A Project report on

Enhancement of Decision Tree for Software Cost Estimation

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

in

Computer Science and Engineering

Submitted by

B. MANICHANDANA
(20H51A05N0)

M. LAYA
(20H51A05P4)

Under the esteemed guidance of

Dr. V. VENKATAIAH
Associate Professor of CSE
& Additional Controller of
Examination

Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING& TECHNOLOGY

(UGC Autonomous)

*Approved by AICTE *Affiliated to JNTUH *NAAC Accredited with A^+ Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2020- 2024

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Major Project report entitled **"Enhancement of Decision Tree for Software Cost Estimation"** being submitted by B. Manichandana (20H51A05N0), M. Laya (20H51A05P4) in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

Dr. V. Venkataiah	**Dr. Siva Skandha Sanagala**	**External Examiner**
Associate Professor of CSE & Additional controller of examination Dept. Of CSE	**Associate Professor & hod Dept. Of CSE**	

ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

We are grateful to **Dr. V Venkataiah,** Associate Professor & Additional controller of examination, Department of Computer Science and Engineering for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank **Dr. Siva Skandha Sanagala,** Head of the Department of Computer Science and Engineering, CMR College of Engineering and Technology, who is the major driving forces to complete my project work successfully.

We are very grateful to **Dr. Ghanta Devadasu**, Dean-Academics, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Major Dr. V A Narayana,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of Information Technology for their co-operation

We express our sincere thanks to **Shri. Ch. Gopal Reddy,** Secretary, CMR Group of Institutions, and **Shri Ch Abhinav Reddy,** CEO, CMR Group of Institutions for their continuous care and support.

Finally, We extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

B. Manichandana - 20H51A05N0

M. Laya - 20H51A05P4

TABLE OF CONTENTS

List of Figures

ABSTRACT

Effort estimation of the software is one of the biggest problems faced by software industry. It includes planning, supervising projects, analysing viability of the system development, preparing proposals and proposals presentation to clients and among this estimation of the effort for planning is one of the most critical responsibilities. It is necessary to have good effort estimation in order to conduct well budget. The accuracy of the effort estimation of software projects is vital for the competitiveness of software companies. For the forecasting of software effort, it is important to select the correct software effort estimation techniques. Numbers of have been proposed like algorithm, non-algorithm, Machine Learning algorithms and so on.

Generally, Decision Trees have drawn the attention of the researchers and changed the direction of Effort Estimation towards computational intelligence.in this project we are using decision tree concept which is a simple yet powerful approach and it can be taken as the base for Random Forest, which is known as a group of decision trees. Decision Trees facing data over fitting problem. To overcome this problem enhancement of Decision Tree with ensemble method proposed in this project and also for evaluate performance of this proposed method using COCOMO 81 dataset and to evaluation metrics like Mean Square Error (MSE) and Root Mean Square Error (RMSE). Comparative study shows that proposed technique, achieves better results than previous techniques.

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1. Problem Statement

In this project, the main aim is to improve software effort estimation by constructing a decision tree that makes use of several machine learning methods on software effort estimation datasets. In this project using COCOMO 81 dataset, we are estimating and calculate the accurate data as compare with other approaches. Software quality is driven by many aspects in software development. In software development organizations, the main aim is to start a project and finish it within acceptable schedule and budget. The main drivers affecting the budget are the number of employees and time. These two come together to form a measure called effort. Effort is the most important factor in a software project determining the cost. Estimating effort is a very important factor affecting the quality of software. At early stages of software development, effort must be estimated to come up with a planned schedule and budget. Traditional software cost estimation methods lack accuracy and reliability. The need for more effective and data-driven approaches is evident. That the reason we come up with this idea.

1.2. Research Objective

Enhancing software cost estimation precision involves utilizing advanced techniques like AdaBoost and pruning within Decision Trees, particularly when faced with limited project information. Compared to BASIC COCOMO, Decision Trees offer superior accuracy by capturing nuanced relationships in the data. AdaBoost combines weak learners to form a robust classifier, while pruning optimizes tree structure, reducing estimation errors. Evaluation using the COCOMO 81 dataset focuses on error reduction, aligning estimated and actual costs. However, beyond error reduction, factors such as interpretability, scalability, and robustness are crucial. Decision Trees provide transparency in estimations, aiding stakeholder understanding, and are scalable for projects of varying complexities. Integrating these factors aims not only to improve estimation precision but also to enhance overall decision-making and project management efficiency.

1.3. Project Scope

The scope of this project is to enhance the accuracy of software cost estimation utilizing decision tree (DT) algorithms, specifically addressing challenges associated with limited project information during early development stages. The project will focus on implementing enhancement techniques such as AdaBoost and pruning to mitigate issues such as overfitting and improve the predictive capabilities of the DT model. The COCOMO 81 dataset will be utilized for evaluation, providing a comprehensive benchmark for performance assessment.

1.4. Limitations

Dependency on the quality and completeness of the COCOMO 81 dataset: The effectiveness of the proposed enhancement techniques may be influenced by the quality and relevance of the dataset used for evaluation.

Simplified representation of software projects: Decision tree models inherently simplify complex relationships, which may overlook certain intricacies of software development projects.

Sensitivity to parameter tuning: The performance of AdaBoost and pruning techniques may vary based on parameter settings, requiring careful optimization.

Generalization to diverse project domains: The applicability of decision tree-based methods may vary across different software development domains and project types.

Computational complexity: Implementing ensemble methods like AdaBoost may introduce computational overhead, especially for large datasets, which could impact scalability.

CHAPTER 2
BACKGROUND WORK

CHAPTER 2
BACKGROUND WORK

2.1 Software Cost Estimation

2.1.1. Introduction

Software Cost Estimation (SCE) is a crucial step in software development, predicting how much effort, time, and money will be needed to finish a project. It's like planning how long and how expensive building a house will be before starting construction. SCE considers factors like the size of the project, measured in lines of code, and its complexity. This estimation helps in planning the project, deciding how many people are needed, and setting a budget. It's important because software projects can be unpredictable, and accurate estimation helps avoid running out of money or time. For example, SCE ensures that the cost of the project matches what the client expects. By providing a roadmap for the project's financial needs, SCE supports effective decision-making and successful project completion.

2.1.2. Merits

- **Budget Planning:** SCE allows for accurate budgeting by predicting the financial resources required for a software project, preventing overspending and financial surprises.
- **Resource Allocation:** It helps in allocating the right number of human resources, time, and tools needed for the project, ensuring optimal utilization and efficiency.
- **Risk Management:** SCE enables identifying potential risks and uncertainties early on, allowing for proactive risk management strategies to be implemented to mitigate them.
- **Decision Making:** It facilitates informed decision-making by providing stakeholders with realistic expectations regarding the project's cost and timeline.

2.1.3. Demerits

- **Uncertainty:** Software projects are inherently unpredictable, making it challenging to accurately estimate costs due to changing requirements, technology, and unforeseen obstacles.
- **Complexity:** SCE becomes more difficult with the complexity of modern software systems, which involve intricate architectures, integrations, and dependencies.
- **Subjectivity:** Estimations can be subjective and biased, influenced by factors such as individual expertise, assumptions, and external pressures.

2.1.4. Implementation of Software Cost Estimation:

The implementation of Software Cost Estimation (SCE) involves several steps and methodologies to predict the resources, time, and budget required for software development projects. Here's a simplified outline of the implementation process:

- **Data Collection:** Gather relevant data about past software projects, including project size (measured in Kilo Lines of Code or function points), complexity, development team capabilities, software tools used, and other relevant factors.
- **Select Estimation Model:** Choose an appropriate estimation model based on the nature of the project and available data. Common models include COCOMO (Constructive Cost Model), Function Point Analysis, and Use Case Points.
- **Parameter Identification:** Identify the parameters and factors that influence software cost estimation, such as project scope, requirements, technology stack, and risk factors.
- **Quantify Parameters:** Assign numerical values or weights to the identified parameters based on historical data and expert judgment. This step may involve normalization or scaling of data to ensure consistency.
- **Estimation Calculation:** Utilize the selected estimation model to calculate the effort, time, and cost required for the software development project based on the quantified parameters.
- Validation and Calibration: Validate the estimated values against actual project outcomes from historical data. Adjust the estimation model parameters if necessary to improve accuracy and reliability.

- **Iterative Refinement:** Continuously refine the estimation process based on feedback from previous projects and updated data. Incorporate lessons learned and new methodologies to enhance the accuracy of future estimations.
- **Documentation and Reporting:** Document the estimation process, assumptions made, and the rationale behind parameter selection. Present the estimated values along with associated risks and uncertainties to stakeholders for decision-making.
- **Monitoring and Control:** Monitor actual project progress and compare it with estimated values regularly. Implement corrective actions if deviations occur to ensure project objectives are met within the defined constraints.
- **Continuous Improvement:** Foster a culture of continuous improvement by analyzing estimation accuracy, identifying areas for enhancement, and updating estimation models and processes accordingly.

Figure 1: System architecture of Software Cost Estimation

2.2 Software Cost Estimation Using Decision Tree

2.2.1. Introduction

Software Cost Estimation (SCE) employing Decision Trees is a methodological approach that leverages decision tree algorithms to predict the effort, time, and financial resources required for software development projects. Decision trees, esteemed for their simplicity and interpretability, serve as predictive models in this context. By analyzing historical project data, such as project size, complexity, and development resources, decision tree algorithms learn patterns and relationships to make accurate predictions about future software endeavors. This approach offers a pragmatic and efficient strategy for estimating software costs, aiding in informed decision-making and resource allocation throughout the software development lifecycle.

2.2.2. Merits

- **Interpretability:** Decision trees offer transparency and ease of understanding, allowing stakeholders to comprehend the factors influencing cost estimates.
- **Flexibility:** Decision trees can handle both numerical and categorical data, making them adaptable to various types of software projects.
- **Scalability:** Decision tree models can be scaled to accommodate large datasets, enabling efficient analysis of extensive project information.

2.2.3. Demerits

- **Overfitting:** Decision trees are prone to overfitting, especially with complex datasets, leading to inaccurate predictions on unseen data.
- **Sensitivity to Data:** Decision trees are sensitive to variations in the training data, which can result in different tree structures and outcomes.
- **Lack of Continuity:** Decision trees make discrete predictions, which may not capture the continuous nature of software cost estimation accurately.

