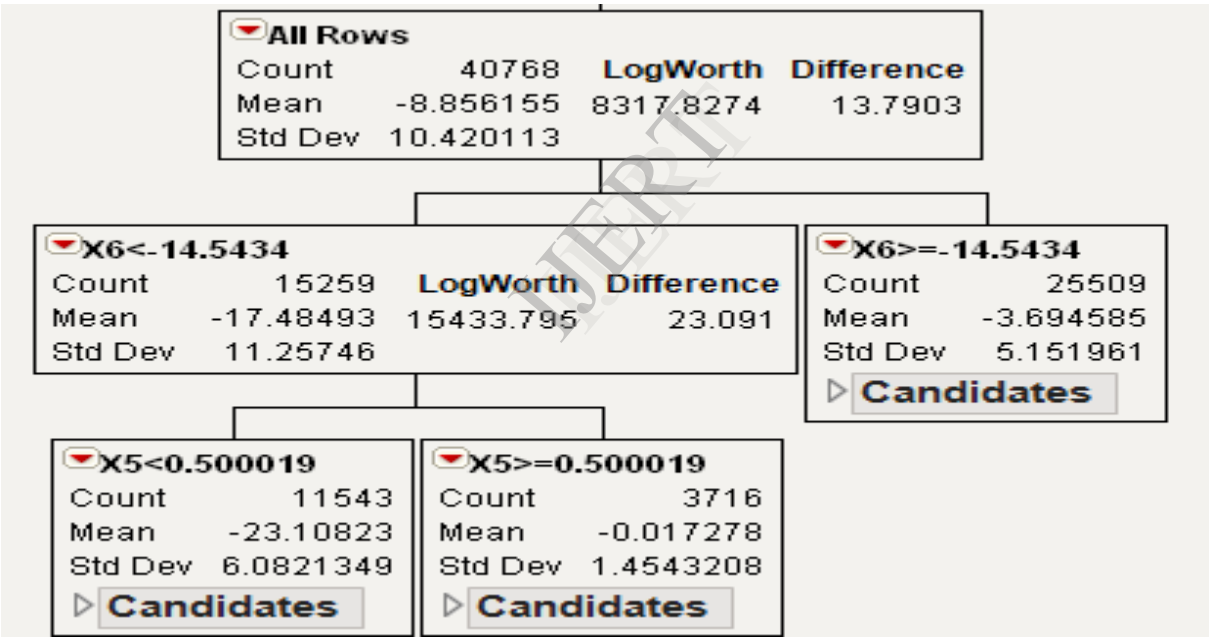


Figure 5.4: second split of decision tree.

The 15259 observations from the previous left leaf are now split into two parts:

•A left leaf, corresponding to x5 ≥ 0.500019, 3716 has observations.
•A right leaf, corresponding to x5 < 0.500019, has 11543 observations.
The 506 original observations are now split into three parts:
•A leaf corresponding to x6 < 14.5434 and x5 ≥ 0.500019.
•A leaf corresponding to x6 < 14.5434 and lstat < 0.500019.
•A leaf corresponding to room's ≥ 14.5434.

The predicted value for the observations in each leaf is the average response. The plot is divided into three sections, corresponding to the three leafs. These predicted values are shown on the plot with black lines. The points are put into random horizontal positions in each section. The vertical position is based on the response produces a plot of actual values by predicted values. This is for continuous responses only.

The following figure 5.5 shows Plot Actual by Predicted of MV-dataset.

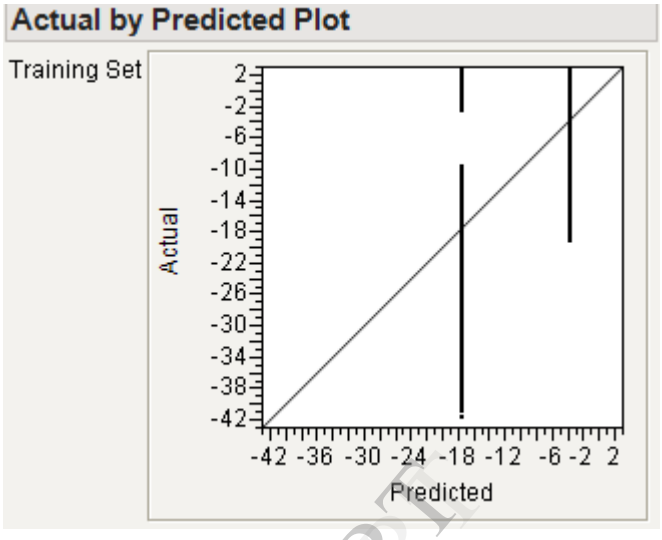


Figure 5.5: Plot Actual by Predicted of MV-dataset

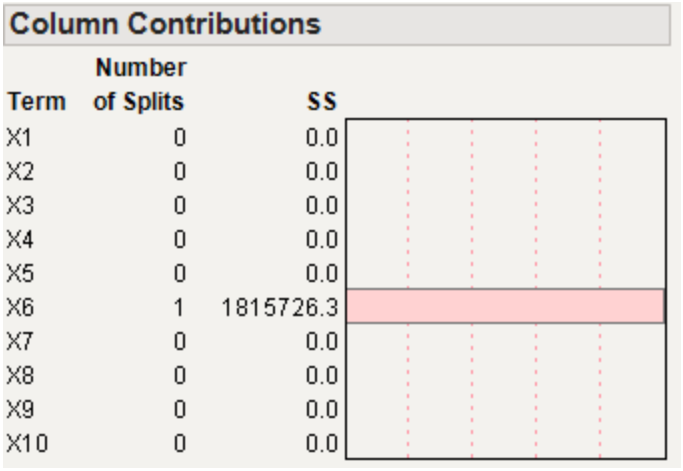
Gives the mean and count or rates for the bottom-level leaves of the report. The following figure 5.6 shows Leaf Report of MV-dataset.

