IMPLEMENTATION OF EFFICIENT EFFORT AND COST ESTIMATION TECHNIQUE FOR AGILE SOFTWARE

Thesis Submitted For the Award of the Degree of

DOCTOR OF PHILOSOPHY

in

Computer Science and Engineering

By

Mohit Arora

41500188

Supervised by
Dr. Sahil Verma

Co-Supervised by
Dr. Aman Singh

LOVELY PROFESSIONAL UNIVERSITY
PUNJAB

2022

DECLARATION

I hereby affirm that the thesis entitled **Implementation of Efficient Effort and Cost Estimation Technique for Agile Software** submitted by me for the Degree of Doctor of Philosophy in Computer Science and Engineering is the result of my original and independent research work carried out under the guidance of Supervisor **Dr. Sahil Verma** and Co-Supervisor **Dr. Aman Singh**, and it has not been submitted for to any university or institute for the award of any degree or diploma.

Mohit Arora

School of Computer Science and Engineering

Lovely Professional University

Jalandhar- Delhi GT Road (NH-1)

Phagwara, Punjab, INDIA

Date: 10th November, 2021

[illegible]

Signature of the candidate

CERTIFICATE

This is to certify that the thesis entitled **Implementation of Efficient Effort and Cost Estimation Technique for Agile Software** submitted by Mr. Mohit Arora, School of Computer Science and Engineering, Lovely Professional University, for the award of the degree of Doctor of Philosophy, is a record of bonafide work carried out by him under my supervision, as per the code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

[illegible]

Dr. Sahil Verma

Supervisor

Aman

Dr. Aman Singh

Co-Supervisor

ABSTRACT

Waterfall to Agile is the most well-known transformation in the last two decades. This paradigm shift from heavyweight to lightweight process models addresses many of the roadblocks to software project development. Any competitive IT industry cannot avoid underestimating the effort, cost, and length of their projects. Approximately 43 percent of projects are often completed late and reach crises as a result of over budgeting and less necessary functions. Estimation is a critical component of Software Project Management and according to the International Society for Parametric Analysis (ISPA), the International Cost Estimating and Analysis Association (ICEAA), and the Standish group chaos manifesto, several IT ventures have been hampered by unreliable estimates of effort and related costs. Improper and unreliable evaluation of software projects, as well as uncertainty in software specifications, contributes to failure and must therefore be taken into account in full letter and spirit. When Agile principles-based process models (e.g., Scrum) were introduced, it resulted in a dramatic shift in project management. This cultural shift proves to be beneficial in terms of improving the relationship between developer and customer. Estimation is easier in conventional methodologies because they use plan-driven methods, but in Agile, requirements are volatile, so effort estimation is the most difficult. This cause raises awareness among potential researchers all over the world to begin working on addressing the issue of unreliable effort prediction. There are numerous explanations for the difference between estimated and actual effort, including project, people, and resistance variables, incorrect use of cost drivers, ignorance of regression testing effort, understandability of user story size and its related difficulty, and so on. We examined the work of various authors and potential researchers who were attempting to bridge the gap between real and estimated effort. According to the related literature, machine learning models outperform non-machine learning and conventional estimation techniques. For various agile methodologies, researchers investigated and applied estimation types, techniques, and tools ranging from conventional to machine learning estimation. Software projects are dynamic and potentially unpredictable, which adaptive models can accommodate well. At the moment, the majority of IT managers working on Agile projects rely on conventional estimation methods such as planning poker, expert judgment, and so on, which suffer from individual bias. To deal with volatile situations, an expert system is necessary. Our proposed approach is based on the concepts of adaptive networks and neuro-fuzzy to assist managers in determining suitable project resources. At the moment, there is a lack of scrum project datasets in public repositories, making it difficult for any researcher to present their work without jeopardizing its validity. To show the system's effectiveness, Scrum project data has been seeded into the knowledge base. IT stakeholders use issue monitoring systems such as JIRA, which offers a comprehensive environment for managing, integrating, and collaborating on end-to-end IT services but lacks Machine Learning supported estimation.

While researchers used various effort estimation techniques such as use case points, adjusted use case points, story points, analogy-based estimation, and some soft computing techniques, ideal estimation accuracy remains a myth. According to Collabnet VersionOne agile state of the art study, scrum is the most widely used agile technique in software industries, but it is afflicted by estimation problems. We developed a hybrid Adaptive Neuro-Fuzzy Inference System (ANFIS) model tuned by the novel Energy-Efficient BAT (EEBAT) and Cost Estimating BAT (CEBAT) algorithms to handle complex specificities and accurate estimation challenges in a scrum context. The system has been evaluated against various state-of-the-art and practice meta-heuristic and Machine Learning (ML) algorithms such as Fireworks, Ant Lion Optimizer (ALO), BAT, Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and others, and it not only produces promising effort estimation results but also outperforms them for homogeneous data sets. On over 200 Agile projects, we used our proposed techniques ANFIS-EEBAT and ANFIS-CEBAT and achieved significant estimation accuracy.

Dedicated

To

My Beloved Parents,

My wife, Shivali Chopra,

My daughter, Parinaaz Arora

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor Dr. Sahil Verma and co supervisor Dr. Aman Singh for their continuous support throughout my literature work. Their guidance helped me a lot to address all the issues and challenges that came in my way.

My family has always been a support to me. It is because of them I have been able to get time for research.

(Mohit Arora)

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
ACM	Association for Computing Machinery
ALO	Ant Lion Optimization
AMBA	Adaptive multi-swarm bat algorithm
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Network
AR	Association Rule
ASD	Agile Software Development
BatDNN	Bat Deep Neural Network
BN	Bayesian Network
BRE	Balance Relative Error
BU	Bottoms Up
CART	Classification And Regression Trees
CatBoost	Categorical Boosting
CBA	Chaotic Bat Algorithm
CBR	Case-Based Reasoning
CCNN	Cascade Correlation Neural Network
CDF	Cobb-Douglas's Function
CEBAT	Cost Estimation Bat

CESP	Cost Efficient Scrum Process
CFASEE	Consistent Fuzzy Analogy-based Software Effort Estimation
CFP	COSMIC Function Points
CMMI	Capability Maturity Model Integration
COCOMO	Constructive Cost Model
CORADMO	Constructive Rapid Application Development Model
CPU	Central Processing Unit
DABA	Directed Artificial Bat Algorithm
DABC	Directed Artificial Bee Colony
DBN	Deep Belief Network
DLBA	Double-subpopulation Lévy flight Bat Algorithm
DT	Decision Trees
DVBA	Dynamic Virtual Bats Algorithm
ECS	Evolutionary Cost-Sensitive
EEBAT	Energy Efficient Bat
EEP	Effort Estimation Problem
EJ	Expert Judgement
FASEE	Fuzzy Analogy-based Software Effort Estimation
FCM	Fuzzy C-Means
FIS	Fuzzy Inference System
FLANN	Functional Link Artificial Neural Network

FP	Function Points
GA	Genetic Algorithms
GD	Gradient Descent
GLM	Generalized Linear Model
GMDH	Group Method of Data Handling
GNB	Gaussian Naïve Bayes
GP	Genetic Programming
GPU	Graphics Processing Unit
GRNN	General Regression Neural Network
IBA	Island multipopulational parallel Bat Algorithm
ICEAA	International Cost Estimating and Analysis Association
IEEE	Institution of Electrical and Electronics Engineers
IFPUG	International Function Point User Group
ISBSG	International Software Benchmarking Standards Group
ISPA	International Society of Parametric Analysis
IT	Information Technology
KLOC	Kilo Lines of Code
KNN	K Nearest Neighbours
KOINS	Kobena Information System
LBA	Levy flight-based Bat Algorithm
LM	Levenberg–Marquardt

LMT	Logistic Model Tree
LoC	Lines of Code
LR	Linear Regression
LSE	Least Squares Estimation
LSTM	Long Short-Term Memory
MAPE	Mean Absolute Percentage Error
MATLAB	Matrix Laboratory
MBE	Mean Balance Error
MBRE	Mean Balance of Relative Error
MDELP	Multilayer Dilation-Erosion-Linear Perceptron
MdMRE	Medium Magnitude of Relative Error
ME	Mean Error
MER	Magnitude of Error Relative
MF	Membership Function
MIBRE	Mean Inverted Balance Relative Error
ML	Machine Learning
MLPANN	Multi-Layer Perceptron Artificial Neural Network
MLR	Multiple Linear Regression
MMER	Mean Magnitude of Error Relative
MMRE	Mean Magnitude of Relative Error
MOBA	Multi-Objective Bat Algorithm

MBT	Model Based Testing
MRE	Magnitude of Relative Error
MSE	Mean Squared Error
MUCP	Modified Use Case Points
NASA	National Aeronautics and Space Administration
NB	Naïve Bayes
NF	Neuro-Fuzzy
NFIS	Neuro Fuzzy Inference System
NFR	Non-Functional Requirements
NN	Neural Networks
OBMLBA	Opposition Based Modified Levy flight Bat Algorithm
OP	Object Points
PC	Product Customization
PNN	Probabilistic Neural Network
PP	Planning Poker
PRED	Percentage Relative Error Deviation
PRINCE2	Project In Controlled Environment
PSO	Particle Swarm Optimization
RBFN	Radial Basis Function Network
RBFNN	Radial Basis Function Neural Network
RBM	Restricted Boltzmann Machine

RF	Random Forests
RHWN	Recurrent Highway Neural Network
RMSE	Root Mean Squared Error
RQ	Research Question
SAP	Systems Programming and Advanced Data Processing
SBA	Shrink factor Bat Algorithm
SBO	Satin Bowerbird Optimization
SDEE	Software Development Effort Estimation
SDLC	Software Development Life Cycle
SGB	Stochastic Gradient Boosting
SGD	Stochastic Gradient Descent
SLIM	Software Lifecycle Management
SLOC	Source Lines of Code
SMOTE	Synthetic Minority Over Sampling Technique
SP	Story Points
SPSS	Statistical Package for Social Sciences
SVM	Support Vector Machine
SVR	Support Vector Regression
TD	Top Down
TE	Team Experience
TF-IDF	Term Frequency-Inverse Document Frequency

TLBAC	Teaching–Learning-Based Artificial Bee Colony
TLBO	Teaching–Learning-Based Optimization
TOSEM	Transactions on Software Engineering and Methodology
TS	Task Size
UCP	Use Case Points
UFP	Unadjusted Function Point
US	User Stories
WEKA	Waikato Environment for Knowledge Analysis
WOA	Whale Optimization Algorithm
XGBoost	eXtreme Gradient Boosting
ZKmS	Zia K-means SMOTE

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Waterfall to Agile: A Paradigm Shift

In the last few decades, there have been significant paradigm shifts in the software engineering culture, the most notable of which is the transition from waterfall to Agile. As depicted in Figure 1.1, the key feature of agile development is that resources and planning are set throughout the project, leaving only requirements to vary. This means that resources and planning are fixed throughout the development cycle, and after some functionality is delivered, the requirements can be adjusted based on the needs. This is a distinct perspective from traditional software development when the final product is unknown.

Traditional software development relies on up-front requirements analysis and breaking it down into milestones and success measures whereas agile development focuses on delivering the correct value based on a prioritized requirements backlog. Testing and working with the functionality, as well as a more fluid means of altering requirements, replace the usually written milestones.

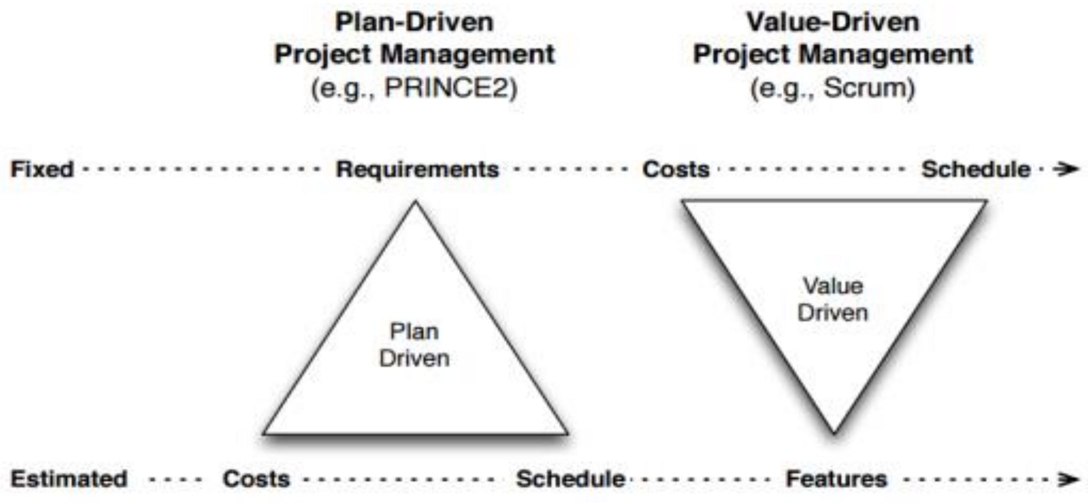


Figure 1.1: Waterfall vs Agile

1.2 Agile umbrella methodologies

ASD is an umbrella term for a collection of approaches and activities based on the Agile Manifesto's values and principles. In the domain of software engineering, the term "Agile" is frequently used. With technological developments, Agile has become the standard in Fortune 500 firms. Figure 1.2 depicts that requirements are initially elicited from the clients but not all requirements have been considered for inclusion in the earliest iterations. The requirements stack is first prioritized and then the most important ones are sent for development.

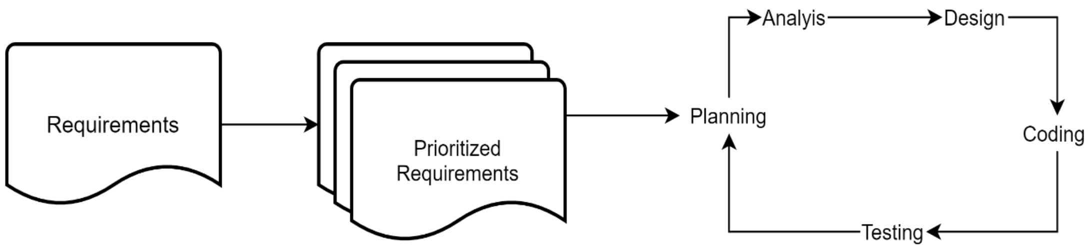
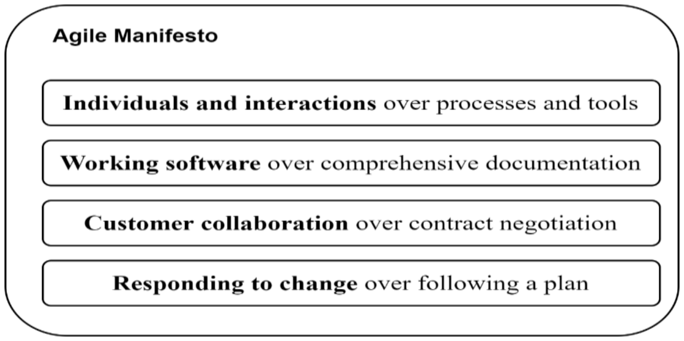


Figure 1.2: Essence of Agile

1.2.1 The manifesto for Agile software development

Agile methodologies are inspired from its manifesto laid by Snowbird'17 to pave the path for IT industries adopting a 'continuous' revolution. Agile is neither an approach nor a technique, it is more of a concept with four fundamental values. The manifesto in its original form is shown in Figure 1.3.



Agile Manifesto

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

Figure 1.3: Agile Manifesto

1.2.2 Agile software development principles

The twelve principles of ASD are depicted in Figure 1.4 and are made to be followed in true letter and spirit by all the organizations embracing agile.

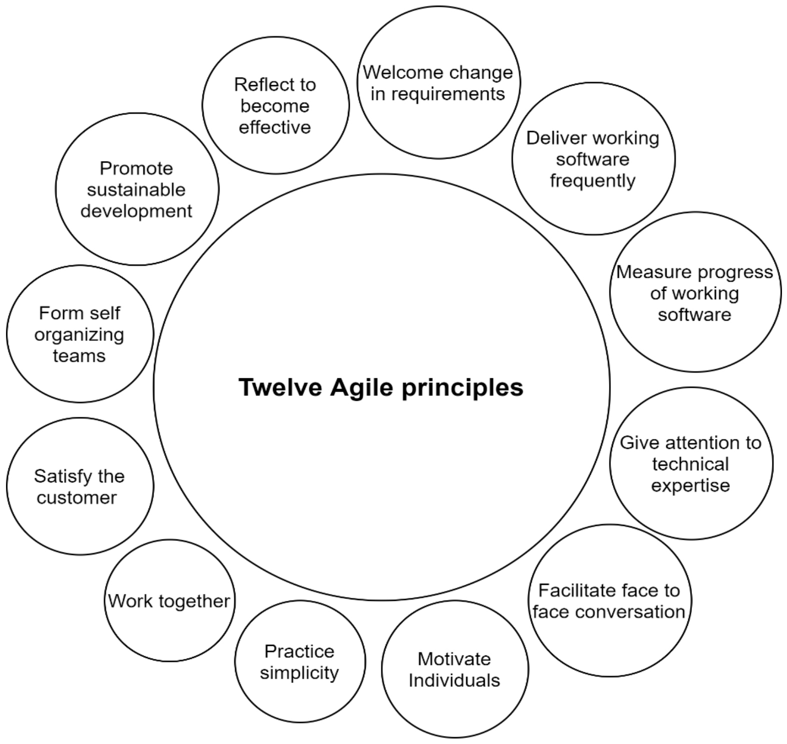

Twelve Agile principles
Welcome change in requirements
Deliver working software frequently
Measure progress of working software
Give attention to technical expertise
Facilitate face to face conversation
Motivate Individuals
Practice simplicity
Work together
Satisfy the customer
Form self organizing teams
Promote sustainable development
Reflect to become effective

Figure 1.4: Agile Principles

1.2.3 Agile software development methodologies

Agile has a buffet of methodologies in its parasol and each considers different thrust areas. The various popular agile methodologies are shown in Figure 1.5.

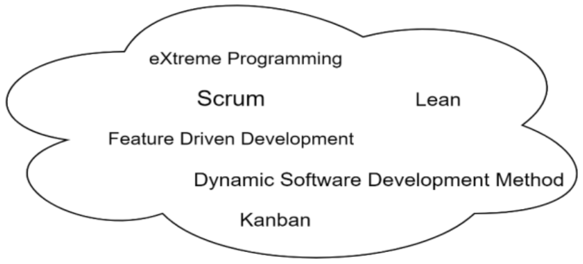

eXtreme Programming
Scrum
Lean
Feature Driven Development
Dynamic Software Development Method
Kanban

Figure 1.5: Agile Umbrella

1.3 Scrum framework

Scrum is an agile framework that takes in user stories as input requirements and accomplished the same in short, fixed-length, and time-boxed iterations called sprints. A shippable product is delivered as the sprint ends. The entire cycle beginning with Sprint Planning and ending at Retrospectives is called a Sprint. The detailed steps are given below:

- **Product Backlog:** Customer prioritizes requirements as per business value and lists them in the product backlog. The representation of requirements is made using the US. As customer prioritizes their requirements, the project team also called the scrum team provides a high-level estimate for each user story.
- **Sprint Planning:** As soon as a sprint begins, the team conducts a sprint planning meeting. All the stakeholders participate in the meeting. The prioritized product backlog is used by the team to pick up stories for implementation throughout a sprint during the meeting. The available capacity in person-hours and the team's productivity is tantamount to address the count of user stories targeted. The customers must prioritize the product backlog. Prioritization ensures that the features developed first are of the highest value. The sprint planning meeting normally takes about half a day.
- **Sprint Backlog:** The Scrum team utilizes sprint tasks which are defined as the definitive development activities, vital in resolving the requirement. They are obtained as breakdowns of product backlog's requirements. A sprint backlog is an outcome after a spring planning meeting. The sprint backlog details the tasks and task-level estimates of the selected stories. On Sprint Backlog completion, the estimated total work is matched against original high-level Product Backlog estimates.
- **Implementation Cycle:** Once the team is ready with the sprint backlog, implementation of stories commences. The Implementation cycle involves the

activities of design, coding, and testing. The progress of the team is monitored through visual controls like Story Boards and Effort Burn-down charts.

- **Daily Scrum**: Every day the daily scrum meeting is conducted at a pre-determined time – typically done at the beginning of the day. This is a short meeting carried out without deviating from technical issues. It is mandatory for the team to "stand up" during these meetings so that the stipulated time is not exceeded. Each team member shares the status of their work by responding to the three crucial questions: what I achieved yesterday, what I shall accomplish today, and what impediments stall my progress? The sprint backlog is updated – with addition, deletion, and modification to the planned tasks and the remaining efforts for the same as the daily scrum meeting wrap-up.
- **Sprint Review:** The output of the sprint is a potentially shippable product, which is demonstrated to all the stakeholders and their feedback is sought – this is called the sprint review meeting. All enhancements, bugs, or defects identified by the customer are added to the product backlog and are addressed based on their priority. The abstract view of the Scrum process has been shown in Figure 1.6.
- **Retrospective:** Consequently, Retrospective succeeds Sprint Review. The team assesses what went well, what did not and identifies the changes needed to make the process better.

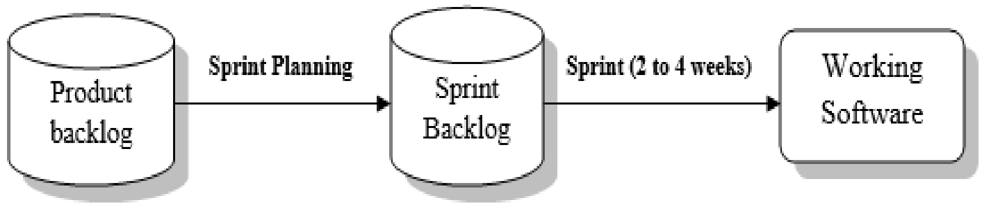


Figure 1.6: Scrum process

It allows a team to inspect and adopt. The basic steps of ASD are shown in Figure 1.7. The detailed supposition stages of ASD present an iterative loop wherein customer collaboration plays a crucial role in framing decisions, choosing and prioritizing requirements, incorporating feedback, etc.

The iterative and incremental development resolve associated risks, thus laid a strong foundation of process models.

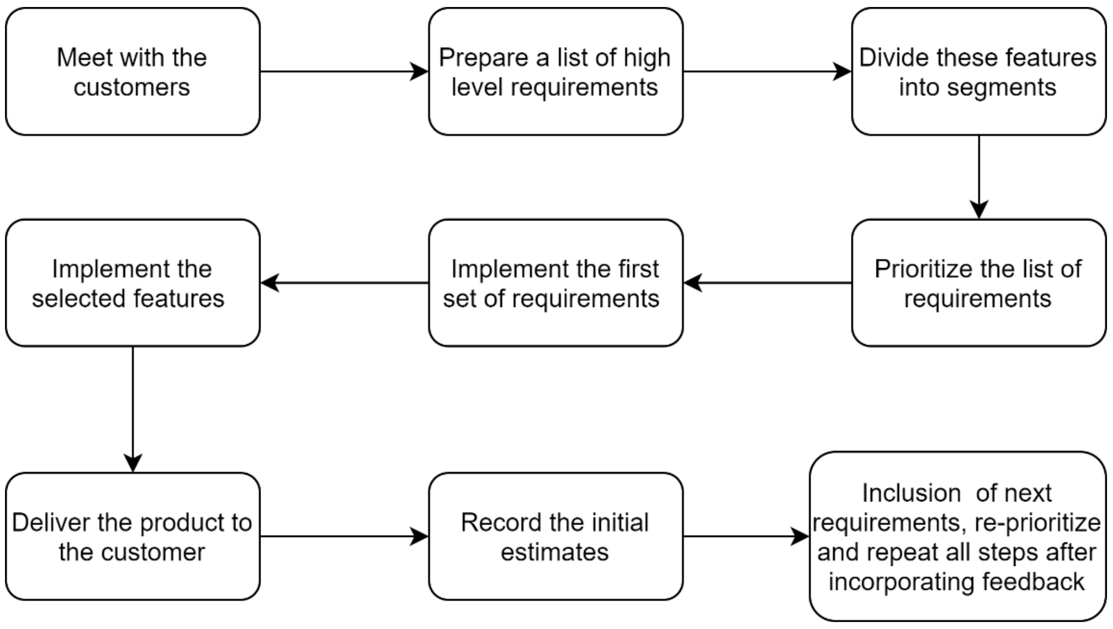

Meet with the customers
Prepare a list of high level requirements
Divide these features into segments
Prioritize the list of requirements
Implement the first set of requirements
Implement the selected features
Deliver the product to the customer
Record the initial estimates
Inclusion of next requirements, re-prioritize and repeat all steps after incorporating feedback

Figure 1.7 Steps followed in Agile Software Development

1.4 Effort estimation approaches and framework

For the successful and effective execution of the project, SDEE plays an indispensable role in SDLC. It is the process of anticipating the amount of effort required to develop the software at the beginning of the project. It is very much essential for Software Project Management. The estimating framework is given in Figure 1.8.

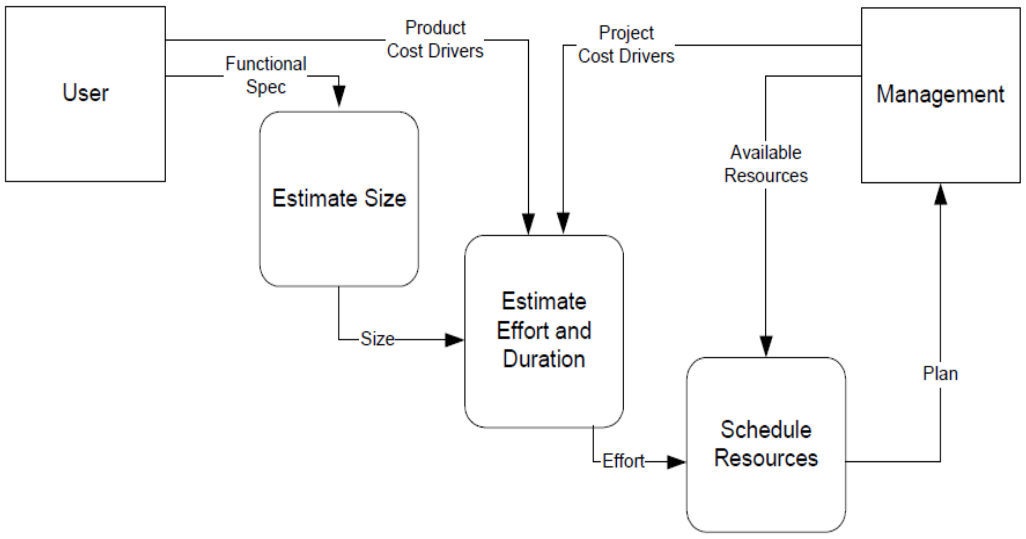


Figure 1.8 Software estimation process flow

1.4.1 Size estimating methods

As per a survey, among SP, UCP, FP, OP, and LoC, SP is the preferred choice in the IT industry. The details for size estimation techniques and their usage are given in Table 1.1

Table 1.1 Usage trend for size estimation techniques [1]

Size estimation techniques	% Usage in industry
SP	61
UCP	16
FP	28
OP	1
LoC	11

SP approach is mostly used in the IT industry as compared to other techniques. Various estimating methods are being used and employed by industries during the estimation process. As shown in Table 1.2, various metrics are used to measure the requirements in context.

Table 1.2 Requirement size estimation metric

Size metric	Definition and industrial presence
SP	ASD uses SP as a measuring metric and is a relative unit of measurement. It estimates the difficulty of implementing the user story. Many people use the Fibonacci sequence for estimating. PP can also be used to estimate SP. It is a variation in Delphi. Agile has an advantage over others as it takes units in their relative form instead of absolute thus giving an edge of comparing things easily.
FP	FP is based on the count of functions on which the size of the software depends. Function points can be calculated with a total of five characteristics i.e., Number of inputs, number of outputs, Inquiries count, External interfaces, and external logical files.
UCP	UCP extends FP. It is based on use case analysis. It has three types of actors – Simple, Average, and Complex, and the weighting factor is assigned to them as 1, 2, and 3 respectively.
OP	OP is used in COCOMO-II wherein objects are considered as modules and reports of the programming language for which effort estimate is desired.
LoC	LoC simply refers to estimate the project by tallying the lines of code in some source code.

1.4.2 Effort estimation approaches

Effort estimation can be carried out in various ways. The majority of the IT industries rely on empirical estimation wherein an educated guess is made to get the desired effort estimate. Few traditional and machine learning estimation techniques have been categorized in Table 1.3 with their associated usage, pros, and cons.

Table 1.3: Comparison of effort estimation approaches

Estimation techniques	Category	Usage	Pros	Cons
Estimation by Analogy	Formal estimation model/non-Algorithmic	Weighted micro function points	The estimation result is unequivocal.	Improbable estimates; Dependable on preceding projects
PP	Expert estimation	Group estimation	Most popular/ accepted in industries	Less research is done, so little empirical evidence is available.
EJ	Empirical estimation	An educated guess	Fast result	Suffers from individual bias
Delphi estimation	Group estimation	Wideband Delphi	The collective opinion of estimators.	No Analytic foundation
COCOMO and COCOMO-II	Heuristic approach	Parametric models	Clear results	Much data is required for estimation.
UCP	Formal estimation model	Size based	Useful for predicting initial estimates	Product backlog oblivious to few conditions.
MUCP	Formal estimation model	Size based	Based on requirements from UCP, Project managers, etc.	Same issues as UCP.

LR, RBFNN	Parametric/ Model-Based	Broad-spectrum	Compatibility with existing data	Accuracy declines due to low historical data availability.
NN (SVM)	Parametric/ Model-Based	Broad-spectrum	High performance during on clamorous input data	NN is a Black box with representation.
TD and BU estimation	EJ	Project management software	Historical data feasibility with excellent performance	Less empirical evidence, TD is better in contrast to BU.

1.5 Agile Estimation: Inception to Transition

Agile is flexible, so it is a bit difficult to carry out effort estimation in it. We have already discussed that in agile, customer requirements are listed in the form of issues/user stories. An issue is a high-level definition of a requirement holding enough information for its estimation, development, and testing. However, a user story must be implementable in a single iteration otherwise if not, the requirement is broken down into smaller stories. The collection of stories is known as Product Backlog. The units of work can be categorized in two ways viz., Real-time units (hrs, days, etc.) and abstract units (SP, Ideal days).

The customer prioritizes the stories. The priorities could be in terms like "high, medium, low", "Definitely needed, needed, and nice to have" or just numbers with higher numbers indicating higher priority. The responsibility of estimating the effort to implement the stories rests on the project team. Once prioritization and estimation are done, the customer's need to go to time-to-market is decided upon, and a tentative release date is arrived at.

1.5.1 A decade of Agile estimation techniques

Considering a decade of agile effort estimation research and the comparative accuracies (in general) is achieved by different estimation techniques is given in Table 1.4.