2.2.4. Implementation of Software Cost Estimation using Decision Tree:

- **Data Collection:** Gather historical data on software development projects, including attributes like project size, complexity, development time, and actual costs.
- **Data Preprocessing:** Clean the data by handling missing values, outliers, and inconsistencies. Convert categorical variables into numerical format if needed.
- **Feature Selection:** Identify the most relevant features that significantly impact software development costs. This step helps improve the accuracy and efficiency of the decision tree model.
- **Model Training:** Split the dataset into training and testing sets. Train the decision tree model on the training data, using algorithms like CART (Classification and Regression Trees) or ID3 (Iterative Dichotomiser 3).
- **Model Evaluation:** Assess the performance of the decision tree model using evaluation metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or Mean Absolute Error (MAE). Validate the model's effectiveness in predicting software development costs.
- **Deployment:** Once the decision tree model demonstrates satisfactory performance, deploy it in real-world scenarios for software cost estimation. Continuously monitor and refine the model to enhance its accuracy over time.

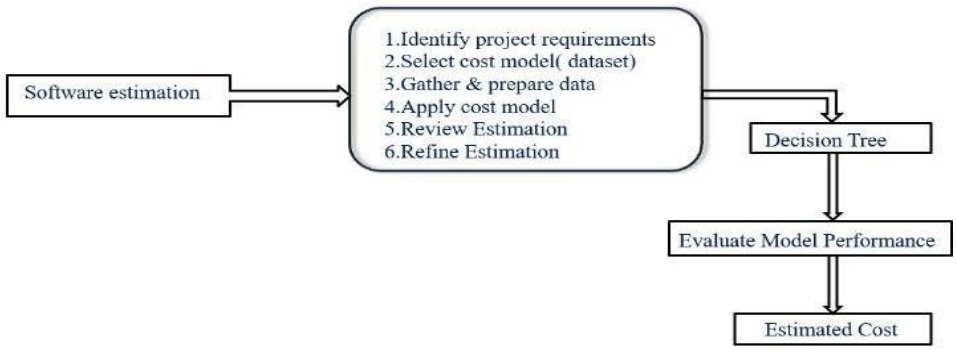


Figure 2: Block Diagram of Software Cost Estimation using Decision Tree

2.3 Enhancing Software Cost Estimation with COCOMO81 Data

2.3.1. Introduction

Software cost estimation is vital for effective project planning and resource allocation in software development. Leveraging the COCOMO81 dataset, this study aims to enhance the accuracy of cost estimation methodologies. COCOMO81 provides essential attributes like project size, complexity, and resource requirements, facilitating a comprehensive analysis. By utilizing advanced techniques such as decision trees, this research endeavors to refine cost estimation models. Improving accuracy in software cost estimation can mitigate budget overruns and project delays, enhancing overall project management effectiveness. This introduction sets the stage for exploring how the COCOMO81 dataset can be utilized to enhance software cost estimation methodologies.

2.3.2. Merits

- **Comprehensive Attributes:** COCOMO81 dataset offers a wide range of attributes, including project size, complexity, and resource requirements, providing a detailed insight into software development projects.
- **Established Framework:** COCOMO81 follows a well-defined framework for estimating software development effort and cost, providing a structured approach that has been widely used and tested in the industry.
- **Historical Data:** The dataset includes historical data from various software projects, enabling benchmarking and comparison to improve the accuracy of cost estimation.

2.3.3. Demerits

- **Limited Scope:** COCOMO81 may not capture all factors influencing software development costs, leading to potential inaccuracies in estimation, especially for projects with unique characteristics.
- **Assumption-Based:** Like any estimation model, COCOMO81 relies on certain assumptions about project characteristics and development processes, which may not always align with real-world scenarios.

2.3.4. COCOMO 81 Dataset:

rely	data	cplx	time	stor	virt	turn	acap	aexp	pcap	vexp	lexp	modp	tool	sced	loc	actual
0.88	1.16	0.7	1	1.06	1.15	1.07	1.19	1.13	1.17	1.1	1	1.24	1.1	1.04	113	2040
0.88	1.16	0.85	1	1.06	1	1.07	1	0.91	1	0.9	0.95	1.1	1	1	293	1600
1	1.16	0.85	1	1	0.87	0.94	0.86	0.82	0.86	0.9	0.95	0.91	0.91	1	132	243
0.75	1.16	0.7	1	1	0.87	1	1.19	0.91	1.42	1	0.95	1.24	1	1.04	60	240
0.88	0.94	1	1	1	0.87	1	1	1	0.86	0.9	0.95	1.24	1	1	16	33
0.75	1	0.85	1	1.21	1	1	1.46	1	1.42	0.9	0.95	1.24	1.1	1	4	43
0.75	1	1	1	1	0.87	0.87	1	1	1	0.9	0.95	0.91	0.91	1	6.9	8
1.15	0.94	1.3	1.66	1.56	1.3	1	0.71	0.91	1	1.21	1.14	1.1	1.1	1.08	22	1075
1.15	0.94	1.3	1.3	1.21	1.15	1	0.86	1	0.86	1.1	1.07	0.91	1	1	30	423
1.4	0.94	1.3	1.11	1.56	1	1.07	0.86	0.82	0.86	0.9	1	1	1	1	29	321
1.4	0.94	1.3	1.11	1.56	1	1.07	0.86	0.82	0.86	0.9	1	1	1	1	32	218
1.15	0.94	1.3	1.11	1.06	1	1	0.86	0.82	0.86	1	0.95	0.91	1	1.08	37	201
1.15	0.94	1.3	1.11	1.06	1.15	1	0.71	1	0.7	1.1	1	0.82	1	1	25	79
1.15	0.94	1.65	1.3	1.56	1.15	1	0.86	1	0.7	1.1	1.07	1.1	1.24	1.23	3	60
1.4	0.94	1.3	1.3	1.06	1.15	0.87	0.86	1.13	0.86	1.21	1.14	0.91	1	1.23	3.9	61
1.4	1	1.3	1.3	1.56	1	0.87	0.86	1	0.86	1	1	1	1	1	6.1	40
1.4	1	1.3	1.3	1.56	1	0.87	0.86	0.82	0.86	1	1	1	1	1	3.6	9
1.15	1.16	1.15	1.3	1.21	1	1.07	0.86	1	1	1	1	1.24	1.1	1.08	320	11400
1.15	1.08	1	1.11	1.21	0.87	0.94	0.71	0.91	1	1	1	0.91	0.91	1	1150	6600
1.4	1.08	1.3	1.11	1.21	1.15	1.07	0.71	0.82	1.08	1.1	1.07	1.24	1	1.08	299	6400
1	1.16	1.15	1.06	1.14	0.87	0.87	0.86	1	1	1	1	0.91	0.91	1	252	2455
1.15	1	1	1.27	1.06	1	1	0.86	0.82	0.86	0.9	1	0.91	1	1.23	118	724
1.15	1	1	1.08	1.06	1	1	0.86	0.82	0.86	0.9	1	1	1	1.23	77	539
0.88	1	0.85	1.06	1.06	1	0.87	1	1.29	1	1.1	0.95	0.82	0.83	1	90	453
1.15	1.16	1.3	1.15	1.06	1	0.87	0.86	1	0.86	1.1	1	0.82	0.91	1.08	38	523
0.94	1	0.85	1.07	1.06	1.15	1.07	0.86	1	0.86	1.1	1	0.91	1.1	1.08	48	387
1.15	0.94	1.15	1.35	1.21	1	0.87	1	1	1	1	1	0.82	1.1	1.08	9.4	88
1.15	1.08	1.3	1.11	1.21	1.15	1.07	0.86	1	0.86	1.1	1.07	1.1	1.1	1	13	98
0.88	1	1	1	1	1	1	1.1	1.29	0.86	1	1	0.91	0.91	1.23	2.14	7.3
0.88	1	1	1	1	1	1	1	1.29	0.86	1	1	0.91	0.91	1.23	1.98	5.9
1.4	1.08	1	1.48	1.56	1.15	1.07	0.86	0.82	0.86	1.1	1.07	1	1	1	62	1063
0.88	1.08	0.85	1	1	1	1	0.71	0.82	1	1	1	1.1	1.1	1	390	702
1.4	1.08	1.3	1.48	1.56	1.15	0.94	0.86	0.82	0.86	0.9	1	0.91	0.91	1	42	605
1.15	1.08	1	1.06	1	1	0.87	1	1	1	1	1	0.91	1.1	1.23	23	230
0.75	0.94	1.3	1.06	1.21	1.15	1	1	0.91	1	1.1	1	1.24	1.24	1	13	82
0.88	1.05	0.81	1	1	0.87	0.87	1.19	1	1.17	0.9	0.95	1	0.91	1.04	15	55
0.88	0.94	0.7	1	1.06	1	1	0.86	0.82	0.86	1	1	1	1	1	60	47
1	1	1.15	1	1	0.87	0.87	0.71	0.91	1	0.9	0.95	0.82	0.91	1	15	12
1	1	1.15	1	1	0.87	1	0.71	0.82	0.7	1	0.95	0.91	1.1	1	6.2	8
1	0.94	1.3	1	1	1	0.87	0.86	0.82	1.17	1	1	1.1	1	1	3	8
0.88	0.94	1	1	1	0.87	0.87	1	0.82	0.7	0.9	0.95	0.91	0.91	1	5.3	6
0.88	1.04	1.07	1	1.06	0.87	1.07	0.86	1	0.93	0.9	0.95	0.95	0.95	1.04	45.5	45
1	1.04	1.07	1	1.21	0.87	1.07	0.86	1	1	0.9	0.95	1	1	1.04	28.6	83
0.88	1.04	1.07	1.06	1.21	0.87	1.07	1	1	1	0.9	0.95	1.1	1	1.04	30.6	87
0.88	1.04	1.07	1	1.06	0.87	1.07	1	1	1	0.9	0.95	1	0.95	1.04	35	106
0.88	1.04	1.07	1	1.06	0.87	1.07	1	1	0.86	0.9	0.95	1	1	1.04	73	126
0.75	0.94	1.3	1	1	0.87	0.87	0.71	0.82	0.7	1.1	1.07	1.1	1	1.04	23	36
0.88	0.94	0.85	1	1	0.87	1	1.19	0.91	1.17	0.9	0.95	1.1	1	1.04	464	1272
1	1	0.85	1	1	1	0.87	0.71	1	0.7	1.1	1	0.82	0.91	1	91	156
1.15	1	1	1.3	1.21	1	0.87	0.86	1	0.86	1.1	1	1	1	1	24	176
0.88	1	1	1	1	1	1.15	1.19	1	1.42	1	0.95	1.24	1.1	1.04	10	122
0.88	0.94	0.85	1	1.06	1.15	1	1	1	1	1.1	1.07	1.24	1.1	1	8.2	41
0.88	0.94	1.15	1.11	1.21	1.3	1	0.71	1	0.7	1.1	1.07	1	1.1	1.08	5.3	14
1	0.94	1	1	1.06	1.15	0.87	1	0.82	1	1	0.95	0.91	1.1	1	4.4	20
0.88	0.94	0.7	1	1	0.87	0.87	0.86	0.82	1.17	0.9	0.95	1.1	1	1	6.3	18
1.15	0.94	1.3	1.3	1.21	1	1	0.86	0.91	1	1.1	1.07	1.1	1.1	1.08	27	958
1	0.94	1.15	1.11	1.21	1.3	1	1	1	1	1.1	1.07	1.1	1.1	1.23	17	237
1.4	0.94	1.3	1.66	1.21	1	1	0.71	0.82	0.7	0.9	0.95	0.91	1	1	25	130
1	0.94	1.15	1.06	1.06	1	0.87	1	1	1	1	1	0.91	1	1	23	70
1.15	0.94	1.3	1.11	1.06	1	1	0.86	1.13	0.86	1.1	1.07	1.1	1.1	1.08	6.7	57
1	0.94	1.15	1	1	0.87	0.87	0.86	1	0.86	0.9	1	0.82	1	1	28	50
0.88	0.94	1.3	1.11	1.21	1.15	1	0.78	0.82	0.7	1.21	1.14	0.91	1.24	1	9.1	38
1	0.94	1.15	1	1	1	0.87	0.71	0.82	0.86	1	1	0.82	1	1	10	15