Leaf Report

Leaf Label	Mean	Count
X6<-14.5434	-17.48493	15259
X6>=-14.5434	-3.6945851	25509

Figure 5.6: Leaf Report of MV-dataset

Brings up a report showing how each input column contributed to the fit, including how many times it was split and the total G2 or Sum of Squares attributed to that column. The following figure 5.7 shows Column Contributions of MV-dataset.



Column Contributions

Term	Number of Splits	SS
X1	0	0.0
X2	0	0.0
X3	0	0.0
X4	0	0.0
X5	0	0.0
X6	1	1815726.3
X7	0	0.0
X8	0	0.0
X9	0	0.0
X10	0	0.0

Figure 5.7: Column Contributions of MV-dataset

Shows a plot of R2 vs. the number of splits. If you use validation, separate curves are drawn for training and validation R2. The following figure 5.8 shows Split History of MV-dataset.

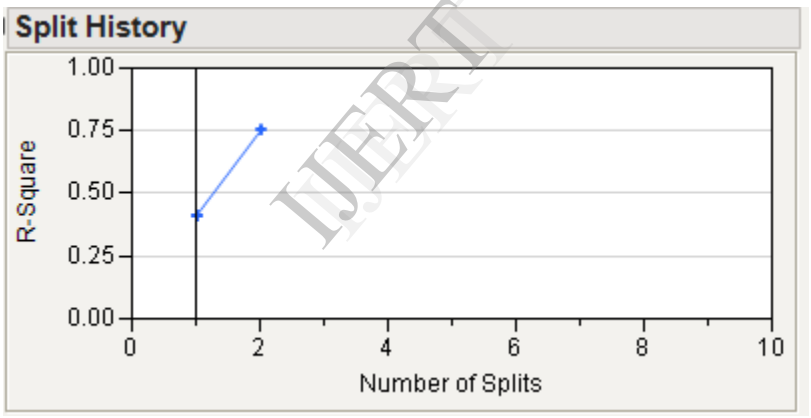


Figure 5.8: Split History of MV-dataset

Shows a Cross validation report, giving fit statistics for both the training and folded sets. The following figure 5.9 shows K Fold Cross validation of MV-dataset.

Crossvalidation

k-fold		SSE	RSquare
5	Folded	2611257.43	0.4101
	Overall	2610703.84	0.4102

Figure 5.9: K Fold Cross validation of MV-dataset

5.2 BOOTSTRAP TREE

5.2.1 INTRODUCTION

Boosting is the process of building a large, additive decision tree by fitting a sequence of smaller trees. Each of the smaller trees is fit on the scaled residuals of the previous tree. The trees are combined to form the larger final tree[105]. The process can use validation to assess how many stages to fit, not to exceed the specified number of stages[100].
The tree at each stage is short, typically 1-5 splits. After the initial tree, each stage fits the residuals from the previous stage. The process continues until the specified number of stages is reached, or, if validation is used, until fitting an additional stage no longer improves the validation statistic[110]. The final prediction is the sum of the estimates for each terminal node over all the stages.
If the response is categorical, the residuals fit at each stage are offsets of linear logits[113]. The final prediction is a logistic transformation of the sum of the linear logits over all the stages.
For categorical responses, only those with two levels are supported.

5.2.2 DATASET DESCRIPTION

The following Table: 5.4 shows different datasets with different parameter values using Bootstrap tree method.

S.NO	DATASETS	FEATURES	INSTANCES	RSquare	RMSE	N
1	AILERONS	35	13750	0.806	0.0001795	13750
2	CALIFORNIA	9	20640	0.678	65442.552	20740
3	ELEVATORS	19	16599	0.647	0.0039924	16599
4	HOUSE	17	22786	0.500	37383.418	22784
5	TIC	86	9822	0.000	0.0000000	9822

Table: 5.4 shows different datasets with different parameter values using Bootstrap tree method

5.2.3 ANALYSIS OF RESULTS

If the Bootstrap Forest method is selected on the platform launch window, the Bootstrap Forest options window appears after clicking OK. Bootstrap Forest Fitting Options shows the window using the Car Poll.jmp data table. The column sex is used as the response, and the other columns are used as the predictors.

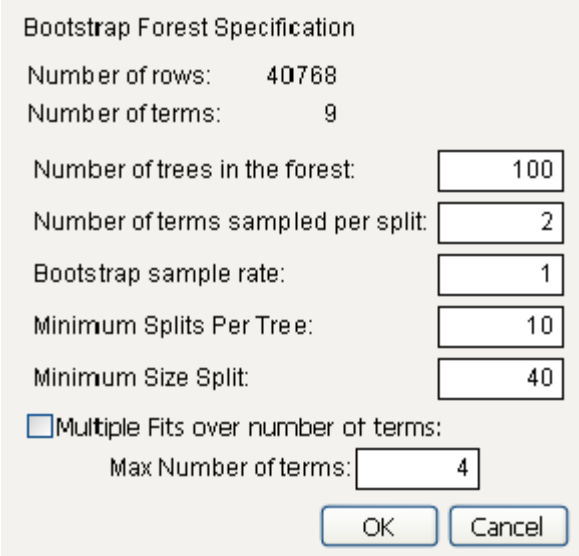


Figure: 5.10 Bootstrap Forest Fitting Options of mv dataset

Provides fit statistics for all the models fit if you selected the Multiple Fits option on the options window. Specifications provide information on the partitioning process.