Table 1.4 Accuracy metrics for various estimation algorithms [2]

Estimation Algorithms	Accuracy Metrics	The accuracy achieved (in % age)
NN	MRE	34.5
	MMRE	41.65
	PRED (100)	91.535
	MAPE	39.78
	MdMRE	89
	R2	77.5
	MSE	1.25
EJ	MRE	27.675
	MdMRE	45
PP/Disaggregation	MRE	52.6
UCP	MRE	15.5
	MMRE	20.75
	PRED (100)	76.955
	R2	90.6
	MSE	8
	MMER	72.85
MUCP	MRE	9.45
	MMRE	10
	PRED (10)	90
	PRED (20)	60
	MdMRE	9
LR using MUCP	MRE	34

	MMRE	24.86
	PRED (100)	94.35
	MdMRE	62.76
	MMER	65.066
Wideband Delphi (using LR)	MRE	8.85
BU/TD	MRE	39

1.5.2 Environment for different estimation algorithms

The estimation algorithms have different impacts in different environments. For Project A the estimation technique E may result differently than Project B as there may be some factors considered in Project A that are not needed in Project B. Also, the in-house resources impact the effort. Team members' coupling and commitment to meet the estimated effort play a major role in the successful and timely competition of a project. There may be a case when a team member leaves his job in between the project so re-work is required to re-estimate the effort. The estimation algorithms and their associated challenges are given in Table 1.5

Table 1.5 Challenges in various estimation algorithms [2]

Estimation algorithms	Challenges
NN	Due to a lack of explanation ability and neural networks' inherent black box characteristic, the estimated effort may not be accepted by IT stakeholders. Also, in some cases, the performance is below par vis-à-vis expert judgment technique.
EJ	IT expert's technical experience is subject to their learning curve and cannot be generalized. IT projects effort and costs estimated by an expert are biased due to their different perception of project size and complexity.

PP	PP has been a widely used technique for Agile estimation but does not yield accurate estimates.
UCP	Due to Agile Scrum project characteristics, UCP cannot be utilized in its standard form.
MUCP	Due to Agile Scrum project characteristics, UCP cannot be utilized in its standard form.
LR	Regression analysis heavily relies on past project data which is a challenge in Agile as very little data is available for experimental analysis on the public repositories.
Wideband Delphi	Delphi cost estimation introduces bias in estimation-related decisions.
BU/TD	Not much evidence is available in the literature.

1.5.3 Accuracy parameters

The various accuracy parameters used to check the efficacy of effort and cost estimation techniques are given below [3].

- Magnitude of Relative Error (MRE) is the most common criteria for estimation techniques. It assesses every project in a dataset individually. The following equation represents MRE.

$$MRE_i = \frac{|Actual\ Effort - Estimated\ Effort|}{Actual\ Effort} \tag{1.1}$$

- Mean Magnitude of Relative Error (MMRE) measures percentage values of relative errors. The calculated % age value is the average value over the N items. It is given in the following equation.

$$MMRE_i = \frac{1}{N}\sum_{i=1}^{N} MRE_i \tag{1.2}$$

- Median Magnitude of Relative Error (MdMRE) It measures the Median for MRE(s) and has acceptance criteria that is not sensitive to outliers. It is given in the following equation.

$$MdMRE_i = Median\ (MRE_i) \tag{1.3}$$

- Mean Magnitude of Error Relative- It is used for cost estimation and is calculated by the median of Magnitude of Error Relative (MER). It is given in the following equation.

$$MMER_i = \frac{1}{N}\sum_{i=1}^{N} MER_i \tag{1.4}$$

- Mean Absolute Percentage Error- It determines absolute accuracy for different estimation models. The term absolute is considered as the assessment of the cost estimations from the actual recognized costs. MAPE can be calculated using the following equation.

$$MAPE_i = \frac{1}{N}\sum_{i=1}^{N} \left| \frac{Actual\ Effort - Estimated\ Effort}{Actual\ Effort} \right| * 100 \tag{1.5}$$

  In this, the first summation is done for each estimated point, divided by the number of suitable points N.

- Mean Squared Error- For calculating MSE, the following

$$MSE_i = \frac{1}{N}\sum_{i=1}^{N} (Actual\ Effort - Estimated\ Effort)^2 \tag{1.6}$$

  where N is the available data in the dataset. It is directly proportional to PRED (x).

- Balance Relative Error- It is more balanced than MRE in terms of overestimation and underestimation and can be calculated using the following equation.

$$BRE = \frac{|Actual\ Effort - Estimated\ Effort|}{min(Actual\ Effort, Estimated\ Effort)} \tag{1.7}$$

- Squared Correlation Coefficient- It is used and defined to assess the efficacy of regression. It can be represented using the following equation.

$$R^2 = 1-\frac{\sum_{i=1}^{N}(Actual\ Effort - Estimated\ Effort)^2}{\sum_{i=1}^{N}(Actual\ effort - Mean(Actual\ Effort))^2} \tag{1.8}$$

- Prediction (PRED (x)) – In mathematical definition, PRED(x) is mathematically determined as:

$$PRED(x) \equiv \sum_{i=1}^{N}[MRE_i \leq x]) \mid N > 0 \tag{1.9}$$

PRED(x) value is calculated using the following equation.

$$PRED(x) = \frac{K}{N} \tag{1.10}$$

Here, ‘N’ represents the total of projects and ‘K’ is the count of projects having MRE below or equal to x. The value of x can be either 0.25, 0.50. 0.75 or 1.0. If a common value of x is 0.50, then PRED (0.50) refers to the % of projects whose MRE is less than or equal to 50%. Measuring the accuracy of estimation in scrum is an essential activity and determines its superiority with self and others.

1.5.4 Effort estimation factors

The agile effort is affected by work and project resources which are given in Table 1.6. The story point estimation factors and their tiers are given in Table 1.7. Table 1.8 and Table 1.9 show the velocity rating and complexity rating factors respectively. Table 1.10 describes the story size scales while Table 1.11 describes the User story/Issues complexity scale. Table 1.12 and Table 1.13 list the decelerating factors and dynamic force factors respectively.

Table 1.6 Work-Resource and Project factors [4]

Estimation Factors	
Work Resource Related factors	Project-related factors
Project type	Communication Skill
Non-Functional Requirements (NFR)	Familiarity in the team
Software and Hardware Requirements	Management skill
Operational feasibility analysis	Safety and Security

Complexity	Work shift hours
Information transaction	Past project experience
Site info	Technical skills

Table 1.7 Effort Estimation factors [5]

Story point estimation factors	Tiers
TE	Fresher, Intermediate, Advance, Expert
TS	Small, Medium, Large, Extra-Large
Task Complexity	Easiest, easy, moderate, complex, arduous
Estimation Accuracy	Over-estimated, well-estimated, under-estimated

Table 1.8 Velocity rating factors [6]

Factor level	Rating	Type of project
Low	0.94—0.98	The project is simple. For example, requirements are very straightforward, no volatility of requirements, all business and technical requirements are very clear to the team with no uncertainty, no research required in the project and it requires basic programming skills to complete.
Medium	0.90—0.94	The project is Moderately complex. For example, it requires little or no research and the team has strong expertise in allotted work.
High	0.85—0.89	The project is extremely complex and demands accurate estimates by consideration of all the factors at a high level.

Table 1.9 Complexity rating factors [6]

Level of factor	Rating	Type of Project
Low	1	The project is simple. For example, requirements are very straightforward no volatility of requirements, no research required in the project, only rudimentary programming skills for completion with a clear understanding of technical requirements. There is no product uncertainty, process uncertainty, and resource uncertainty. A team of the right ability and experience is available.
Medium	3	The project is moderately complex. For example, it requires little or no research, and the team has strong expertise in the allotted work.
High	5	The project is extremely complex and demands accurate estimates by consideration of all the factors at a high level.

Table 1.10 Story size scales [6]

Value	Guidelines
5	The value "5" signifies some too large story to be accurately estimated. The story has to be fragmented into smaller stories and suitable to be divided a candidate for a new project.
4	The value "4" signifies a very large story that requires a developer's efforts for an extended time (beyond a week).
3	The value "3" signifies a moderately large story that requires approx. 2-5 days of work to be completed.
2	The value "2" signifies a medium-sized story that requires 1-2 days of work

1	The value "1" signifies a very short story that can be completed in few hours and requires very little effort.

Table 1.11 User story/Issues complexity scale [6]

Value	Guidelines
5	The value "5" signifies an extremely complex story with high system interdependencies and vital expertise lacking in the developer group. Accurate interpretation of a story is tough due to the presence of large unknowns. Thus, extensive refactoring and research, and delicate judgment calls are required. The story is significantly impacting itself externally.
4	The value "4" signifies a very complex story with high system interdependencies and vital expertise weakly present in the developer group. The story described by the product owner may not be accurately represented due to the presence of large unknowns. Thus, the story should have comparatively wider refactoring, research, strong programming skills, and delicate judgment calls. The story has a moderate impact externally on itself.
3	The value "3" signifies a moderately complex story with moderate system interdependency with the presence of a strong skill set or experience within the developer group. It is a little challenging for the product owner in providing an accurate description of the story due to the presence of some unknowns. Thus, little refactoring, intermediate programming skills, research, and potent judgment calls. The story has a minimal external impact on itself.
2	The value "2" signifies a coherent technical and business requirement that requires minimal research as the presence of unknowns are few. Thus, rudimentary programming skills are required to complete the story description. The story is localized to itself.

1	The value “1” signifies a completely accurate story description with unambiguous technical and business requirements and minimal unknowns. Thus, there is no research required, and basic skills in programming can be utilized. The story is localized to itself.

Table 1.12 Decelerating factors [6]

Decelerating factors	Normal	Volatile	Highly Volatile	Extremely Volatile
Team Composition	1.00	0.99	0.94	0.89
Process	1.00	0.99	0.95	0.91
Environmental Factors	1.00	0.98	0.96	0.95
Team Dynamics	1.00	0.99	0.89	0.86

Table 1.13 Dynamic force factors [6]

Variable Factor	Normal	High	Very High	Extremely High
Expected Team Changes	1.00	0.99	0.97	0.95
Introduction of New Tools	1.00	0.98	0.96	0.94
Vendor’s Defect	1	0.99	0.96	0.89
Team member’s responsibilities outside the project	1	0.98	0.97	0.95
Personal Issues	1	0.98	0.97	0.96
Expected Delay in Stakeholder response	1	0.99	0.98	0.96
Expected Ambiguity in Details	1	0.99	0.98	0.97

Expected Changes in environment	1	0.98	0.97	0.90
Expected Relocation	1	0.98	0.96	0.92

1.6 Estimation in Scrum

Scrum is flexible, so effort estimation becomes challenging. Customer requirements in Scrum are listed in the form of user stories. A user story represents requirements in some standard form. However, a user story must be implementable in a single iteration otherwise if not, the requirement is broken down into smaller stories. The collection of stories is known as Product Backlog. The units of work can be categorized in two ways viz., Real-time units (hours, days, etc.) and abstract units (SP, Ideal days). The customer prioritizes the stories. The priorities could be in terms like "high, medium, low", "Definitely needed, needed, and nice to have" or just numbers with higher numbers indicating higher priority. The responsibility of estimating the effort to implement the stories rests on the project team. Once prioritization and estimation are done, the customer’s need to go to time-to-market is decided upon, and a tentative release date is arrived at. A typical scrum estimation process is shown in Figure 1.9.

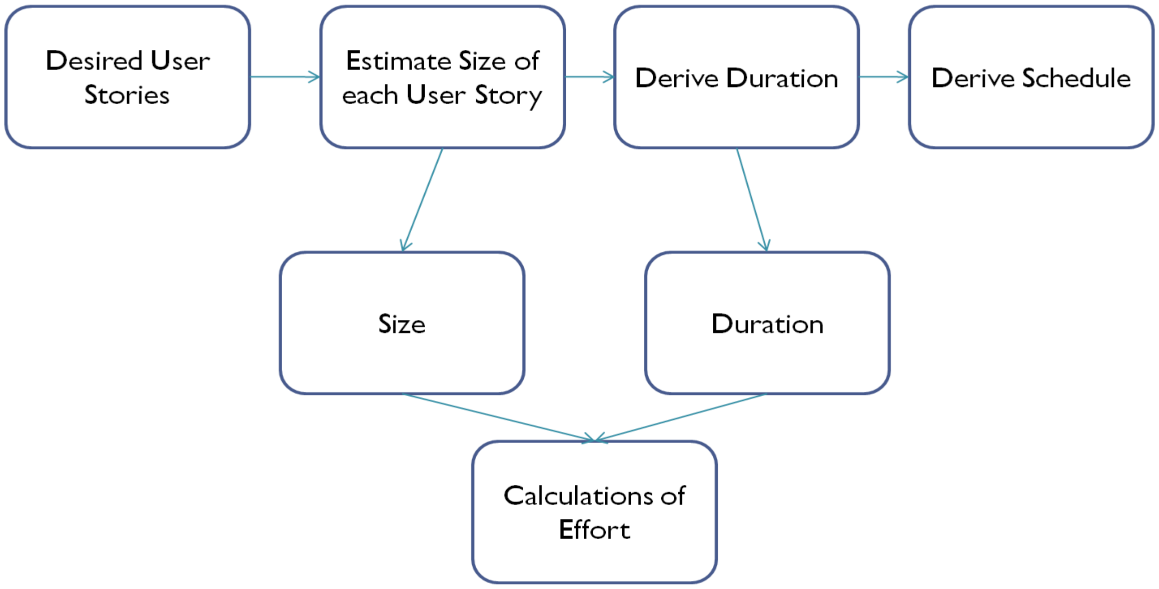


Figure 1.9 Estimation in Scrum (Overview)

Estimation in scrum is carried out in two ways i.e., (i) using numerical inputs, (ii) text data. The most important inputs in the first category are the calculated story points and the scrum team's velocity. Velocity is a measure of the efforts completed in a single sprint through the scrum team. A sprint in scrum represents iterations. In the second case, the input is user story text/words which can be extracted from issue tracking systems.

Based on one survey conducted by M. Usman et al. [7] data from sixty agile practitioners was collected to find which is the most used estimation technique and is tabulated in Table 1.14.

Table 1.14 Widely used effort estimation techniques in industries [7]

Parameters of Study	Techniques	Percentage practiced by Agile practitioners
Estimation Techniques	PP	63%
	Analogy based estimation	47%
	EJ	38%
Size metrics	SP (used solo or in combination with FP)	62%
Cost Drivers	Team's experience and expertise level	Mostly used

1.6.1 Scrum estimation approaches

As per the standard industry norms and related literature, scrum estimation has been carried out by various approaches given below:

- Expert-based approaches: It relies on human expertise like expert judgment.
- Model-based approaches: It relies on the previous data to predict new data.
- Hybrid approaches: It combines both expert and model-based approaches.

Scrum estimation as cited above primarily relies on the story point metric. It is the widely used metric for estimations in the Agile industry. ML models of estimation are performing well as compared to an educated guess by an IT expert. Various ML models have been used like SVR, RF, DT, ANN, NF, DBN, optimization techniques (like GA, SBO, Adaptive Fireworks, ALO, and PSO), etc. This classical problem of estimation in scrum has been categorized into two parts given below:

- Numerical data (Data points are numerical values as inputs)
- Text classification (Data points are text/words as inputs) [8]

In both scenarios, labels will always be a numerical value.

1.6.2 Scrum estimation challenges

We can't rely on fixed models of ML-based estimation for estimating the effort of Scrum projects. They give good results in the initial estimates but differ to vary after successive sprints. The various estimation challenges in the field of scrum projects are given below:

- User stories or Issue volatility.
- No standard scale for issue size and its complexity.
- No continuous estimation.
- Non-inclusion of effort of quality requirements.
- Issue's complexity
- Too many parameters affecting effort.

1.6.3 Traditional estimation to Machine Learning assisted estimation

The major reasons for this transition are given below:

- Suffered from individual bias
- Poor estimation accuracy leads to wrong client commitments
- More time to reach a consensus
- User story complexity

1.7 Motivation

Software Project Management is an important aspect of software development. However, it has always been manual; hence it is prone to human error. The state of software project management currently suffers from inaccurate estimation, bias, and non-adaptiveness to rapid changes, either in requirements or environment. As a result, the time and cost of development increases, burdening the Agile team to fast-track the delivery process. Even though Agile is very flexible to the rapid changes, it is largely dependent on human expertise and efforts. Experience is a subjective quality that depends entirely on human factors. We cannot train two or more humans for the same experience in the same time frame and be assured that they will have the same learning. An experienced software engineer who has been in various business environments will be more robust than those who work in a single domain.

Our motivation stems primarily from these factors:

- The human factors can be automated with modern computation techniques.
- Human errors can be lessened by choosing to depend on computer logic.

To develop such a system of estimation for assisting a human is the need of the day. Fortunately, with the advancements in internet infrastructure, hardware, and software, such a system is feasible. The modern machine learning and deep learning techniques are a boon for us to automate the system and develop an intelligent assistant algorithm to aid the software development process.

1.8 Research objectives and main contributions

The research objectives are given below.

Research Objective 1

To review and analyze existing effort estimation techniques for Agile-based projects.

Research Objective 2

Design and development of a hybrid effort estimation model for agile-based software projects.

Research Objective 3

Design and development of a cost estimation model for agile-based software projects.

Research Objective 4

Comparative analysis of existing effort and cost estimation approaches with the proposed models.

Our main contribution is to the field of Agile software project estimation, using hybrid machine learning algorithms. An in-depth analysis of the existing effort estimation techniques reflects on the current state of Agile project estimation. We have utilized a hybrid of machine learning and fuzzy logic called an NF system. The most prominent is ANFIS, which is capable of modeling human logic and human learning efficiency, to automate complex human tasks. Thus, we have contributed to widening the applicability scope of ANFIS. We also contributed to the field of meta-heuristic optimization by developing a nature-inspired optimization algorithm to improve the learning of standard ANFIS. We have compared our implementation to several state-of-the-art algorithms and provided substantial information to assess our model's advantages over them.

1.9 Thesis Organization

This section provides an overview of the chapter contents and their organization in the thesis.

Chapter 2 describes the background study for our research. This chapter provides an outlook of the current scenario in our domain of interest: Scrum project estimation. We have discussed the machine learning techniques used in the domain; the background of Adaptive Neuro-Fuzzy Inference Systems, its architecture, and implementation; the description of standard Bat algorithm, its implementation, shortcomings, and the various hybrids developed;

Chapter 3 presents the literature review of the effort and cost estimation in Scrum projects using a variety of approaches. This chapter details the study selection process, inclusion and exclusion criterion of the literature, and the data and literature sources description; literature review summarizing the work of several authors on the usage of traditional, machine learning, fuzzy system and hybrid techniques to predict the effort or cost, their advantages, and disadvantages; formulation and discussion of research questions.

Chapter 4 presents our developed technique for effort estimation. This chapter meticulously describes in various sections our architecture from data preparation to the estimated value of effort; the dataset description, feature selection, data synthesis and transformation; the challenges in adopting standard Bat algorithm for effort estimation; our novel structure modification to standard Bat algorithm in developing its hybrid; its pseudo-code and implementation in ANFIS; and the results of our developed technique against several state-of-the-art algorithms.

Chapter 5 presents our developed technique for cost estimation. This chapter thoroughly describes in various sections our architecture for estimating the cost of a Scrum project; feature selection and transformation of the dataset; the challenges in adopting standard Bat algorithm for cost estimation; description of our novel feature that addresses the challenge of adopting modern machine learning techniques for cost estimation; its pseudo-code and implementation in ANFIS; and the results of our developed technique against several similar optimization algorithms implemented in ANFIS.

Chapter 6 presents the conclusion of our thesis and the future scope of the research conducted. This chapter summarizes the challenges in traditional techniques of estimation; the transition to machine learning techniques, present literature of work and results; advantages and disadvantages of using ANFIS; requirement of meta-heuristic optimization algorithms in assisting with hyper parameter tuning; design and development of EEBAT, its integration and results relevant to effort estimation; design and development of CEBAT, its integration and results relevant to cost estimation; time and space complexity analysis against several other algorithms; and the future scope of work in the software development estimation field.

CHAPTER 2

BACKGROUND AND PRELIMINARIES

The underlying architecture of the proposed approach has been inspired by the universal estimator i.e., ANFIS. Standard ANFIS [9] has promising solutions for problems of heavy weight process models in context to software estimation. ANFIS has some inherent pros and cons, which makes it a little less efficient for estimating in an Agile environment if applied as a standard. Some shortcoming of ANFIS includes high computational cost due to structural complexity and gradient learning hence for large inputs it will be slow, type, location and no. of membership functions, the curse of dimensionality and complexity-interpretability trade-off. As Agility means injecting 'change', a de-facto ingredient in reshaping the culture of software engineering, it becomes a mandate to optimized ANFIS hyper parameters to predict and adjust the Scrum project's effort during all prominent sprints. EEBAT technique will tune the ANFIS parameters. Related background work has been discussed in this chapter.

2.1 Machine Learning techniques

Various techniques are being used over the years for estimating the effort in both agile and non-agile environments. In agile projects, two very commonly used metrics influence the project's growth efforts, one that defines the size and complexity of the project called the story points, and the other that defines the total number of story points that can be conveyed by the team in a sprint called the project team's velocity. We can estimate efforts that are required by the software project using the agile approach efficiently based on the two factors- number of story points and team velocity. The estimation process for user stories in the backlog is discussed in a sprint planning meeting, after which, the product owner prioritizes the item effectively based on the team's velocity. A very important factor in this process is to reduce any kind of influence on the team and to successfully practice the exercise. Several machine learning techniques/hybrids that have been used in Agile project estimation are briefly discussed:

- **XGBoost** [10]**:** The XGBoost algorithm is built upon tree boosting algorithms. It is an end-to-end scalable, ensemble system that uses data compression and gradient boosting framework. It comparatively uses lesser resources as compared to other ensemble learning methods. It is widely accepted to be an improvement over DT and RF and thus preferred in machine learning challenges.
- **CatBoost** [11]**:** CatBoost is developed by Yandex Inc as a new gradient boosting toolkiSt that used ordered boosting, based on permutation for processing categorical features. CatBoost can utilize both CPU and GPU natively thus extends scalability over traditional machine learning techniques. It rivals XGBoost in performance and accuracy.
- **FLANN-WOA** [12]**:** It is a hybrid algorithm that combines FLANN having 3 layers with a set number of nodes and then those nodes are being multiplied by the calculated weight vector by combining WOA.
- **RBFN-WOA** [12]**:** RBFN has 3 layers just like the FLANN algorithm. The first layer of RBFN has input neurons that provide input data whereas the Gaussian RBF generates the middle layer. The final output is obtained as the weighted sum which is calculated by WOA and then multiplied by the nodes of the middle layer.
- **DBN-ALO** [13]**:** The number of knots in the proposed DBN-ALO architecture has five RBM's for traditional inputs whereas 3 for agile inputs. Just one node is available to attempt the output layer. The input is linked and evaluated according to the training algorithm in the visible DBN layer. Three RBM stacks were relocated. The effort is calculated as a linear amount of the final RBM output at the output layer.
- **SVR-RBF** [14]**:** SVR has been designed to address regression problems. Oliveira initially investigated the application of SVR to estimate the cost of software projects. Kernel learning algorithm uses a popular kernel function in machine learning called the radial base function which is also used for the classification of vector machines.

- **ABC-PSO** [15]**:** A novel method was proposed for the calculation of the commitment in agile development projects focused on velocity and the story points. A mixed variant of ABC and PSO algorithms had been implemented as well for getting better results.

2.2 Standard Adaptive Neuro-Fuzzy Inference Systems [9]

ANFIS, popularly known as a universal estimator is based on Takagi-Sugeno Fuzzy System and makes use of potentials of neural network and fuzzy logic altogether in a package. It is computationally more efficient than Mamdani, which mostly depends on expert knowledge. The architecture of a standard ANFIS is given in Figure 2.1 and its equations are given in Table 2.1. It has primarily five layers with functionalities as follows:

- **Fuzzifying Layer**: All neurons are adaptive nodes including premise parameters.
- **Implication Layer**: Each neuron contains the product of all inputs.
- **Normalizing Layer**: Each neuron is fixed.
- **Defuzzifying Layer**: All neurons are adaptive nodes including consequence parameters.
- **Combining Layer**: One neuron containing the sum of all inputs.

Table 2.1 Standard ANFIS equations [9]

Error Measure	$E_p = \sum_{m=1}^{L} \left(T_{m,p} - O_{m,p}\right)^2$ (2.1)
Error rate for the output node	$\frac{\partial E_p}{\partial O_{i,p}^{L}} = -2\left(T_{i,p} - O_{i,p}^{L}\right)$ (2.2)
Generic parameter alpha	$\Delta\alpha = -\eta \frac{\partial E}{\partial \alpha}$ (2.3)
Learning rate with ‘k’ step size	$\eta = \frac{k}{\sqrt{\sum_{\alpha}\left(\frac{\partial E}{\partial \alpha}\right)^2}}$ (2.4)
Unknown Vector	$X_{i+1} = X_i + S_{i+1}a_{i+1}(b_{i+1}^{T} - a_{i+1}^{T}X_i)$ (2.5)
Covariance Matrix	$S_i = S_i - \frac{S_i\, a_{i+1}a_{i+1}^{T}S_i}{1+ a_{i+1}^{T}S_i a_{i+1}}, i = 0,1, \ldots p-1$ (2.6)
Gaussian Function	$\mu_{A_i}(x) = a_i \exp\left[-\left(\frac{x-c_i}{a_i}\right)^2\right]$ (2.7)

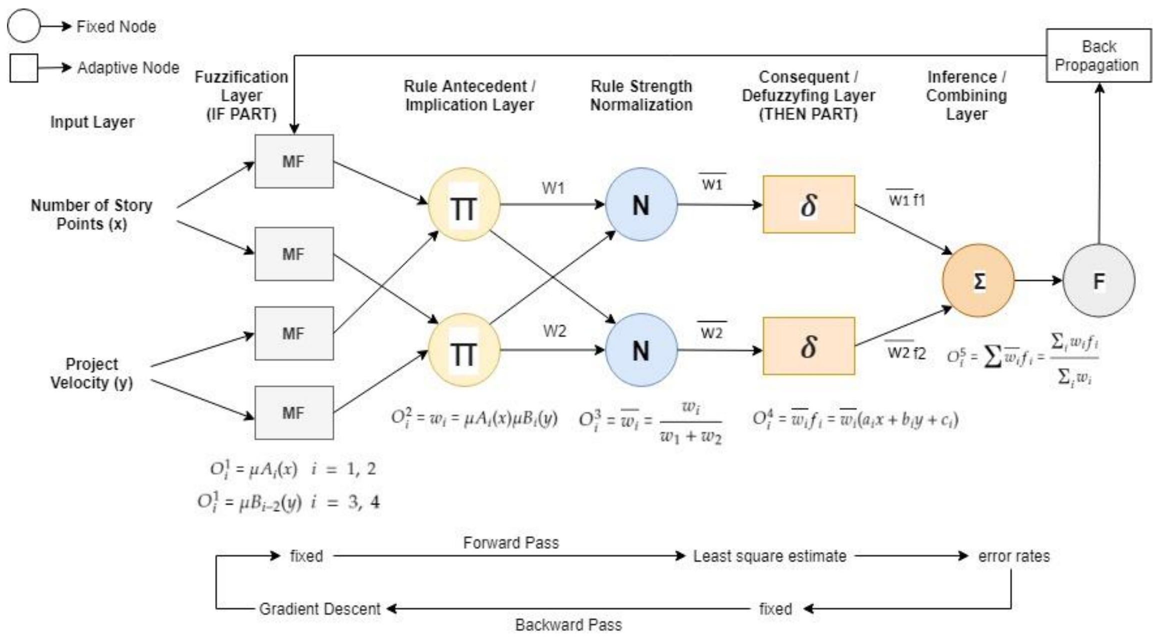

Fixed Node
Adaptive Node
Input Layer
Fuzzification Layer (IF PART)
Rule Antecedent / Implication Layer
Rule Strength Normalization
Consequent / Defuzzyfing Layer (THEN PART)
Inference / Combining Layer
Back Propagation
Number of Story Points (x)
Project Velocity (y)
MF
Π
N
δ
Σ
F
W1
W2
$\overline{W1}$
$\overline{W2}$
$\overline{W1}$ f1
$\overline{W2}$ f2
$O_i^1 = \mu A_i(x) \quad i = 1, 2$
$O_i^1 = \mu B_{i-2}(y) \quad i = 3, 4$
$O_i^2 = w_i = \mu A_i(x)\mu B_i(y)$
$O_i^3 = \overline{w_i} = \frac{w_i}{w_1 + w_2}$
$O_i^4 = \overline{w_i} f_i = \overline{w_i}(a_i x + b_i y + c_i)$
$O_i^5 = \sum \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$
fixed
Forward Pass
Least square estimate
error rates
Gradient Descent
Backward Pass
fixed

Figure 2.1. Standard ANFIS Architecture with defined inputs

The detailed supposition stages of the ANFIS model are described below and shown in Figure 2.2.

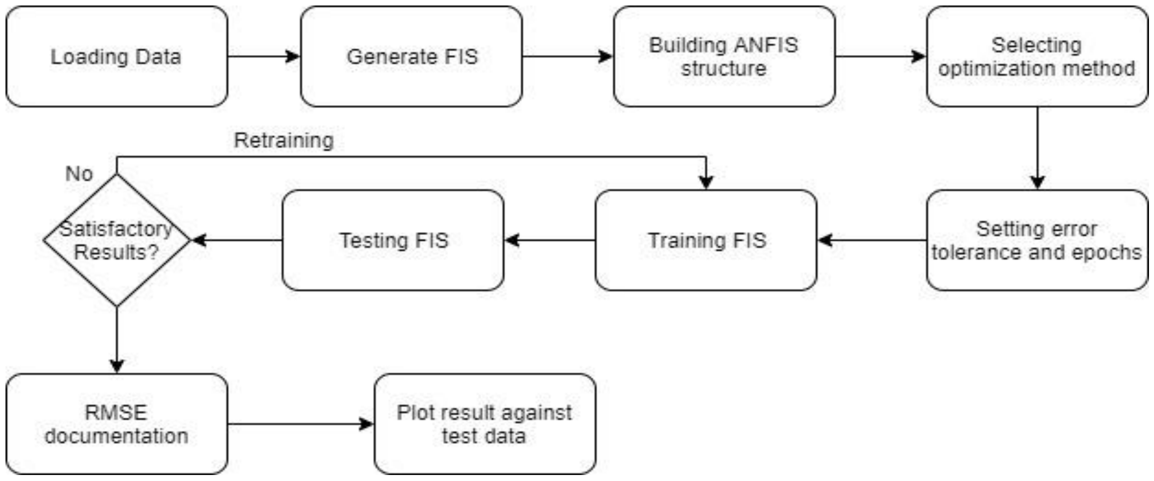


Figure 2.2 Stages of ANFIS

This model includes two stages given below.

Stage 1: Obtain the optimal parameters for ANFIS

- Loading training and testing data

- Generate base FIS
- Setting base FIS parameters
- Parameter adjustment of base FIS
- Outputting optimal values as the result

Stage 2: Using the optimized parameters in ANFIS

Updated parameter values obtained from optimization algorithm added in the fuzzy system and calculation of the system error metrics. Table 2.2 describes the standard ANFIS hybrid learning method.