Figure 3: COCOMO 81 Dataset

2.4. Essentials of Decision Trees: A Fundamental Overview

2.4.1. Introduction

A decision tree is a flowchart-like tree structure where each internal node denotes the feature, branches denote the rules and the leaf nodes denote the result of the algorithm. It is a versatile supervised machine-learning algorithm, which is used for both classification and regression problems. It is one of the very powerful algorithms. And it is also used in Random Forest to train on different subsets of training data, which makes random forest one of the most powerful algorithms in machine learning.

2.4.2. Fundamentals of Decision Tree: Decision trees are a foundational concept in machine learning and data analysis, offering a systematic approach to decision-making and prediction. At their core, decision trees organize data into a hierarchical structure resembling inverted trees. Each internal node in the tree represents a decision based on a feature attribute, while each branch represents an outcome of that decision. Ultimately, the tree leads to leaf nodes, which provide the final decision or prediction. This structure makes decision trees intuitive and easy to interpret, as the decision-making process follows a clear path from the root to the leaves.

A key aspect of decision trees is the selection of splitting criteria, which determines how the dataset is divided at each node. Various criteria, such as Gini impurity and information gain, are used to maximize the homogeneity of resulting subsets. Additionally, decision trees employ feature selection algorithms to identify the most informative attributes for optimal splitting. As the tree grows, decisions are made recursively based on these criteria, leading to the creation of a predictive model.

While decision trees offer interpretability and simplicity, they also face challenges. Overfitting is a common issue, where the tree captures noise in the training data, leading to poor generalization. Techniques like pruning are used to address overfitting by simplifying the tree. Furthermore, decision trees may struggle with complex datasets or those with high dimensionality, requiring advanced algorithms or ensemble methods for improved performance.

2.4.3. Introduction to decision tree terminology: Decision tree terminology refers to the fundamental concepts and terms used to describe the structure and functioning of decision trees in machine learning. It includes terms like root node, internal node, and leaf node, which define the key components of a decision tree. Understanding these terms is essential for interpreting and building decision tree models effectively.

- **Root Node:** The topmost node in a decision tree from which the tree originates. It represents the entire dataset and is used to make the initial split based on a selected feature.
- **Internal Node:** Nodes within the decision tree that are not leaf nodes. Internal nodes represent decision points where the dataset is split into subsets based on specific feature values.
- **Leaf Node:** Also known as terminal nodes, leaf nodes are the endpoints of a decision tree where the final prediction or classification is made. Each leaf node corresponds to a class label or a regression value.
- **Splitting Criterion:** The criteria used to divide the dataset at each internal node of the decision tree. It is based on features such as entropy, Gini impurity, or information gain, aiming to maximize homogeneity within subsets.
- **Decision Path:** The sequence of decisions made from the root node to a leaf node in a decision tree, determining the classification or prediction for a particular instance.
- **Pruning:** The process of removing branches or subtrees from a decision tree to prevent overfitting and improve generalization performance on unseen data.

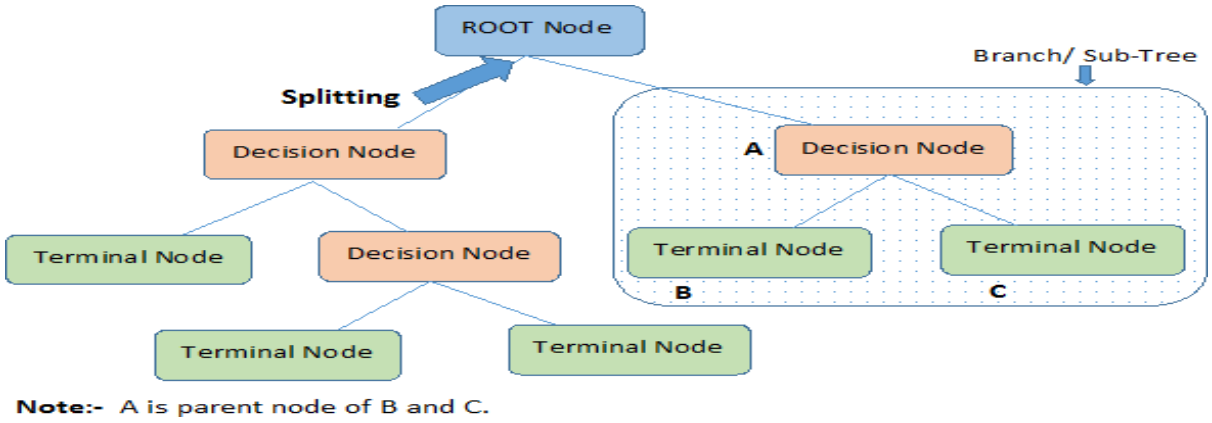


Figure 4: Example of Decision Tree

2.4.4. Merits

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

2.4.5. Demerits

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
- For more class labels, the computational complexity of the decision tree may increase.

2.4.6. Applications of Decision Tree: Decision Tree is one of the basic and widely-used algorithms in the fields of Machine Learning. It's put into use across different areas in classification and regression modelling. Due to its ability to depict visualized output, one can easily draw insights from the modelling process flow. Here are a few examples wherein Decision Tree could be used,

- Business Management
- Customer Relationship Management
- Fraudulent Statement Detection
- Energy Consumption
- Healthcare Management
- Fault Diagnosis

2.5. Essential Metrics for Software Cost Estimation

2.5.1. Introduction

Effective software cost estimation is crucial for project planning and management in the realm of software development. Essential metrics play a vital role in this process by providing valuable insights into project complexity, resource requirements, and development effort. Metrics such as lines of code, function points, and complexity measures are commonly used to quantify project size and intricacy. By leveraging these metrics, organizations can make informed decisions, allocate resources efficiently, and mitigate risks associated with cost overruns and project delays. This document explores the significance and application of essential metrics in software cost estimation, highlighting their importance in ensuring project success.

2.5.2. Evaluating Metrics

Mean Square Error (MSE): Mean Square Error (MSE) is another widely used metric in evaluating predictive models, particularly in regression analysis. It measures the average squared difference between the predicted values and the actual values in a dataset. MSE provides a measure of the variability or dispersion of the errors in the predictions.

The formula for calculating MSE is similar to that of RMSE:

$$\mathbf{MSE = \Sigma (yi - \hat{y}i)^2 / n}$$

Here, yi represents the actual values in the dataset, ŷi represents the predicted values generated by the model, and n is the total number of data points.

To compute MSE:

1. Calculate the squared difference between each actual value (yi) and its corresponding predicted value (ŷi).
2. Sum up all these squared differences.
3. Divide the sum by the total number of data points (n).

MSE is useful for comparing the performance of different models or assessing the improvement of a model over time. Like RMSE, a lower MSE value indicates better predictive performance, while a higher MSE suggests poorer performance. MSE is often used alongside RMSE to provide a comprehensive evaluation of a model's accuracy and effectiveness in making predictions.

Root Mean Square Error (RMSE) : Root Mean Square Error is a commonly used metric for evaluating the accuracy of a predictive model, particularly in the context of regression analysis. It measures the average magnitude of the errors between predicted values and actual values in a dataset. RMSE is a popular choice because it penalizes larger errors more heavily than smaller ones, making it sensitive to outliers.

The formula for calculating RMSE is as follows:

RMSE = $\sqrt{(\Sigma (yi - \hat{y}i)^2 / n)}$

Where:

- yi represents the actual values in the dataset.
- $\hat{y}i$ represents the predicted values generated by your model.
- n is the total number of data points in the dataset.

Here's how you can calculate RMSE step by step:

1. For each data point in your dataset, calculate the squared difference between the actual value (yi) and the predicted value ($\hat{y}i$).
2. Sum up all these squared differences.
3. Divide the sum by the total number of data points (n) to calculate the mean squared error.
4. Take the square root of the mean squared error to obtain the RMSE.