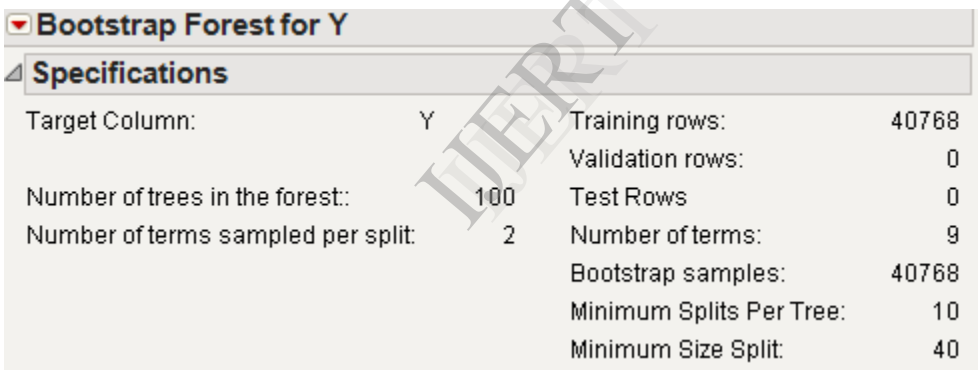


Figure: 5.11 Overall Statistics Model Validation - Set Summaries of mv dataset

Overall Statistics provides fit statistics for both the training and validation sets.

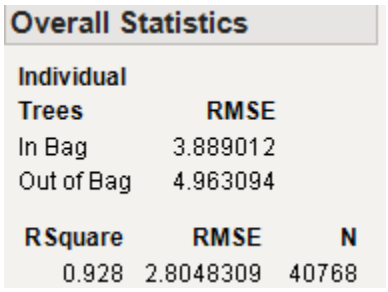


Figure: 5.12 Overall Statistics of mv dataset

Per-Tree Summaries gives summary statistics for each tree.

Per-Tree Summaries

Tree	Splits	Rank	OOB Loss	OOB Loss/N	RSquare	IB SSE	IB SSE/N	OOB N	OOB SSE	OOB SSE/N
1	93	43	175216.61	11.746102	0.8958	291490.99	7.1499949	14917	174203.7	11.6782
2	441	54	249982.4	16.793121	0.8568	401957.28	9.8596271	14886	246300.51	16.545782
3	9	94	1243938.1	82.951324	0.2369	2132052.9	52.297216	14996	1244076.3	82.960544
4	7	72	369060.43	24.576175	0.7778	622529.48	15.270052	15017	369099.65	24.578788
5	239	56	259743.94	17.307032	0.8445	436190.22	10.699328	15008	258066.79	17.195282
6	199	34	151010.73	10.10849	0.9127	244181.32	5.9895339	14939	149889.32	10.033424
7	227	45	178619.79	11.877105	0.8904	305881.33	7.5029761	15039	177412.38	11.79682
8	415	73	381624.49	25.462002	0.7767	619717.51	15.201077	14988	376581.39	25.125527
9	839	29	130135.3	8.6866896	0.9302	196168.56	4.811827	14981	126603.76	8.4509553
10	625	28	121320.55	8.067599	0.9307	193451.33	4.7451759	15038	118119.31	7.8547219
11	17	89	1206083.3	81.305332	0.2378	2149048.2	52.714094	14834	1205831.5	81.288356
12	123	85	721647.79	48.380785	0.5606	1229799.4	30.165801	14916	720445.01	48.300148
13	565	17	87859.022	5.9184252	0.9485	146348.95	3.5897997	14845	86818.366	5.8483237
14	601	26	115344.36	7.7479919	0.9414	164713.61	4.0402671	14887	112644.32	7.5666232
15	365	33	144453.39	9.6276586	0.9125	244470.55	5.9966284	15004	142122.01	9.4722746

Figure: 5.13 Per-Tree Summaries of mv dataset

Plot Actual by Predicted provides a plot of actual versus predicted values. This is only for continuous responses.

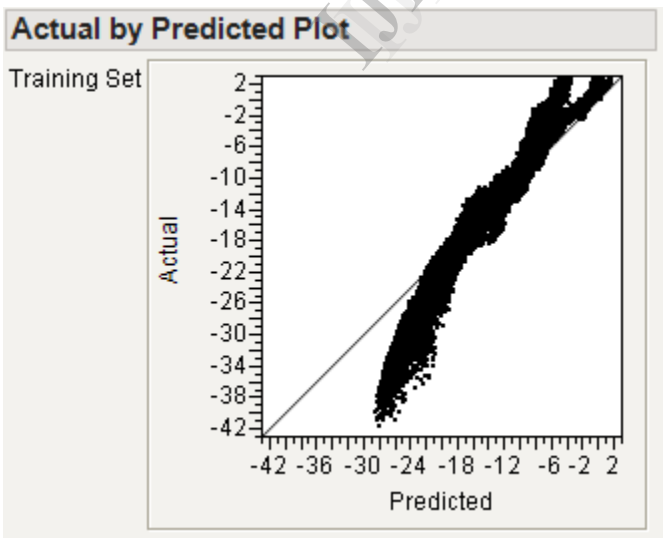


Figure: 5.14 Plot Actual by Predicted of mv dataset

Column Contributions brings up a report showing how each input column contributed to the fit, including how many times it was split and the total G2 or Sum of Squares attributed to that column.