Table 2.2 Forward and Backward pass in ANFIS hybrid learning method

Legends	Forward Pass	Backward Pass
Antecedent Parameters	Fixed	GD
Consequent Parameters	LSE	Fixed
Signals	Node Outputs	Error rates

2.3 Standard Bat algorithm

There are many meta-heuristic algorithms but we made use of the Bat algorithm for our proposed work. Bat algorithm is a metaheuristic optimization algorithm developed by Xin-She Yang [16] in 2010. The biological inspiration for the algorithm is the hunting instincts of micro-bats for food in some regions using one type of sonar, called echolocation. Bat species emit waves in the ultrasonic spectrum of waves, having some frequency and loudness, at some rate, towards the food which hits it and reflects to form an echo. Through the use of echoes, the bats analyze the distance of the food from its position. This is because the wave emitted at time step t reflects towards the bat at some time step t' and the difference in time step (t' - t) is used to assess the distance of the food from the bat. An individual bat from the population randomly flies towards a position P_i with velocityV_i, varying wavelengthλ, loudnessL_0, fixed frequencyf_{min}, and pulse emission rate$r \in [0, 1]$. The parameters – wavelength and rate of pulse emission rate can be automatically adjusted based on a target's proximity. Loudness $L \in [L_0, L_{min}]$ where L_0 is a largely positive and L_{min} is minimum constant. The difference between obstacles and targets is an inherent property of the bat species. The standard Bat algorithm is shown in Figure 2.3. The steps of the algorithm are as follows:

- **Initialization of the bat population:** The search space is a region of many targets. Every iteration of the algorithm requires the bat to search for optimum target locations for their satisfaction. These locations being unknown, are initialized using the random generation of values through vectors (dimension d and population size n) with subsequent evaluation of their quality.

$$P_{i,j} = P_{min} + \vartheta(P_{max} - P_{min}) \tag{2.8}$$

where $\in [1,2, \dots n]$; $j \in [1,2, \dots d]$; P_{max} and P_{min} are the corresponding highest and lowest boundaries of dimension d; $\vartheta \in [0, 1]$ randomly.

- **Frequency, Velocity, and New Solutions Generation**: The quality of the food sources evaluated during the initialization phase, influence the movements of the bats. The equations for generating velocity, frequency, and position are as follows.

$$F_i = f_{min} + (f_{max} - f_{min})\theta \tag{2.9}$$

$$V_i^t = V_i^{t-1} + (P_i^t - P_*)f_I \tag{2.10}$$

$$P_i^t = P_i^{t-1} + V_i^t \tag{2.11}$$

where f_i is the frequency of ith bat; f_{min} is the minimum frequency; f_{max} is the maximum frequency; θ's value is generated randomly; P_* is the global best solution obtained among the n bats until current iteration; V_i^t is the velocity of bat i at time t.

- **Intensification capability**: The solution so obtained during the exploration phase is used as the central point for local searching of solutions.

$$P_{next} = P_{previous} + \omega\bar{L}^t \tag{2.12}$$

where $P_{previous}$ is one of the elected high-quality solutions through any selection method; $\bar{L}^t$ is the average loudness of all bats at time step t; $\omega \in [-1, 1]$ generated randomly.

- **Updating loudness and pulse emission rate:** Due to the proximity of the target, the Loudness L and pulse emission rate epr are modified and are given as follows.

$$L \propto \frac{1}{epr} \tag{2.13}$$

The equations for updating the parameters are as follows.

$$L_i^{t+1} = \alpha L_i^t \tag{2.14}$$

$$epr_i^{t+1} = epr_i^0 [1 - \mathrm{e}^{-\gamma t}] \tag{2.15}$$

where α, γ are constants.

Standard BAT algorithm [16] has certain inherent issues like failure to converge to global optima, multimodal optimization, poor exploration, slow rate of convergence, and no population diversity. To address these issues, various BAT variants have been introduced by researchers across the globe and are listed in Table 2.3 with their strengths.

The list of inferences that have been deduced from the variants cited in Table 2.3 are:

- Handling trade-off between exploration and exploitation
- Converging to global optima instead of being trapped in local minima
- Flexibility in the integration of the bat variants in different models
- Diversity factor to maintain the distinctness of population
- Improvising the algorithm for multimodal functions

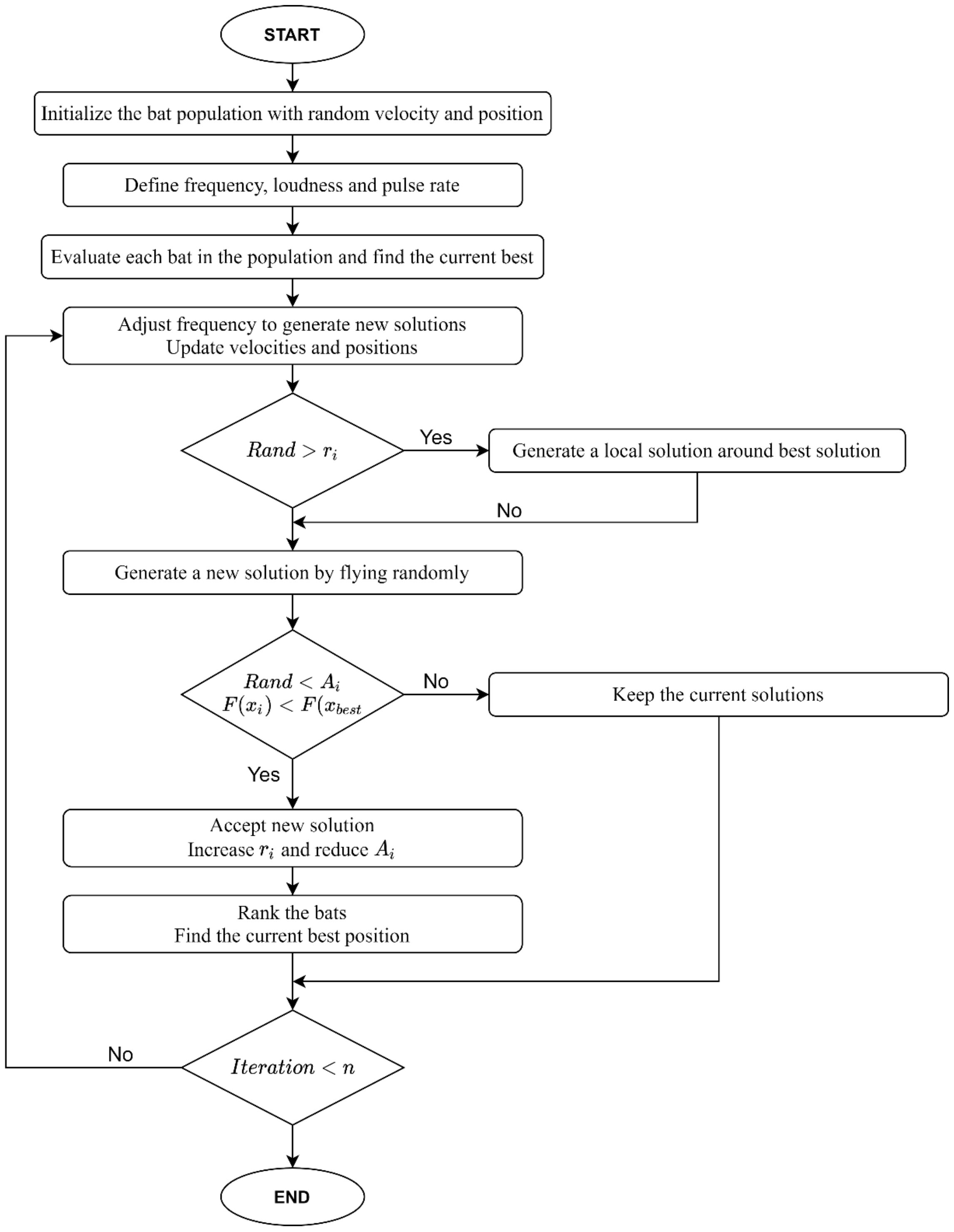

START
Initialize the bat population with random velocity and position
Define frequency, loudness and pulse rate
Evaluate each bat in the population and find the current best
Adjust frequency to generate new solutions
Update velocities and positions
$Rand > r_i$
Yes
Generate a local solution around best solution
No
Generate a new solution by flying randomly
$Rand < A_i$
$F(x_i) < F(x_{best}$
No
Keep the current solutions
Yes
Accept new solution
Increase r_i and reduce A_i
Rank the bats
Find the current best position
No
$Iteration < n$
END

Figure 2.3 Standard Bat algorithm flowchart

Table 2.3 BAT Variants with their strengths

Bat Variants	Strengths
AMBA [17]	Diversification using Opposition Based Learning technique, Swarm selection based on - random, best average fitness, worst average fitness, and farthest swarm best; Swarms can exchange information; Diversification is maintained
Bat with Mutation [18]	Mutation operator increases population diversity that results in improved exploration and a faster global rate of convergence
BatDNN [18]	The sigmoid function used leads to a higher rate of convergence
Binary Bat Algo [19]	Fitness function based on optimum-path classifier improves the convergence rate. Search space divided into n-cube lattice improves exploration and exploitation capabilities, Suitable for feature selection
CBA [20]	Chaotic mapping increases convergence and rate of convergence
Differential Operator & Levy flights Bat [21]	Differential operator improves the rate of convergence; Levy flights creates diversification leading to avoidance of premature convergence; Performs well on complex high-dimensional problems
DABA [22]	Direction scope increases exploration and rate of convergence; Deep checks in the limited area increases exploitation
DLBA [18]	Dynamic Weight model and Levy flight model increase exploration and exploitation; Mutation Probability increases diversity; Adaptive memory ability increases the rate of convergence
DVBA [23]	Division of population into Explorer and Exploiter bat improve convergence, Roles can be interchanged, Exploration and exploitation capabilities are controlled using incremental rate divisor which is dependent on the location of prey
Improved Bat Algo (cost estimation) [24]	Automatic shifting of a global search to local search Random movement prevents local search while in exploration hence increasing the rate of convergence
IBA [25]	Diversity increases exploration capability
LBA [21]	Levy flight behavior improves the exploitation capability, convergence and prevents falling into local optima
LogisticBatDNN [26]	Polynomial mapping is used leading to a very good rate of convergence
MeanBatDNN [26]	Updated equation of velocity based on p_{best} and g_{best} has higher convergence
Modified Bat Algorithm (ANN) [26]	Alpha parameter updates loudness and increasing exploration capability; Diversification leads to a better rate of convergence; Suitable for complex problems

Modified Bat Algorithm (Stability Analysis) [27]	Additional parameter ‘w’ adds stability by lessening restrictions on ‘f’ thus improving convergence; Exploration and exploitation capabilities are balanced
MOBA [28]	Combines the objective functions using weighted sum into a single objective function for solving multimodal problems
Novel bat algorithm with multiple strategies coupling (mixBA) [29]	Multiple strategy autonomous selection strategy using probability
OBMLBA [30]	Levy flight improves exploitation ability; Opposition based Learning improves exploitation ability; Sinusoidal equation allows flexibility in modification of frequency
Piecewise-BatDNN [26]	Piecewise linear chaotic map leads to a higher rate of convergence
SBA [25]	Contraction factor maintains diversity; improves convergence efficiency and prevents falling into local optima
Simplified Adaptive Bat based on the frequency [31]	Frequency adjustment improves convergence and prevents falling into local optima
SinBatDNN [26]	The sinusoidal mapping used leads to a higher rate of convergence

2.4 Summary

The transition from traditional techniques of estimation in Scrum-based projects to soft computing techniques such as machine learning and NF systems is evolutionary. Traditional techniques are human-dependent, as they require considerable experience and related domain knowledge to properly estimate the effort and cost of a project. However, there is an inherent disadvantage of such techniques, which is the primary individual bias that sways the sprint planning and clearance of backlog as a result. It has been concluded that machine learning techniques outperform traditional techniques of estimation.

Though several popular ML techniques are applied in the application of cost and effort estimation in Scrum projects, their performance lags when the non-linearity in data increases. Several authors created hybrids of such techniques which altered the performance positively. We have presented evidence that ensemble estimation techniques win over a single approach of estimation. However, hyperparameter tuning is a primary concern that needs to be addressed when adapting such algorithms to this domain.

The usage of optimization techniques for hyperparameter tuning results in bridging actual and predicted values of effort estimation as compared to manual tuning. Grid Search and Random Search are the most common methods of hyperparameter tuning in machine learning. However, they are time-consuming and require high computational resources. This presents opportunities for developing optimization algorithms that have given promising results. DBN-ALO has maximum accuracy (PRED) and the lowest MMRE on the most used Agile dataset. Using the ALO optimization algorithm, DBN is automatically tuned to provide the best set of estimations of effort in effort and cost estimation.

The advent of NF systems is a boon to the research community as it is most capable of replacing complex human tasks with intelligent automation. We have detailed the most prominent NF system called ANFIS that can be used in Scrum project estimation. We chose this system to develop a more efficient and faster algorithm than DBN-ALO. ANFIS has three fundamental concerns: selection of type and number of membership function, the curse of dimensionality, and interpretability-complexity trade-off. These can be mitigated by substituting another optimization algorithm in the standard ANFIS algorithm to adapt it for Scrum project estimation. We propose the use of a standard BAT algorithm for this purpose.

The standard BAT algorithm is one of the best optimization algorithms that are suitable for enhancing the ANFIS performance. The standard BAT algorithm is very popular among researchers due to its enthralling performance on various datasets. Since it has also suffered from problems like convergence failure for global optima, multimodal optimization, poor rate of exploration, slow rate of convergence, lack of population diversity, and forgetfulness, its variants have been proposed by several authors. We have provided a summary of the best variants to sensitize the readers of its potential in several domains.

CHAPTER 3

REVIEW OF LITERATURE

3.1 Related work

Ravi Kiran et al. (2021) [32] presented various estimation challenges and issues faced by the IT companies incorporating Agile based practices. They conclude, planning poker as the most used technique/model of agile estimation. Authors also listed various pointers leads to inaccurate estimation and software crises.

Athanasios Karapantelakis et al. (2021) [33] made use of MBT for teams incorporating scrum to estimate costs. Authors have created and presented a set of models that can estimate MBT adoption costs based on a number of baseline criteria such as staff competence and availability, as well as historical data utilization.

Neha Gupta et al. (2021) [34] uses the integration technique to evaluate dynamic development efforts based on features and user input. The distributed method's results suggest that the energy expenditures produced from user data are valuable.

Rene Avalloni de Morais (2021) [35] proposed deep learning-based algorithms for estimating story points in agile projects. For story point estimation, 16 open-source projects proposed and train different algorithms on a big dataset. Furthermore, author also employed natural language processing techniques to extract more useful characteristics from software requirements expressed as user stories.

Zainab Rustum Mohsin (2021) [36] in his paper deduced that the artificial neural network technique was used to model software development effort COCOMO was one of the datasets used in the estimation. The MMRE and correlation R were utilized as evaluation tools. Following the construction and testing of the ANN model, and a comparison of the ANN model's test results with those of the SLIM, Function Points, and COCOMO-basic models, it was determined that the ANN was a suitable model for estimating effort. It is also suggested that ANN be utilized as a predictive model for estimating software development effort.

Simone Briatore et al. (2021) [37] in their paper describes a pilot validation experiment of an unique Agile framework for the development of hardware systems, which includes a parametric tool for more rigorously estimating task effort than standard confidence votes. The validation of electrical hardware design task estimation and overall project performance is presented. Experiments with teams of junior engineering students are used to validate the system. When using the offered tool during the planning phases of the development, the validation experiment revealed an improvement from a minimum of 8% to a maximum of 18%.

Kasi Periyasamy et al. (2021) [38] explains the design and execution of a project tracking tool for software projects built using the agile method Scrum. The tool's users may keep a close eye on the progress of user stories, sprint tasks, and test cases that have been added to a scrum board. The tool provides a measure of difficulty to implement in terms of story points for each user story, as well as the expected completion time for each sprint job. For ongoing monitoring of efforts based on sprint tasks, the solution employs machine learning support. Three separate graduate course projects were used to test the tool's effectiveness.

Przemyslaw Pospieszny et al. (2018) [39] use three approaches for effort estimation in the ISBSG dataset: SVM, MLPANN, GLM. They inferred SVM outperforms ANN and GLM by having lesser MMRE and higher PRED (25). Even when the dependent variable is log-transformed, GLM outperforms ANN, but SVM still wins. According to the authors, results may vary due to the presence of several project data in ISBSG, and things may be different if applied to a homogeneous set, such as PROMISE or from source forge.

Morakot Choetkiertikul et al. (2017) [8] estimated the effort of user stories rather than the whole project. The authors made use of hybrid LSTM and RHWN and mined 16 open-source projects. Their developed deep learning algorithm outperformed random guessing, median, and mean techniques of estimation. Because the model is recurrent, the features will be consistent across all layers, eliminating over-fitting. They developed an end-to-end model in which words are provided as input and story points are calculated as a final output. To validate results, they employed non-parametric testing. When utilized inside a team, story points are useful, but they should not be used to compare projects.

Jasem M. Alostad et al. (2017) [5] introduced a Mamdani FIS type model for story point estimation that utilized team experience, narrative size, and narrative complexity as inputs to the FIS and estimated accuracy as the final result. The final results of the performance metrics MMRE and PRED are 0.28 and 50% respectively, which is directly proportional to the team's experience as the sprints increase.

Habibi Arifin et al. (2017) [40] proposed linear regression models for estimating both effort size (relative) and effort time (absolute). They used data from Atlassian JIRA repositories to develop an estimator that is evidence-based.

Saurabh Bilgaiyan et al. (2017) [3] reviewed cost estimation literature in agile-inspired projects. The authors have answered various RQs like - What are the most common agile estimate approaches? In what scenario may it be used? What percentage of people succeeds and how many people fail? They concluded that NN, EJ, PP, LR, Wideband Delphi, UCP, and MUCP are frequently searched terms. They also addressed the challenges they faced and concluded that scrum is the most popular methodology.

Murat Salmanoglu et al., (2017) [41] compared the cosmic functional point to the narrative point on three industrial projects in an agile framework. As a consequence, they observed that cosmic gives more objective estimations than relative SP (which is based on the team's expertise) and that cosmic has a better forecast than SP with an underlying fact function. They concluded that cosmic FP-based regression models outperformed SP-based regression models.

Ricardo Araujo et al. (2017) [42] developed an MDELP model to handle software effort estimation concerns. However, because it is not used in agile projects, no conclusions can be drawn about its effectiveness. They choose dilation and erosion operators based on pessas concepts. They claimed that the hybrid technique outperformed the existing in terms of PRED.

Dragicevic Srdjana et al. (2017) [43] emphasized how the success of agile projects is dependent on the elicitation of strong issues. Their developed approach could be used in any independent agile project to estimate effort. They used data from 160 projects. They employed the performance metric RMSE to check the discrepancy between actual and estimated effort.

Vlad-Sebastian Ionescu et al. (2017) [44] used the TF-IDF, SVR, and GNB techniques to estimate effort for traditional techniques. When compared to the existing literature, the results appear to be favorable.

Maciej Abdzki et al. (2017) [45] discussed the OSW project, the TOPO system, and the FOODIE system, along with the many challenges inherent in them, in their paper. They used the agile estimates literature to analyze everything and established some interesting conclusions, such as Planning Poker's capacity to produce superior results.

Lavazza et al. (2017) [46] developed a novel technique that provides good estimation accuracy before production deployment. There were discussions of several accuracy prediction factors, and a standardized accuracy measure was used to assess the model's accuracies. The authors also compared various data sets of alternative models.

Janeth Martnez et al. (2017) [47] demonstrated how to utilize a BN model in a Scrum context to generate estimation criteria based on the complexity and importance of user stories. They validated their developed model using correlation tests. This approach will help all newcomers’ transition to scrum-based projects.

Mohd. Owais and R. Ramakishore (2017) [48] provided a technique for effort, cost, and time duration estimation in agile-based projects. No ML algorithm is used in this technique. They conclude their technique is fundamental thus any empirical proof that the current best techniques will be improved is absent.

Seyyed Hamid Samareh Moosavi et al. (2017) [49] developed an SBO algorithm for FIS parameter optimization. The authors claim that their developed algorithm provides optimum parameters to the FIS. They used the ISBSG dataset and the comparison was made using existing techniques such as CART, MLR, and others. 0.235 values for MMRE were identified, which is smaller than the group chosen for comparison.

Shashank Mouli Satapathy et al. (2017) [50] demonstrated the usage of the SP technique to increase the accuracy of effort estimation. The authors compared internal and external results using three machine learning algorithms: DT, SGB, and RF. For training and testing the models, they used data from Zia et al. They next used logarithmic modifications to standardize the data they had collected. They give the ML model issue counts and velocity as inputs, to output the predicted effort. As a result, SGB performs better than the competition. BN can be employed in the future.

Aditi Sharma and Ravi Ranjan (2017) [51] clarified some of the pressing queries, such as NFIS used in effort estimation and their rate of success based on several performance metrics. The concluding statements are incoherent in the context of Agile. It is suggested that NFIS be used with COCOMO, FP in the future.

Binish Tanveer (2017) [52] mentioned in his work that it is necessary to have accompanying guidelines before setting up effort estimates to have a high success probability. It's designed for assessing the impact of change. After consulting with specialists, a framework for guidelines is created.

Non-algorithmic models are applicable for agile-based tasks, according to Sufyan Basri et al., (2016) [53] . Because requirements are unpredictable in agile, they must be factored into the estimated effort and added to the final effort. A table of change type values is also provided, along with the percentage effort necessary for modifications in various phases. MRE's outcome is unsatisfactory.

Saurabh Bilgaiyan et al. (2016) [54] provided an overview of soft computing strategies used in the last decade in agile effort estimation. In their conclusion, BN appears to be more promising than other models, with a 62.8% accuracy rate, when compared to regression-based models, composite models, expert opinion, and PP.

Anjali Sharma et al. (2016) [55] compare RF to various NNs and claim that RF outperforms GRNN, PNN, CCNN, and GMDH. They've used effort as a source of information for the RF approach.

Zia et al. (2016) [56] proposed a model based on expert systems and compared it with COCOMO-II and Function Point Analysis and found quite promising results i.e., 9% increased accuracy as compared to both. However, there is no detailed reasoning mentioned about the intelligent system and is not made in context to agile.

Kayhan Moharreri et al. (2016) [57] developed for Agile-based projects a story point’s auto-estimation model. It inputs a dataset, selects and extracts features, then does a cost estimation analysis. RF, PP, NB, LMT, and their hybrids are some of the methods that have been used. Each one generates a confusion matrix. Conclusively, PP is outperformed by the hybrid.

Binish Tanveer et al., (2016) [58] focused on Agile-based project effort estimation as an industrial case study. By the survey conducted amongst 3 SAP teams, impact, team expertise, and task complexity are three key factors that influence agile project effort and the industry largely rely on PP and SP for estimation.

Aditi Panda et al. (2015) [59] compared multiple NNs used in Agile effort estimation on Zia et al. dataset. GRNN, PNN, GMDH, CCNN, and polynomial NN have been studied. With a PRED of 94%, they discovered that CCNN outperformed all others. They suggest SGB, RF with SP approach for future scope of work.

Muhammad Usman et al. (2015) [60] conducted a state-of-the-practice survey for determining accuracy through an industrial perspective. They mined data from sixty agile practitioners. They concluded that PP is the most frequent method of estimation (63%) followed by SP estimation (62%) and the most frequent size metrics and cost drivers used at what stage of the SDLC.

Hind Zahraoui et. al (2015) [61] described scales and characteristics influencing user stories, and how they alter them for increased accuracy. Their approach is based on determining the story's priority as a multiple of its urgency and business value then finally creating a scale to accommodate it.

Vachik S. Dave et al. (2015) [62] examined a decade of literature in which NN was used to estimate effort. They looked at 21 articles and concluded with several key findings which include a comparison of ANN to several techniques like PP, ANN having superior accuracy compared to FP and SLIM. However, the analysis was not in the context of agile.

Ali Bou Nassif et al. (2015) [63] compared the effort for non-agile-based projects using the RBFNN, GRNN, CCNN, and MLP. Five datasets from ISBSG were used with CCNN outperforming the others.

Manga I, Blamah (2014) [64] proposed the PSO framework in their study, which delivers improved percent accuracy. With adaptive learning, a comparison is made, but facts are missing.

L.R. Nerkar, P.M. Yawalkar (2014) in their paper [4] reviewed the existing cost estimation techniques wherein algorithmic models like COCOMO, Putnam, FP model, and non-algorithmic i.e., analogy based, Parkinson’s law, price to win, and EJ comparisons are drawn. They also proposed a web cost model without any evidence that it is good.

Rashmi Popli, Naresh Chauhan (2014) in their paper [65] presents the algorithmic estimation method based on the effect of various people and project factors. The author explains why it is necessary to include these factors and what problems peeps in if we don’t include these factors in estimation. The author discussed fourteen factors which include types of projects, quality requirements, etc. The algorithm begins with the calculation of unadjusted values, quality factors, and time factors. Based on these calculations, estimated story points and estimated time for the project. The future work states the inculcation of other factors which affect the estimation process to make the process of estimation more accurate and efficient.

Govind Singh Rajput, Ratnesh Litoriya (2014) in their paper [66] presented a novel cost estimation method for agile online projects. They used hybrid CORADMO for RAD projects which can be used in agile projects which are named CORAD_AGILE. It explains three new cost drivers which are substituted with three old cost drivers. These new cost drivers are replaced with personnel, collaboration support, and prepositioning assets. Effort, schedule, and personal productivity are calculated based on this model.

Rashmi Popli, Narsh Chauhan (2014) [67] lay the foundation for an algorithm to estimate Effort and Cost in Agile projects. It is a related work to the previous paper that takes into account the concept of story points. This paper explains a mathematical estimation technique. The author also explained the life cycle of agile and explains the reason for the necessity of effort estimation in any project. The major causes which are responsible for inaccurate estimation in agile development are also discussed which include the methodology adopted, the political forces like managerial pressure, management control issues such as uncertainty and self-knowledge. Then the existing agile estimation techniques which are available are given along with their problems. Then the author proposed their method for the estimation using the story point approach. Total story points are calculated followed by the calculation of velocity which is the value computed by the SP completed in one iteration divided by SP in one US. Then, frictional velocity is calculated followed by estimated development time, effort and cost. Then a case study is done using hypothetical values of the various factors. The future work of the paper states that various correlating factors must be added to improve the estimation accuracy.

Shashank Mouli Satapathy et al. (2014) [68] introduced SVR strategies for improving the effort estimation accuracy using the SP methodology. A comparative analysis of various SVR kernels has been carried out and it has been concluded after experimental results that RBF outperforms linear, polynomial, and sigmoid including kernels. SGB, RF, and other techniques can be used to make further improvements.

Usman, M., et al., (2014) [69] discussed various traditional and ML estimation techniques and compared their prediction accuracies. In their work, MUCP and NN appear as good candidates. The authors also identified various size metrics and cost drivers such as team expertise, size, etc.

Zhamri Che Ani, Shuib Basri (2013) in their paper [70] has investigated how to estimate the effort for the software development in Agile Environment using UCP. Calculating the initial efforts in Agile Projects is a challenge because of the volatile requirements in these projects. And implementing UCP is difficult in agile projects due to two reasons. First, the product backlog contains short descriptions of user stories that don't fit into the documentation standards of use case points. Secondly, none of the studies has clearly described how to use Agile Product Backlog with this approach. So, the authors have successfully implemented this method despite its limitations. For the implementation of this method, KOINS data was taken for analysis. The steps for UCP are followed which involve determining and computing unadjusted use case points, technical complexity factors, environmental complexity factors, productivity factors, and estimated number of hours. The estimated result was near to the actual result stating that the UCP approach is suitable for estimating the efforts for software development at the early stages. The future work states that other estimation models are needed to be compared with this method like COCOMO on Agile projects. The major challenge in this paper was the relation of agility with the calculation is not clearly explained. The concept of user stories which is the baseline of agile projects seems to be disappeared in this paper. So, the problem definition can be formed by merging the concept of story points with use case points. The major challenge against this statement is the availability of the data set for the analysis.

Abhilasha, Ashish Sharma (2013) in their paper [71] explain the concept of regression testing and test effort estimation for the regression testing. Test effort estimation turns out to be costly if all test cases need to be executed. So, there are various techniques used for the selection of test cases that minimally needs to be executed. An approach for the calculation of the Test effort estimation is proposed.

Ziauddin, Shahid Kamal Tipu, Shahrukh Zia (2012) [6] This paper presents a model for the effort estimation for agile projects. The author first discusses some cost estimation techniques and explains agile software development and its characteristics. Then techniques for effort estimation in agile that can be used are discussed. These estimating techniques include numeric sizing, t-shirt inspired sizes, the Fibonacci sequence, etc. It clearly states that the estimation is done by the team members in the sprint planning meeting for the stories of a product backlog. The story size scale is built which is an assessment of the work's relative size in terms of actual development effort. The complexity of the project which may be because of user stories or technical complexity is measured on the complexity scale. These two values, the effort can be calculated for a particular user story which in turn can be summed up for the total effort. Then the concept of velocity is used for the calculation of effort that the team can accomplish in one sprint. Then this velocity is optimized by taking into account the friction forces and the dynamic forces which reduces the projected velocity. Then completion time is calculated followed by the calculation of development cost based on the data collected from the 14 CMMI level 3 companies. Then the experimental analysis is made from the empirical data which is based on data of 21 software projects. The outcome shows that estimated results are near to the actual results. This paper opens several research problems for a future investigation like the use of the number of scales for the estimation of efforts like ranking scale or use of Fibonacci sequence. Moreover, the other factors that are affecting the velocity other than that are mentioned in the paper can be analyzed and more optimized results can be obtained. An improvement can be made on the estimation method by analyzing the major factors that seem missing in this approach.

Evita Coelho, Anirban Basu (2012) in their paper [72] discussed the most acceptable approach in agile methodology – Story Points. Story points are the unit of measurement of user stories that expresses their overall size. The effort and duration that is required for the delivery of features to the customer are estimated by the team member. The traditional methods of effort estimation are not appropriate for effort estimation. For the story points approach, the estimation of the schedule and effort starts by understanding the customer’s conditions of success and failure for the product backlog followed by the estimation of user stories and selection of iteration length, then the estimation of velocity, prioritization of user stories and then estimation of the delivery date.

Ratnesh Litoriya, Narendra Sharma, Abhay Kothari (2012) in their paper [73] investigated the behavior of various cost drivers responsible for the prediction of the cost of any project and then substitution with closest values which will result in the decrease of cost of any project. The investigation is done on the 60 NASA past project data which already contains the actual efforts. And this data is then put into the WEKA tool and K-mean clustering is applied to the data set which results in the formation of clusters. Then the value of these clusters is used to analyze the values of the cost drivers or in other words, the value of cost drivers gets optimized with the formation of clusters. So, the effect of the reduced values of cost drivers has a direct impact on the cost of the project. This reduction of values of cost drivers which results in the reduction of cost of the project is calculated by the use of the online freely available web-based tool AGILE COCOMO-II which was developed by the University of Southern California. The future work of the paper says that other data mining algorithms such as apriori etc. can be used to determine the better optimization of the cost drivers. And these optimized values can be applied on the web-based AGILE COCOMO-II. This paper must have served a great problem definition by incorporating data mining techniques in Cost estimation and new combinations of these fields have come out.