A lower RMSE value indicates that the model's predictions are closer to the actual values, suggesting better predictive performance. Conversely, a higher RMSE value implies that the model's predictions are further from the actual values, indicating poorer predictive performance. RMSE is often used in the fields of machine learning, statistics, and data analysis to assess the quality of regression models, such as linear regression, decision trees, and neural networks. It provides a single value that summarizes how well a model fits the data, making it a useful tool for model selection and evaluation.

2.6 Software Cost Estimation Using Ant Colony Optimization

2.6.1. Introduction

In contemporary software development, accurate cost estimation is paramount for effective project planning and resource allocation. Various machine learning techniques, including Decision Trees, Support Vector Machines, and Neural Networks, have been applied to this end. However, there remains ample opportunity for optimization. This paper proposes the utilization of Ant Colony Optimization (ACOT) to predict software cost estimation, leveraging datasets from literature sources such as ISBSG, IBMDPS, and COCOMO 81. By employing ACOT, this study aims to enhance predictive accuracy and efficiency in software cost estimation.

2.6.2. Merits

Integration of Ant Colony Optimization offers a novel approach to software cost estimation.

- Utilization of multiple datasets ensures comprehensive evaluation and validation.
- Comparative analysis against existing techniques provides insights into the effectiveness of ACOT.

2.6.3. Demerits

- Complexity of ACOT implementation may require specialized expertise.
- Limited scope for customization compared to more flexible machine learning algorithms.
- Performance heavily dependent on parameter tuning and dataset characteristics.

2.6.4. Implementation of Software Cost Estimation using Ant Colony Optimization:

Data Collection and Preprocessing: Collect datasets from literature sources, including ISBSG, IBMDPS, and COCOMO 81. Perform data cleaning to handle missing values and outliers. Normalize features using techniques such as min-max scaling to ensure uniformity.

Ant Colony Optimization Algorithm: Implement the ACOT algorithm using a suitable programming language like Python or Java. Define parameters such as alpha, beta, pheromone persistence, and initial pheromone levels. Design the ant movement rules, including probability calculations for selecting next moves.

Model Training and Testing: Split the dataset into training and testing sets, typically using techniques like k-fold cross-validation. Train the ACOT model on the training set, adjusting parameters to optimize performance. Validate the model's performance on the testing set to ensure generalization.

Performance Evaluation: Compute evaluation metrics such as Root Mean Square Error (RMSE) and Mean Square Error (MSE) to quantify the accuracy of the model's predictions. Compare the performance of the ACOT model against baseline methods or other machine learning algorithms.

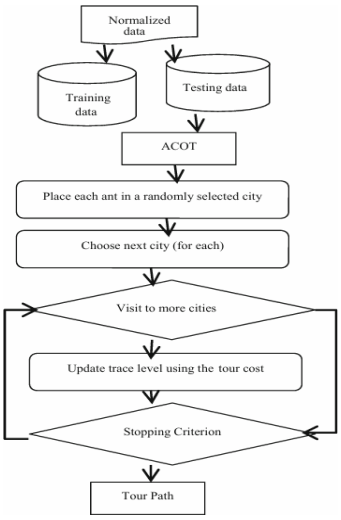


Fig5.Flow chart for testing phase of ACOT

CHAPTER 3
PROPOSED SYSTEM

CHAPTER 3

PROPOSED SYSTEM

3.1. Objective of Proposed Model

The objective of the proposed model is to enhance the accuracy of software cost estimation (SCE) using Decision Trees (DT) in conjunction with AdaBoost and pruning techniques. By leveraging the COCOMO 81 dataset attributes related to software projects' reliability, data complexity, time constraints, and storage requirements, the model aims to refine and improve the predictive capabilities of DT for SCE. The primary goal is to reduce errors in cost estimation by applying advanced machine learning techniques while maintaining simplicity and interpretability. Through the evaluation metrics of Root Mean Square Error (RMSE) and Mean Square Error (MSE), the model seeks to demonstrate significant improvements in accuracy compared to traditional approaches. Ultimately, the objective is to provide a more reliable and effective method for estimating software development costs, thereby aiding in project planning and resource allocation.

3.2. Algorithms used for Proposed Model

The proposed model integrates Decision Trees, AdaBoost, and pruning techniques to enhance software cost estimation accuracy, addressing challenges in project planning. These algorithms collectively refine predictive capabilities, offering a robust solution for reliable software development effort predictions.

[1] Decision Trees (DT): Decision Trees are a popular machine learning algorithm used for both classification and regression tasks. In the context of software cost estimation, Decision Trees are employed to create a predictive model that learns from the data by recursively partitioning it into subsets based on the values of input features. At each step, the algorithm selects the feature that best separates the data, aiming to maximize predictive accuracy. Decision Trees offer advantages such as interpretability, ease of visualization, and the ability to handle both numerical and categorical data.

[2] AdaBoost (Adaptive Boosting): AdaBoost is an ensemble learning technique that combines multiple weak learners, typically Decision Trees, to create a strong predictive model. In the context of software cost estimation, AdaBoost is applied to improve the performance of Decision Trees by iteratively training a series of Decision Trees, with each subsequent tree focusing on correcting the errors made by its predecessors. By giving more weight to misclassified instances, AdaBoost helps the model adapt and learn from its mistakes, ultimately enhancing predictive accuracy.

[3] Pruning: Pruning is a technique used to prevent overfitting in Decision Trees by removing unnecessary branches and nodes from the tree structure. Overfitting occurs when the model captures noise or irrelevant patterns in the training data, leading to poor generalization performance on unseen data. Pruning helps simplify the Decision Tree model by eliminating branches that do not significantly contribute to predictive accuracy. By reducing the complexity of the tree, pruning improves its ability to generalize to new data and enhances overall model performance.

3.3. Designing

3.3.1. Block Diagram

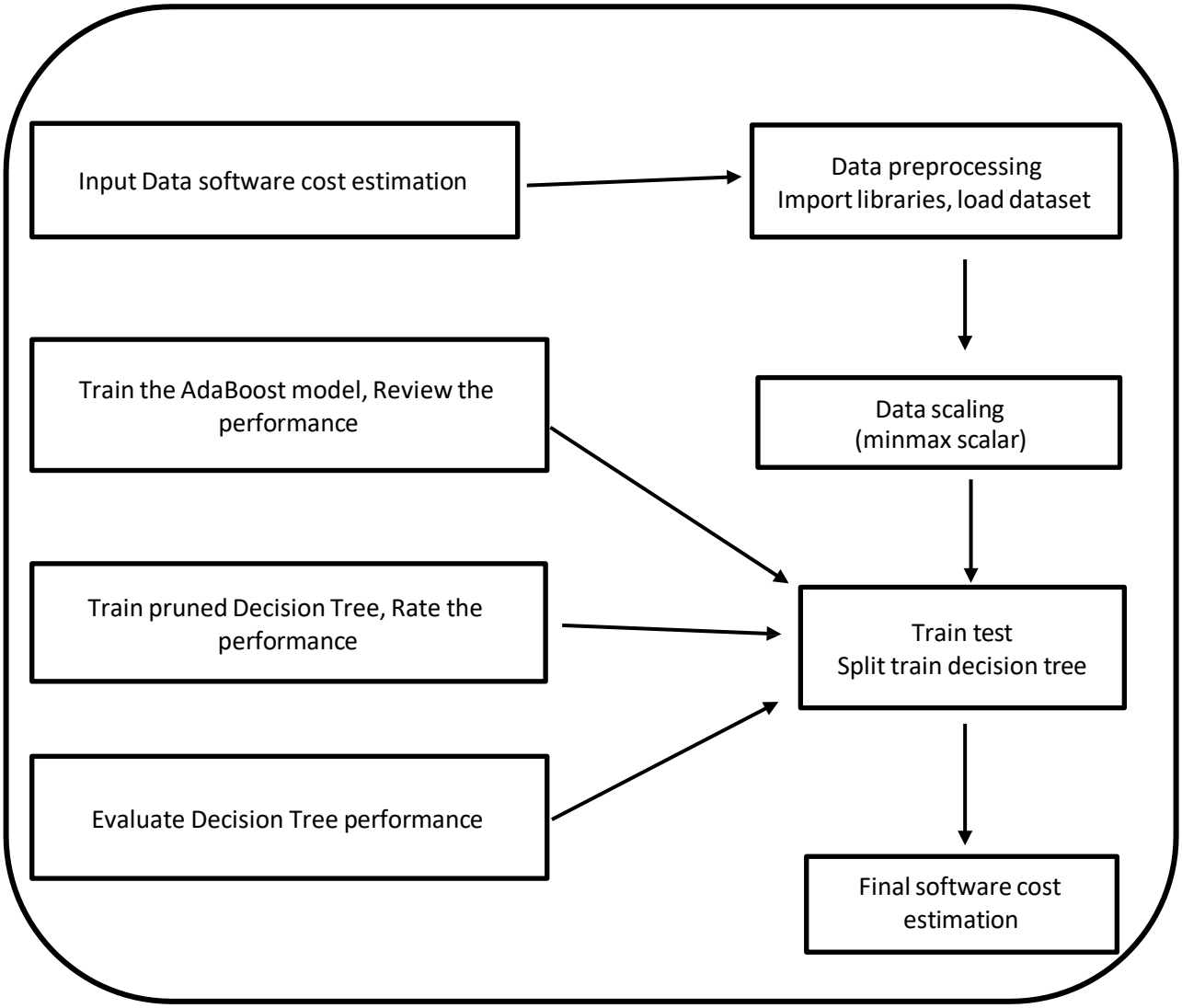


Fig6. Block Diagram of Enhancement of Decision Tree for SCE

3.4. Implementation

Input Data: The input data consists of software cost estimation data, which includes attributes such as reliability, data complexity, time constraints, and storage requirements. The dataset is pre-processed to handle any missing values or outliers.

Data Preprocessing: The data preprocessing step involves importing necessary libraries such as pandas, NumPy, and scikit-learn. The data is then scaled using MinMaxScaler to ensure all features fall within a uniform range of 0 to 1.

Train and Test Split: The dataset is split into training and testing sets using the train_test_split function from scikit-learn. This separation is crucial for training the model on one subset and evaluating its performance on another unseen subset.