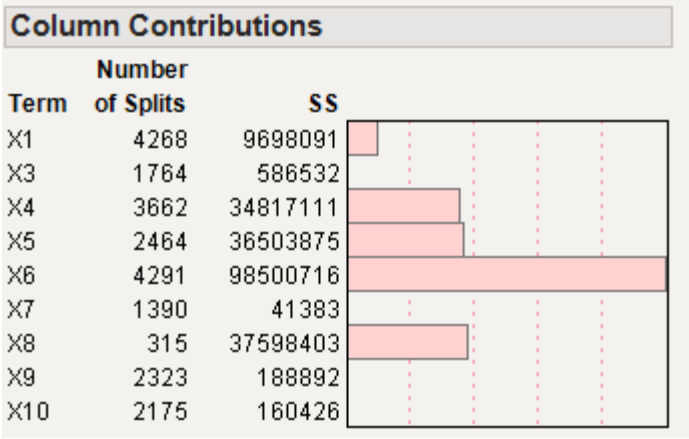


Figure: 5.15 Column Contributions of mv dataset

5.3 BOOTSRAP FOREST

5.3.1 INTRODUCTION

The Bootstrap Forest method makes many trees, and averages the predicted values to get the final predicted value [73]. Each tree is grown on a different random sample (with replacement) of observations, and each split on each tree considers only a random sample of candidate columns for splitting [84]. The process can use validation to assess how many trees to grow, not to exceed the specified number of trees.

5.3.2 DATASET DESCRIPTION

The following Table5.5 shows different datasets with different parameter values using Bootstrap Forest method

S.NO	DATASETS	FEATURES	INSTANCES	RSquare	RMSE	N
1	AILERONS	35	13750	0.843	0.0001615	13750
2	CALIFORNIA	9	20640	0.797	51955.496	20640
3	ELEVATORS	19	16599	0.749	0.0033665	16599
4	HOUSE	17	22786	0.651	31238.959	22784
5	TIC	86	9822	0.275	0.2017361	9822

Table:5.5 Different datasets with different parameter values using Bootstrap Forest method

5.3.3 ANALYSIS OF RESULTS

If the Boosted Tree method is selected on the platform launch window, the Boosted Tree options window appears after clicking OK.

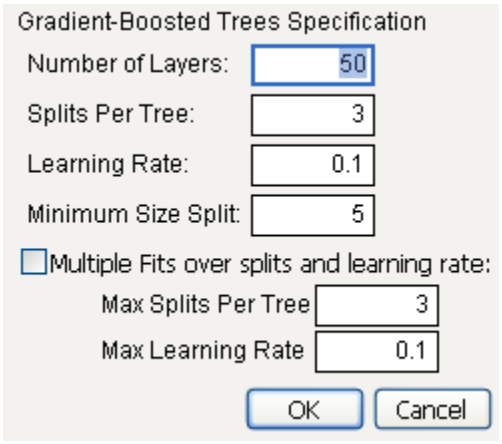


Figure: 5.16 Bootstrap Forest Fitting Options of mv dataset

Provides fit statistics for all the models fit if you selected the Multiple Fits option on the options window. Specifications provide information on the partitioning process.

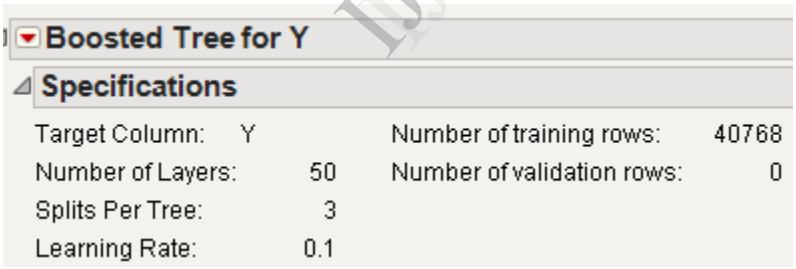


Figure: 5.17 Model Validation - Set Summaries of mv dataset

Overall Statistics provides fit statistics for both the training and validation sets.

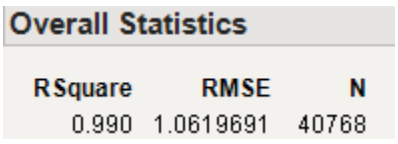


Figure: 5.18 Overall Statistics of mv dataset

Plot Actual by Predicted provides a plot of actual versus predicted values. This is only for continuous responses.

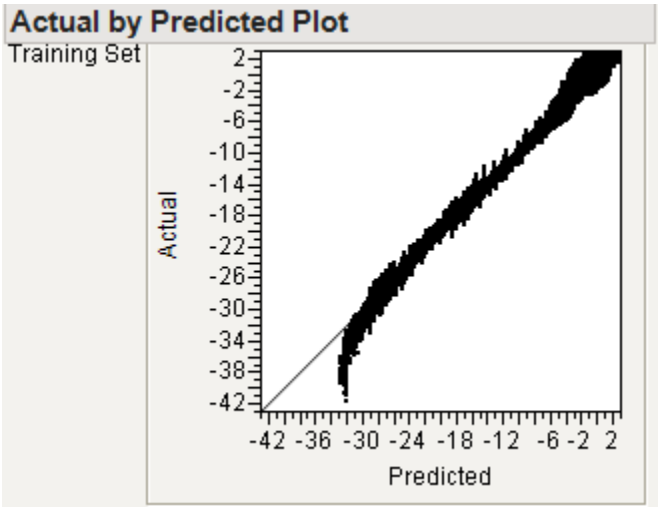


Figure: 5.19 Plot Actual by Predicted of mv dataset

Column Contributions brings up a report showing how each input column contributed to the fit, including how many times it was split and the total G2 or Sum of Squares attributed to that column.