Ritesh Tanmrakar, Magne Jorgensen (2012) in their paper [74] gives the study of the effect of the use of Fibonacci numbers concerning the linear numbers for effort estimation. Two case studies performed for analyzing the same provide significant insights. In the first case study, a group of students was divided into two groups – one for linear and the other for Fibonacci scale for effort estimation. The result showed a large difference in the values of effort estimations whereas the linear scale shows a higher value of the estimation. The second case study was performed with a set of experienced developers. The difference between the values of Fibonacci and linear scale estimations differs with a smaller value. So, the use of Fibonacci is considered to be better than for the use of linear scale.

Ehab E. Hassanein and Salma A. Hassanien (2020) [75] proposed CESP method would have a global project view, which will aid in reducing resource waste in the earlier stages by rearranging the tasks that will be needed in those stages. As a result, the total project time and expense are reduced. This while retaining the versatility and flexibility that the Agile technique needs.

Aiman Khan Nazir et al. (2017) [76] investigated how agile methodology affects various aspects of software project management According to the literature review, agile methodology aids in software project management, which contributes to software success.

Croix, Benjamin (2018) [77] aimed to examine the impact of Agile manifesto on the implementation of partial and tailored agile approaches in the related literature from 2001 to 2017. They were convinced that the real nature of agility had been lost as a result of the industrialization of agile practices. Agile methods should be used within a specified structure that adheres to specific criteria based on principles, rather than as a simple process or procedure.

P. Suresh Kumar et al. (2020) [78] discussed the use of various ANN for effort estimation. It has been discovered that using ANN to forecast machine effort is more accurate. As compared to conventional approaches such as function point, use-case methods, and so on, this approach is more accurate and superior. In the case of COCOMO projects when compared to statistical models, neural network-based models are more competitive as compared to the conventional regression models.

Muhamad Yusnorizam Ma’arif et al. (2018) [79] aimed to share observations on the complexities of implementing Agile projects in Malaysia. This article will highlight the problems that a company faces if the Agile Scrum approach is implemented by conducting a few sets of interviews with domain experts. The key goal of the Agile process in managing IS development projects is to achieve rapid implementation and execution.

Faisal Hayat et al. (2019) [80] conducted a survey of various software companies and the results indicated that almost every software organization uses agile development (Scrum), which has a positive effect on software project management.

Ali E. Akgun (2020) [81] categorized software development on various bases such as conceptual, community, and organizational. He took about two hundred software development teams and recognized the importance of the team wisdom process in the entire software development process. He showed the research model in the research work and also showed the correlation between different parameters. He then showed the results of the hypothesis.

Asad Ali and Carmine Gravino (2019) [82] showed a systematic review based on their analysis of papers. They also answered 6 research questions with relevant tables and figures. They took the list of different datasets used in the paper and showed which dataset was widely used. They performed a secondary search in which they attempted to refer to the sources of the related studies, and they discovered several additional papers that had been missing in the primary search.

Abdullah Altaleb and Muna Altherwi (2020) [83] provided a detailed analysis of the variables that influence the precision of effort estimation. The estimate of effort from previous research, variables were gathered and validated with 20 professionals in the field of mobile application growth from 18 organizations. The detailed factors and predictors provided in this research work guide the estimation value assignment for user stories. These variables were gathered from previous research on the Agile process and the creation of mobile apps.

Emanuel Dantas et al. (2018) [84] proposed and tested a decision tree-based approach for estimating effort in agile projects They predefined its accuracy and usability, and the tool was tested by gathering data from four projects. They also compared the values of MRE from the teams' forecasts to the values provided by the tool to assess their accuracy. They used the Technology Acceptance Model, which was simple to use, to determine. The preliminary results showed that the method can be used both effectively and efficiently.

Magne Jørgensen et.al (2020) [85] took 362 software professionals in their 1st experiment for estimating the effort of 3 larger tasks of equal scale tasks. All of the tasks in the two experiments were assigned in random order. The likelihood of a bias against estimates of insufficient effort can be reduced.

Anupama Kaushik et al. (2020) [13] proposed a hybrid model that combines DBN and ALO. The research work also included the time taken to predict effort as to deal with ambiguous estimation. The developed algorithm, DBN-ALO performs well as compared to other algorithms applied in the field of estimation on the Zia dataset with the lowest MMRE.

Thanh Tung Khuat et. al (2018) [15] combines two algorithms which were ABC and PSO to make a hybrid model which produced favorable results comparatively. They also mentioned the related work with different parameters. They also showed the whole process starting with the estimation of story points. This latest algorithm outperformed various types of ANNs in previous studies.

Onkar Malgonde and Kaushal Chari (2019) [86] took seven algorithms to predict a story's efforts. They also conducted different computer experiments to show that the ensemble-based benchmarking worked better than other ensemble-based benchmarking. They compared all the related work based on different parameters and also showed their proposed method. They used the ensemble learning model in their paper for the comparative analysis.

Ali Bou Nassif et al. (2019) [87] initially, the regression analysis was carried out. The results revealed that data heteroscedasticity affected model efficiency. The linear output inference method worked better than the other modes. They answered three questions in the paper and also showed the related work on fuzzy models for getting an idea about the current work done. They took four datasets and did the graphical representation for comparison.

Soufiane Ezghari and Azeddine Zahi (2018) [88] proposed strengthening the FASEE by applying quality requirements to address the aforementioned drawbacks. The model includes two capabilities known as CFASEE. Both fuzzy estimates and crisp estimation are derived from the current estimation model. An experimental study was also carried out.

Hosahalli Mahalingappa Premalatha and Chimanahalli Venkateshavittalachar Srikrishna (2019) [89] introduced a DBN-based model for the prediction of effort in any agile technique. There is no effect of the ECS-DBN method on agility because it uses simple inputs. At m level, prediction to evaluate the model's accuracy. Nearly 99% precision was achieved by the ECS-DBN method as compared to the other referred techniques.

Claudio Ratke et al. (2019) [90] introduced an automated model based on narrative texts to estimate the effort of development. For the extraction of verbs and nouns, and linguistic reduction, and the standardization of keywords by synonyms, this paper suggested techniques for the symbolic study of natural language. The validation showed an accuracy of over 81%. To validate the algorithm, they separated the basis of the stories that were already calculated, provided in two halves where one was used to compose the words and the other was used to validate the results.

Gajendra Sharma et al. (2019) [91] conducted and gathered empirical evidence from Nepal software development companies. The minimum size of the business was 30, while the maximum size of the company was 200. After that, the study of the case was performed by conducting a set of structured questionnaire interviews. The findings from the case study were compared with the results of the case study. He also did a literature review and discovered that there are practices for verification, validation, and testing cost/effort estimates based on empirical evidence. He concluded that the estimation of test effort follows the same pattern as the estimate of the software development project.

Martin Shepperd et al. (2007) [92] proposed a new framework for the evaluation of competing prediction systems based on unbiased statistics, standardized accuracy, testing the likelihood of results relative to the random 'predictions' baseline technique, which is guessing, and calculating effect sizes.

Brijendra Singh and Shika Gautam (2016) [93] provided a systematic literature review based on situational factors influencing the software process. They also analyzed how situational variables are involved in the Software processes. During the whole process of software development, various managerial and technical problems arise. To solve these problems, software processes are used to develop a product of quality. There are different kinds of situational variables that influence the software process. In the paper, situational factors were identified based on the literature review influencing the software process. Also, they analyzed five key significant situational factors.

Fabián Ugalde et al. [94] compared the four methods of functional size estimation as the basis for estimation of effort in the context of a start-up company that uses an agile methodology to develop mobile applications. Software size measurements, expressed in USP, IFPUG Function, UCP, UFP, and CFP were taken from one project in the business for a set of requirements. Models of effort estimation were then derived from the using regression and their precision was determined by two accuracy parameters.

Laura Diana Radu (2019) [95] proposed a model using BN for agile software development project prediction. They identified two main categories of factors that influence the effort needed based on literature review and the knowledge of practitioners: the quality of teamwork and user stories.

Hrvoje Karna et al.(2018) [96] investigated data based on five distinct software empirical projects originating in the same setting that had been used for performing a formal experiment. The assessment of the results obtained during the process of data mining uses established criteria.

Anureet Kaur and Kulwant Kaur (2019) [97] presented different factors to estimate effort for the mobile apps. The indices of different popular accuracy parameters are used to measure the model's accuracy and results indicate that the proposed model provides a good forecast as well as prediction.

Ahlam Alhaddad et al. (2005) [98] research has shown that most of the current research has used MMRE, MRE, and PRED for accuracy of effort estimation measurement, where NASA93 and COCOM81 were the most widely used datasets. In addition, most of the studies reviewed attempted to use methods of machine learning, whereas other studies have proposed models for hybrids. Concerning size metrics, most of the studies reviewed used a line of (KLOC/ LOC / SLOC) code.

Mayank Jha and Richa Jha (2020) [99] showed the model to represent the variation in bias and the accuracy of the technology estimates of an enterprise test attempt to conclude CDF, NF approach, and methods of Genetics. The motive of this study is to reduce the cost of software and to explain how to apply these concepts to the general system with divisions. Simple algorithms are provided - Cobb Douglas, ANFIS approach to genetic algorithms and decide which algorithm is most appropriate for finding the best algorithm as precise as possible.

Pinar Efe and Onur Demirors (2019) [100] proposed a method to measure change and subsequent cost of rework and evolution, to accurately monitor software projects. In five different businesses, five case studies were performed to explore the usability of the proposed model.

Washington Almeida (2021) [101] described a process model for calculating metrics that can be used in agile contracts rather than structured FPs, which have already proven to be troublesome and difficult to handle for contractors.

Muhammad Ijaz Khan et al. (2019) [102] investigated the characteristics of user stories that can affect the estimation of effort in agile projects. They also showed the different search strategies and answered one research question in their paper. In the results, they showed the papers selected, a summary of the papers when collected year-wise, and characteristics of different user stories in their paper.

According to ISPA [1], approx. 67% of software projects decline to meet deadlines while staying within budget. The unpredictability of system and software requirements is one of the primary reasons for software project failures. The other is incorrect estimations of job size, cost, and personnel requirements. Two of the most difficult parts of estimating scrum-based projects are change and sprint-wise estimation. The vast majority of IT firms have adopted hybrid process models, which are mostly based on Agile umbrella methodologies. The shift in effort estimation approaches has occurred from the transition of process models from heavy weight models such as an iterative waterfall to lightweight models [103] such as Agile. Traditional estimation processes [104] such as EJ, TD estimating, Delphi-Cost estimation and others are well suited for high weight process models in some way, but they fall short of bridging the estimated and real effort gap that Agile methodologies need. Researchers began exploring alternatives as a result of the changing nature of Agile-based project needs, finally settling on soft computing techniques [3]. Providing rules for unpredictable issues is one of the most well-known applications of NF frameworks [51]. However, because software projects are inherently unpredictable and complicated, the data supplied at the beginning of the activity is insufficient, and the issue of story point estimation is entirely unknown. Fuzzy and NF models can be used to mitigate the vulnerability while enhancing estimation precision in this instance. Due to the unpredictability of the effort estimation issue and the complexity of the project and human characteristics relationship analysis, the optimization process is critical. The optimization [49] can be connected directly to effort estimating techniques like quality weighting in analogy-based estimation, or indirectly to machine learning algorithms. It is suitable for parameter weighting, NN optimization, and ANFIS hyperparameter tuning.

Jorgensen and Shepperd [92] published a comprehensive review in the 1980s that found more than 10 approaches for estimating effort, with regression-based tactics beating empirical alternatives. Despite the vast number of comprehensive research on ML models in the estimation of software projects, contradictory results have been accounted for in terms of the estimated exactness of these models, regardless of the process model method. When a comparable ML model is built with various datasets [39], [104] or circumstances [68], the accuracy of the estimation changes. The authors in [104] declared that the ML model is superior to the regression model, however, they reasoned in [105] that the regression model surpasses the ML model. According to the correlation amongst various ML models such as ANN and case-based reasoning, researchers in [39] inferred the former beats the latter while those in [106] detailed the contrary outcome. Experts may be deterred from adopting ML models due to the difference in current empirical judgments. Furthermore, ML systems have a far more complex theory than traditional estimate procedures. To promote the use of ML techniques in SDEE, it is important to actively condense empirical proof on ML models in continuous research and testing. More than ML, industry experts rely on EJ and Delphi cost estimating methodologies. CBR, BN, SVR, ANN, DT, GA, AR, GP, and other ML approaches have been utilized for SDEE [69][107][108][109], although most have yet to be employed for estimation in Agile projects. The aforementioned ML systems can be used as standalone or in conjunction with other ML or non-ML approaches. For instance, GA has been integrated with CBR, ANN, and SVR for highlight weighting and selection. For execution, fuzzy logic [5] was used with CBR, ANN, and DT. Other datasets were utilized for estimation, such as ISBSG, JIRA Repositories, PROMISE data repository, and so on. Holdout, n times overlay Cross-Validation and Leave One Out Cross-Validation [104], [110] are the most popular approval methods. MMRE, PRED (25), and MdMRE [3] are prominent accuracy metrics. For estimating projects that incorporated both traditional and lightweight methods in some scenarios, BN [43], [106] was shown to have the poorest MMRE across all ML Techniques, compared to CBR, DT, SVR, ANN, AR, and GP. COCOMO estimates, EJ, and FPA [41] were also researched. CBR and ANN, according to studies, are more accurate than ML-based regression models. We have determined from our literature survey that ML models outperform non-ML approaches. Because the performance of estimation models varies from one dataset to the next, rendering them prone to inaccuracies, analyst's advice [111], [112] that identifying the best model in a given context rather than the best single model is more productive. The base models separate homogeneous [42], [62], [112] (e.g., MLP, ANFIS, CART, LR, CBR, SVR, RBF, RF, SGB, and so forth) and heterogeneous effort estimation systems. Single ML techniques, according to studies, are the most trustworthy for producing ensembles. It was discovered that DT-based homogeneous ensembles are the most accurate, followed by CBR-based ensembles, and SVR-based ensembles. NF, SVR, DT, Regression are the most prominent techniques used as solos, hybrid, ensembles for estimation. It has also been inferred from the literature that the most common blend rules namely mean, median, and weighted mean is deduced for integrating the base effort models. MLP, SVM, CART, FIS with C-means clustering, and subtractive grouping are among the most widely utilized non-linear concepts. All of the approaches mentioned are derived from generic estimation procedures.

3.2 Research Questions

The research questions have been drafted within the ML context as per the following:

- Research review include criterion
- Research review exclude criterion
- Data repositories details
- Study selection criteria

The various research questions formulated are defined below:

RQ1: What ML algorithms are applied for estimation in Scrum projects?

RQ2: What level of estimation accuracy is achieved by ML techniques in Scrum project estimation?

RQ3: In ML techniques, do meta-heuristic algorithms increase the estimation accuracy significantly?

RQ4: Which Scrum project data is available online?

RQ5: Do ensemble methods outperform other ML techniques?

3.2.1 Include and Exclude Criterion

This research includes papers that link different soft computing approaches for ASD estimation. Papers were gathered from a variety of internet sites, journals, conferences, and other items that have previously been circulated. Papers and data that aren't related to the topic of analysis are excluded from the study.

3.2.2 Data and Literature Sources description

Papers from TOSEM (ACM), IEEE Transactions, Science Direct, Google Scholar, Springer, and other sources were utilized in the study. Papers from these online databases have been searched using search strings: software ∧ (effort ∨ cost) ∧ (estimate) ∧ (learning ∨ ML) ∨ machine ∨ CBR ∨ DT ∨ regression analysis ∨ NN ∨ BN ∨ SVM ∨ SVR ∨ Deep ∨ Learning ∨ fuzzy ∨ NF ∨ ANFIS ∨ Metaheuristic ∨ scrum ∨ Agile ∧ Software ∧ Development ∨ GA ∨ analogy ∨ EJ ∨ PP.

3.2.3 Study Selection Process

The following processes were used to determine which studies should be included and excluded based on the criteria given below:

- Electing abstract and title: The review approach resulted in the submission of a few research articles, some of which were picked based on their titles and modified works.
- Electing complete article: A considerate volume of papers and articles have been thoroughly evaluated and examined.

3.3 Research Question responses

What ML algorithms are applied for estimation in Scrum projects? (RQ1)

ASD and its umbrella methodologies have extensive applicability for many ML models. It can be inferred from Figure 3.1 [113][114][5][59][50][43][115][116][8][64] that many authors have transitioned to increasingly powerful ML techniques with the increase in years.

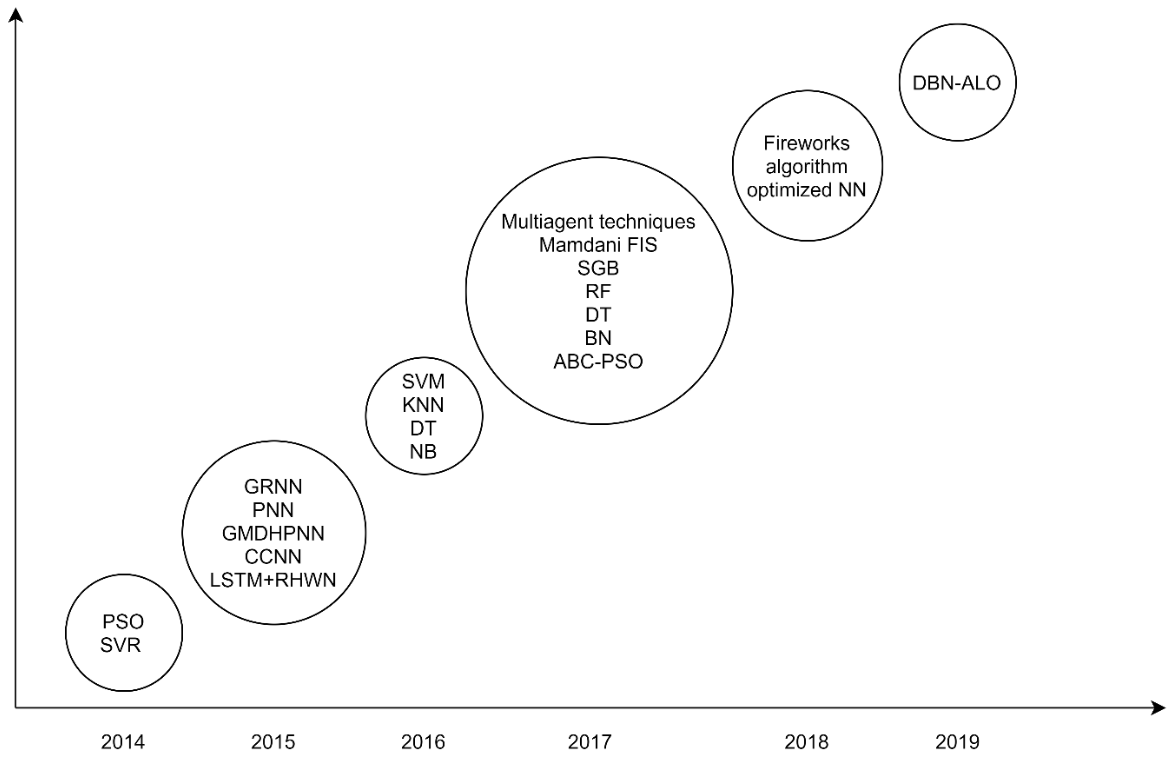


Figure 3.1 ML in Scrum estimation techniques

What level of estimation accuracy is achieved by ML techniques in Scrum project estimation? (RQ2)

ML techniques are superior as compared to traditional ones. Table 3.1 shows the estimation accuracy of various ML techniques in Scrum project estimation. We have used the MMRE accuracy parameter for comparative analysis. Several techniques outperform others on the same dataset and performance metric.

Table 3.1 Accuracy parameter score of ML estimation techniques

Estimation techniques	Accuracy parameters (MMRE)	Dataset used	Outperformed
Fireworks algorithm optimized NN [113]	0.0293	Zia	TLBO, TLBABC, DABC, LM
DBN-ALO [13]	0.0225	Zia	Zia, Fireworks algorithm optimized NN
Multiagent Techniques [114]	0.1	Twelve Web projects	Delphi, PP
Mamdani FIS [5]	Sprint1 – 0.28 Sprint2 – 0.15 Sprint3 – 0.09	Three sprints of Agile projects	Comparison with actual estimates
GRNN [117]	0.3581	Zia	Zia, PNN
PNN [117]	1.5776	Zia	Zia
GMDHPNN [118]	0.1563	Zia	GRNN, PNN
CCNN [118]	0.1486	Zia	GRNN, PNN, GMDHPNN
SGB [68]	0.1632	Zia	RF, DT
RF [119]	0.2516	Zia	DT
DT [50]	0.3820	Zia	Zia
BN [43]	Above 90% accuracy	Real agile projects	Comparison with actual estimates.

ABC-PSO [15]	0.0569	Zia	ABC, PSO, GRNN, PNN, GMDPNN, CCNN
SVM, NB, KNN, DT [116]	SVM – 0.50, NB – 0.85, KNN – 0.70 DT – 0.98	699 and 5607 issues from 8 open projects	Comparison with actual estimates
NB [44]	2.044	10 developer groups from IBM.	Not Available
LSTM + RHWN [8]	58% accuracy	16 projects from 9 open-source repositories	Traditional techniques of estimation.
PSO [57]	0.1988	Zia	Zia
SVR Linear kernel, SVR Polynomial kernel, SVR RBF kernel, SVR Sigmoid kernel [118]	0.1492 0.4350 0.0747 0.1929	Zia	SVR Linear, Polynomial, and Sigmoid kernel

From Table 3.1 DBN-ALO with 0.0225% MMRE currently stands as the best technique when applied to Zia dataset. However, the other techniques using other datasets have different accuracies thus cannot be used to compare on the same level. The compiled MMRE results can be seen in Figure 3.2.

In ML techniques, do meta-heuristic algorithms increase the estimation accuracy significantly? (RQ3)

The usage of meta-heuristic algorithms for estimation of effort and cost in scrum-based projects is fairly low, as per the literature review. It can also be seen from Table 3.1 that only two such techniques: the Fireworks algorithm [113] and ABC-PSO [115] have been used in this domain. The fireworks algorithm has the most superior accuracy among all ML techniques used in Scrum project estimation while ABC-PSO is second in the performance criterion. This evidence supports our research question that using meta-heuristic algorithms significantly increases estimation accuracy.

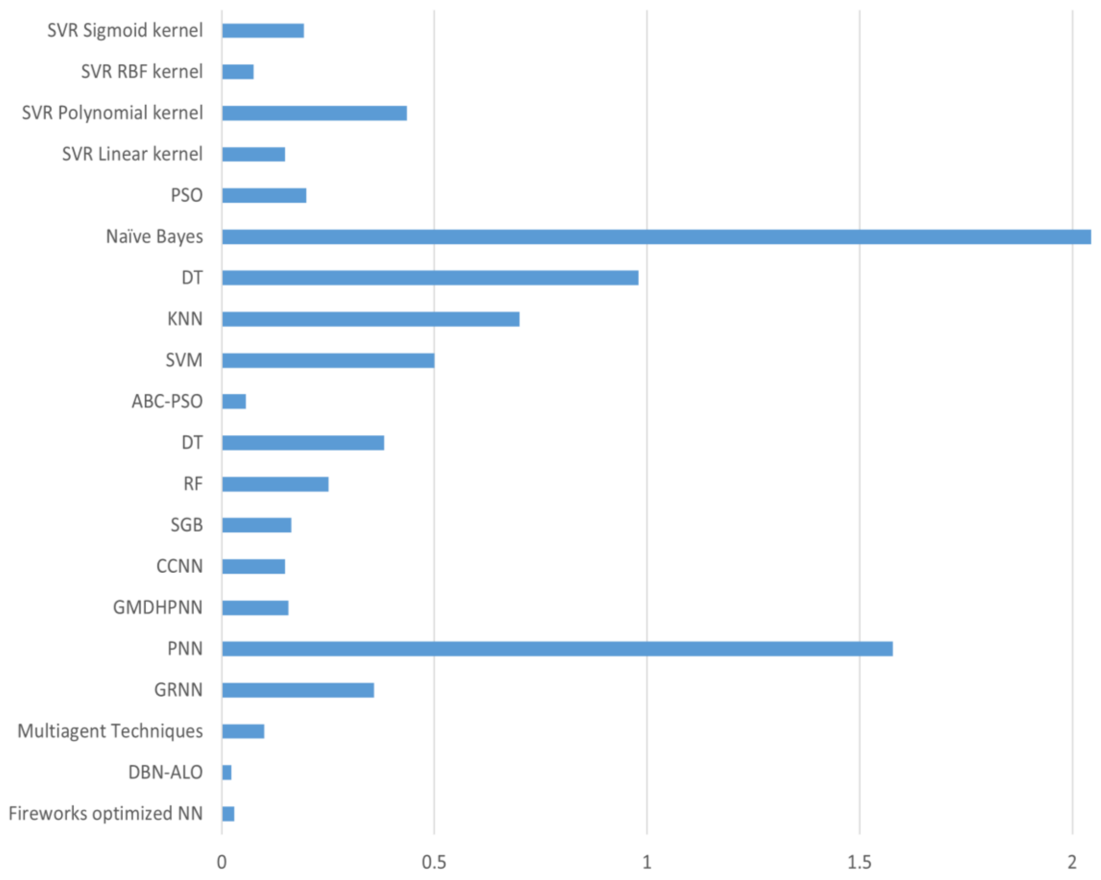


Figure 3.2 MMRE of several ML techniques trained on Scrum datasets

Which Scrum project data is available online? (RQ4)

Various online repositories can be used to find the datasets for Scrum projects in Table 3.2.

Table 3.2 Agile datasets

Dataset Name	Dataset Links/References
Story Point Dataset [120]	https://seanalytics.github.io/DeepSoft.html https://seanalytics.github.io/ https://github.com/SEAnalytics/datasets
Zia et. al	[6]
Twelve Web projects	[121]
Three sprints of Agile projects	[5]
ISBSG	[49]
699 and 5607 issues from 8 open-source projects	[116]

Do ensemble methods outperform other ML techniques? (RQ5)

According to the literature study, ensemble estimation strategies produce better outcomes than single estimate methods. DBN-ALO is having remarkable accuracy as compared to PSO.

3.4 Summary

We have surveyed a large volume of articles, book chapters, and papers, to present the most updated survey on the advancements in the field of effort and cost estimation for Scrum projects. We have provided the inclusion and exclusion criterion, study selection process, description of the data and literature sources, and then formulated a set of vital research questions. In the results and discussions section, we answer these research questions based on our literature review. We have provided the Scrum projects datasets used by several authors for training and testing their ML techniques. We then provide a list of the ML algorithms that have been gaining prominence within recent years. Further, we have also tabulated the set of accuracies for these ML algorithms. Many algorithms use the same dataset. Most authors have chosen to use MMRE and PRED as the performance metrics for defining the accuracy of their techniques. We discuss the relevance of meta-heuristic algorithms in the Agile project domain. Our literature review indicates that such adoption is low, with only two techniques namely Fireworks algorithm [113] and ABC-PSO applied for Scrum project estimation. However, their performance is far superior to ML techniques that are manually tuned. Thus, we infer that the usage of meta-heuristic optimization algorithms significantly improves the performance of ML techniques. We have also established the fact that ensemble methods are superior to other ML techniques. Finally, we provide a list of all the factors that influence effort estimation.

CHAPTER 4

SCRUM EFFORT ESTIMATION USING ANFIS-EEBAT ALGORITHM

4.1 Introduction

ANFIS provides increased learning, adapting, and non- linear abilities, as it makes use of combined advantages of NN and FIS and thereby can be trained without an explicit empirical knowledge pool. Despite carrying strong estimation capabilities, ANFIS architecture needs parameter adjusting and tuning. The objective function of the ANFIS-EEBAT approach is to optimize parameters of ANFIS using an energy-efficient BAT algorithm. To begin with, the system needs its food to start estimating the effort of new projects. Our approach depends on the training of certain project parameters which will be primarily inserted in the knowledge base. However, the data needs to be understandable, so before training, it is being passed from the data preparation module. This chapter discusses our proposed algorithm ANFIS-EEBAT in context to effort estimation.

4.2 Methodology

The input data to ANFIS-EEBAT has been taken from the six software houses incorporating Agile practices. The algorithm of the proposed methodology is presented as four broad categories given below.

- **Data Preparation**
    - Loading the Agile project dataset.
    - Perform a feature selection using ANFIS based exhaustive search.
- **Data Set Partitioning and Model Selection**
    - Partitioning of transformed data into training and testing sets in the ratio 80:20.
    - Training ANFIS-EEBAT model using training data.