Decision Tree Training: A Decision Tree Regressor model is trained on the training set to learn patterns and relationships within the data. The trained model is then used to make predictions on the testing set, and performance metrics such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are computed to assess its effectiveness.

AdaBoost Model Training: An AdaBoost Regressor is trained on the training set, utilizing the Decision Tree Regressor as a base estimator. The AdaBoost model's performance is evaluated by making predictions on the test set and computing performance metrics like MSE and RMSE.

Pruned Decision Tree Training: A pruned Decision Tree Regressor with limited depth is trained on the training set to prevent overfitting and simplify the model. The performance of the pruned Decision Tree is evaluated on the test set, and performance metrics (MSE and RMSE) are calculated.

Evaluation of Decision Tree Performance: The performance of the Decision Tree model is assessed separately, including MSE and RMSE calculations, to compare it with the AdaBoost and pruned Decision Tree models.

Final Software Cost Estimation: The final output showcases the error evaluation before and after enhancements of the Decision Tree model, specifically addressing the reduction of overfitting. The analysis demonstrates how the model of decision trees is refined and improved, ultimately enhancing its predictive capabilities for software cost estimation.

3.5. Code

Import Libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.tree import export_text
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score
```

- These lines import necessary libraries and modules for data manipulation, machine learning modeling, and evaluation.

Read Dataset

```
data = pd.read_csv("C:\\jupyter-notebook\\cocomo81.csv")
```

- Reads the CSV file named "cocomo81.csv" into a pandas DataFrame called `data`.

Obtain No. of rows and columns

```
num_rows, num_columns = data.shape
print("Number of rows:", num_rows)
print("Number of columns:", num_columns)
print("Column names:", data.columns)
```

- Obtains the number of rows and columns in the dataset and prints them, along with the column names.

Apply Normalization

scaler = MinMaxScaler()

data_scaled = scaler.fit_transform(data)

data_scaled_df = pd.DataFrame(data_scaled, columns=data.columns)

- Initializes a MinMaxScaler object to scale the data between 0 and 1. Fits the scaler to the data and transforms it. Finally, creates a DataFrame `data_scaled_df` with the scaled data.

Drop Effort

data_scaled_df.columns = data_scaled_df.columns.str.strip()

- Removes leading and trailing whitespaces from the column names.

target_column = 'Effort1'

- Defines the target column name.

X = data_scaled_df.drop(columns=['Effort1'])

Y = data_scaled_df['Effort1']

Split the Data

- Splits the DataFrame into features (X) and the target variable (Y).

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

- Splits the data into training and testing sets with 80% for training and 20% for testing, using a random state for reproducibility.

Initialize Decision Tree Regressor

dt_model = DecisionTreeRegressor(random_state=42)

dt_model.fit(X_train, Y_train)

Y_pred = dt_model.predict(X_test)

- Initializes a Decision Tree Regressor model, fits it to the training data, and predicts the target variable for the test data.

tree_regressor = DecisionTreeRegressor(max_depth=20, min_samples_split=2, random_state=42)

tree_regressor.fit(X_train, Y_train)

Y_pred = tree_regressor.predict(X_test)

Calculate MSE & RMSE for Decision Tree

```
mse = np.mean((Y_test - Y_pred)**2)
rmse = np.sqrt(np.mean((Y_test - Y_pred)**2))
print(f"MSE: {mse}")
print(f"RMSE: {rmse}")
```

- Calculates mean squared error (MSE) and root mean squared error (RMSE) between the predicted and actual values, then prints them.

Initialize AdaBoost Technique

```
base_regressor = DecisionTreeRegressor(max_depth=5, random_state=42)
adaboost_regressor = AdaBoostRegressor(base_regressor, n_estimators=1, random_state=42)
Y_train = Y_train.ravel()
adaboost_regressor.fit(X_train, Y_train)
y_pred_boosted = adaboost_regressor.predict(X_test)
```

- Initializes a base Decision Tree Regressor and an AdaBoost Regressor using it. Fits the AdaBoost model to the training data and predicts the target variable for the test data.

Calculate MSE & RMSE for AdaBoost

```
mse = np.mean((Y_test - y_pred_boosted)**2)
rmse = np.sqrt(np.mean((Y_test - y_pred_boosted)**2))
print(f"MSE: {mse}")
print(f"RMSE: {rmse}")
```

- Calculates MSE and RMSE between the predicted and actual values for the AdaBoost model, then prints them.

Initialize Pruned Decision Tree

```
pruned_dt_regressor = DecisionTreeRegressor(max_depth=2 ,random_state=42)
pruned_dt_regressor.fit(X_train, Y_train)
Y_pred_pruned = pruned_dt_regressor.predict(X_test)
```

- Initializes and fits a pruned Decision Tree Regressor with specified hyperparameters to the training data and predicts the target variable for the test data.

Calculate MSE & RMSE for Pruned Decision Tree