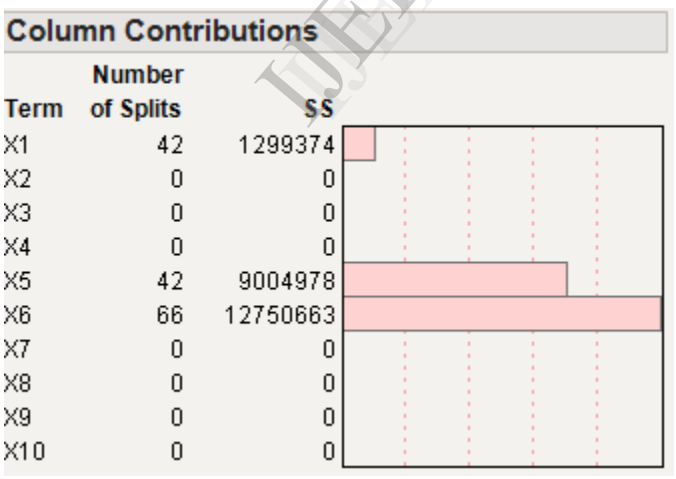


Figure: 5.20 Column Contributions of mv dataset

6. CONCLUSIONS

The accurate and reliable cost estimation effort is very important for software project development. Our proposed is concerned with constructing software cost effort estimation model based on partitioning techniques. In this paper, we are comparing these techniques with parameters such as Rsquare error, Root mean square errors using different datasets (such as California, Ailerons, Elevators, House, Tic**)**.The results show that the bootstrap tree has the lowest RMSSE value i.e. 1.0619691 and the second best performance is shown by bootstrap forest software estimation system with 2.8378287 RMSSE value. Finally experimental results are showed using MV dataset. Hence the proposed partitioning techniques can be used efficient for software cost effort estimation..

REFERENCES

[1]. A. J. Albrecht, and J.E. Gaffney, "Software function, source lines of codes, and development effort prediction: a software science validation," IEEE Trans Software Eng. SE-9, 1983, pp. 639-648.

[2]. J.D. Aron, Estimating Resource for Large Programming Systems, NATO Science Committee, Rome, Italy, October 1969.

[3]. R.K.D. Black, R.P. Curnow, R. Katz and M.D. Gray, BCS Software Production Data, Final Technical Report, RADC-TR-77-116, Boeing Computer Services, Inc., March 1977.

[4]. B.W. Boehm, Software engineering economics, Englewood Cliffs, NJ: Prentice-Hall, 1981.

[5]. B.W. Boehm et al "The COCOMO 2.0 Software Cost Estimation Model," American Programmer, July 1996, pp.2-17.

[6]. L.C. Briand, K. El Eman, F. Bomarius, "COBRA: A hybrid method for software cost estimation, benchmarking, and risk assessment," International conference on software engineering, 1998, pp. 390-399.

[7]. G. Cantone, A. Cimitile and U. De Carlini, "A comparison of models for software cost estimation and management of software projects," in Computer Systems: Performance and Simulation, Elisevier Science Publishers B.V., 1986.

[8]. W.S. Donelson, "Project planning and control," Datamation, June 1976, pp. 73-80.

[9]. N. E. Fenton and S. L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, PWS Publishing Company, 1997.

[10]. G.R. Finnie, G.E. Wittig, AI tools for software development estimation, Software Engineering and Education and Practice Conference, IEEE Computer Society Press, pp. 346-353, 1996.

[11]. M. H. Halstead, Elements of software science, Elsevier, New York, 1977.

[12]. P.G. Hamer, G.D. Frewin, "M.H. Halstead's Software Science – a critical examination," Proceedings of the 6th International Conference on Software Engineering,Sept. 13-16, 1982, pp. 197-206.

[13]. F.J. Heemstra, "Software cost estimation," Information and Software Technology vol. 34, no. 10, 1992, pp. 627-639.

[14]. J. Hihn and H. Habib-Agahi, "Cost Estimation of software intensive projects: a survey of current practices," International Conference on Software Engineering, 1991, pp. 276-287.

[15]. ISBSG, International software benchmarking standards group, http://www.isbsg.org/au.

[16]. D.R. Jeffery, G. C. Low, "A comparison of function point counting techniques," IEEE Trans on Soft. Eng., vol. 19, no. 5, 1993, pp. 529-532.

[17]. C. Jones, Applied Software Measurement, Assuring Productivity and Quality, McGraw-Hill, 1997.

[18]. C.F. Kemerer, "An empirical validation of software cost estimation models," Communications of the ACM, vol. 30, no. 5, May 1987, pp. 416-429

[19]. R.D. Luce and H. Raiffa, Games and Decisions. New York: Wiley, 1957.

[20]. R. Nelson, Management Handbook for the Estimation of Computer Programming Costs, AD-A648750, Systems Development Corp., 1966

[21]. R.E. Park, "PRICE S: The calculation of within and why," Proceedings of ISPA Tenth Annual Conference, Brighton, England, July 1988.

[22]. G.N. Parkinson, Parkinson's Law and Other Studies in Administration, Houghton-Miffin, Boston, 1957.