- **Testing Part**
    - Performing prediction using a trained model.
    - Comparing prediction results with the original dataset.
- **Performance Evaluation**
    - Calculate the loss function i.e., MSE.
    - Perform model comparison using various performance metrics.
    - Compare the output of the above-defined metrics

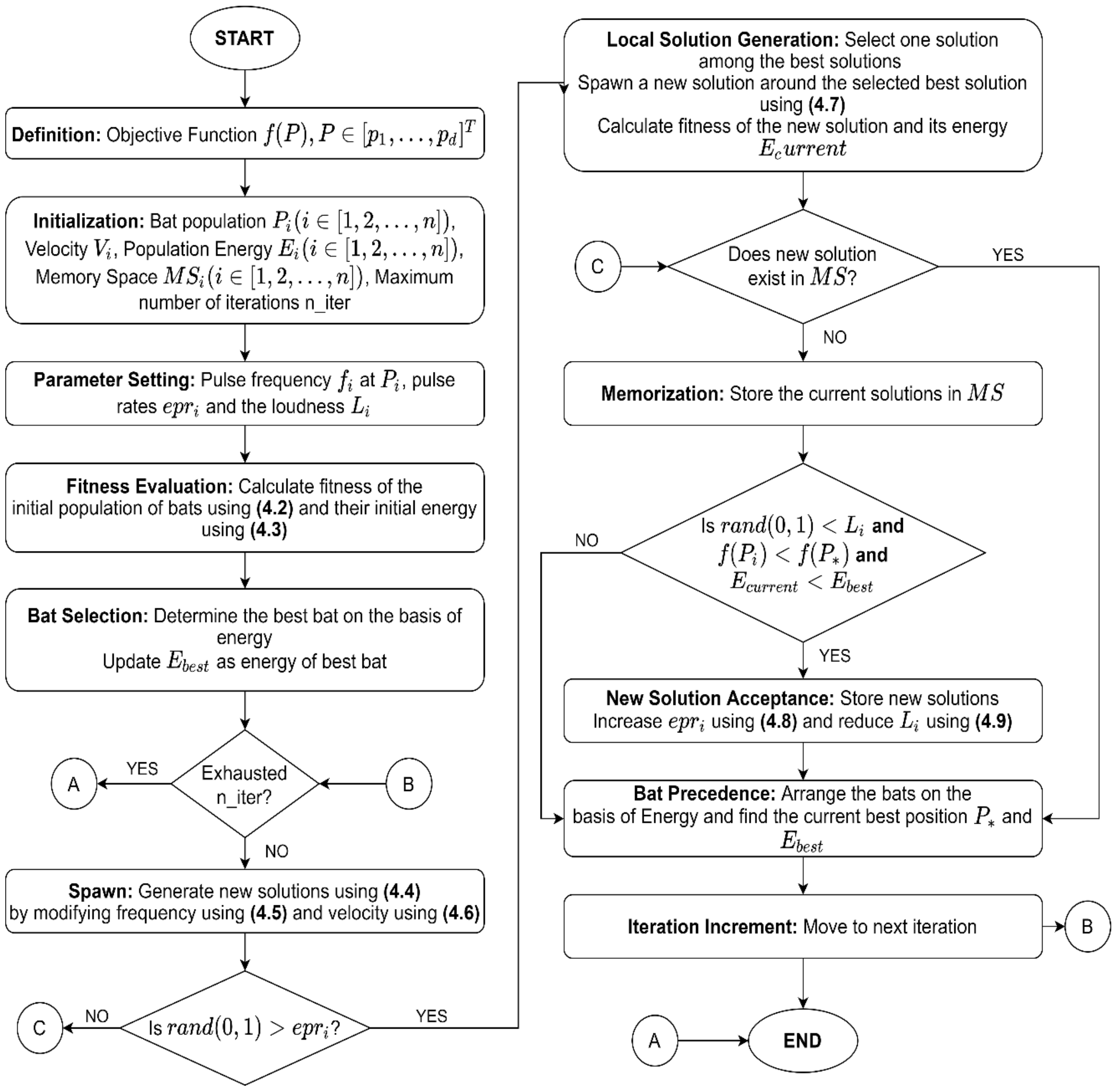
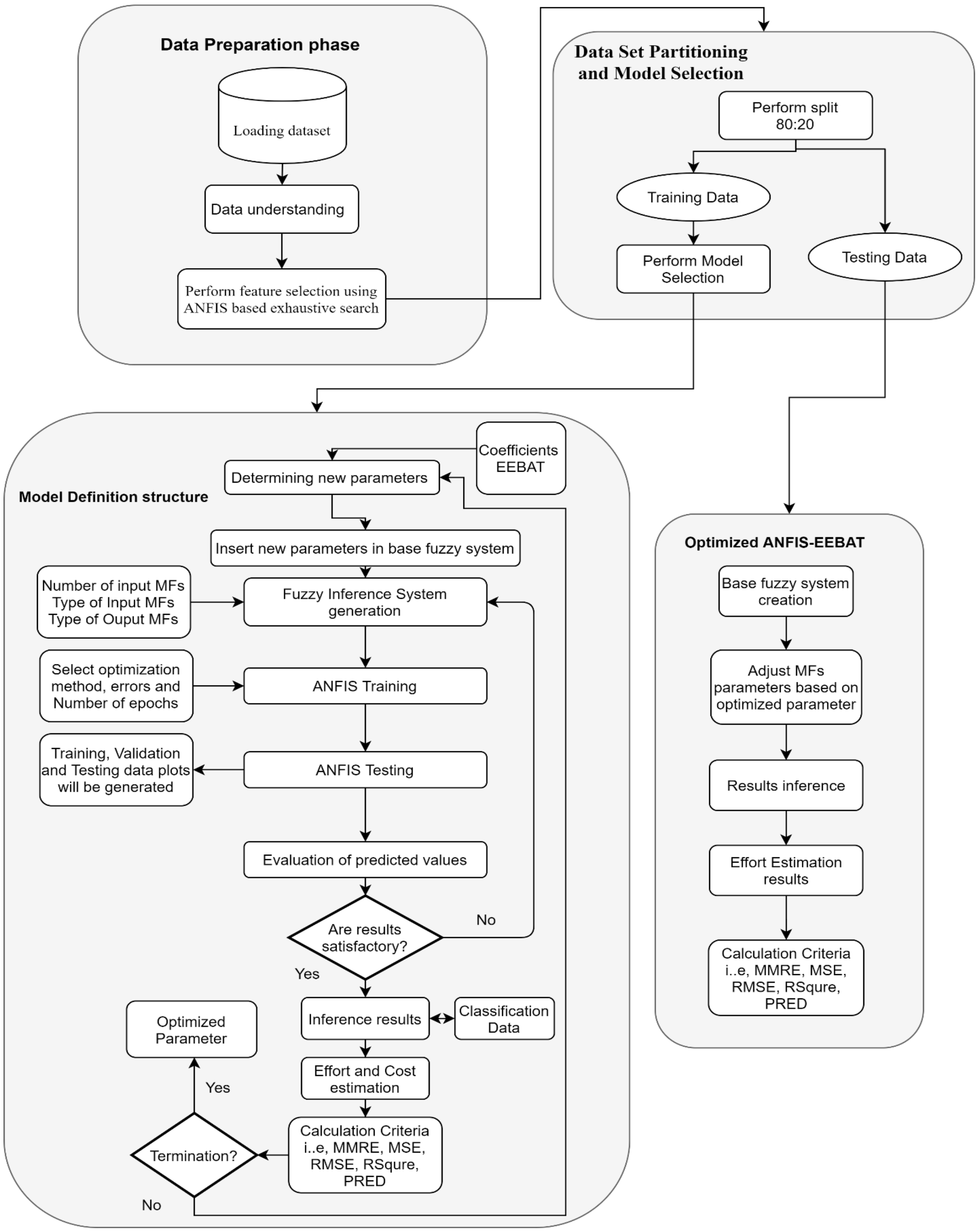

START
Definition: Objective Function $f(P), P \in [p_1, \ldots, p_d]^T$
Initialization: Bat population $P_i (i \in [1, 2, \ldots, n])$, Velocity V_i, Population Energy $E_i (i \in [1, 2, \ldots, n])$, Memory Space $MS_i (i \in [1, 2, \ldots, n])$, Maximum number of iterations n_iter
Parameter Setting: Pulse frequency f_i at P_i, pulse rates epr_i and the loudness L_i
Fitness Evaluation: Calculate fitness of the initial population of bats using (4.2) and their initial energy using (4.3)
Bat Selection: Determine the best bat on the basis of energy Update E_{best} as energy of best bat
Exhausted n_iter?
YES
A
B
NO
Spawn: Generate new solutions using (4.4) by modifying frequency using (4.5) and velocity using (4.6)
Is $rand(0,1) > epr_i$?
NO
C
YES
Local Solution Generation: Select one solution among the best solutions Spawn a new solution around the selected best solution using (4.7) Calculate fitness of the new solution and its energy $E_current$
C
Does new solution exist in MS?
YES
NO
Memorization: Store the current solutions in MS
Is $rand(0,1) < L_i$ and $f(P_i) < f(P_*)$ and $E_{current} < E_{best}$
NO
YES
New Solution Acceptance: Store new solutions Increase epr_i using (4.8) and reduce L_i using (4.9)
Bat Precedence: Arrange the bats on the basis of Energy and find the current best position P_* and E_{best}
Iteration Increment: Move to next iteration
B
A
END

Figure 4.1 EEBAT Flowchart



Data Preparation phase
Loading dataset
Data understanding
Perform feature selection using ANFIS based exhaustive search
Data Set Partitioning and Model Selection
Perform split 80:20
Training Data
Testing Data
Perform Model Selection
Model Definition structure
Coefficients EEBAT
Determining new parameters
Insert new parameters in base fuzzy system
Number of input MFs Type of Input MFs Type of Ouput MFs
Fuzzy Inference System generation
Select optimization method, errors and Number of epochs
ANFIS Training
Training, Validation and Testing data plots will be generated
ANFIS Testing
Evaluation of predicted values
Are results satisfactory?
No
Yes
Inference results
Classification Data
Effort and Cost estimation
Calculation Criteria i..e, MMRE, MSE, RMSE, RSqure, PRED
Termination?
Yes
No
Optimized Parameter
Optimized ANFIS-EEBAT
Base fuzzy system creation
Adjust MFs parameters based on optimized parameter
Results inference
Effort Estimation results
Calculation Criteria i..e, MMRE, MSE, RMSE, RSqure, PRED

Figure 4.2 Flowchart depicting the process of effort estimation using ANFIS-EEBAT

In our proposed algorithm, we update the standard bat algorithm by introducing a new parameter called Energy which will update the position and velocity of the bat based on its distance from the prey. We propose two new factors for the energy parameter - eagerness and magnitude of work, that dynamically get updated for controlling exploration and exploitation trade-off. It becomes exhaustive for a bat or pair of bats to search for its target or prey due to continuous echolocation (lack of cognitive ability), exploration (failure to converge), and exploitation (trapping in local optima). To address these concerns, EEBAT is proposed. The distinctive features of the proposed algorithm are - the energy parameter and memory capability. The flowchart of EEBAT is shown in Figure 4.1.

- **The Energy Parameter, E**

The energy parameter denoted by E is as follows,

$$E = fitness_i * mean(P_i^t) \tag{4.1}$$

where $fitness_i$, is the fitness of the current bat. The population diversity due to energy lets the bat intelligently assess its capability thus improving time complexity and convergence. The mean of the best positions is taken to find a convergence junction, as every bat in the population finds a different position for one value of the parameter. These positions are the best solutions as evident by the fitness value calculated so the collective energy of these deduced positions determines their optimality.

- **The memory capability of the bat**

The population in the standard bat has no history of the previous solutions encountered by the previous bats hence, novel solutions are left and premature convergence occurs. To solve this gap of the standard bat, the second improvement proposed is the introduction of memory capability. We store the position of bats in a special space called Memory Space (MS) in every iteration. This capability improves exploration as previously encountered solutions are prevented from being explored and exploited, hence improving the rate of convergence. This prevents trapping of the population in local optima and decreases time complexity for the algorithm.

Pseudo-code of EEBAT

The pseudo-code of the EEBAT is given below.

```
Define the objective function $(P)$ , $P \in [P_1, \dots, P_d]^T$
Initialize the bat population $p_i$ $(i \in [1,2,\dots,n])$, Velocity $V_i$, Population Energy $E_i$ $(i \in [1,2,\dots,n])$, Memory Space $MS_i$ $(i \in [1,2,\dots,n])$ and Maximum Number of Iterations n_iter.
Define parameters pulse frequency $f_i$ at $P_i$, pulse rates $epr_i$ and loudness $L_i$
Calculate fitness of the initial bat population using (4.2) and their initial energy using (4.3)
```

$$P_{i,j} = P_{min} + \vartheta(P_{max} - P_{min}), \text{ where } \begin{cases} i = 1,2,\dots n \\ j = 1,2,\dots d \\ P_{max} = Highest\ bounds\ of\ dimension \\ P_{min} = Lowest\ bounds\ of\ dimension \\ \vartheta = Random\ value\ between\ [0,1] \end{cases} \quad (4.2)$$

$$E = fitness_i * mean(P_i^t), \text{ where } \begin{cases} i = 1,2,\dots n \\ fitness_i = Fitness\ of\ the\ current\ bat \end{cases} \quad (4.3)$$

```
Determine the best bat based on Energy and set $E_{best}$ as the energy of this bat
while t is lesser than n_iter
Spawn new solutions using (4.4) by adjusting frequency using (4.5) and updating velocities using (4.6)
```

$$P_i^t = P_i^{t-1} + V_i^t, \text{ where } \begin{cases} V_i^t = Velocity\ of\ bat\ i\ at\ time\ step\ t \\ P_i^{t-1} = Position\ of\ bat\ i\ at\ time\ step\ t-1 \end{cases} \quad (4.4)$$

$$f_i = f_{min} + (f_{max} - f_{min})\theta, \text{ where } \begin{cases} f_i = frequency\ of\ ith\ bat \\ f_{min} = Minimum\ frequency \\ f_{max} = Maximum\ frequency \\ \theta = Random\ value\ between\ [0,1] \end{cases} \quad (4.5)$$

$$V_i^t = V_i^{t-1} + (P_i^t - P_*)f_i, \text{ where } \begin{cases} P_* = Global\ best\ solution\ until \\ \quad current\ iteration \end{cases} \quad (4.6)$$

```
        if (rand (0,1) > $epr_i$) then
            Elect a solution among the best solutions
            Spawn a local solution near the elected best solution using (4.7)
            $P_{next} = P_{previous} + \omega \bar{L}^t$
            where $\begin{cases} P_{previous} = \text{Elected high quality solution via any selection method} \\ \omega = \text{Random value} \in [-1, 1] \\ \bar{L}^t = \text{Average loudness of all bats at time step t} \end{cases}$      (4.7)
            Calculate fitness of local solution and its Energy, $E_{current}$
        end if
        if (local solution does not exist in $MS$) then
            Memorize the current solutions in MS
            Fly randomly and spawn a new solution
            if (rand (0, 1) < $L_i$ and $f(P_i) < f(P_*)$ and ($E_{current} < E_{best}$)
                Accept the new solutions and store them in $MS$
                Increase $epr_i$ using (4.8) and reduce $L_i$ using (4.9)
                $L_i^{t+1} = \alpha L_i^t$, where $\{\alpha = Constant$      (4.8)
                $epr_i^{t+1} = epr_i^0[1 - e^{-\gamma t}]$, where $\{\gamma = Constant$      (4.9)
            end if
        end if
        Rank the bats on the basis of Energy and find $P_*$ and $E_{best}$
    end while
```

4.3 Deducing optimal parameters from EEBAT

The proposed system after the default initialization process will undergo tuning of base fuzzy system parameters by EEBAT. The inherent training algorithm of ANFIS will be replaced by EEBAT. The parameters of the base FIS will be adjusted based on low values of fitness/error function. We choose MMRE as our fitness function. genfis is used as a base FIS with fuzzy c-means clustering to create rules and input MFs in the forward pass. EEBAT will minimize the error in the backward pass run. The detailed supposition stages of effort estimation are given in Figure 4.2 and forward and backward pass parameter settings of ANFIS-EEBAT are given in Table 4.1.

Table 4.1 Learning paradigm for ANFIS-EEBAT

	Forward Pass	Backward Pass
Antecedent parameters	Fixed	EEBAT
Consequent parameters	EEBAT	Fixed
Signals	Node Outputs	Accuracy maximization

4.4 ANFIS optimization using EEBAT algorithm

Forward Pass:

Step 1: Initialize the parameters and initial population of the EEBAT algorithm.

Step 2: Set the number of rules for the membership functions and the error tolerance.

Step 3: Update the consequent parameters using the EEBAT algorithm.

Step 4: Predict the values of effort and evaluate them.

Step 5: If the values satisfy the PRED, then go to Step 7. Else, go to Step 3.

Backward Pass:

Step 6: Update antecedent parameters using the EEBAT algorithm.

Step 7: Calculate the MMRE of the predicted values of effort.

Step 8: If the error values are within the threshold, then STOP. Else, go to Step 7.

4.5 Experimental results and discussion

The accuracy achieved by the system depicts the efficacy of the proposed system. This chapter presents the results recorded so far.

4.5.1 Dataset profiling

Each row in Table 4.2 represents an Agile Project. Project names are not revealed in the referred dataset. The *Effort* represents the Number of Story Points. V_i is Initial Team Velocity. *D* is Deceleration, which affects the Team Velocity. *V* is the Final adjusted team velocity, calculated by $V = (V_i)^D$. *Sprint Size* is the size of a typical Sprint in a Scrum Project. *Team Size* is the size of the Scrum Team. *Team Salary* is the salary of Scrum Team Members. The *Actual Time* field value represents the Number of Days, for instance, in the first row 156 SP took 63 days to complete. *Est. Time* and *Estimated Cost* fields signify estimated time and cost calculated by Zia. The *Actual Cost* is the cost of the Scrum Project. *Time MRE* is calculated as:

$$\mathrm{MRE_{Time}} = \frac{|\text{Actual Time} - \text{Estimated Time}|}{\text{Actual Time}} \tag{4.10}$$

Cost MRE is calculated as:

$$\mathrm{MRE_{Cost}} = \frac{|\text{Actual Cost} - \text{Estimated Cost}|}{\text{Actual Cost}} \tag{4.11}$$

Table 4.2 Zia dataset sample [6]

Effort	**V_i**	**D**	**V**	**Sprint Size**	**Team Size**	**Team Salary**	**Actual time**	**Est. Time**	**Actual Cost**	**Est. Cost**	**Time MRE**	**Cost MRE**
156	4.2	0.687	2.7	10	22	230000	63	58	1200000	1023207.14	7.93	14.73
202	3.7	0.701	2.5	10	21	260000	92	81	1600000	1680663.89	11.95	5.04
173	4	0.878	3.3	10	22	250000	56	52	1000000	992269.51	7.14	0.77
331	4.5	0.886	3.8	10	22	300000	86	87	2100000	2002767.22	1.16	4.63
124	4.9	0.903	4.2	10	22	300000	32	29	750000	676081.32	9.375	9.84
339	4.1	0.903	3.6	10	22	400000	91	95	3200000	2895132.85	4.39	9.52
97	4.2	0.859	3.4	10	22	250000	35	29	600000	540113.84	17.14	9.98
257	3.8	0.833	3	10	22	250000	93	84	1800000	1614078.94	9.67	10.32

84	3.9	0.646	2.4	10	22	190000	36	35	500000	507264.58	2.77	1.45
211	4.6	0.758	3.2	10	22	250000	62	66	1200000	1267179.55	6.45	5.59
131	4.6	0.758	3.2	10	22	250000	45	41	800000	786732.223	8.88	1.65
112	3.9	0.773	2.9	10	22	200000	37	39	650000	597142.61	5.4	8.13
101	3.9	0.773	2.9	10	22	200000	32	35	600000	538494.68	9.375	10.25
74	3.9	0.773	2.9	10	22	200000	30	26	400000	394545.65	13.33	1.36
62	3.9	0.773	2.9	10	22	200000	21	22	350000	330561.22	4.76	5.55
289	4	0.742	2.8	10	22	250000	112	103	2000000	1971485.44	8.03	1.42
113	4	0.742	2.8	10	22	250000	39	40	800000	770857.32	2.56	3.64
141	4	0.742	2.8	10	22	250000	52	50	1000000	961866.44	3.84	3.81
213	4	0.742	2.8	10	22	250000	80	76	1500000	1453032.29	5	3.13
137	3.7	0.758	2.7	10	22	220000	56	51	800000	854347.55	8.92	6.79
91	3.7	0.758	2.7	10	22	220000	35	34	550000	567484.33	2.85	3.17

4.5.2 Renaming, identification, and selection of features and labels

We have renamed few fields of the dataset and performed ANFIS based exhaustive search to find the best combination of fields that are chosen as inputs aka features and are matched against output aka label. This exhaustive search has been carried out in MATLAB. Fields named “Effort”, “V” and “Actual Time” from Table 4.1 are renamed to “No. of Story Points”, “Velocity” and “Actual Effort” respectively. Table 4.2 shows that our label “Actual Effort” is mostly affected by “No. of Story Points” and “Velocity” with a minimum value of Train error i.e., 0.6504. The other pairs (No. of Story Points-Team Size) and (Velocity-Team Size) have not been selected as the value of the training error is more vis-à-vis chosen pair. The errors of the feature sets are shown in Figure 4.3. These assist managers in making better decisions of feature selection.

The least indispensable feature selection minimizes complexity and produces software effort estimation results in less time. The deduced features and label after renaming is given in Table 4.4.

Table 4.3: Features Analysis Table

Features	Train error
No. of Story Points, Velocity	0.6504
No. of Story Points, Team Size	4.9212
Velocity, Team Size	15.7069

From Figure 4.4, we can validate our results from ANFIS based Exhaustive search.

Table 4.4: Dataset features and labels

Features		Labels
No. of Story Points	Velocity	Actual Effort
156	2.7	63
202	2.5	92
173	3.3	56
331	3.8	86

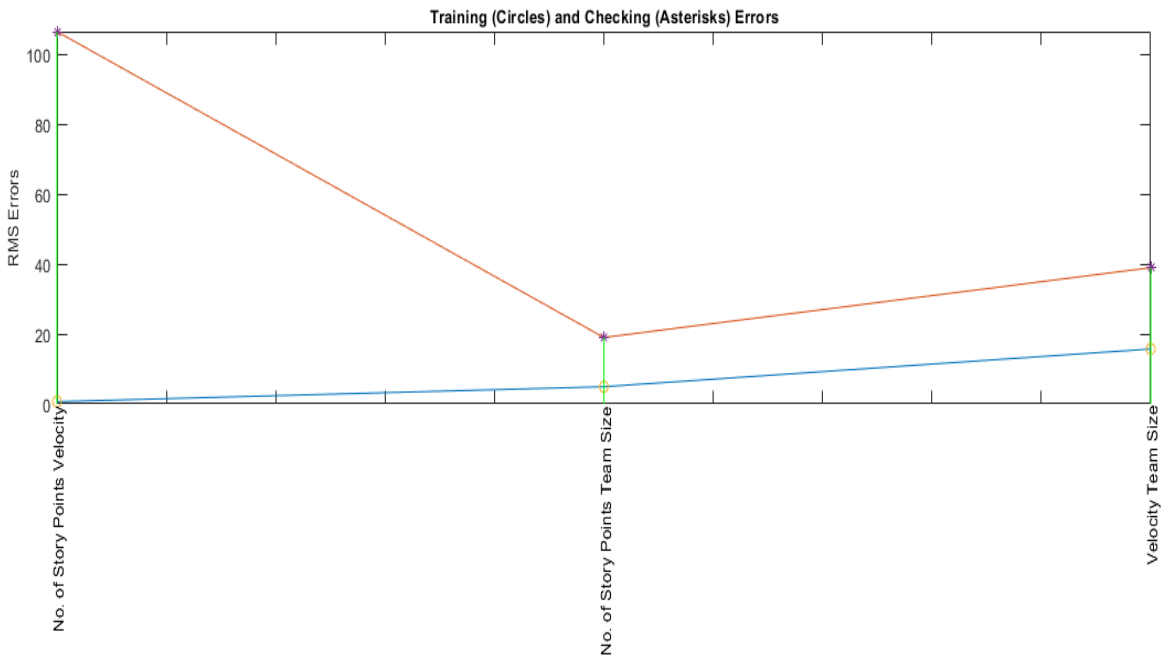


Figure 4.3 RMSE errors for features set in ANFIS exhaustive search

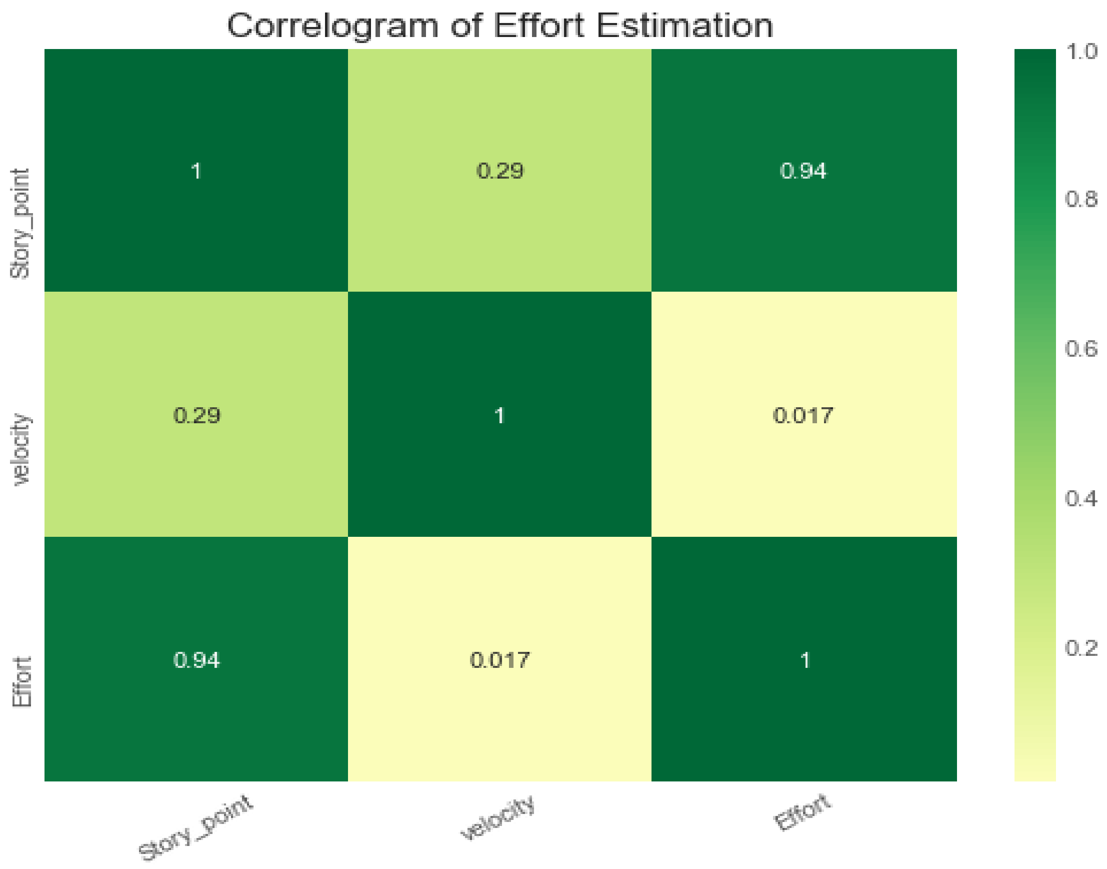


Figure 4.4 Correlogram of Effort estimation

4.5.3 Expansion of dataset using k means SMOTE

We have applied k-means based SMOTE [122], [123], a data augmentation technique, on the Zia dataset, to generate synthetic values of features and labels. The purpose of this step is to address the issues of a modest amount of data for training and testing.

$$x' = x + rand(0,1) * |x - x_k| \tag{4.12}$$

Here, x is the element of minority class set A, x_k is the element of a set A1 which is calculated using k nearest neighbors of x, sampled at some rate N. The new dataset is labeled as ZKmS (Zia K-means SMOTE) and is being used in our ANFIS-EEBAT model.

4.5.4 Descriptive characteristics of the dataset

The descriptive statistics of ZKmS have been given in Table 4.5. It includes count (number of projects in the dataset), mean, standard deviation, the minimum and maximum value of “No. of Story Points”, “Velocity” and “Actual Effort” in the dataset. The statistics “Count” with a value of 162 signifies that ZKmS contains 162 project data. “Mean” represents the average value of the fields. “Std” is the standard deviation which represents the difference of the field values from the Mean value. “Min” represents minimum value and “Max” represents the maximum value.

A detailed profile description of “Number of Story points” is given in Figure 4.5, for “Velocity” in Figure 4.6 and “Actual Effort” in Figure 4.7.

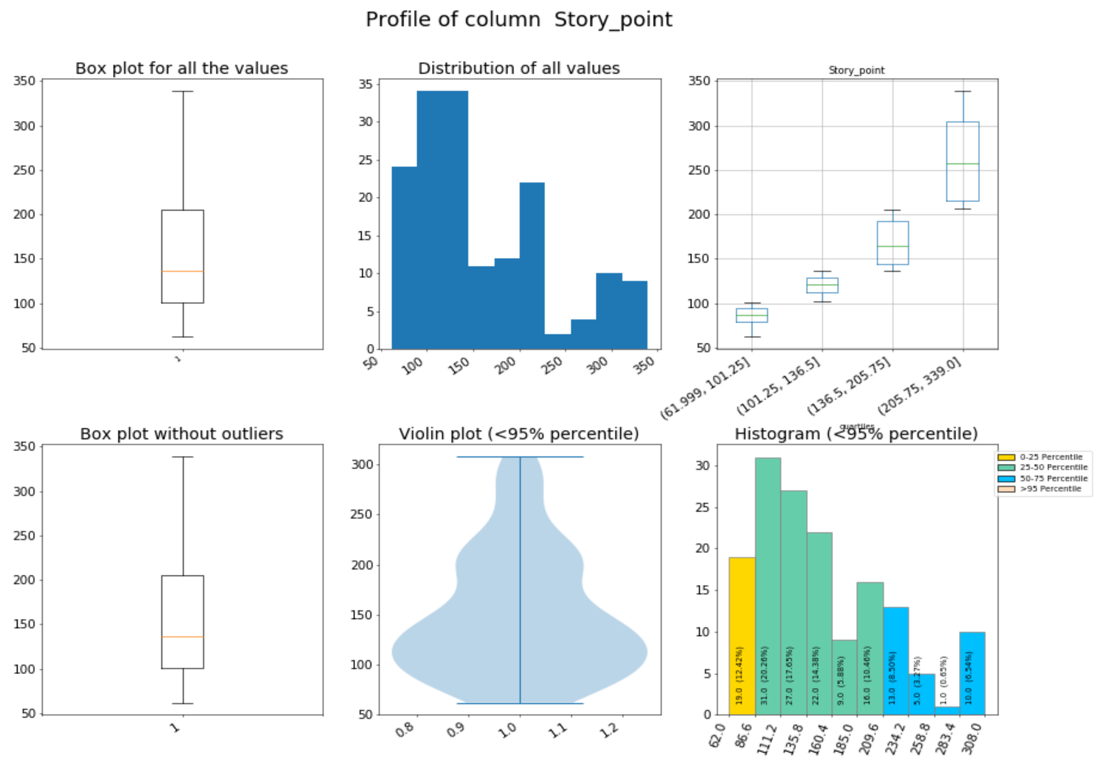


Figure 4.5 Feature profile for No. of Story Points

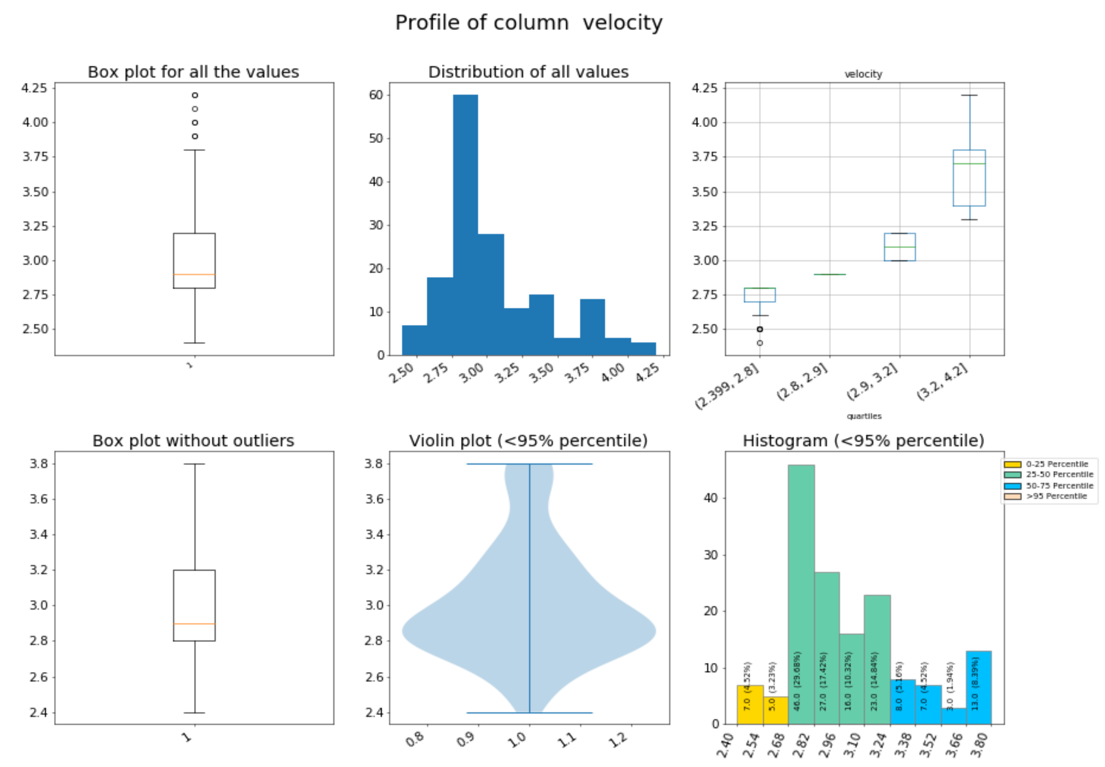


Figure 4.6 Feature profile for Project Velocity

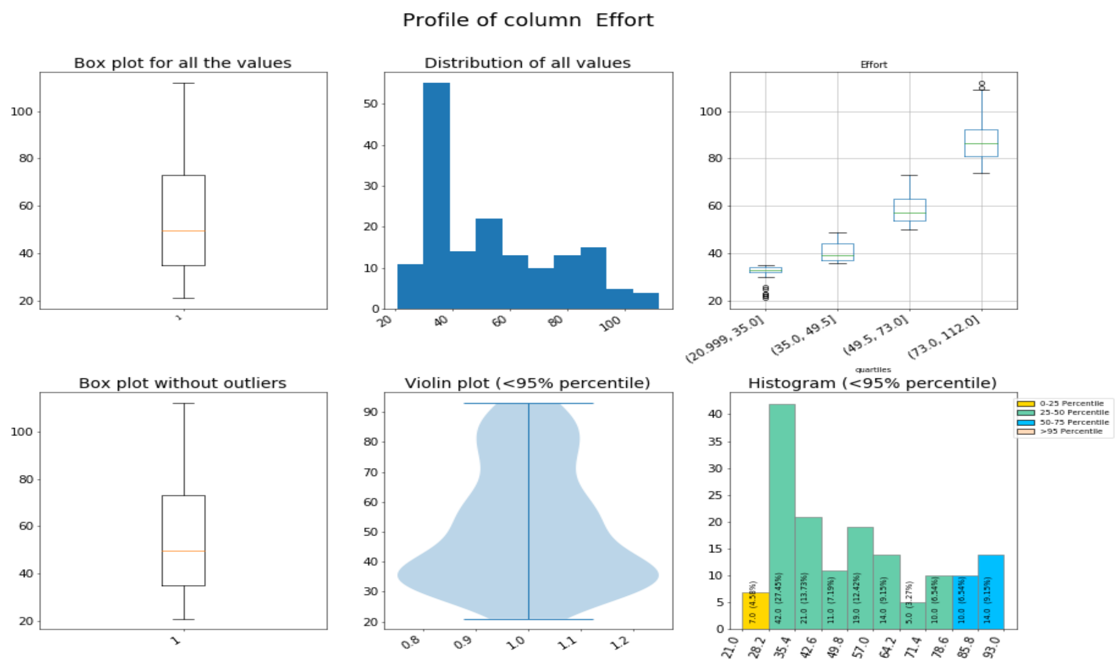


Figure 4.7 Feature profile for Actual Effort

Table 4.5: Descriptive statistics of dataset

Statistics	No. of Story points	Velocity	Actual Effort
Count	162.000000	162.000000	162.000000
Mean	159.648148	3.054938	54.333333
Std	72.914182	0.384328	23.046806
Min	62.000000	2.400000	21.000000
Max	339.000000	4.200000	112.000000

