Efficient Software Cost Estimation Using Partitioning Techniques

K. Subba Rao*1 L.S.S Reddy2 Nagaraju Devarakonda3
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 resources 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.
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.

![Figure 2.1: Classical view of the algorithmic cost estimation process](image_url)
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 COS 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: \( EV = \text{Prob (success)} \times \text{Payoff (successful OS)} + \text{Prob (failure)} \times \text{Payoff (unsuccessful OS)} \)[55][57]. These techniques are better than decision making under complete uncertainty, but they still involve a great deal of risk if the \( \text{Prob (failure)} \) 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.
- **Condition 5:** 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):
5 PROPOSED 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 (x₁ to x₁₀, y). There are a total of 40698 samples.

<table>
<thead>
<tr>
<th>DATASET</th>
<th>Performance Criteria</th>
<th>Model Used</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RSQUARE</td>
<td>RMSE</td>
</tr>
<tr>
<td>MV</td>
<td>0.410</td>
<td>8.0023787</td>
</tr>
</tbody>
</table>

Table 5.1: Computed RSQUARE and RMSE Criterion for All Models of sample dataset MV

The following table 5.2 shows samples of MV-dataset with attribute values

<table>
<thead>
<tr>
<th>S.N O</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>X7</th>
<th>X8</th>
<th>X9</th>
<th>X10</th>
<th>Y</th>
</tr>
</thead>
</table>

Figure 4.1: Cost Estimation Process Model (Preliminary)
Table 5.2: 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 Table 5.3 shows calculation of different parameter with different datasets values using Decision tree method.

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

Table 5.3 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.

![Decision Tree Initial Report](image_url)

Figure 5.1: 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.

![Candidates Columns](image)

**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.

![First Split of Decision Tree](image)

**Figure 5.3: first split of decision tree.**
The original 40768 observations are now split into two parts:

- A left leaf, corresponding to $x_6 < 14.5434$, has 15259 observations.
- A right leaf, corresponding to $x_6 \geq 614.5434$, has 25509 observations.

For the left leaf, the next split would happen on the column $x_5$, which has a SS of 149829.321. For the right leaf, the next split would happen on the column $x_6$, 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.](image)

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

- A left leaf, corresponding to $x_5 \geq 0.500019$, 3716 has observations.
- A right leaf, corresponding to $x_5 < 0.500019$, has 11543 observations.

The 506 original observations are now split into three parts:

- A leaf corresponding to $x_6 < 14.5434$ and $x_5 \geq 0.500019$.
- A leaf corresponding to $x_6 < 14.5434$ and $lstat < 0.500019$.
- A leaf corresponding to room’s $\geq 14.5434$. 
The predicted value for the observations in each leaf is the average response. The plot is divided into three sections, corresponding to the three leaves. 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.

![Actual by Predicted Plot](image)

**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.

<table>
<thead>
<tr>
<th>Leaf Label</th>
<th>Mean</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>X6&lt;14.5434</td>
<td>-17.48493</td>
<td>15259</td>
</tr>
<tr>
<td>X6&gt;=14.5434</td>
<td>-3.6945851</td>
<td>25509</td>
</tr>
</tbody>
</table>

**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.
Figure 5.7: Column Contributions of MV-dataset

Shows a plot of $R^2$ vs. the number of splits. If you use validation, separate curves are drawn for training and validation $R^2$. 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.

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.

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

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.