[23]. N. A. Parr, "An alternative to the Raleigh Curve Model for Software development effort," IEEE on Software Eng. May 1980.

[24]. L.H. Putnam, "A general empirical solution to the macro software sizing and estimating problem," IEEE Trans. Soft. Eng. May 1980

[25]. W. Royce, Software project management: a unified framework, Addison Wesley, 1998

[26]. V. Y. Shen, S. D. Conte, H. E. Dunsmore, "Software Science revisited: a critical analysis of the theory and its empirical support," IEEE Transactions on Software Engineering, 9, 2, 1983, pp. 155-165.

[27]. M. Shepperd and C. Schofield, "Estimating software project effort using analogy," IEEE Trans. Soft. Eng. SE-23:12, 1997, pp. 736-743.

[28]. K. Srinivasan and D. Fisher, "Machine learning approaches to estimating software development effort", IEEE Trans. Soft. Eng., vol. 21, no. 2, Feb. 1995, pp. 126-137.

[29]. D. St-Pierre, et al, Full Function Points: Counting Practice Manual, Technical Report 1997-04, University of Quebec at Montreal, 1997

[30]. S. S. Vivinanza, T. Mukhopadhyay, and M. J. Prietula, "Software-effort estimation: an exploratory study of expert performance," Information Systems Research, vol. 2, no. 4, Dec. 1991, pp. 243-262.

[31]. C.E. Walston and C.P. Felix, "A method of programming measurement and estimation," IBM Systems Journal, vol. 16, no. 1, 1977, pp. 54-73.

[32]. R.W. Wolverton, "The cost of developing large-scale software," IEEE Trans. Computer, June 1974, pp. 615-636.

[33].Bode, J., 1997. Decision support with neural networks in themanagement of research and development: concepts and application to cost estimation. Information and Management, 34, 33–40.

[34].Bode, J., 2000. Neural networks for cost estimation: simulations and pilot application. International Journal of Production Research, 38 (6), 1231–1254. Deng, S., Chin, H., and Yeh, T.H., 2009. Using artificial neural networks to airframe wing structural design cost estimation. Journal of Chung Cheng Institute of Technology, 38 (1), 97–106.

[35].Deng, S. and Yeh, T.H., 2009. Applying machine learning methods to the airframe structural design cost estimation – a case study of wing-box. In: 19th Annual International INCOSE Symposium, 20–23 July, Singapore. Red Hook, NY: Curran Associates, Inc.

[36].Duan, K., Keerthi, S.S., and Poo, A.N., 2003. Evaluation of simple performance measures for tuning SVM hyperparameters. Neurocomputing, 50, 41–59. Duran, O., Rodriguez, N., and Consalter, L.A., 2009. Neural networks for cost estimation of shell and tube heat exchangers. Expert Systems with Applications, 36, 7435–7440.

[37].Garza, J. and Rouhana, K., 1995. Neural networks versus parameter-based application in cost estimating. Cost Engineering, 37 (2), 14–18.

[38].Gunn, S., 1998. Support vector machines for classification and regression. Boston, MA: MIT Press.

[39].Hagan, T., Demuth, H., and Beale, M., 1996. Neural network design. Boston, MA: PWS Publishing.

[40].Haykin, S., 1999. Neural networks – a comprehensive foundation. 2nd ed. Singapore: Prentice Hall. International Society of Parametric Analysis (ISPA). 2008. Parametric estimating handbook. 4th ed. Vienna: ISPA/ SCEA Joint Office.

[41].Jolai, F. and Ghanbari, A., 2010. Integrating data transformation techniques with Hopfield neural networks for solving travelling salesman problem. Expert Systems with Applications, 37, 5331 5335.

[42].Layer, A., et al., 2002. Recent and future trends in cost estimation. International Journal of Computer Integrated Manufacturing, 15 (6), 499–510. Lippmann, R.P., 1987. An introduction to computing with neural nets. IEEE ASSP MAGAZINE, 4, 4– 22.

[43].Mamdouh, R., 2007. Data preparation for data mining using SAS. San Francisco, CA: Morgan Kaufmann Publishers.

[44].Montgomery, D.C., 2001. Design and analysis of experiments. 5th ed. New York: Wiley.

[45].Prasad, P.W.C. and Beg, A., 2009. Investigating datapreprocessing methods for circuit complexity models. Expert Systems with Applications, 36, 519–526.

[46].Shtub, A. and Versano, R., 1999. Estimating the cost of steel pipe bending, a comparison between neural networks and regression analysis. International Journal of Production Economics, 62, 201–207.

[47].Sigurdsen, A., 1992. CERA: an integrated cost estimating program. Cost Engineering, 34 (6), 25–30.

[48].Smith, A.E. and Mason, A.K., 1997. Cost estimation predictive modeling: regression versus neural network. The Engineering Economist, 42 (2), 137–161.

[49].Smola, A.J. and Scho¨ lkopf, B., 2002. Learning with kernels – support vector machines, regularization, optimization, and beyond. Boston, MA: MIT Press.

[50].Suykens, J.A.K., et al., 2002. Least square support vector machines. Singapore: World Scientific.

[51].Tay, E.H. and Cao, L., 2001. Application of support vector machines in financial time series forecasting. Omega, 29, 309–317.

[52].Vapnik, V., 1995. The nature of statistical learning theory. New York: Springer. Verlinden, B., et al., 2008. Cost estimation for sheet metal parts using multiple regression and artificial neural networks: a case study. International Journal of Production Economics, 111, 484–492.