4.5.5 Transformation of Features

The features for effort estimation using ANFIS-EEBAT have been transformed using Box-Cox transformation. The Box-Cox transformation uses lambda λ as the exponent. The best value of λ is found from the following equation:

$$y(\lambda) = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & if\ \lambda \neq 0 \\ \log y, & if\ \lambda = 0 \end{cases} \quad \forall\, \lambda \in [-5,5] \qquad (4.11)$$

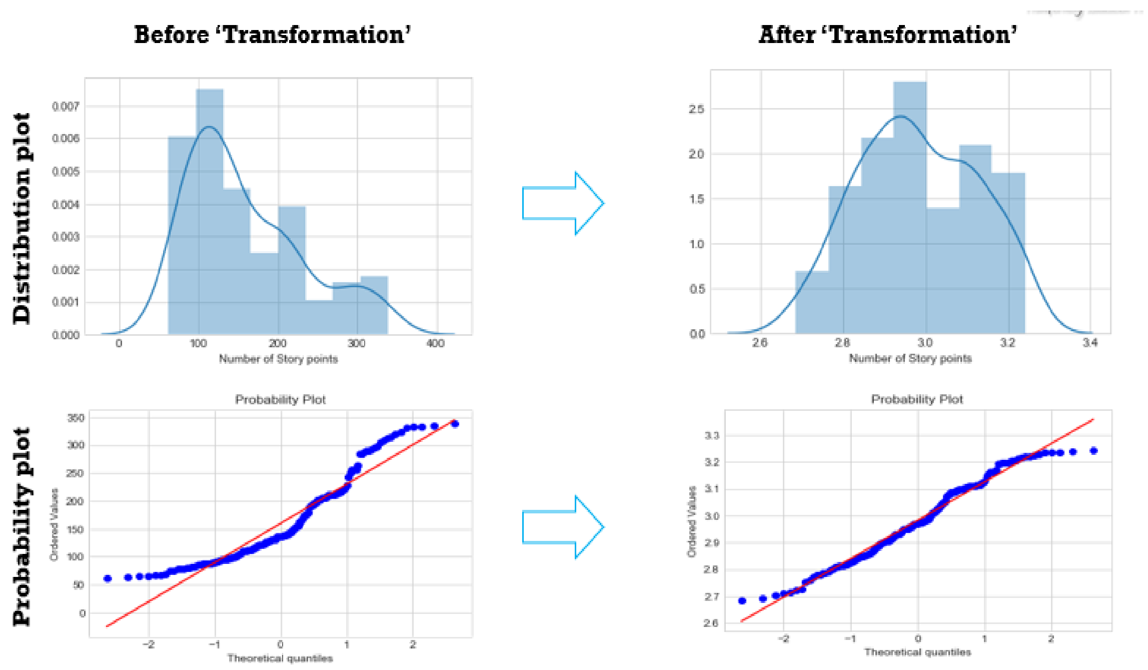


Figure 4.8 Box cox transformation of Number of story points

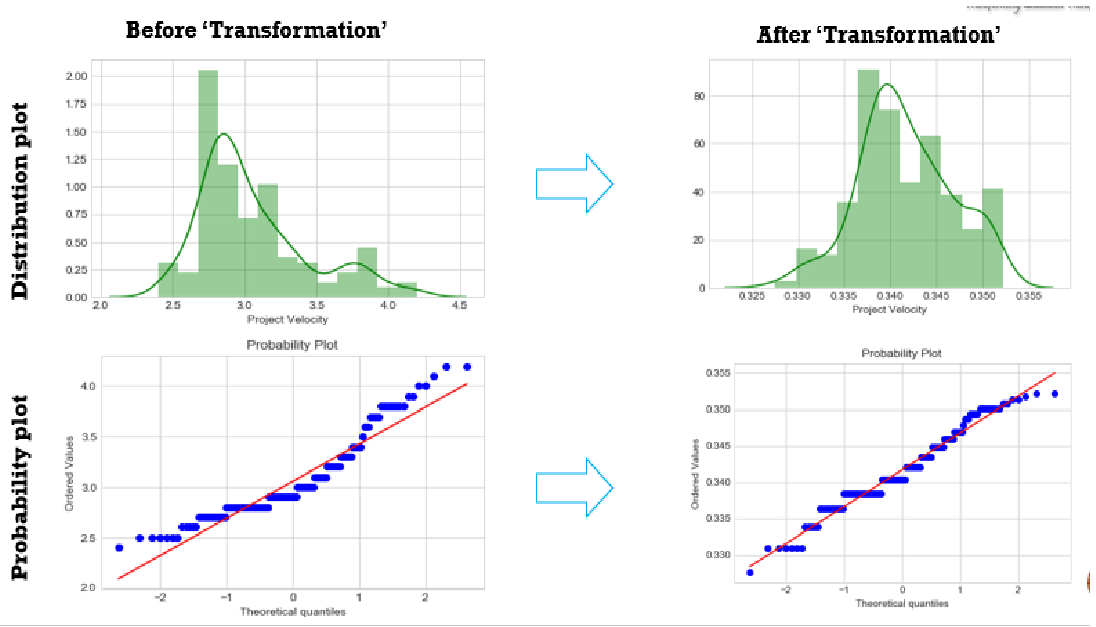


Figure 4.9 Box cox transformation of Project velocity

4.5.6 Model selection

ANFIS-EEBAT has been applied to the features from the dataset as per the step given below.

Data loading and Generate Fuzzy Inference system

After we input features in the proposed ANFIS-EEBAT model, the antecedent layer creates the input MFs. The initial set of parameters for ANFIS and EEBAT are given in Table 4.6 and Table 4.7 respectively.

The number of inputs is "2" which are "No. of Story Points" and "Velocity". The Number of outputs is "1" which is "Actual Effort". The Learning algorithm is "EEBAT". The value "4" in the Number of inputs MFs parameter signifies that there exists 4 gaussian MFs for each input with a unique set of gaussian parameters. "Fuzzy C-Means" Partitioning method has been employed which is used to create a base FIS. The input MF is "gaussmf (gaussian)" which represents our data in normal distribution and the output MF is "linear" which produces a singular value. The base fuzzy system is created using the "genfis3" functionality of MATLAB. The "And" method signifies the product of weights of neuro-fuzzy system with the inputs. The “Or” method utilizes “probor (probabilistic or)” which is the algebraic sum of the previous layers. The implication and aggregation are set to “min” and “max” respectively. “wtaver” i.e., weightage average is used for defuzzification. The training iterations aka epochs are set to 100 as after this value over fitting occurs. The iterations have been validated against several trials. The error tolerance is set to 1e-5.

Table 4.6 FIS parameters for effort estimation

No. of inputs	2
No. of outputs	1
Learning algorithm	EEBAT
No. of input MFs	[4 4]
Partitioning method	Fuzzy C-Means
Input MF	Gaussmf
Output MF	Linear
Base fuzzy system	genfis3
And Method	Prod
Or Method	Probor
Implication	Min
Aggregation	Max
Defuzzification	wtaver
Maximum Iterations	100
Error Tolerance	1e-5

The size of the initial BAT population is kept as “40”. The max no. of iterations is “100”. Pulse rate signifies optimal solution searching precision of the algorithm. The tuning parameters of ANFIS are the optimal solution. Loudness controls the speed of convergence of the algorithm. The value of F_{min} and F_{max} determines the range of frequency that assists in global searching capability. Alpha and gamma are constants. The values for each parameter are obtained during several exhaustive trials.

Table 4.7 EEBAT parameters

Population size	40
Max Iterations	100
Pulse Rate	0.3
Loudness	0.9
F_{min}	0
F_{max}	0.1
Alpha (α)	0.9
Gamma (γ)	0.9

Building ANFIS-EEBAT model structure

After setting up the initial parameters, the proposed model's structure is shown in Figure 4.10.

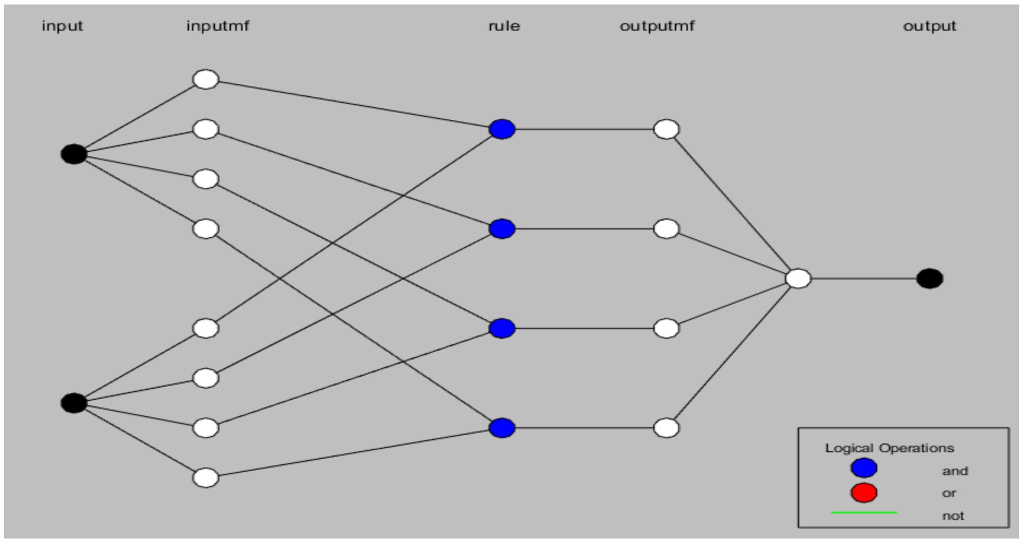


Figure 4.10 ANFIS-EEBAT structure

It contains five layers, as discussed in Figure 2.1. There are two inputs, four pairs of input MFs, four sets of rules, four output MFs and one output. The two inputs are "No. of Story

Points" and "Velocity". "Estimated Effort" is the final output. The operations performed at different layers are synonymous with the description in Figure 2.1. There are three basic logical operations, "and", "or", "not" depicted in the figure with three color codes "blue", "red" and "green" respectively. The rules are created using logical "and" operations in our case. The logical "or" and "not" operations are not used.

ANFIS-EEBAT MFs and Rules view

After the training and testing, membership function parameters are adjusted using EEBAT and can be seen in Figure 4.11(a) and Figure 4.11(b). The rules for the same are shown in Figure 4.12.

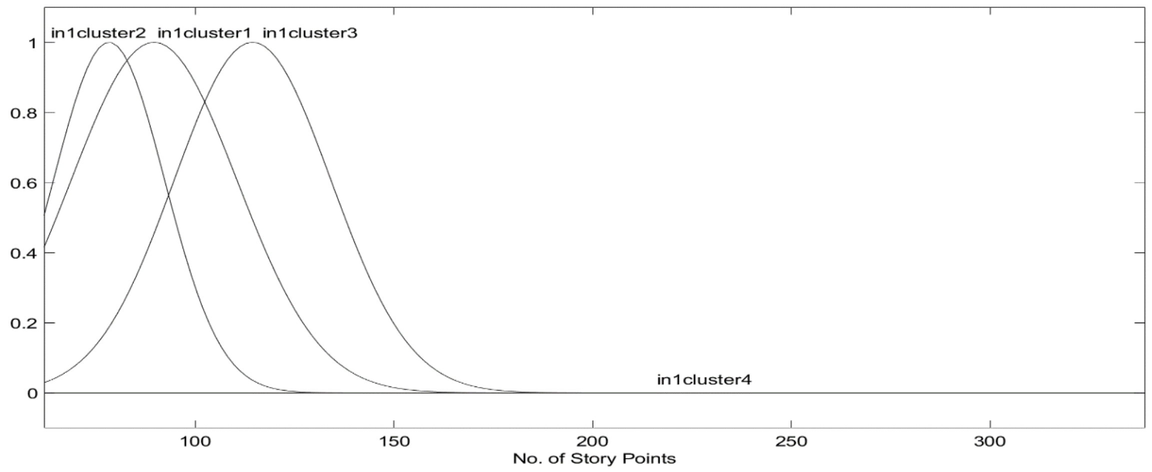


Figure 4.11(a): Membership function for No. of Story Points in ANFIS-EEBAT

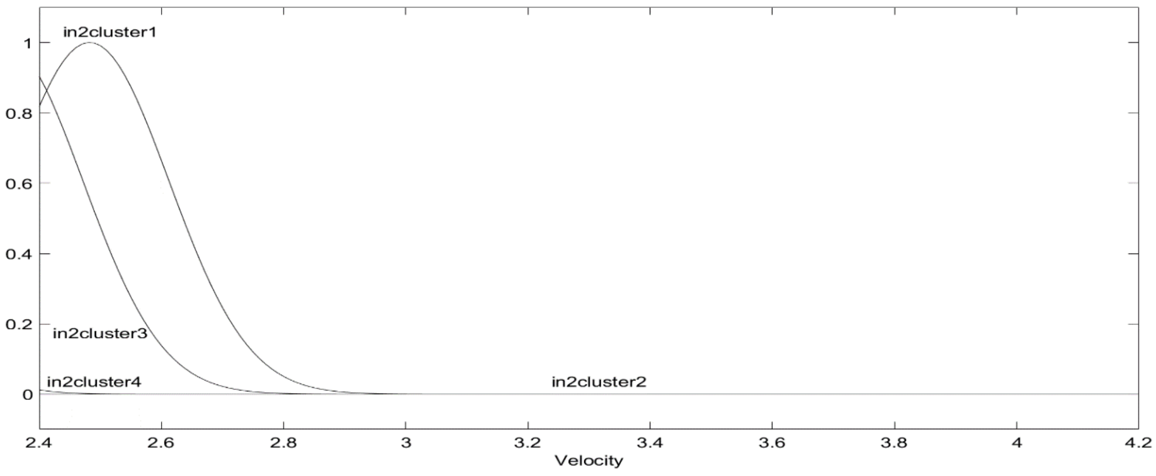


Figure 4.11(b): Membership function for Velocity in ANFIS-EEBAT

The x-axis and y-axis represent "No. of Story Points" and membership values of input1 respectively in Figure 4.11(a). The set of four unique MFs created are represented as curves (in1cluster1, in1cluster2, in1cluster3, in1cluster4). The red color marks the selected curve. in1 signifies Input1. The x-axis and y-axis represent "Velocity" and membership values of input2 respectively in Figure 4.11(b). The set of four unique MFs created are represented as curves (in1cluster1, in1cluster2, in1cluster3, in1cluster4). in2 signifies Input2. The curves are overlapped due to the minute values of "Velocity" which is ranging from 2.4 to 4.2.

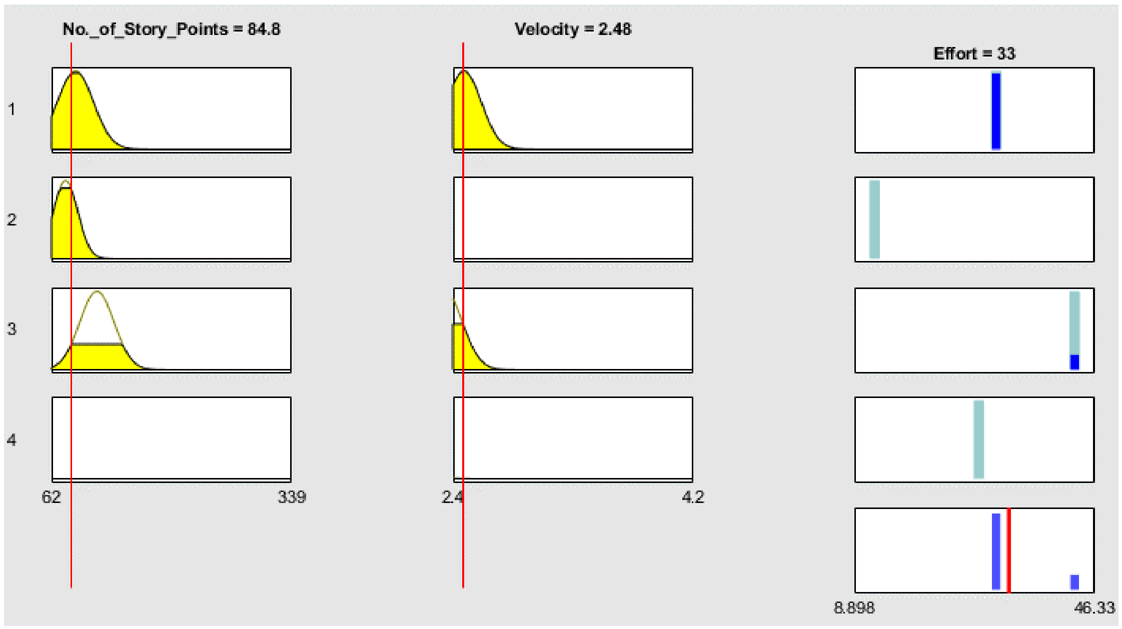


Figure 4.12: ANFIS-EEBAT Rules View.

Each row of the plot represents a rule. There are a set of four rules based on input and output membership functions. The red line is a slider for selecting the input values. For instance, we have selected the value of "No. of Story Points" as 84.8 and "Velocity" as 2.48. The yellow color in the plots depicts how the input variable is used in the rules. The blue color in the output membership function, "Effort", signifies how the output is utilized in the rules. The output of each rule is combined and defuzzified to create an aggregated output in the bottom-right plot. The estimated effort, 33 is the output and is shown by the red color line.

ANFIS-EEBAT Surface Plot

The surface plot shown in Figure 4.13 depicts the mapping of the features with the labels. It can be deduced from the surface plat that for our features, the output is linear which is following Takagi-Sugeno Type 3 FIS.

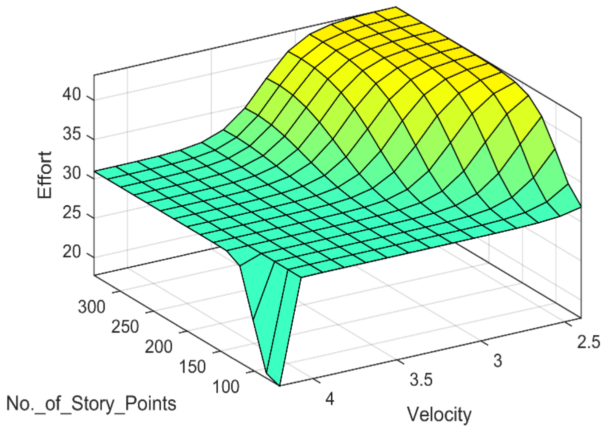


Figure 4.13 ANFIS-EEBAT surface plot

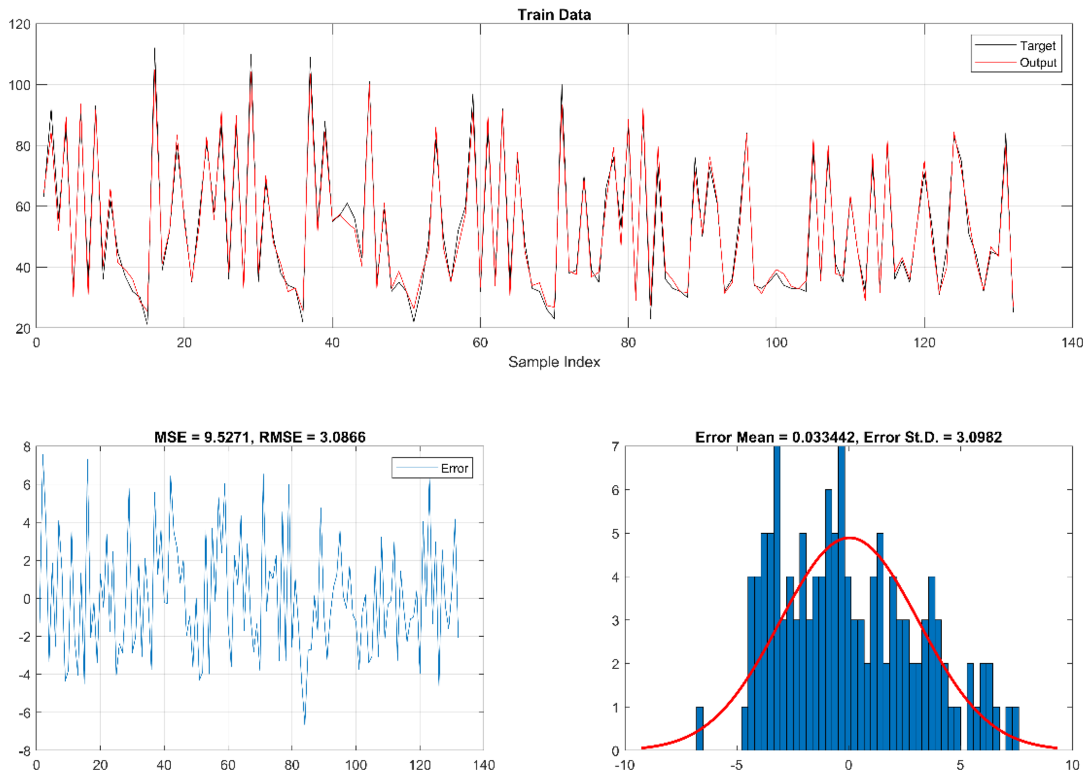


Figure 4.14(a) Training plots for ANFIS-EEBAT

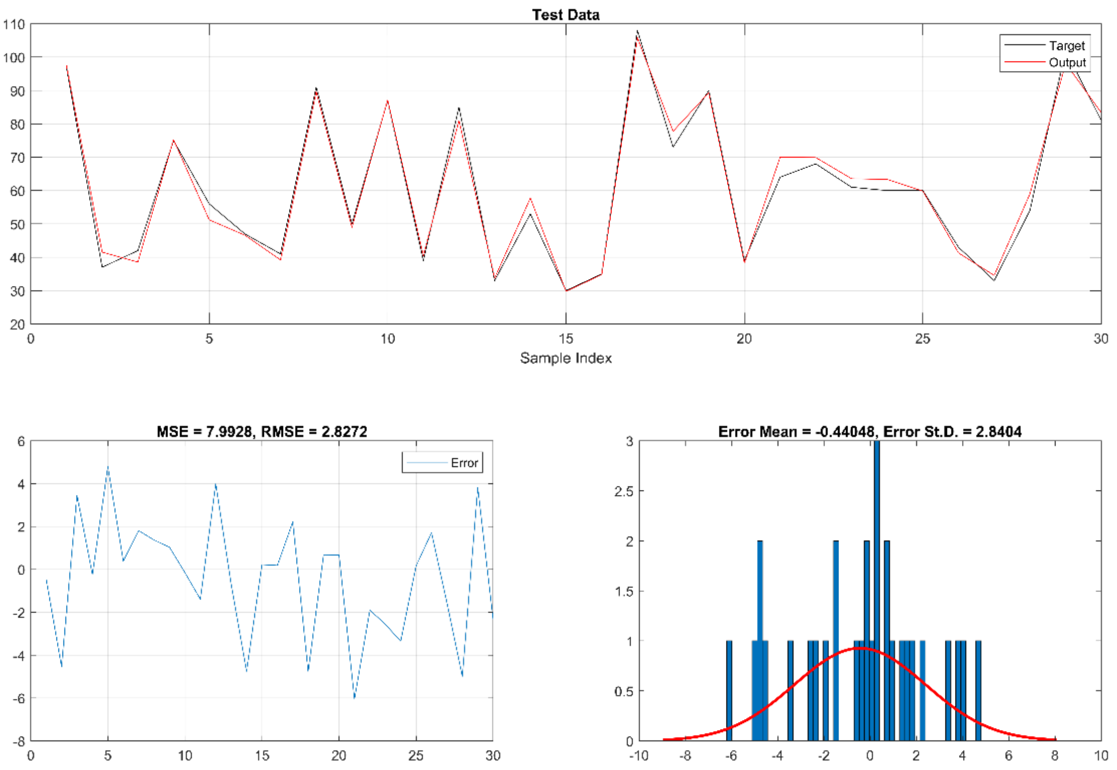


Figure 4.14(b) Testing plots for ANFIS-EEBAT

4.5.7 ANFIS-EEBAT performance evaluation and comparative analysis

ANFIS-EEBAT model's performance has been evaluated using various metrics like R^2, MSE, RMSE, MAE, MAPE, MMRE, and PRED and is given in Table 4.8 for ZKmS and Zia datasets. ANFIS-EEBAT has also been compared with other state-of-the-art models on the aforementioned datasets and summarized in Table 4.9 and Table 4.10. The training and testing plots of ANFIS-EEBAT on ZKmS are shown in Figure 4.14(a) and Figure 4.14(b) respectively.

Our approach is accurate to 98.47% and 99.93% on ZKmS and Zia datasets respectively and will assist the IT industry stakeholders in getting accurate estimates of their respective projects. It also provides 100% estimation accuracy up to 2.4% for PRED.

Table 4.8 ANFIS-EEBAT performance metric evaluation

Data set	R^2	MSE	RMSE	MAE	MAPE	MMRE	PRED (25%)	PRED (2.4%)
ZKmS	0.984723	7.992858	2.827164	0.440483	0.971148	3.910372	100	100
Zia [6]	0.999349	0.556203	0.74579	0.35558	1.019133	1.518311	100	100

Table 4.9 Results on ZKmS with other techniques

Techniques	R^2	MSE	RMSE	MAE	MAPE	MMRE	PRED (15%)
ANFIS	0.982857	9.25933	3.042915	0.57697	0.896473	4.310884	100
ANFIS-GA	0.973329	18.2304	4.269707	2.025023	3.525099	6.568641	96.67
ANFIS-PSO	0.977309	11.83014	3.439497	0.366967	0.857856	4.498164	96.67
ANFIS-BAT	0.955252	24.72921	4.972847	1.386903	2.748092	5.78877	86.67
Random Forest	0.812542	98.71803	9.935695	0.325292	1.394628	13.38908	66.67
SVR	0.294153	377.0704	19.4183	3.678706	0.965226	20.50747	46.67
SGB	0.955736	22.65617	4.759849	0.35478	0.825001	5.676487	93.33
ANFIS-EEBAT	0.984723	7.992858	2.827164	0.440483	0.971148	3.910372	100

The lowest MMRE and highest PRED (15%) signify the efficacy of ANFIS-EEBAT over other techniques. Various techniques are employed on the ZKmS dataset for comparative analysis. Standard ANFIS uses hybrid (back propagation and LSE) learning for training. In ANFIS-GA, ANFIS-PSO, and ANFIS-BAT the default learning algorithm of ANFIS has been replaced by GA, PSO, and BAT respectively. GA, PSO, and BAT are well-known nature-inspired meta-heuristic algorithms. Their innate ability to find optimal solutions provides valuable feedback in exploration and comparison. RF is one of the ensemble learning algorithms which performs the mean prediction of singular trees for estimation. SVR with RBF kernel has been used. SGB has also been employed for estimation. It is a well-known algorithm that inculcates randomness and variation in boosting which increases robustness in learning complex data.

Table 4.10: Results on Zia with other techniques

Techniques	R^2	MAE	MMRE	PRED (25%)	PRED (2.4%)
ANFIS	0.982857	0.57697	4.310884	100	40
ANFIS-GA	0.973329	2.025023	6.568641	100	20
ANFIS-PSO	0.977309	0.366967	4.498164	100	40
ANFIS-BAT	0.955252	1.386903	5.78877	100	40
Zia regression [6]	Not Available*	Not Available	7.19	57.14	0
Fireworks algorithm [113]	0.9946	Not Available	2.9339	Not Available	Not Available
DBN-ALO [13]	Not Available	Not Available	2.225	98.4321	Not Available
ANFIS-EEBAT	0.99935	0.35558	1.518	100	100

* The data of performance metrics are not available in the referred research papers. This comparison has been performed on real Agile projects. It can be inferred that despite good accuracies by Fireworks optimized NN and DBN-ALO, a gap of actual and estimated effort is still present. This gap has been further narrowed down using the ANFIS-EEBAT approach with a PRED of 100 closes to 2.4%. Figure 4.15 shows the box plot of our proposed approach with other models on the homogeneous dataset. It can be inferred that ANFIS-EEBAT has the lowest value of the median.