```
mse = np.mean((Y_test - Y_pred_pruned)**2)
rmse = np.sqrt(np.mean((Y_test - Y_pred_pruned)**2))
print(f"MSE: {mse}")
print(f"RMSE: {rmse}")
```

- Calculates MSE and RMSE between the predicted and actual values for the pruned Decision Tree model, then prints them.

The provided code performs regression analysis on software effort estimation data using decision tree-based models. It first preprocesses the data, scales it, and splits it into training and testing sets. Three decision tree regressors are trained: one with default parameters, one with boosted AdaBoost, and one pruned for simplicity. The code evaluates each model's performance using Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) metrics. Finally, it prints the MSE and RMSE values for each model. Overall, the code demonstrates the application of decision tree regressors and their variants in software effort estimation, comparing their performance.

CHAPTER 4
RESULTS AND DISCUSSION

CHAPTER 4

RESULTS AND DISCUSSION

The results of applying AdaBoost and pruning techniques to Decision Trees for software cost estimation (SCE) demonstrate notable improvements in accuracy. AdaBoost, by enhancing the performance of Decision Trees, achieved a remarkable reduction in both Root Mean Square Error (RMSE) and Mean Square Error (MSE). After AdaBoost application, the RMSE decreased to 0.043 and the MSE to 0.0019, showcasing the effectiveness of boosting in refining the predictive capability of the model. On the other hand, pruning, which simplifies Decision Trees to mitigate overfitting, also led to a reduction in errors, albeit to a slightly lesser extent compared to AdaBoost. Post-pruning, the RMSE was observed at 0.050, with an MSE of 0.0025.

The comparison between the two techniques highlights AdaBoost as the more effective approach in this SCE context. Its ability to significantly enhance the accuracy of Decision Trees suggests its suitability for improving SCE models. However, the choice between AdaBoost and pruning should consider additional factors beyond just error reduction. Factors such as computational complexity, interpretability of the model, and specific objectives of SCE must be taken into account. While AdaBoost offers superior performance, it may introduce higher computational overhead. Pruning, on the other hand, simplifies the model, making it more interpretable but may not achieve the same level of accuracy improvement as AdaBoost. Thus, the decision on which technique to employ should be based on a careful consideration of these trade-offs and alignment with the goals of the SCE process. Overall, the results underscore the importance of advanced techniques like AdaBoost in enhancing the accuracy of SCE models while emphasizing the need for a nuanced approach considering various practical constraints and objectives.

4.1. Performance metrics:

The software cost estimation (SCE) project leverages Decision Tree-based machine learning models and investigates the impact of enhancing these models using ensemble methods such as AdaBoost and pruning techniques. Utilizing the COCOMO 81 dataset, the models' performance is evaluated based on Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) metrics.

Decision Tree (Before Enhancement)

The initial Decision Tree Model serves as a baseline for SCE, providing a starting point for comparison. With a RMSE of 0.051, this model demonstrates moderate accuracy in predicting software development effort.

AdaBoost-Enhanced Decision Tree Model

MSE Reduction: The introduction of AdaBoost results in a significant reduction in MSE, dropping from 0.0026 to 0.0019. This suggests that the AdaBoost-enhanced model produces more accurate predictions compared to the baseline.

RMSE Reduction: Correspondingly, the RMSE experiences a substantial decrease, falling from 0.051 to 0.044. This reduction indicates an improvement in the model's ability to generalize to unseen data.

Accuracy Improvement: The observed percentage improvement in RMSE, approximately 14.70%, highlights the efficacy of AdaBoost in enhancing predictive accuracy. This enhancement is crucial for achieving more reliable SCE outcomes.

Pruned Decision Tree Model:

MSE Reduction: The pruned decision tree model shows a slight decrease in MSE, from 0.0026 to 0.0025. While the reduction is not as substantial as with AdaBoost, it still contributes to improved model performance.

RMSE Reduction: Similarly, the RMSE experiences a modest decrease, dropping from 0.051 to 0.050. This indicates that pruning helps mitigate overfitting, leading to more generalizable predictions.

Accuracy Improvement: The percentage improvement in RMSE is approximately 1.93%, indicating a more modest improvement compared to AdaBoost. However, pruning still plays a valuable role in simplifying the model while maintaining competitive performance.

Observation.

AdaBoost Effienecy: AdaBoost emerges as a highly effective technique for enhancing Decision Tree models in SCE. By iteratively improving the model's performance, AdaBoost significantly reduces errors and enhances predictive accuracy.

Pruning Rule: Pruning, although resulting in a less complex model, still contributes to improved performance. This suggests that simplifying the model architecture can lead to more interpretable results without sacrificing accuracy.

Consideration: When selecting between AdaBoost and pruning techniques, factors such as computational complexity and specific SCE objectives must be considered. While AdaBoost offers substantial improvements, pruning provides a simpler alternative with competitive performance.

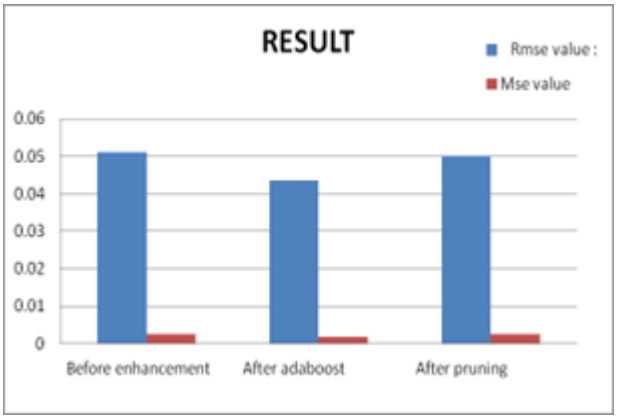


Fig7 . Result Analysis

CHAPTER 5
CONCLUSION

CHAPTER 5
CONCLUSION

5.1. Conclusion

In today's dynamic software development landscape, where the costs of complex systems continue to escalate, accurate cost estimation early in the project lifecycle is paramount. However, achieving perfect prediction accuracy remains a challenging task in software engineering. This study addresses this challenge by proposing a strategy based on the COCOMO model and leveraging decision tree algorithms to enhance cost estimation accuracy. By applying Decision Tree models alongside AdaBoost and pruning techniques on the COCOMO 81 dataset, the research demonstrates notable improvements in accuracy. AdaBoost emerges as a particularly effective method, although pruning offers a more sophisticated approach. The findings underscore the significance of accurate software cost estimation for effective project planning and resource allocation.

The results of this study indicate a substantial enhancement in cost estimation precision, with the RMSE decreasing from 0.051 to 0.043 after implementing AdaBoost and pruning techniques. This improvement highlights the efficacy of Decision Trees in software cost prediction, further enhanced by the application of AdaBoost and pruning. Notably, both AdaBoost and pruning lead to a noticeable decrease in error rates, underscoring their role in enhancing cost estimation precision. Overall, our approach utilizing Decision Trees, AdaBoost, and pruning presents a promising solution for improving software cost estimation accuracy, essential for successful project management in today's competitive software development landscape.

5.2. Future Enhancement:

In the realm of software cost estimation using Decision Trees, there are several avenues for future enhancement and exploration:

- **Exploration of Other Ensemble Methods:** While AdaBoost has shown promising results in enhancing Decision Tree models, there are other ensemble methods worth exploring, such as Gradient Boosting or Random Forest. These methods may offer alternative ways to improve model accuracy and robustness by leveraging different strategies for combining multiple weak learners into a strong predictor.
- **Experimentation with Advanced Pruning Techniques:** Pruning techniques play a crucial role in simplifying decision tree structures and preventing overfitting. Future research could delve into advanced pruning techniques that go beyond simple depth constraints, such as cost-complexity pruning (also known as weakest link pruning) or reduced error pruning. These techniques aim to optimize decision tree structures by selectively pruning branches based on their contribution to model performance.
- **Investigation of Feature Engineering and Selection:** Feature engineering and selection play a vital role in building predictive models. Future work could focus on identifying and engineering relevant features that capture the underlying relationships in software cost estimation datasets more effectively. Additionally, exploring advanced feature selection methods, such as recursive feature elimination or feature importance ranking, could help identify the most informative attributes for model training.

REFERENCES

REFERENCES

[1] International Journal of Computer Applications (0975 – 8887) Volume 104 – No 12, October 2014

[2]J.Caper, "Estimating Software Costs (English) 2nd Edition", Tata McGraw - Hill Education, 2007.

[3] Z. A. Khalifehlou, F. S. Gharehchopogh, "A Survey of Data Mining Techniques in Software Cost Estimation", AWER Procedia Information Technology & Computer Science Journal, Vol:1, pp.331- 342, 2012

[4] Z.A. Khalifelu, F.S. Gharehchopogh, "Comparison and Evaluation Data Mining Techniques with Algorithmic Models in Software Cost Estimation", Elsevier Press, Procedia-Technology Journal, ISSN: 2212-0173, Vol: 1, pp. 65-71, 2012.

[5] Abdel Karim Baareh / Journal of Computer Science 2019, 15 (3): 321.331 DOI: 10.3844/jcssp.2019.321.331

[6] Baareh, A. K. (2019). Optimizing Software Effort Estimation Models Using Back-Propagation Versus Radial Base Function Networks. Journal of Computer Science, 15(3), 321-331. https://doi.org/10.3844/jcssp.2019.321.331

[7] International Journal of Innovation and Applied Studies ISSN 2028-9324 Vol. 5 No. 1 Jan. 2014, pp. 72 81 © 2014 IJCRT2403825 International Journal of Creative Research Thoughts (IJCRT) www.ijcrt.org g913 www.ijcrt.org © 2024 IJCRT | Volume 12, Issue 3 March 2024 | ISSN: 2320-2882 Innovative Space of Scientific Research Journals http://www.issr-journals.org/ijias/

[8] Aitkenhead MJ (2008) A co-evolving decision tree classification method. Exp Syst Appl 34: 18–25

[9] T. Chai1,2 and R. R. Draxler1 1NOAA Air Resources Laboratory (ARL), NOAA Center for Weather and Climate Prediction, 5830 University Research Court, College Park, MD 20740, USA .

[10] Allan H. Murphy Department of Atmospheric Sciences, Oregon State University, Corvallis, Oregon DOI:
https://doi.org/10.1175/15200493(1988)116%3C2417:SSBOT M%3E2.0.CO;2
[11] https://www.spiceworks.com/tech/artificial- intelligence/articles/what-is-decision-tree/
[12] Prediction of Software Cost Estimation Using Spiking Neural Networks | SpringerLink
[13]Improving-Shared-Cache-Performance-Using-Variation-of-Bit-Set-Insertion-Policy.pdf (researchgate.net)

GitHub Link

https://github.com/Bhukya-Manichandana/Enhancement-of-decision-tree-for-Software-Cost-Estimation

RESEARCH PAPER

&

CERTIFICATIONS

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Ref No : IJCRT/Vol 12/ Issue 3 / 825

To,
B Manichandana

Subject: Publication of paper at International Journal of Creative Research Thoughts.

Dear Author,