<table>
<thead>
<tr>
<th>Tree</th>
<th>Splits</th>
<th>Rank</th>
<th>OOB Loss</th>
<th>OOB Loss N</th>
<th>RSquare</th>
<th>IB SSE</th>
<th>IB SSE N</th>
<th>OOB N</th>
<th>OOB SSE</th>
<th>OOB SSE N</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>93</td>
<td>43</td>
<td>175210.061</td>
<td>11.746102</td>
<td>0.8956</td>
<td>231490.99</td>
<td>7.1493949</td>
<td>14917</td>
<td>174203.7</td>
<td>11.6782</td>
</tr>
<tr>
<td>2</td>
<td>141</td>
<td>54</td>
<td>249992.4</td>
<td>16.792121</td>
<td>0.6588</td>
<td>401967.28</td>
<td>9.956271</td>
<td>14886</td>
<td>246300.51</td>
<td>16.545782</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>94</td>
<td>1243936.9</td>
<td>82.951324</td>
<td>0.2369</td>
<td>2132052.92</td>
<td>52.237216</td>
<td>14996</td>
<td>1244076.3</td>
<td>82.930544</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>72</td>
<td>389080.43</td>
<td>24.576175</td>
<td>0.7778</td>
<td>422526.48</td>
<td>16.270052</td>
<td>15017</td>
<td>369006.65</td>
<td>24.579788</td>
</tr>
<tr>
<td>5</td>
<td>238</td>
<td>56</td>
<td>259743.94</td>
<td>17.307032</td>
<td>0.6449</td>
<td>436190.22</td>
<td>10.639328</td>
<td>15008</td>
<td>250866.79</td>
<td>17.135262</td>
</tr>
<tr>
<td>6</td>
<td>199</td>
<td>34</td>
<td>151010.73</td>
<td>10.10849</td>
<td>0.9127</td>
<td>244181.32</td>
<td>6.9995339</td>
<td>14939</td>
<td>149869.32</td>
<td>10.033424</td>
</tr>
<tr>
<td>7</td>
<td>227</td>
<td>45</td>
<td>178919.79</td>
<td>11.67105</td>
<td>0.6904</td>
<td>305981.33</td>
<td>7.5023761</td>
<td>15039</td>
<td>177412.38</td>
<td>11.79682</td>
</tr>
<tr>
<td>8</td>
<td>415</td>
<td>73</td>
<td>381824.49</td>
<td>25.462002</td>
<td>0.7787</td>
<td>619717.51</td>
<td>15.201077</td>
<td>14988</td>
<td>379581.39</td>
<td>25.125527</td>
</tr>
<tr>
<td>9</td>
<td>339</td>
<td>29</td>
<td>130135.3</td>
<td>9.666686</td>
<td>0.9302</td>
<td>196166.55</td>
<td>4.811927</td>
<td>14861</td>
<td>126603.79</td>
<td>6.4539553</td>
</tr>
<tr>
<td>10</td>
<td>525</td>
<td>38</td>
<td>121320.55</td>
<td>8.067599</td>
<td>0.9307</td>
<td>133451.33</td>
<td>4.7451759</td>
<td>15038</td>
<td>118119.31</td>
<td>7.8547719</td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>69</td>
<td>1209093.3</td>
<td>81.363328</td>
<td>0.2378</td>
<td>2149046.2</td>
<td>52.714994</td>
<td>14834</td>
<td>1205031.5</td>
<td>81.238356</td>
</tr>
<tr>
<td>12</td>
<td>123</td>
<td>85</td>
<td>721647.79</td>
<td>48.367185</td>
<td>0.5608</td>
<td>1229706.4</td>
<td>30.1895801</td>
<td>14816</td>
<td>720445.01</td>
<td>48.300146</td>
</tr>
<tr>
<td>13</td>
<td>585</td>
<td>17</td>
<td>87959.022</td>
<td>5.916452</td>
<td>0.9465</td>
<td>146969.95</td>
<td>3.5997997</td>
<td>14645</td>
<td>368160.365</td>
<td>5.9433237</td>
</tr>
<tr>
<td>14</td>
<td>601</td>
<td>26</td>
<td>115344.36</td>
<td>7.747919</td>
<td>0.9214</td>
<td>164713.81</td>
<td>4.0402671</td>
<td>12887</td>
<td>112644.32</td>
<td>7.5986232</td>
</tr>
<tr>
<td>15</td>
<td>385</td>
<td>33</td>
<td>144453.39</td>
<td>9.6276586</td>
<td>0.6125</td>
<td>244470.55</td>
<td>6.9965204</td>
<td>15004</td>
<td>142122.01</td>
<td>9.4722746</td>
</tr>
</tbody>
</table>

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.

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.
5.3 Bootstrap 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 Table 5.5 shows different datasets with different parameter values using Bootstrap Forest method.

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

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.

![Gradient-Boosted Trees Specification](image)

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.

![Boosted Tree for Y Specifications](image)

Figure: 5.17 Model Validation - Set Summaries of mv dataset

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

![Overall Statistics](image)

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.

![Actual by Predicted Plot](image1)

**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.

![Column Contributions](image2)

**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


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