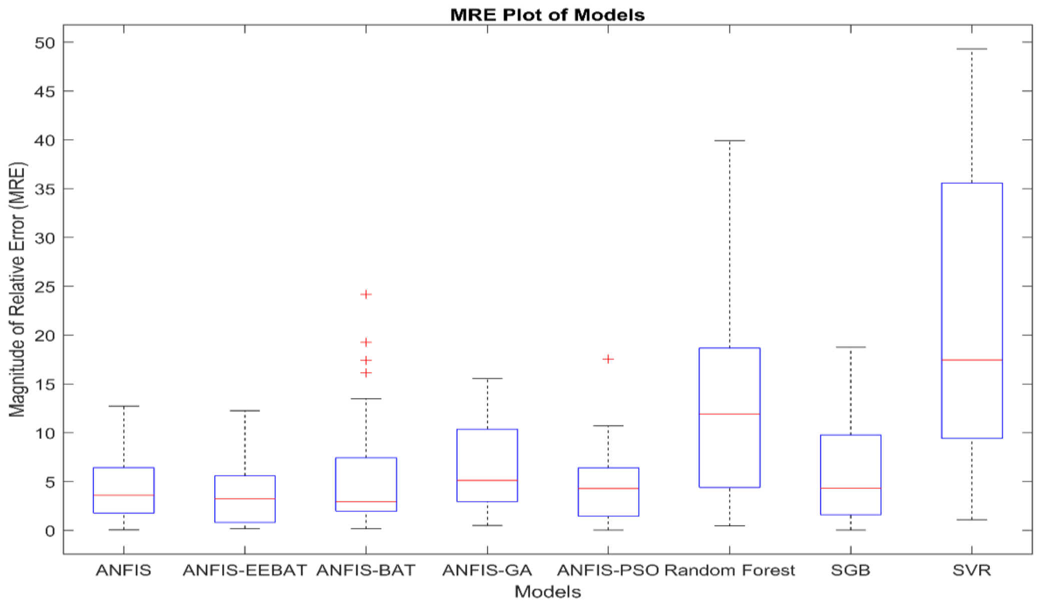


Figure 4.15: Box Plot of ANFIS-EEBAT with other models on ZKmS dataset

4.6 Statistical validations

Performance evaluation of our proposed approach with other models is based on statistical validations. As per the nature of our data, non-parametric tests such as Friedman[124] have been applied to the ZKmS dataset using SPSS. This test performs the average ranking of the models and detailed test statistics have been shown in Table 4.11 and Figure 4.12 for the Friedman test.

We have considered our null hypothesis that all the models are similar. The standard chi-square value for 4 degrees of freedom, *df*, and alpha = 0.05 is 9.488 and our chi-square value is 10.133, which rejects our null hypothesis. The value of Asymp. Sig. (0.038) is less than the significance value of alpha (0.05) which ascertains those models are dissimilar. As per Figure 4.16, the mean rank of ANFIS-EEBAT is 2.47 as compared to ANFIS (2.87), ANFIS-PSO (2.93), ANFIS-GA (3.73), and ANFIS-BAT (3.0).

Table 4.11. Mean Ranking of algorithms

Ranks	
Algorithms	Mean Rank
ANFIS	2.87
ANFIS-PSO	2.93
ANFIS-GA	3.73
ANFIS-BAT	3.00
ANFIS-EEBAT	2.47

Table 4.12. Friedman test results on ZKmS dataset

Test Statistics	
N	30
Chi-Square	10.133
df	4
Asymp. Sig.	0.038

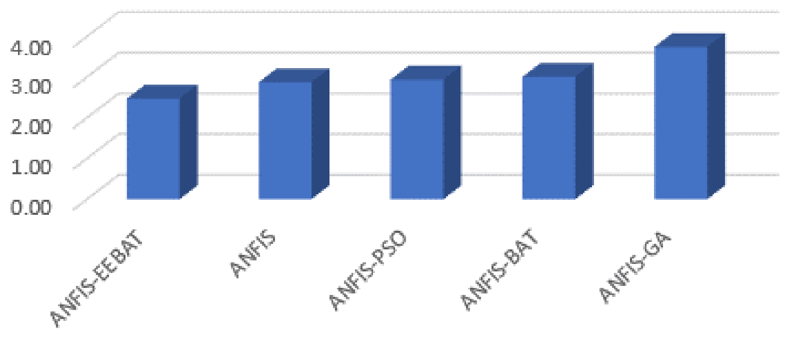


Figure 4.16: Friedman test rank comparison on ZKmS

4.7 Summary

We have developed a nature-inspired algorithm called Energy Efficient BAT algorithm, to tune the parameters of ANFIS and titled it ANFIS-EEBAT, for Scrum Effort estimation. Our developed algorithm outperforms various state-of-the-art algorithms applied in the field of estimation including DBN-ALO, ABC-PSO, ANFIS-PSO, ANFIS-GA and ANFIS-BAT, and several others. The key novelty of our algorithm is Energy Factor and Memory Space. Energy Factor is paramount to let the stronger bats search for global areas while the weaker bats are utilized to search locally in one of the global areas. This approach is robust for Scrum effort estimation as the bats have a defined role based on their energy. Essentially, the bats search both local and global areas for the best value of effort in parallel. Memory Space helps in keeping a track of areas already explored, preventing the bats from repeated searching. This means that for a similar set of story points and velocity, the effort need not be searched again, as the best estimate has already been stored. In technical terms, the goal for the bats is to search for local and global optima. One set of bat population searches one of the global solutions locally to find the best solution among them while the other set scours other unexplored global solutions, for deeper exploration. This allows our algorithm to constantly look for the best solutions in a wider search space. As the bats have Memory Space, this means that convergence to local optima will never occur and there will be no loss in computation due to the search for already existing solutions within the search space. These factors improve the time and space complexity of the algorithm, as evident by the analysis of results on the dataset. For effort estimation in Scrum projects, this is vital to include little variations in story points and velocity.

As we move to experimental results and discussion, there is descriptive information about various characteristics of the dataset and the generated ANFIS model. The selection of our features and labels is made using ANFIS Exhaustive search in MATLAB. This can be justified from the correlogram that indicates a high correlation among these features. The dataset profiling illustrates our features and labels in terms of mean, deviation, outliers, etc. through the plots. Since the data is scarce, the dataset has been expanded with the help of K-means SMOTE. It is worth noting that we implemented the algorithm by tuning the k-value to prevent a repetition of data values.

The model selection for ANFIS is crucial as it has to be manually generated and validated. ANFIS requires manual selection of the type and number of membership functions. We used the Gaussian function as the choice of our membership function with 4 rules each for one input. The Gaussian function is one of the types of bell-curved functions that suits well for real-world data with substantially large values. For each input, 4 rules will be generated to assimilate all the required cases for Scrum project estimation. We randomly tested for 4 rules and found them suitable for our purpose. All details have been illustrated, from the ANFIS-EEBAT structure to the membership functions. Finally, we have comparatively analyzed our developed algorithm against other state-of-the-art algorithms, on various performance metrics suitable for regression type of problems and statistically validated these results.

CHAPTER 5

SCRUM EFFORT ESTIMATION USING ANFIS-CEBAT ALGORITHM

5.1 Introduction

To be competitive in this fast-moving market place organization need to drive innovations through cross key functional and business units. Continuous change is the new norm in software engineering. This industry is primed for a paradigm shift as numerous potentially market-changing technologies link up on every front. The transition from heavy-weight process models (e.g., waterfall) to digital singularity (e.g., Agile, DevOps) is the most noted one. Cost estimation is one of the key factors to determine the success of a typical IT project. It is an important metric to assists project managers to take firm decisions in context to budgeting and resource management. Estimating costs in Scrum projects require a more collaborative, iterative, and incremental approach than in traditional techniques. To estimate the cost of a Scrum project, its US size must be known. The US is a high-level requirement and is measured as story points. The SP is a relative metric and performs a significant role in estimation. As per ICEAA and Standish group chaos manifesto, nearly two-third of the software projects entered crises as a result of inaccurate predictions of effort and its associated costs. The empirical estimation techniques like EJ, Delphi-cost estimation, COCOMO, etc. have their inherent limitations of learning knowledge base and may not be used in Scrum projects. Most of the Agile teams use PP and other empirical estimation techniques to estimate the effort of scrum projects. These traditional techniques suffered from individual bias and lead to inaccurate estimates.

Scrum projects faced critical issues of inaccurate cost estimation over the past few years which lead to software crises. These inaccurate estimates create ripples in subsequent iterations of a typical Scrum project. We have created a hybrid ANFIS model tuned by

Cost-Estimating Bat which will assist project managers to make better decisions in context to cost commitments to clients. As per our in-depth study, there is no single expert system that exists in the literature that estimated the cost of scrum projects. ANFIS-CEBAT approach improves the present state of cost estimation due to its unique learning capability. We discuss the novelty of CEBAT in this chapter.

5.2 Methodology

The steps of the proposed methodology are given below:

Data Preparation

- Loading the Agile project dataset.
- Perform a feature selection using ANFIS based exhaustive search.
- Data Transformation using Quantile transformer.

Data Set Partitioning and Model Selection

- Partitioning of transformed data into training and testing sets in the ratio 80:20.
- Training ANFIS-CEBAT model using training data.

Testing Part

- Performing prediction using a trained model.
- Inverse transformation of the predicted value.
- Comparing prediction results with the original dataset.

Performance Evaluation

- Calculate output from the loss function i.e., MSE.
- Perform model comparison using various performance metrics.
- Compare the output of the above-defined metrics

Algorithm of CEBAT

The standard BAT algorithm is susceptible to getting trapped in local minima or not converging at all. To improve upon the standard BAT algorithm, we have proposed a unique factor called Energy Factor that accounts for a bat’s capability in searching for solutions within the global or local space. The Energy Factor is based on a bat’s eagerness to search for a solution. Hence, the higher the eagerness, the higher the chances of exploration. Less eager bats are allowed to exploit the global spaces that help in finding the best solution in that area. Hence, it solves the problem of trapping in local minima.

We have also introduced another factor called Memory Space. It stores all explored space at every iteration of the algorithm. This improves the rate of convergence and prevents the bat from leaving out a global space for exploration. The time complexity of the algorithm is also increased.bat from leaving out a global space for exploration. The time complexity of the algorithm is also increased.

One key application of the standard BAT algorithm is estimating linear quantities. In Scrum projects, cost estimation is non-linear, as the combination of several direct and indirect components manipulates the cost of a project. To address this issue, we propose another factor that ranges between 0 and 1. The choice of value for Cost Multiplier is human-based, as a means for bridging the differences in components. A value of 0 suggests the bridge is too long thus it is highly unlikely the non-linearity can be mapped onto the algorithm while a value of 1 means that there is no bridge between the components thus the cost can be ascertained easily. The flowchart of CEBAT is shown in Figure 5.1.

Pseudo-code of CEBAT

The pseudo-code of CEBAT is given below:

```
Define the objective function $(p)$ , $p \in [p_1, \ldots, p_d]^T$
Initialize the bat population $p_i$ $(i \in [1,2, \ldots, n])$ using (5.1), Velocity $V_i$,
Population Energy $E_i$ $(i \in [1,2, \ldots, n])$, Memory Space $MS_i$ $(i \in [1,2, \ldots, n])$ and
Maximum Iterations n_iter.
```

$$p_{i,j} = p_{min} + \vartheta(p_{max} - p_{min})$$

$$where \begin{cases} i = 1,2,\dots n \\ j = 1,2,\dots d \\ p_{max} = Highest\ boundary\ of\ dimension \\ p_{min} = Lowest\ boundary\ of\ dimension \\ \vartheta = Random\ value\ between\ [0,\ 1] \end{cases} \quad (5.1)$$

Define parameters pulse frequency f_i at p_i, pulse rates epr_i and loudness L_i

*Calculate fitness of the initial bat population using **(5.2)** and their initial energy using **(5.3)***

$$fitness_i = f(p_i),\ where \quad \begin{cases} i = 1,2,\dots n \\ p_i = Position\ of\ ith\ bat \end{cases} \quad (5.2)$$

$$E = fitness_i * CostMultiplier * mean(p_i)$$

$$where \begin{cases} fitness_i = Fitness\ of\ the\ current\ bat \\ CostMultiplier \in [0,1] \end{cases} \quad (5.3)$$

Determine the best bat based on Energy and set E_{best} as the energy of this bat

***while** t is lesser than n_iter*

*Spawn new solutions using **(5.4)** by adjusting frequency using **(5.5)** and updating velocities using **(5.6)***

$$p_i^t = p_i^{t-1} + V_i^t,\ where \begin{cases} V_i^t = Velocity\ of\ bat\ i\ at\ time\ step\ t \\ p_i^{t-1} = Position\ of\ bat\ i\ at\ time\ step\ t-1 \end{cases} \quad (5.4)$$

$$f_i = f_{min} + (f_{max} - f_{min})\theta,\ where \begin{cases} f_i = frequency\ of\ ith\ bat \\ f_{min} = Minimum\ frequency \\ f_{max} = Maximum\ frequency \\ \theta = Random\ value\ between\ [0,\ 1] \end{cases} \quad (5.5)$$

$$V_i^t = V_i^{t-1} + (p_i^t - p_*)f_i,\ where \begin{cases} p_* = Global\ best\ solution \\ until\ current\ iteration \end{cases} \quad (5.6)$$

```
if (rand (0,1) > $epr_i$) then
Elect a solution among the best solutions
Spawn a local solution near the elected best solution using (5.7)
```

$$p_{next} = p_{previous} + \omega \bar{L}^t$$
$$\text{where} \begin{cases} p_{previous} = \text{Elected high quality solution via any selection method} \\ \omega = \text{Random value between } [-1,\ 1] \\ \bar{L}^t = \text{Average loudness of all bats at time step } t \end{cases} \tag{5.7}$$

```
Calculate fitness of local solution and its Energy, $E_{current}$
end if
if (local solution does not exist in $MS$) then
Memorize the current solutions in $MS$
Fly randomly and spawn a new solution
If (rand (0,1) < $L_i$ and $E_{current} < E_{best}$)
Accept the new solutions and store them in $MS$
Increase $epr_i$ using (5.8) and reduce $L_i$ using (5.9)
```

$$L_i^{t+1} = \alpha L_i^t\ ,\ \text{where } \{\alpha = Constant \tag{5.8}$$

$$epr_i^{t+1} = epr_i^0[1 - e^{-\gamma t}],\ \text{where } \{\gamma = Constant \tag{5.9}$$

```
end if end if
Rank the bats based on Energy and find $p_*$ and $E_{best}$
end while
```

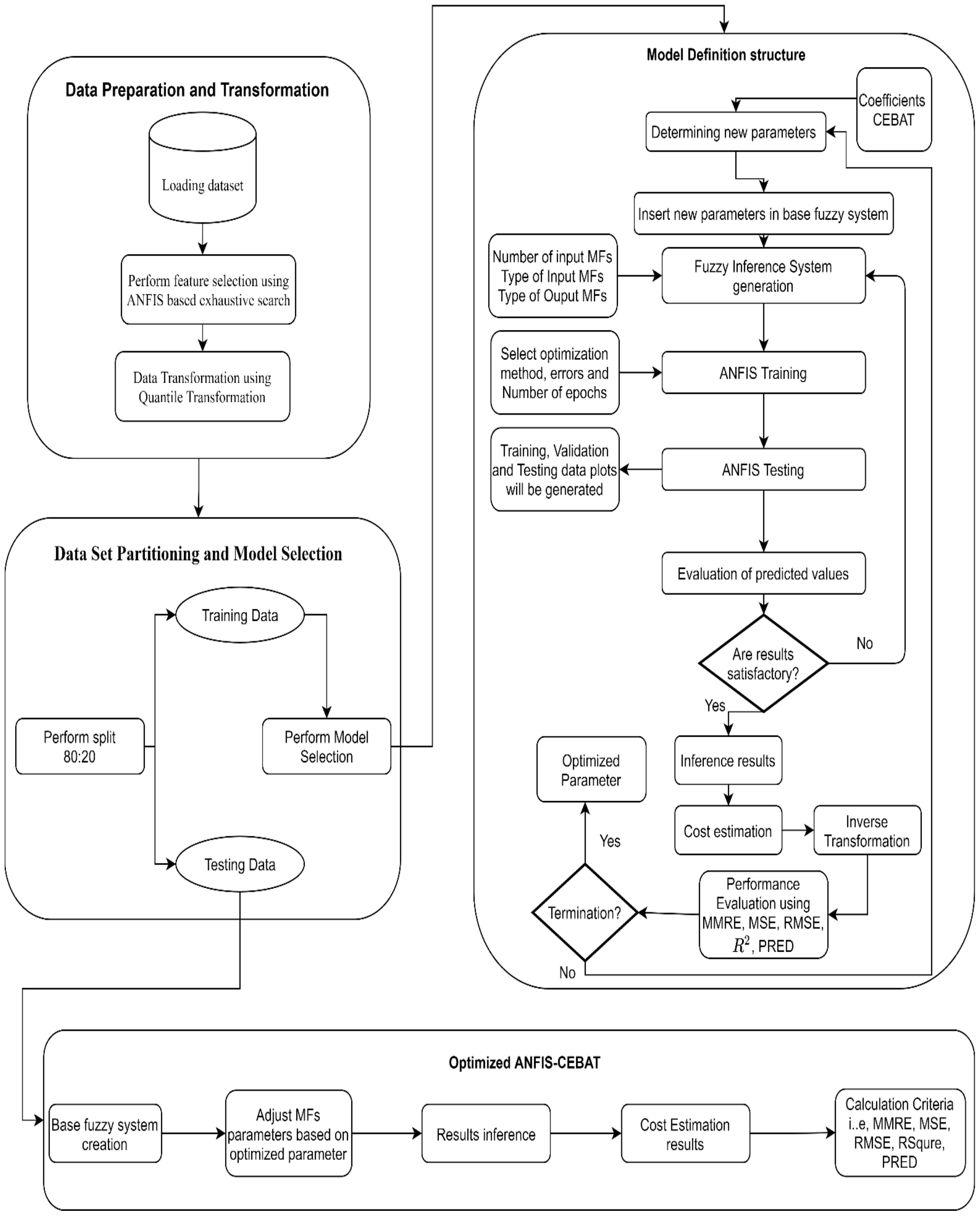

Data Preparation and Transformation
Loading dataset
Perform feature selection using ANFIS based exhaustive search
Data Transformation using Quantile Transformation
Data Set Partitioning and Model Selection
Training Data
Perform split 80:20
Perform Model Selection
Testing Data
Model Definition structure
Coefficients CEBAT
Determining new parameters
Insert new parameters in base fuzzy system
Number of input MFs Type of Input MFs Type of Ouput MFs
Fuzzy Inference System generation
Select optimization method, errors and Number of epochs
ANFIS Training
Training, Validation and Testing data plots will be generated
ANFIS Testing
Evaluation of predicted values
Are results satisfactory?
No
Yes
Optimized Parameter
Inference results
Yes
Cost estimation
Inverse Transformation
Termination?
Performance Evaluation using MMRE, MSE, RMSE, R^2, PRED
No
Optimized ANFIS-CEBAT
Base fuzzy system creation
Adjust MFs parameters based on optimized parameter
Results inference
Cost Estimation results
Calculation Criteria i..e, MMRE, MSE, RMSE, RSqure, PRED

Figure 5.1 CEBAT Flowchart

5.3 Experimental results and discussion

In this section, experimental results have been discussed from inception to the transition of the application of ANFIS-CEBAT.

5.3.1 Renaming, Identification, and Selection of features and labels

We have made use of ANFIS based exhaustive search functionality in MATLAB to find the train and validations errors of a random combination of features of the dataset. These errors help us to find correlation i.e., which combination of features in the dataset is in alignment with the target label i.e., Actual cost. In Table 5.1, it can be inferred that value of train and validation error is minimum for “No. of Story Points, Velocity and Team Salary” and therefore it is used as a feature for our algorithm.

Table 5.1 RMSE error for features set in ANFIS exhaustive search

Features	Train Error	Validation Error
No. of Story Points, Velocity, Team Size	0	0.2751
No. of Story Points, Velocity, Team Salary	0	0.2378
No. of Story Points, Team Size, Team Salary	0.0002	0.1005
No. of Story Points, Team Size, Actual Effort	0.0065	0.1802
Velocity, Team Size, Team Salary	0.0888	3.6809
Velocity, Team Size, Actual Effort	0.0003	0.415
Velocity, Team Salary, Actual Effort	0	0.2702
Team Size, Team Salary, Actual Effort	0.0071	0.4004

Few samples of the deduced features set and labels are given in Table 5.2.

Table 5.2 Sample of features and labels for cost estimation

Features			Labels
No. of Story Points	Velocity	Team Salary	Actual Cost
156	2.7	230000	1200000

202	2.5	260000	1600000
173	3.3	250000	1000000
331	3.8	300000	2100000

Figure 5.1 validates our selection of features and labels, by which it is inferred that a high correlation exists among them. It is worth noting that the "Team Salary" feature highly correlates with "No. of Story Points" and "Team Velocity", which is corroborated by the fact that a software developer's salary is directly dependent on his efforts completed within a due timeframe. Thus, it is evidencing these two features are direct components of "Team Salary". However, in the case of the "Actual Cost" feature, the figure ascertains that "No. of Story Points" highly correlates while "Team Velocity" is contrastingly having a little dependency on "Actual Cost". This implies "Team Velocity" is an indirect component for estimating the cost of a project. Hence, the choice of value for Cost Multiplier is to be judged carefully by discussion, to aid in an accurate cost estimation process.

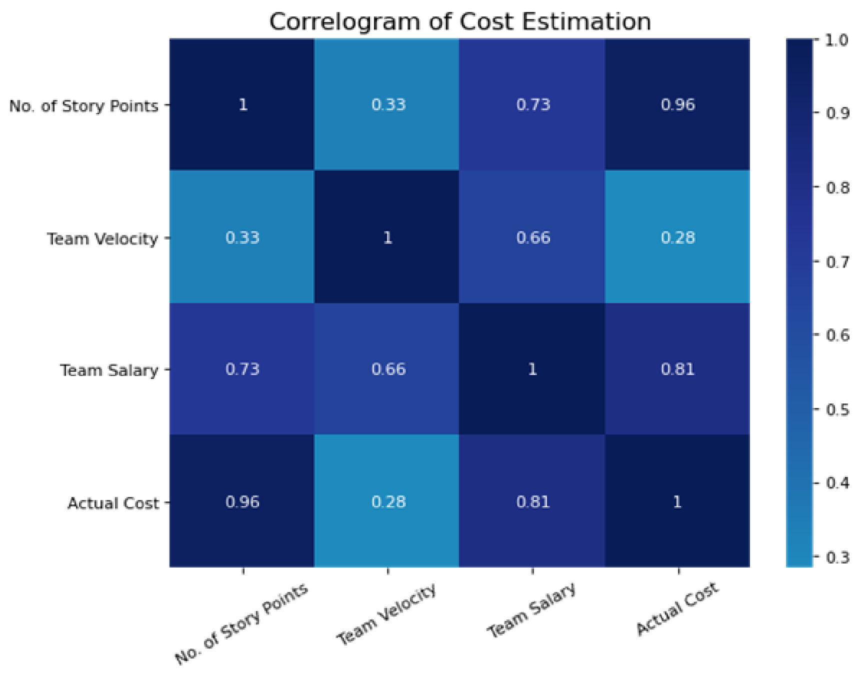


Figure 5.2 Correlogram of Cost estimation

5.3.2 Descriptive statistics of the dataset

Table 5.3 contains the descriptive statistics for Zia. The count (number of projects in the dataset), mean, standard deviation, minimum and maximum values of “No. of Story Points”, “Velocity”, “Team Salary” and “Actual Cost” are all included in the dataset. The statistic “Count” with the value 21 indicates that there are 21 projects in Zia. The average value of the fields is represented by the term "mean." The standard deviation (Std) is the difference between the field values and the Mean value. The values "Min" and "Max" represent the minimum and maximum values, respectively.

Table 5.3 Descriptive statistics for features and labels

Statistics	No. of Story Points	Velocity	Team Salary	Actual Cost
count	21.000000	21.000000	21.000000	21.000000
mean	163.714286	3.023810	246190.48	1114286.00
std	82.743062	0.438069	46419.41	705893.60
min	62.000000	2.400000	190000	350000
25%	101.000000	2.800000	220000	600000
50%	137.000000	2.900000	250000	800000
75%	211.000000	3.200000	250000	1500000
max	339.000000	4.200000	400000	3200000

After the selection of our features and labels, the dataset has been profiled to provide an overview of the various characteristics in the features and labels for cost estimation. Figure 5.3 explains the profile of the “Team Salary” attribute while Figure 5.4 explains the profile of the “Actual Cost” attribute.

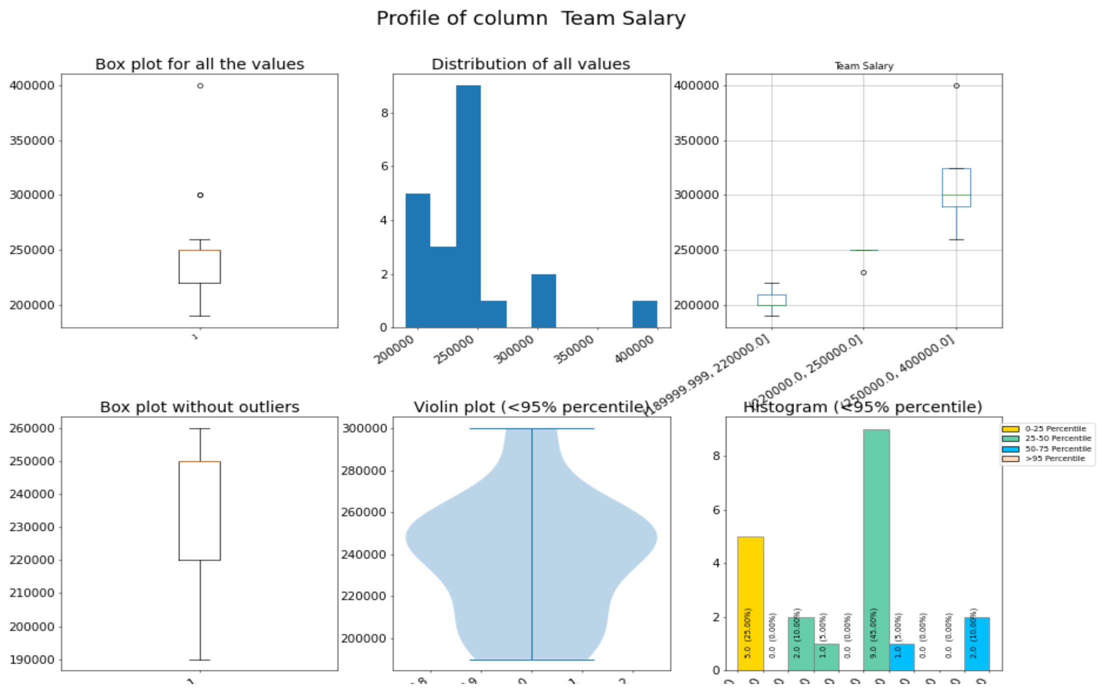


Figure 5.3 Feature profile for Team Salary

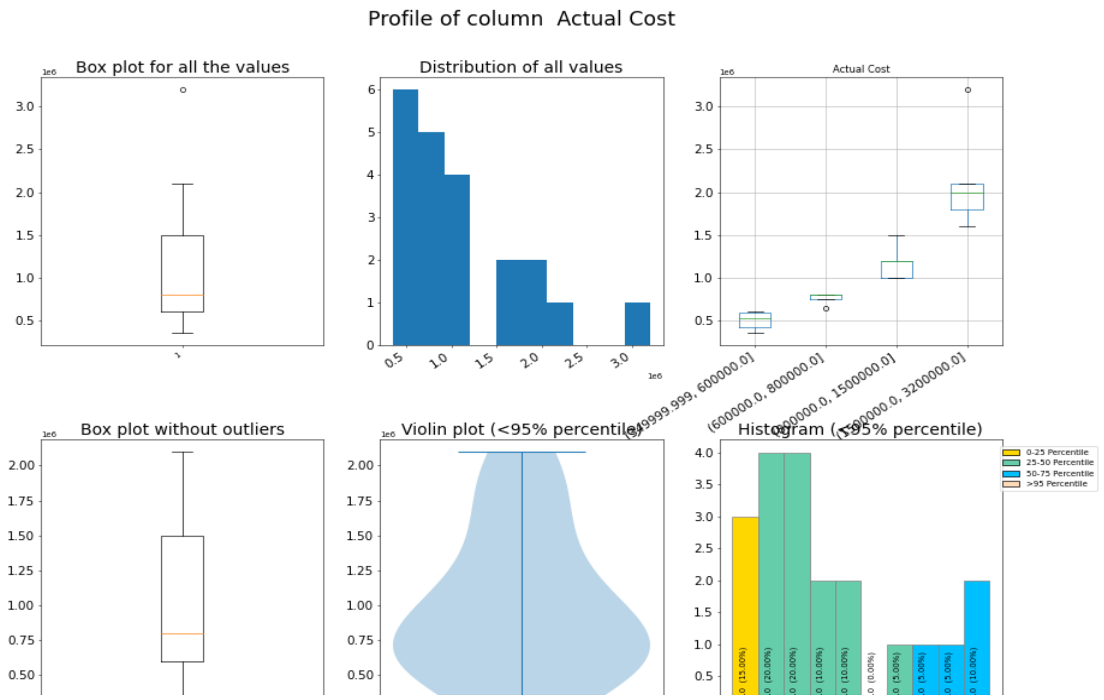


Figure 5.4 Feature profile for Actual Cost

5.3.3 Transformation of Features

The dataset contains proportionally large values for Actual Cost as compared to No. of Story Points and Velocity. It has to be transformed to a Gaussian distribution and made suitable for estimation using machine learning techniques. Hence, we used Quantile Transformation to transform the values ranging from 0 to 1. The formula for Quantile Transformation is given by the following formula:

$$Q(p) = \inf\{x \in \mathbb{R} : p \leq F(x)\} \ \forall \ p \in [0, 1] \tag{5.10}$$

A sample of the transformed values is provided in Table 5.4.

Table 5.4 Features and labels after Quantile transformation

No. of Story Points	Velocity	Team Salary	Actual Cost
0.6	0.15	0.35	0.675
0.7	0.05	0.85	0.8
0.65	0.8	0.6	0.55
0.95	0.95	0.925	0.95
0.4	1	0.925	0.35

Figure 5.5(a) and Figure 5.5(b) portray the before and after the change in values using Quantile Transformation in “Story Points”.

Figure 5.6(a) and Figure 5.6(b) portray the before and after the change in values using Quantile Transformation in “Team Velocity”.

Figure 5.7(a) and Figure 5.7(b) portray the before and after the change in values using Quantile Transformation in “Team Salary”.

Figure 5.8(a) and Figure 5.8(b) portray the before and after the change in values using Quantile Transformation in “Actual Cost”.