With Greetings we are informing you that your paper has been successfully published in the International Journal of Creative Research Thoughts - IJCRT (ISSN: 2320-2882). Thank you very much for your patience and cooperation during the submission of paper to final publication Process. It gives me immense pleasure to send the certificate of publication in our Journal. Following are the details regarding the published paper.

About IJCRT : Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool) , Multidisciplinary, Monthly, Indexing in all major database & Metadata, Citation Generator, Digital Object Identifier(DOI) | UGC Approved Journal No: 49023 (18)
Registration ID : IJCRT_253789
Paper ID : IJCRT2403825
Title of Paper : Enhancement of Decision Tree For Software Cost Estimation
Impact Factor : 7.97 (Calculate by Google Scholar) | License by Creative Common 3.0
Publication Date: 22-March-2024
DOI :
Published in : Volume 12 | Issue 3 | March 2024
Page No : g909-g914
Published URL : http://www.ijcrt.org/viewfull.php?&p_id=IJCRT2403825
Authors : B Manichandana, M Laya, K Dhathri Guptha, V.Venkataiah
Notification : UGC Approved Journal No: 49023 (18)

Thank you very much for publishing your article in IJCRT.

Editor In Chief
International Journal of Creative Research Thoughts - IJCRT
(ISSN: 2320-2882)

IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

ENHANCEMENT OF DECISION TREE FOR SOFTWARE COST ESTIMATION

[#1] B. Manichandana, [#2] M. laya, [#3] K. Dhathri Guptha, [#4] V. Venkataiah

[1,2,3] *UG Student, Department of CSE, CMR College of Engineering & Technology, Hyderabad, Telangana*

[4] Associate Professor, Department of CSE, &Additional controller of examination CMR College of Engineering &Technology, Hyderabad, Telangana

Abstract: Predicting the amount of effort needed for software development is a major challenge encountered by the software industry. It includes planning, supervising projects, analysing the viability of the system development, preparing proposals and proposals presentation to clients, and among this estimation of the effort for planning is one of the most critical responsibilities. It is necessary to have good effort estimation to conduct a well budget. Accurate software project effort estimation is crucial for the competitiveness and success of software companies. For the forecasting of software program attempts, it's far vital to choose the suitable software program attempt estimation techniques. A number of have been proposed like algorithms, non-algorithms, Algorithms over machine learning, and so on. Generally, Decision Trees have drawn the attention of researchers and changed the direction of Effort Estimation towards computational intelligence. We are utilizing the DT concept in our project, which is a straightforward but effective strategy that serves as the foundation for Random Forest, also referred to as a collection of decision trees. Decision Trees facing data overfitting problem. To overcome this problem enhancement of the DT with the ensemble method proposed in this project and also to evaluate the performance of this proposed method using the COCOMO 81 dataset. To evaluate metrics like MSE and RMSE. Research comparing different methodologies indicates that the proposed technique outperforms prior approaches in terms of results.

Index Terms **- Software Cost Estimation (SCE), Decision Tree (DT), AdaBoost, Pruning, COCOMO 81, Mean Square Error (MRE), Root Mean Square Error (RMSE), and Machine Learning (ML).**

I. INTRODUCTION

SCE holds significant importance in system production cycles, guiding project planning and resource allocation. It involves predicting the approximate cost before software development commences, utilizing mathematical algorithms known as cost estimation models. Despite numerous methodologies proposed, achieving complete accuracy remains a challenge due to the uncertainty inherent in early-stage project inception [1]. Accurate estimations are crucial for effective decision-making and successful project management, preventing budget overruns and delays.Given the paramount importance of software cost estimation, many organizations prioritize its precision. Various advanced machine learning techniques, such as Gradient Boosting, Neural Networks, and Random Forest, are employed for this purpose. Decision Trees, renowned for their simplicity, interpretability, and adaptability to diverse data types, particularly excel in this domain [2]. They effectively capture non-linear relationships and facilitate easy visualization, making them invaluable for software cost prediction insights.This paper introduces an enhanced decision tree method tailored to improve the accuracy of software cost estimation, particularly addressing challenges associated with limited project information during early development stages. Through a comparative analysis with BASIC COCOMO, the proposed approach demonstrates superior effectiveness in achieving refined and reliable cost estimations.

II. BACKWORD WORK

Software cost estimation is a critical process in project management, involving the prediction of resources, time, and budget required for the development of a software project. It aims to provide stakeholders with a realistic assessment of the project's financial and temporal requirements. Several factors influence software cost estimation, including project size, complexity, and requirements clarity. Metrics like lines of code, function points, and use case points are commonly used to quantify the size and functionality of the software [3]. The experience and expertise of the development team, as well as the technology and tools employed, play a significant role in determining costs. Risk assessment is crucial, considering identified risks and mitigation strategies. External factors such as regulatory compliance, security requirements, and dependencies on third-party integrations can introduce additional complexities. Indeed, with the growing significance of software systems, estimating software costs accurately has become increasingly vital. Researchers continually strive to address this challenge by introducing new methods each year. Leveraging machine learning algorithms, some modern approaches aim to enhance the precision of cost estimates. These methods represent a promising avenue for improving software cost estimation in response to the evolving nature of modern software systems.

Reza Ahamad and team from Hacettepe University employed the bee colony optimization (BCO) algorithm to refine cost estimation accuracy based on intermediate COCOMO. By categorizing NASA information set projects according to COCOMO assignment types and calibrating recovery parameters through BCO, they achieved enhanced accuracy. Results indicated a significant improvement, with MMARE declining from 0.2371 percent (COCOMO) to 0.1619 percent (proposed method), reflecting a decrease of 0.0762 percent [4].

Abdel Karri Baareh from Al-Balqa Applied University, Ajloun, Jordan. In his paper, He used two NN models, namely the Back-propagation algorithm and the Radial Base algorithm, to compare their effectiveness in software effort and cost estimation. Using a NASA public dataset, the models are implemented, with 45 data samples for training and 15 for testing. The results indicate that the Back-propagation neural network outperforms the Radial Base function in both training and testing cases [5].

Azeeh Ghatasheh and his team used the Firefly Algorithm to optimize parameters in three COCOMO based software effort estimation models. Experimental results demonstrate the algorithm's high accuracy and substantial error minimization compared to other optimization methods. They concluded that the Firefly Algorithm's superior performance [6].

Mohammad Masdari from Iran University presents a paper on estimating software development project costs using a blend of Genetic Algorithm (GA) and Ant Colony Optimization (ACO), specifically tailored for NASA software projects. The model employs GA for testing and ACO for training, showing enhanced results compared to the conventional COCOMO model. The model's effectiveness stems from its consideration of influential factors in estimation [7].

III. EMPPLOYED TECHNIQUES

The main technique used to find the accuracy of SCE is the DT.

A. Decision Tree

A decision tree visually represents decision-making processes within software, serving as a graphical depiction of these processes. Illustrating potential outcomes and the choices leading to them. Widely utilized across machine learning, business, and decision analysis domains, Decision trees play a crucial role in tasks such as classification and regression [8]. These trees are constructed by iteratively segmenting records based on features to form homogeneous subsets. At each node, the algorithm identifies the most informative feature for data splitting, aiming to maximize information gain or minimize impurity. This iterative process persists until a predefined stopping criterion is met, resulting in a finalized structure that can be applied to new data. Decision trees offer numerous benefits, including interpretability, ease of visualization, and the capability to handle both categorical and numerical data.

B. Overfitting

It is in device studying happens whilst a version learns the education information too well, shooting noise and random fluctuations. This leads to poor generalization, where the model may perform exceptionally on the training set but fails to generalize effectively to new, unseen data. One common technique to address overfitting, especially in decision tree based models, is pruning.

C. Pruning

It entails pruning branches from a decision tree that don't significantly enhance its predictive capability. In the realm of decision trees, overfitting frequently leads to excessively intricate trees that capture the noise present in the training data, rather than the underlying patterns. Pruning aims to simplify the tree, eliminating unnecessary

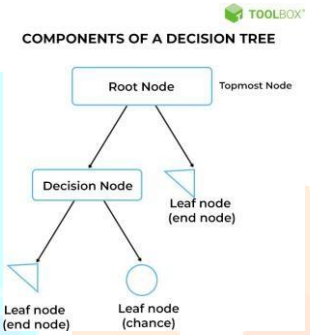
branches and nodes. This process helps prevent the tree from memorizing specific details of the training data that do not represent true patterns, ensuring a more generalized and robust model.

D. Boosting

It's a method in which multiple weak learners are combined to form a robust learner. In the context of decision trees, boosting entails training a sequence of decision trees, with each subsequent tree assigning higher weight to instances misclassified by the preceding trees. The boosting process helps improve the overall performance of the model by iteratively correcting errors made by previous weak learners. Popular boosting algorithms that can be applied to decision trees include AdaBoost and Gradient Boosting (e.g., XGBoost, LightGBM, and CatBoost).

E. AdaBoost

It helps prevent overfitting by paying extra attention to the wrongly guessed examples and forming a strong team of decision trees. It works well with tricky data, gets better at making predictions overall, and is good at handling



noisy information. AdaBoost's special attention to tricky cases makes decision tree models perform better.

Structure of Decision Tree:[11]

To find the accuracy of a software cost estimation, the evaluating Metrics used, are Root Mean Square Error (RMSE) and Mean Square Error (MSE).

B. Error of Root Mean Square (RMSE):