[53].Wang, H.S., 2007. Application of BPN with featurebased models on cost estimation of plastic injection products. Computers and Industrial Engineering, 53, 79– 94.

[54].Yale, K., 1997. Preparing the right data for training neural networks. IEEE Spectrum, 34 (3), 64–66.

[55].Younossi, O., Kennedy, M., and Graser, J.C., 2001. Military airframe cost – the effects of advanced materials and manufacturing processes – MR1370. Santa Monica, CA: RAND.

[56].Zhang, Y.F. and Fuh, J.Y.H., 1998. A neural network approach for early cost estimation of packaging products. Computers and Industrial Engineering, 34 (2), 433– 450.

[57].B. Boehm, Software Cost Estimation with COCOMO II, Prentice Hall PTR, Upper Saddle River, New Jersey, 2000.

[58]. X. Huang, D. Ho, L. Capretz and J. Ren "Novel Neuro-Fuzzy Models for Software Cost Estimation", Proc.of the Third International Conference on Quality Software, IEEE ComputerSociety Press, Dallas, TX, USA, 2003.

[59]. X. Huang, D. Ho, J. Ren and L. Capretz "An Intelligent Approach to Software Cost Prediction", Proc. Of the 18th International Forum on COCOMO and Software Cost Modeling,Los Angeles, CA, USA, 2003

[60] .Haykin S, Neural Networks: A Comprehensive Foundation.Prentice Hall, 1998.

[61] .Zadeh L A, Fuzzy Logic. Computer, Vol 21, pp. 83-93, 1988.

[62]. Huang X, Ho D, Ren J, Capretz L, A Soft Computing Framework for Software Effort Estimation. Soft Computing Journal, Springer, available at www.springeronline.com, 2005.

[63]. Ali Idri, Taghi M. Khoshgoftaar, Alain Abran "Can Neural nets be easily interpreted in Software Cost Estimation", Proc. Of World Congress on Computational Intelligence, Hawaii, 2002.

[64] .A. Abraham, Adaptation of Fuzzy Inference System Using Neural Learning, Springer Berlin, ISSN: 1434-9922 (Print) 1860-0808 (Online), vol. 181, 2005.

[65] A. Abraham and M.R. Khan, Neuro-Fuzzy Paradigms for Intelligent Energy Management, Innovations in Intelligent Systems: Design, Management and Applications, Studies in Fuzziness and Soft Computing, Springer Verlag Germany, Chapter 12, pp. 285-314, 2003.

[66] .B. W. Boehm, Software engineering economics, Englewood Cliffs, NJ: Prentice-Hall, 1981.

[67] C. E. Walston, C. P. Felix, A method of programming measurement and estimation, IBM Systems Journal, vol. 16, no. 1, pp. 54-73, 1977.

[68] G.N. Parkinson, Parkinson's Law and Other Studies in Administration, Houghton-Miffin, Boston, 1957.

[69] L. H. Putnam, A general empirical solution to the macro software sizing and estimating problem, IEEE Trans. Soft. Eng., pp. 345-361, July 1978.

[70] J. R. Herd, J.N. Postak, W.E. Russell, K.R. Steward, Software cost estimation study: Study results, Final Technical Report, RADC-TR77- 220, vol. I, Doty Associates, Inc., Rockville, MD, pp. 1-10, 1977.

[71] J. W. Bailey and V. R. Basili, "A meta model for software development resource expenditure," in Proceedings of the International Conference on Software Engineering, pp. 107–115, 1981.

[72] R. E. Park, PRICE S: The calculation within and why, Proceedings of ISPA Tenth Annual Conference, Brighton, England, pp. 231-240, July 1988.

[73] R. Jang, Neuro-Fuzzy Modeling: Architectures, Analyses andApplications, Ph.D. Thesis, University of California, Berkeley, 1992.

[74] R.K.D. Black, R. P. Curnow, R. Katz, M. D. Gray, BCS Software Production Data, Final Technical Report, RADC-TR-77-116, Boeing Computer Services, Inc., March, pp. 5-8, 1977.

[75] R. Tausworthe, Deep Space Network Software Cost Estimation Model, Jet Propulsion Laboratory Publication 81-7, pp. 67-78, 1981.

[76] W. S. Donelson, Project Planning and Control, Datamation, pp. 73-80, June 1976.

[77].Abdel-Hamid, T. (1989a), "The Dynamics of Software Project Staffing: A System Dynamics-based Simulation Approach," IEEE Transactions on Software Engineering, February.

[78].Abdel-Hamid, T. (1989b), "Lessons Learned from Modeling the Dynamics of Software Development," Communications of the ACM, December.

[79].Abdel-Hamid, T. and S. Madnick (1991), Software Project Dynamics, Prentice-Hall, 1991.

[80].Abdel-Hamid, T. (1993), "Adapting, Correcting, and Perfecting Software Estimates: A Maintenance Metaphor," IEEE Computer, March.

[81].Abdel-Hamid, T. and S. Madnick (1993), "Modeling the Dynamics of Software Reuse: An Integrating System Dynamics Perspective," Presentation to the 6th Annual Workshop on Reuse, Owego, NY, November.

[82].Abts, C. (1997), "COTS Software Integration Modeling Study," Report prepared for USAF Electronics System Center, Contract No. F30602-94-C-1095, University of Southern California.

[83].Abts, C., B. Bailey, and B. Boehm (1998), "COCOTS Software Integration Cost Model: An

[84]/Overview," In Proceedings of the California Software Symposium, 1998.

[85].Albrecht, A. (1979), "Measuring Application Development Productivity," In Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, October, pp. 83–92.

[86].Baird, B. (1989), Managerial Decisions Under Uncertainty, Wiley, New York, 1989.

[87].Banker, R., R. Kauffman, and R. Kumar (1994), "An Empirical Test of Object-Based Output Measurement Metrics in a Computer Aided Software Engineering (CASE) Environment," Journal of Management Information System.

[88].Boehm, B. (1981), Software Engineering Economics, Prentice-Hall.

[89].Boehm, B., B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby (1995), "Cost Models for Future Software Life-cycle Processes: COCOMO 2.0," Annals of Software Engineering 1, Special Volume on Software Process and Product Measurement, J.D. Arthur and S.M. Henry, Eds., 45–60.

[90].Box, G. and G. Tiao, (1973), Bayesian Inference in Statistical Analysis, Addison-Wesley, Reading, MA.

[91].Briand, L., V. Basili, and W. Thomas (1992), "A Pattern Recognition Approach for Software Engineering Data Analysis," IEEE Transactions on Software Engineering 18, 11, November.

[92].Brooks, F. (1975), The Mythical Man-Month, Addison-Wesley, Reading, MA. Cash, J. (1979), "Dallas Tire Case," Harvard Business School.

[93].Chidamber, S. and C. Kemerer (1994), "A Metrics Suite for Object Oriented Design," CISR WP No. 249 and Sloan WP No. 3524-93, Center for Information Systems Research, Sloan School of Management, Massachusetts Institute of Technology.

[94].Chulani, S. (1997), "Modeling Defect Introduction," In California Software Symposium, November.

[95].Chulani, S. (1998), "Incorporating Bayesian Analysis to Improve the Accuracy of COCOMO II and Its Quality Model Extension," Ph.D. Qualifying Exam Report, University of Southern California, February.

[96].Chulani, S., B. Boehm, and B. Steece (1998), "Calibrating Software Cost Models Using Bayesian Analysis," Technical Report, USC-CSE-98-508, June. To appear in IEEE Transactions on Software Engineering, Special Issue on Empirical Methods in Software Engineering.

[97].Clark, B., S. Chulani, and B. Boehm (1998), "Calibrating the COCOMO II Post Architecture Model," In International Conference on Software Engineering, April.

[98].Forrester, J. (1961), Industrial Dynamics, MIT Press, Cambridge, MA.

[99].Gray, A. and S. MacDonell (1996), "A Comparison of Techniques for Developing Predictive Models of Software Metrics," Information and Software Technology 39, 1997. Helmer, O. (1966), Social Technology, Basic Books, NY.

[100].Henderson-Sellers, B. (1996), Object Oriented Metrics – Measures of Complexity, Prentice Hall, Upper Saddle River, NJ.

[101].Jensen R. (1983),"An Improved Macrolevel Software Development Resource Estimation Model," In Proceedings of 5th ISPA Conference, April, pp. 88–92.

[102].Jones, C. (1997), Applied Software Measurement, McGraw Hill.

[103].Judge, G., W. Griffiths, and C. Hill (1993), Learning and Practicing Econometrics, Wiley, New York.

[104].Kauffman, R. and R. Kumar (1993), "Modeling Estimation Expertise in Object Based ICASE Environments," Stern School of Business Report, New York University, January.

[105].Khoshgoftaar, T., A. Pandya, and D. Lanning (1995), "Application of Neural Networks for Predicting Program Faults," Annals of Software Engineering 1.

[106].Leamer, E. (1978), Specification Searches, Ad hoc Inference with Nonexperimental Data, Wiley Series, Wiley, New York.

[107].Madachy, R. (1994), "A Software Project Dynamics Model for Process Cost, Schedule and Risk Assessment," Ph.D. Dissertation, University of Southern California.

[108].Madachy, R. (1999), CS577a class notes, University of Southern California, 1999. Minkiewicz, A. (1998), "Measuring Object Oriented Software with Predictive Object Points," PRICE Systems.

[109].Nelson, E. (1966), "Management Handbook for the Estimation of Computer Programming Costs," Systems Development Corporation, October.

[110].Park R. (1988), "The Central Equations of the PRICE Software Cost Model," In 4th COCOMO Users' Group Meeting, November.

[111].Putnam, L. and W. Myers (1992), Measures for Excellence, Yourdon Press Computing Series.

[112].SELECT (1998), "Estimation for Component-based Development Using SELECT Estimator," SELECT Software Tools, website: http://www.select.com.

[113].Shepperd, M. and M. Schofield (1997), "Estimating Software Project Effort Using Analogies," IEEE Transactions on Software Engineering 23, 12.

[114].Symons (1991), Software Sizing and Estimating – Mark II FPA, Wiley, UK. USC-CSE (1997), "COCOMO II Model Definition Manual," Center for Software Engineering, Computer

[115].Science Department, University of Southern California, Los Angeles, CA, website: http://sunset.usc. edu/COCOMOII/cocomo.html.

[116].Weisberg, S. (1985), Applied Linear Regression, 2nd Edition, Wiley, New York, NY.

[117].Wittig, G (1995), "Estimating Software Development Effort with Connectionist Models," Working Paper Series 33/95, Monash University.

[118] K. Subba Rao ,L.S.S Reddy, Nagaraju Devarakonda, shaik Subhani, K. Raviteja "Software Cost Estimation in Multilayer Feed forward Network using Random Holdback Method" International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 10, October 2013.