RMSE is the square root of the average squared difference between the predicted values and the actual values in a dataset. The decrease in the RMSE, the higher the version suits a dataset. It is a commonly used metric for evaluating the accuracy of a predictive model, particularly in the context of regression analysis. It measures the common significance of the mistakes among anticipated values and real values in a dataset. RMSE is a popular choice because it penalizes larger errors more heavily than smaller ones, making it sensitive to outliers.[9].

It is calculated as **RMSE = √ 1/n*Σ (ŷi – yi)2**

where: Σ is a symbol that means "sum", ŷi is the predicted value for the ith observation, yi is the observed value for the ith observation and n is the sample size.

C. Square root of mean error (MSE):

MSE quantifies the average squared difference between the predicted values and the actual values in a dataset. The decrease in the MSE, the higher the version suits a dataset [10].

The formula for MSE is **MSE = 1/n * Σ (ŷi – yi)2**

where: Σ is a symbol that means "sum", ŷi is the predicted value for the ith observation, yi is the observed value for the ith observation and n is the sample size.

IV. PROPOSED METHODOLOGY

In this paper, the COCOMO 81 data set is collected from the literature which includes attributes related to software projects, such as reliability, data complexity, time constraints, storage requirements, and numerous additional influencing elementssoftware development efforts and cost after that min max scaler is applied to normalize the dataset features, ensuring they fall within a uniform range of 0 to 1. This step is crucial to prevent certain features from disproportionately influencing the machine learning model due to varying scales. Training and testing portions of the dataset are separated. sets using the train_test_split function. This separation is fundamental for training the model on one subset and evaluating its performance on another, unseen subset. A Tree of Decisions Regressor is trained on the training set. It learns patterns and relationships within the data. Algorithm for decision trees:

- Establish node N for software cost estimation.
- If all projects in dataset D share the same cost estimation class, provide back N as a leaf node with that class labelled.
- If the attribute list was empty: -Return N as a leaf node that has the label average or median cost in D.

- Apply Attribute_selection_method (D, attribute_list) to determine the "best" splitting criterion.
- Tag node N with the selected splitting criterion.
- If the the characteristic that separates is discrete-valued and multiway splits are permitted: >Remove the the characteristic that separates from the attribute lists.

>For each outcome of the splitting criterion:

-Let D_partition be the subset of data projects in D satisfying the outcome.

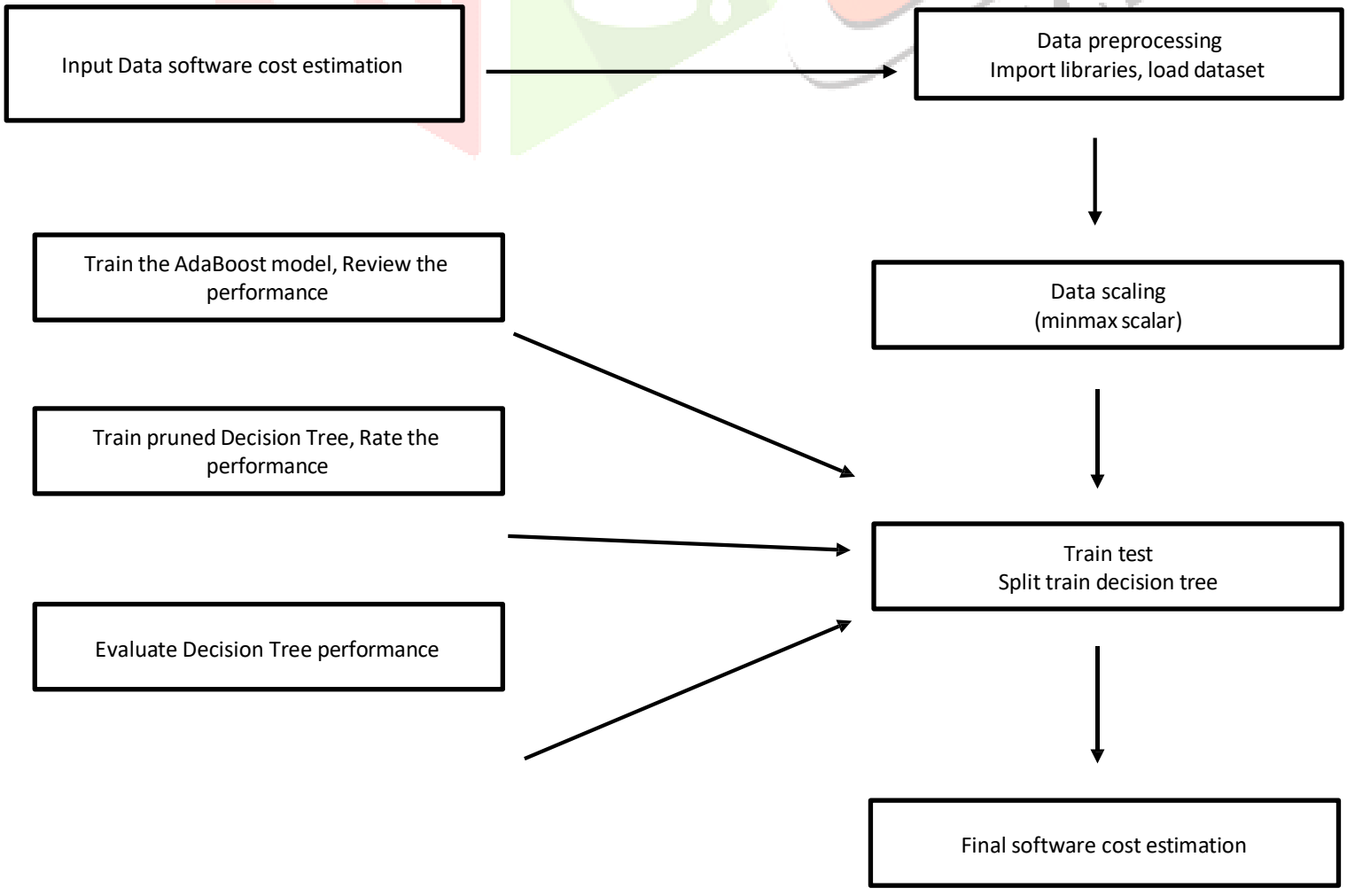
- If D_partition is empty:

-Attach a leaf labeled with the average or median cost in D to node N.

- Else: Attach the node returned by generate_cost_estimation_tree(D_partition,attribute_li st) to node N.
- Return N.

To assess the effectiveness of the Decision Tree model, predictions are made on the test set, and Relevant metrics such as root-mean-squared error (RMSE) and the mean squared error (MSE) are computed. Next, an AdaBoost Regressor is trained on the training set, utilizing the Decision Tree Regressor as a base estimator. AdaBoost blends a number of ineffective learners to create a stronger predictive model. The AdaBoost model's performance is then assessed by making predictions on the test set and computing performance metrics such as MSE and RMSE. Following that, a pruned Decision Tree Regressor with limited depth is trained on the training set. Pruning helps prevent overfitting and simplifies the model. The performance of the pruned Decision Tree is evaluated on the test set, and performance metrics (MSE and RMSE) are calculated. The final software cost estimation output showcases the error evaluation before and after the enhancement of the decision tree model, specifically addressing the reduction of overfitting. This analysis demonstrates how the Model of decision trees is refined and improved, ultimately enhancing its predictive capabilities for software cost estimation.

BLOCK DIAGRAM



IV. RESULTS AND DISCUSSION

In this document, the software cost estimation results demonstrate notable improvements after applying enhancement techniques such as AdaBoost and pruning. Before enhancement, the RMSE stood at 0.05102516893992321, while the MSE was 0.0026035678653477047. Following the application of AdaBoost, the RMSE decreased to 0.04352386357990304, and the MSE reduced to 0.0018943267009220099. Pruning, another enhancement method, resulted in an RMSE of 0.05004060795782995 and MSE of 0.002504062444789234. Both techniques contributed to a decrease in errors, with AdaBoost exhibiting a more pronounced effect. The choice between AdaBoost and pruning should consider factors beyond error reduction, such as computational complexity and the specific objectives of the SCE.

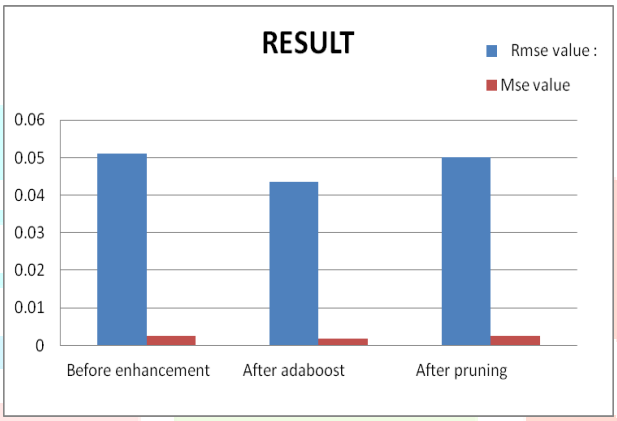


Fig.1: Result analysis

V. CONCLUSION

In today's software development landscape, accurately estimating the costs of complex systems early on is crucial, given the rising expenses associated with software projects. However, achieving perfect prediction accuracy remains an elusive goal in software engineering research. This work tackles the problem by putting forth a strategy based on the fundamental COCOMO model that uses decision tree algorithms to improve cost estimation accuracy. Through the application of Decision Tree models alongside AdaBoost and pruning techniques on the COCOMO 81 dataset, the study demonstrates significant improvements in accuracy. AdaBoost is especially notable for its significant improvement, although pruning provides a more sophisticated method. The results showcase a noticeable uptick in cost estimation precision, with the RMSE dropping from 0.051 percent to 0.043 percent after implementing AdaBoost and pruning.

REFERENCES

[1] International Journal of Computer Applications (0975 – 8887) Volume 104 – No 12, October 2014

[2]J.Caper, "Estimating Software Costs (English) 2nd Edition", Tata McGraw - Hill Education, 2007.

[3] Z. A. Khalifehlou, F. S. Gharehchopogh, "A Survey of Data Mining Techniques in Software Cost Estimation", AWER Procedia Information Technology & Computer Science Journal, Vol:1, pp.331- 342, 2012

[4] Z.A. Khalifelu, F.S. Gharehchopogh, "Comparison and Evaluation Data Mining Techniques with Algorithmic Models in Software Cost Estimation", Elsevier Press, Procedia-Technology Journal, ISSN: 2212-0173, Vol: 1, pp. 65-71, 2012.

[5] Abdel Karim Baareh / Journal of Computer Science 2019, 15 (3): 321.331 DOI: 10.3844/jcssp.2019.321.331

[6] Baareh, A. K. (2019). Optimizing Software Effort Estimation Models Using Back-Propagation Versus Radial Base Function Networks. Journal of Computer Science, 15(3), 321-331. https://doi.org/10.3844/jcssp.2019.321.331

[7] International Journal of Innovation and Applied Studies ISSN 2028-9324 Vol. 5 No. 1 Jan. 2014, pp. 72-81 © 2014

Innovative Space of Scientific Research Journals http://www.issr-journals.org/ijias/

[8] Aitkenhead MJ (2008) A co-evolving decision tree classification method. Exp Syst Appl 34: 18–25

[9] T. Chai1,2 and R. R. Draxler1 1NOAA Air Resources Laboratory (ARL), NOAA Center for Weather and Climate

Prediction, 5830 University Research Court, College Park, MD 20740, USA

[10] Allan H. Murphy Department of Atmospheric Sciences, Oregon State University, Corvallis, Oregon DOI: https://doi.org/10.1175/15200493(1988)116%3C2417:SSBOT M%3E2.0.CO;2

[11] https://www.spiceworks.com/tech/artificial- intelligence/articles/what-is-decision-tree/

Certificate of Publication

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | ISSN: 2320 - 2882

An International Open Access, Peer-reviewed, Refereed Journal

The Board of

International Journal of Creative Research Thoughts

Is hereby awarding this certificate to

B Manichandana

In recognition of the publication of the paper entitled

Enhancement of Decision Tree For Software Cost Estimation

Published In IJCRT (www.ijcrt.org) & 7.37 Impact Factor by Google Scholar

Volume 12 Issue 3 March 2024 , Date of Publication: 22-March-2024

UGC Approved Journal No: 43023 (18)

PAPER ID : IJCRT2403825

Registration ID : 253789

ISSN 2320-2882 IJCRT

EDITOR IN CHIEF

Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool), Multidisciplinary, Monthly Journal

IJCRT | ISSN: 2320-2882 | IJCRT.ORG

Certificate of Publication

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | ISSN: 2320 - 2882

An International Open Access, Peer-reviewed, Refereed Journal

The Board of

International Journal of Creative Research Thoughts

Is hereby awarding this certificate to

M Laya

In recognition of the publication of the paper entitled

Enhancement of Decision Tree For Software Cost Estimation

Published In IJCRT (www.ijcrt.org) & 7.37 Impact Factor by Google Scholar

Volume 12 Issue 3 March 2024 , Date of Publication: 22-March-2024

UGC Approved Journal No: 43023 (18)

PAPER ID : IJCRT2403825

Registration ID : 253789

ISSN 2320-2882 IJCRT

EDITOR IN CHIEF

Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool), Multidisciplinary, Monthly Journal

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | IJCRT

An International Scholarly, Open Access, Multi-disciplinary, Indexed Journal

Website: www.ijcrt.org | Email id: editor@ijcrt.org | ESTD: 2013

IJCRT | ISSN: 2320-2882 | IJCRT.ORG

Certificate of Publication

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | ISSN: 2320 - 2882

An International Open Access, Peer-reviewed, Refereed Journal

The Board of

International Journal of Creative Research Thoughts

Is hereby awarding this certificate to

K Dhathri Guptha

In recognition of the publication of the paper entitled

Enhancement of Decision Tree For Software Cost Estimation

Published In IJCRT (www.ijcrt.org) & 7.37 Impact Factor by Google Scholar

Volume 12 Issue 3 March 2024 , Date of Publication: 22-March-2024

UGC Approved Journal No: 43023 (18)

PAPER ID : IJCRT2403825

Registration ID : 253789

ISSN 2320-2882 IJCRT

EDITOR IN CHIEF

Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool), Multidisciplinary, Monthly Journal

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | IJCRT

An International Scholarly, Open Access, Multi-disciplinary, Indexed Journal

Website: www.ijcrt.org | Email id: editor@ijcrt.org | ESTD: 2013

IJCRT | ISSN: 2320-2882 | IJCRT.ORG

Certificate of Publication

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | ISSN: 2320 - 2882

An International Open Access, Peer-reviewed, Refereed Journal

The Board of

International Journal of Creative Research Thoughts

Is hereby awarding this certificate to

V.Venkataiah

In recognition of the publication of the paper entitled

Enhancement of Decision Tree For Software Cost Estimation

Published In IJCRT (www.ijcrt.org) & 7.37 Impact Factor by Google Scholar

Volume 12 Issue 3 March 2024 , Date of Publication: 22-March-2024

UGC Approved Journal No: 43023 (18)

PAPER ID : IJCRT2403825

Registration ID : 253789

ISSN 2320-2882 IJCRT

EDITOR IN CHIEF

Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool), Multidisciplinary, Monthly Journal

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | IJCRT

An International Scholarly, Open Access, Multi-disciplinary, Indexed Journal

Website: www.ijcrt.org | Email id: editor@ijcrt.org | ESTD: 2013

IJCRT | ISSN: 2320-2882 | IJCRT.ORG