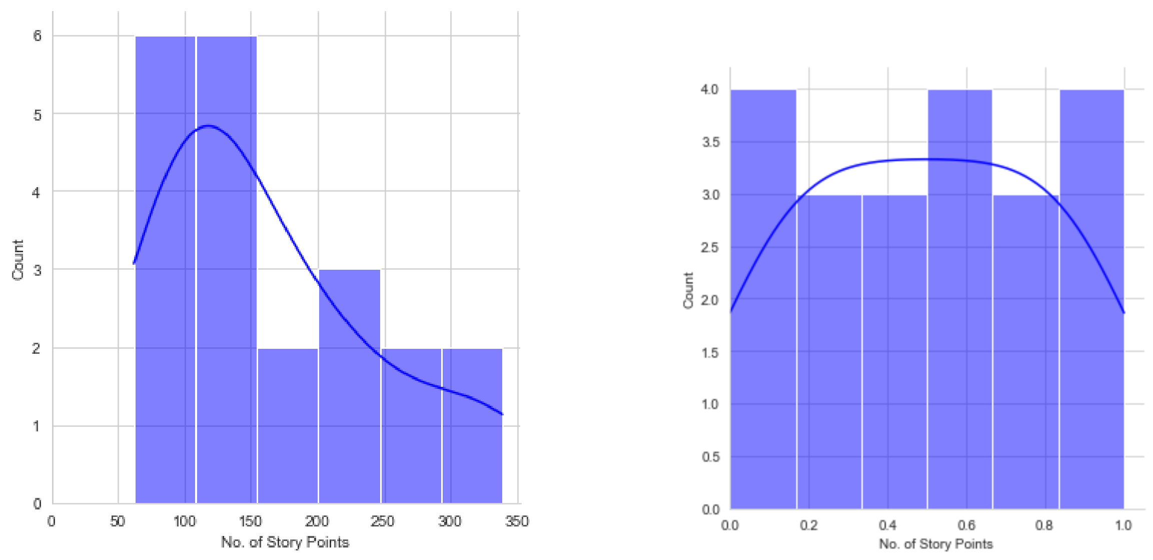


Figure 5.5(a) Quantile transformation of Number of Story Points

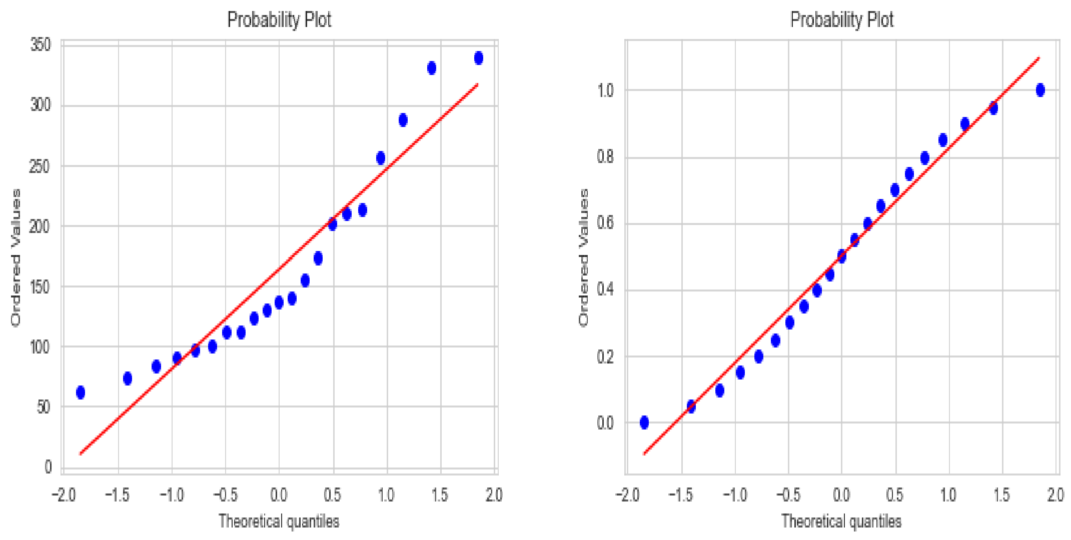


Figure 5.5(b) Probability plots after quantile transformation of Number of Story Points

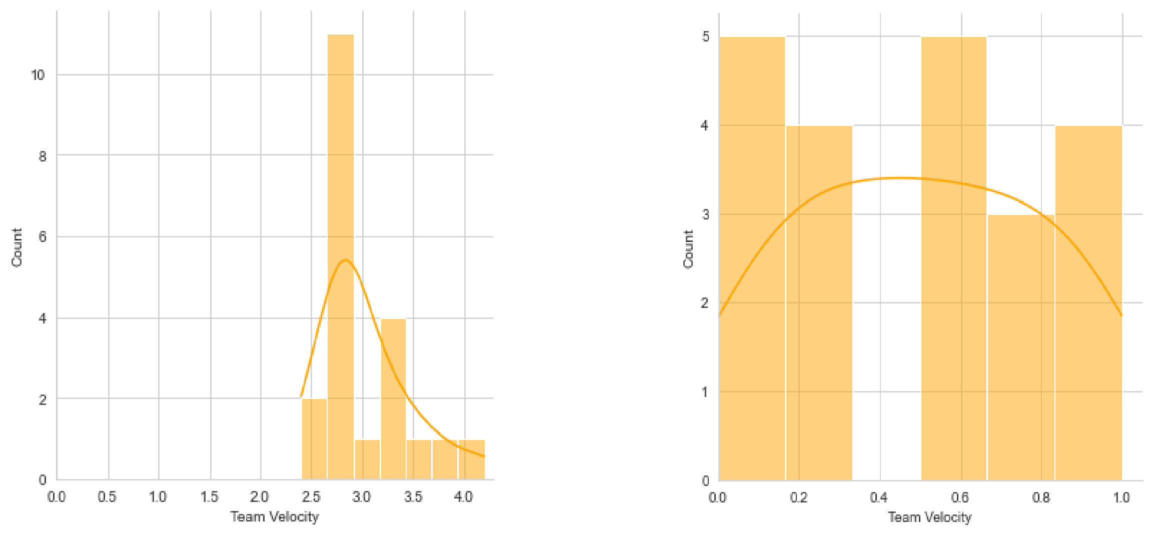


Figure 5.6(a) Quantile transformation of Team Velocity

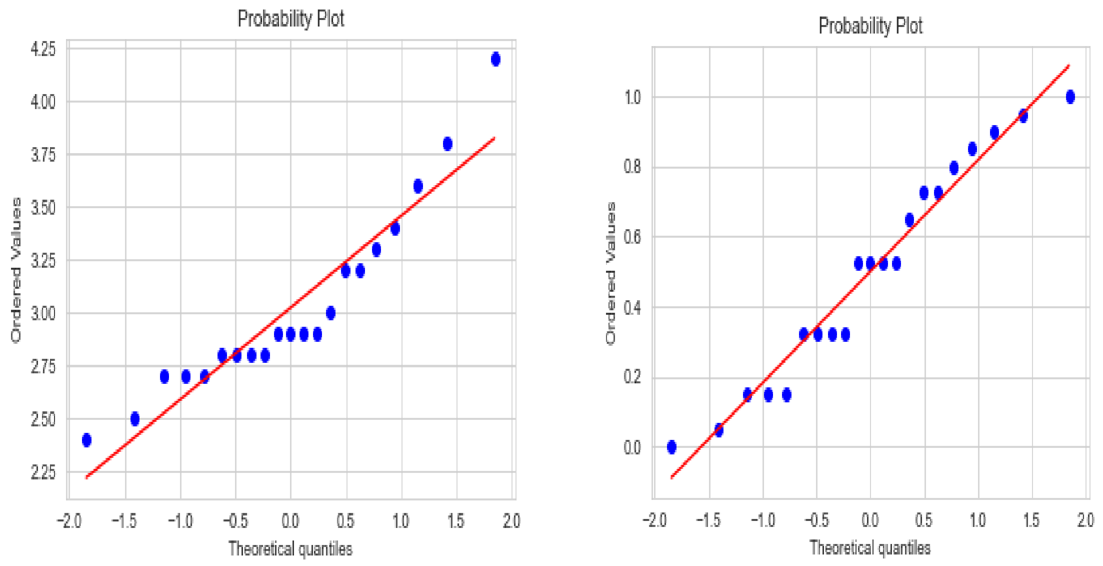


Figure 5.6(b) Probability plots after quantile transformation of Team Velocity

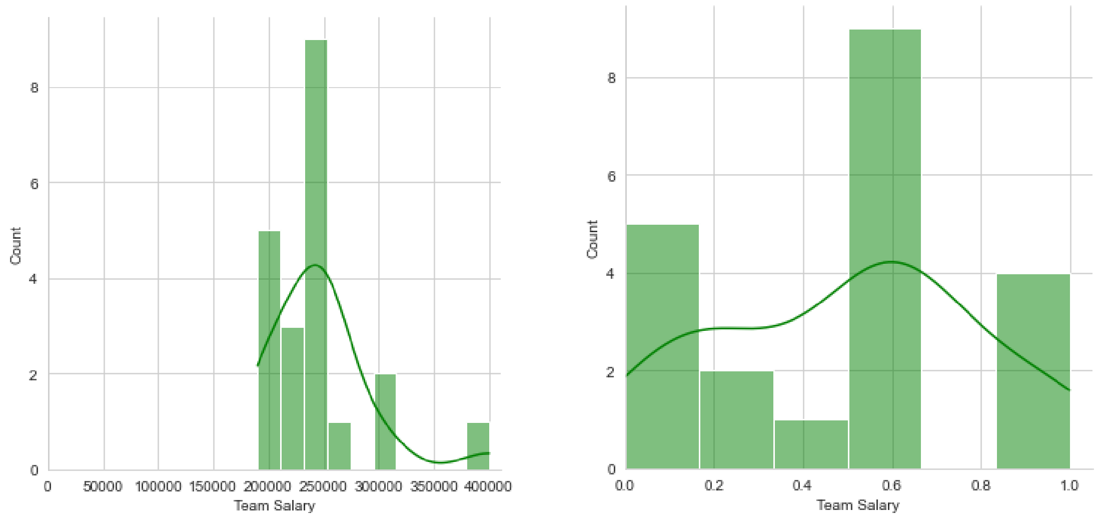


Figure 5.7(a) Quantile transformation of Team Salary

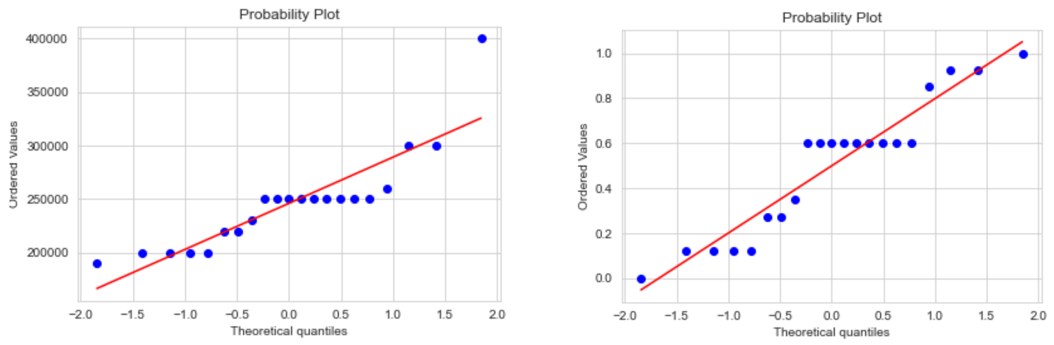


Figure 5.7(b) Probability plots after quantile transformation of Team Salary

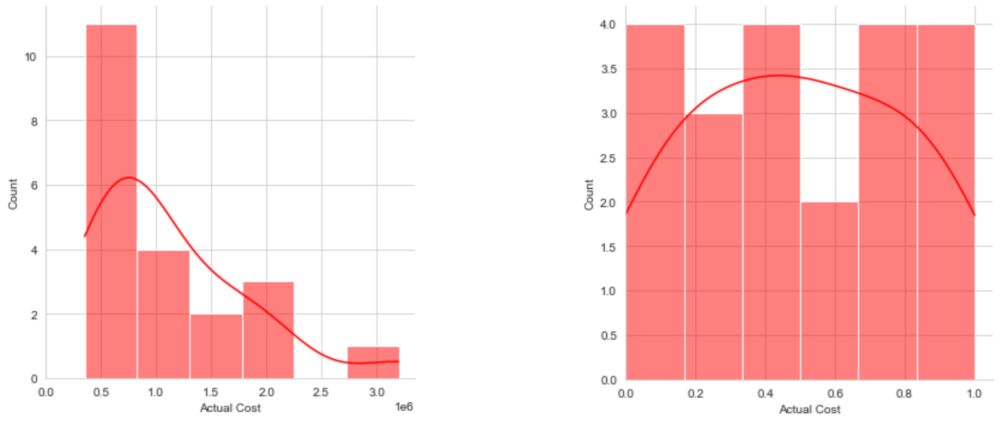
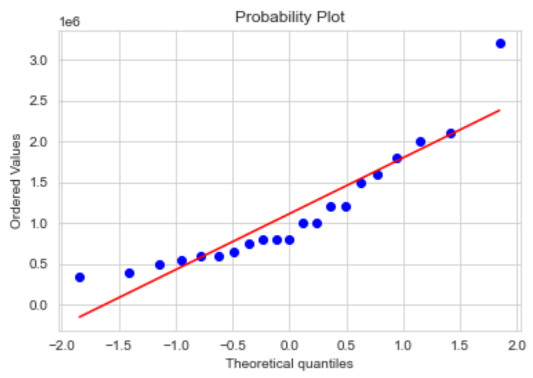


Figure 5.8(a) Quantile transformation of Actual Cost



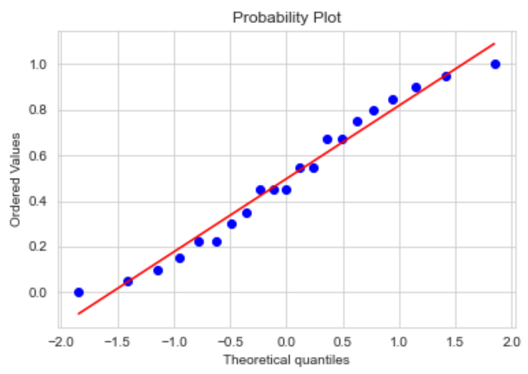


Figure 5.8(b) Probability plots after quantile transformation of Actual Cost

5.3.4 Model selection

Data loading and Generate Fuzzy Inference System

After providing the inputs to the ANFIS-CEBAT model, the antecedent layer creates the input MFs. The initial set of parameters for ANFIS and CEBAT are given in Table 5.5 and Table 5.6 respectively. The values of ANFIS parameters have been optimized using CEBAT. The Number of inputs is “3” which are “No. of Story Points”, “Velocity” and “Team Salary”. The Number of outputs is “1” which is “Actual Cost”. The Learning algorithm is “CEBAT”. The value “10” in the number of inputs MFs parameter signifies that there exists 10 gaussian MFs for each input with a unique set of gaussian parameters. “Fuzzy C-Means” Partitioning method has been employed which is used to create a base FIS. The input MF is “gaussmf (gaussian)” which represents our data in normal distribution and the output MF is “linear” which produces a singular value. The base fuzzy system is created using the “genfis3” functionality of MATLAB. The “And” method signifies the product of weights of neuro-fuzzy system with the inputs. The “Or” method utilizes “probor (probabilistic or)” which is the algebraic sum of the previous layers. The implication and aggregation are set to “min” and “max” respectively. “wtaver” i.e., weightage average is used for defuzzification. The training iterations aka epochs are set to 40 as after this value over fitting occurs. The iterations have been validated against several trials. The error tolerance is set to 1e-5.

Table 5.5 FIS parameters for cost estimation

ANFIS Parameters	
No. of inputs	3
No. of outputs	1
Learning algorithm	CEBAT
No. of input MFs	[10, 10, 10]
Partitioning method	Fuzzy C-Means
Input MF	gaussmf
Output MF	linear
Base fuzzy system	genfis3
And Method	prod
Or Method	probor
Implication	prod
Aggregation	Sum
Defuzzification	wtaver
Max. Iterations	40
Error Tolerance	1e-05

The initial BAT population size is set to “10”. The maximum number of iterations is “40”. Pulse rate signifies optimal solution searching precision of the algorithm. The tuning parameters of ANFIS are the optimal solution. The values for each parameter are obtained during several exhaustive trials.

Table 5.6 CEBAT Parameters

Population Size	10
Max Iterations	40
Pulse Rate	0.3

Loudness	0.9
F_{min}	0
F_{max}	0.1
Alpha (α)	0.9
Gamma (γ)	0.9
Cost multiplier	1.0

Building ANFIS-EEBAT model structure

After setting up the initial parameters, the proposed model's structure is shown in Figure 5.9. It contains five layers, as discussed in Figure 2.1. There are three inputs, ten pairs of input MFs, ten sets of rules, ten output MFs and one output. The three inputs are "No. of Story Points", "Velocity" and "Team Salary". The output is "Estimated Cost". The operations performed at different layers are synonymous with the description in Figure 2.1. There are three basic logical operations, "and", "our", "not" with three color codes "blue", "red" and "green" respectively. The rules are created using logical "and" operations in our case. The logical "or" and "not" operations are not used.

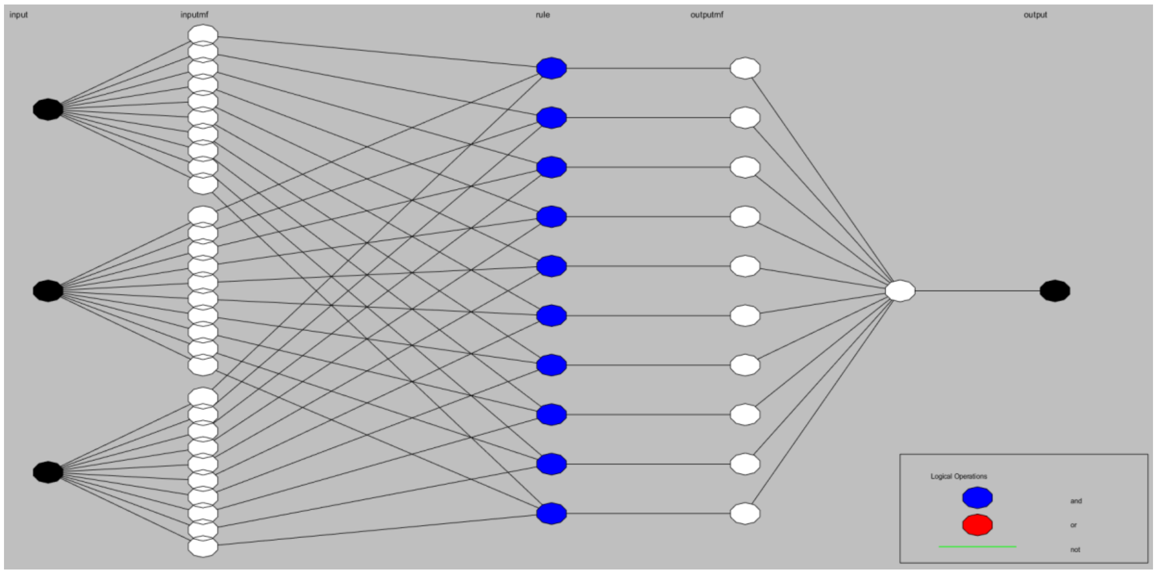


Figure 5.9 ANFIS-CEBAT structure

ANFIS-CEBAT MFs and Rules view

After the training and testing, membership function parameters are adjusted using CEBAT. The rules for the same are shown in Figure 5.11. The x-axis and y-axis in Figure 5.10(a) represent "No. of Story Points" and membership values of input1 respectively. The set of ten unique MFs created are represented as curves (in1cluster1, in1cluster2, in1cluster3, … in1cluster10) in the figure. in1 signifies Input1. The x-axis and y-axis in Figure 5.10(b) represent "Velocity" and membership values of input2 respectively. The set of ten unique MFs created are represented as curves (in1cluster1, in1cluster2, in1cluster3, in1cluster10) in the figure. in2 signifies Input2. The x-axis and y-axis in Figure 5.10(c) represent "Team Salary" and membership values of input3 respectively. The set of ten unique MFs created are represented as curves (in1cluster1, in1cluster2, in1cluster3, …, in1cluster10) in the figure. in3 signify Input3.

Each row of the plot represents a rule. There are a set of ten rules based on input and output membership functions. The red line is a slider for selecting the input values. For instance, we have selected the value of "No. of Story Points" as 0.04, "Velocity" as 0.016 and Team Salary as 0.073. The yellow color in the plots depicts how the input variable is used in the rules. The blue color in the output membership function, "Cost", signify how the output is utilized in the rules. The output of each rule is combined and defuzzified to create an aggregated output in the bottom-right plot. The estimated cost, 0.0425 is the output and is shown by the red color line.

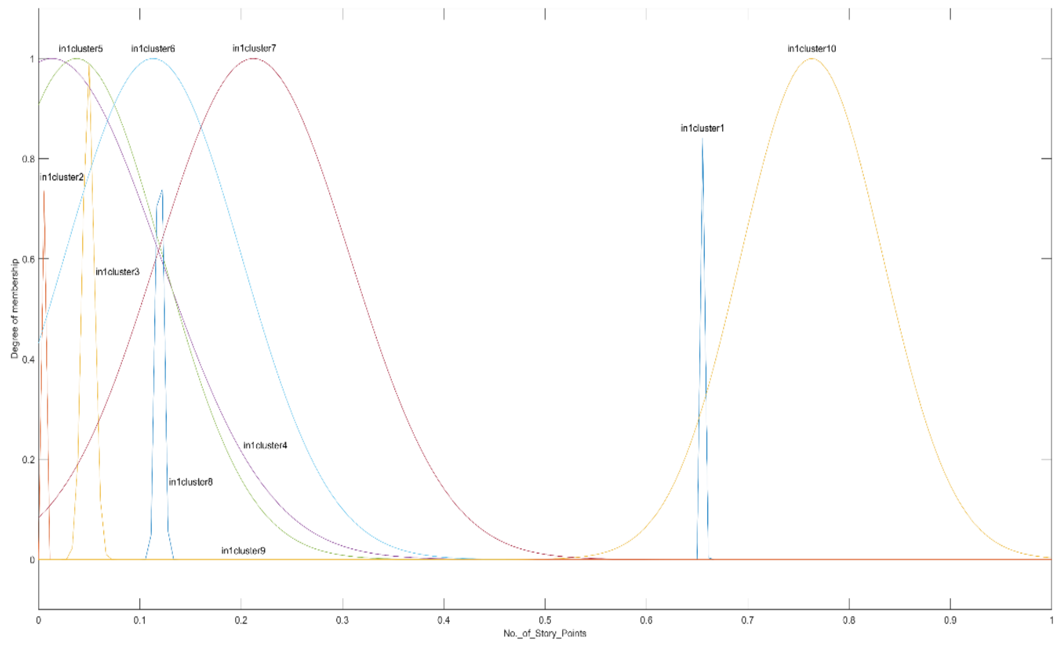


Figure 5.10(a) Membership function for No. of Story Points in ANFIS-CEBAT

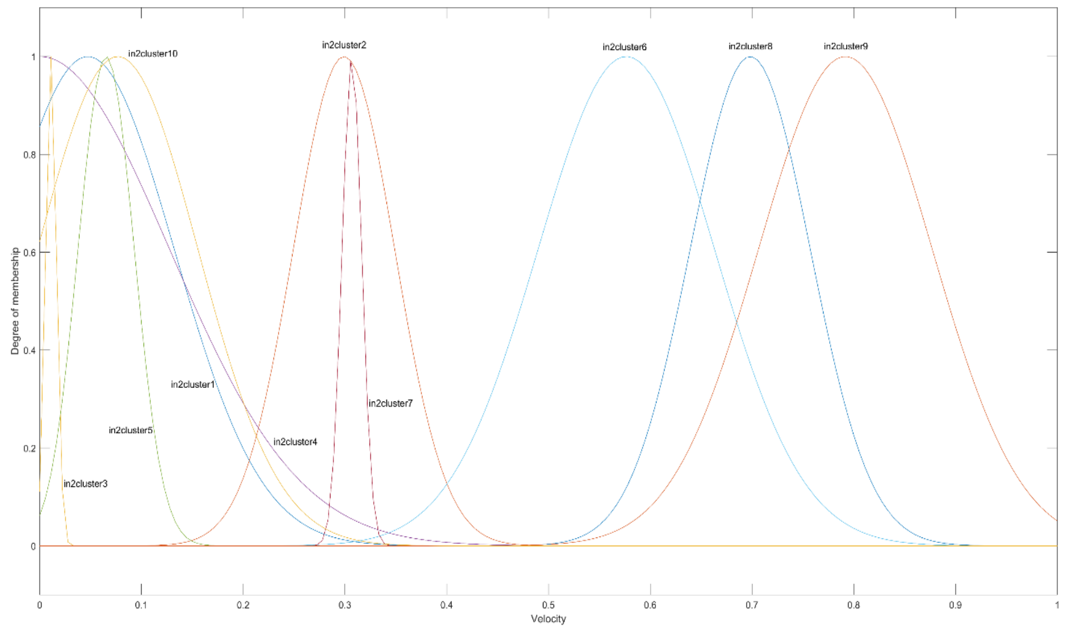


Figure 5.10(b) Membership function for Velocity in ANFIS-CEBAT

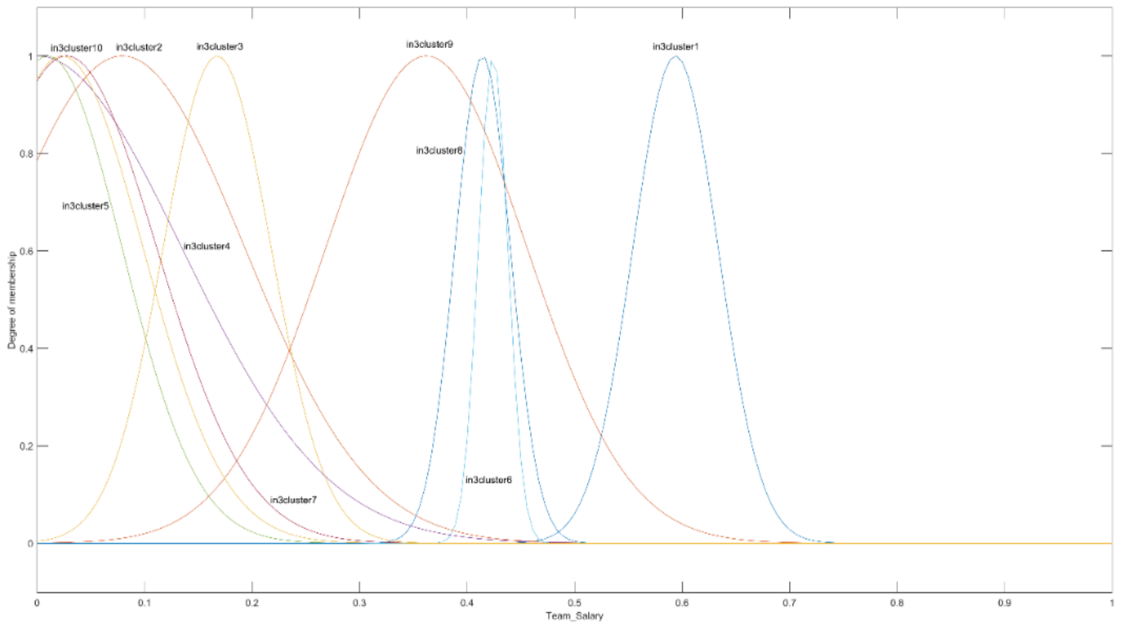


Figure 5.10(c) Membership function for Team Salary in ANFIS-CEBAT

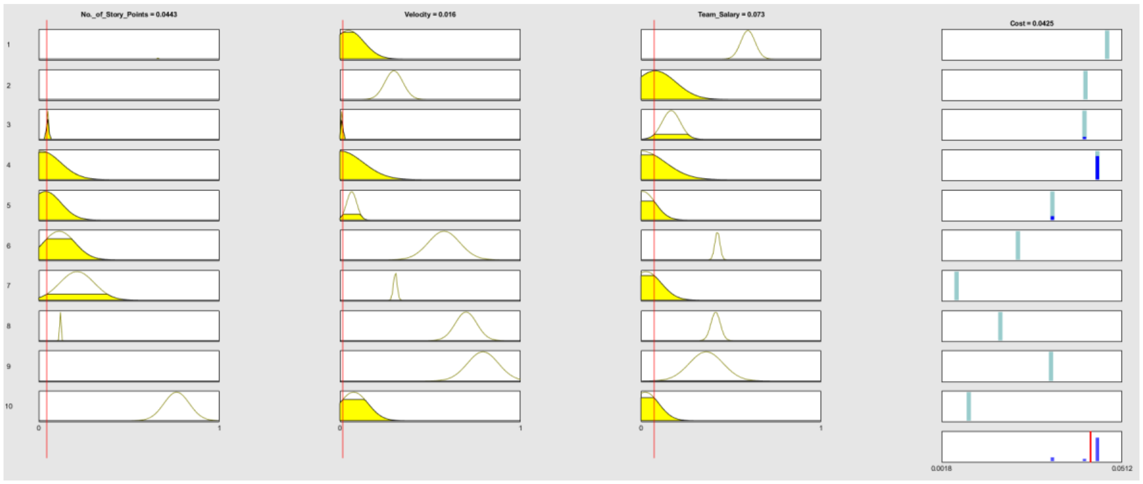


Figure 5.11 ANFIS-CEBAT Rules View

ANFIS-CEBAT Surface Plot

The surface plots are shown in Figures 5.12(a), 5.12(b), and 5.12(c) depicts the mapping of the features with the labels. It can be deduced from the surface plat that for our features, the output is linear which is following Takagi Sugeno Type 3 FIS. Surface Plot renders a 3-dimensional view of one dependent variable (“Actual Cost”) against two independent variables ("No. of Story Points", "Team Velocity", "Team Salary"; set of either two).

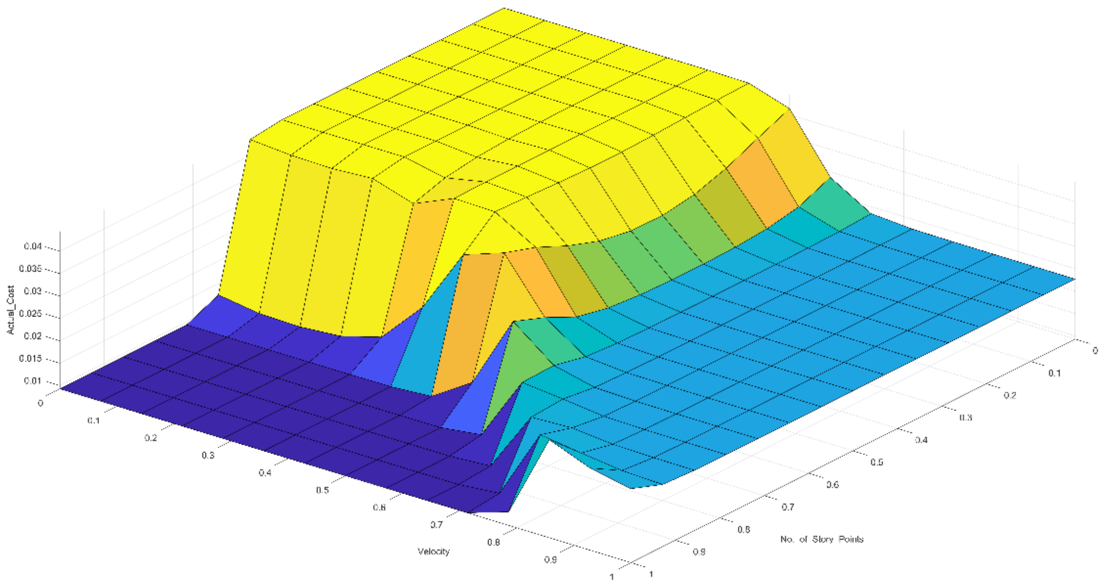


Figure 5.12(a) Surface Plot for No. of Story Points & Velocity vs Actual Cost

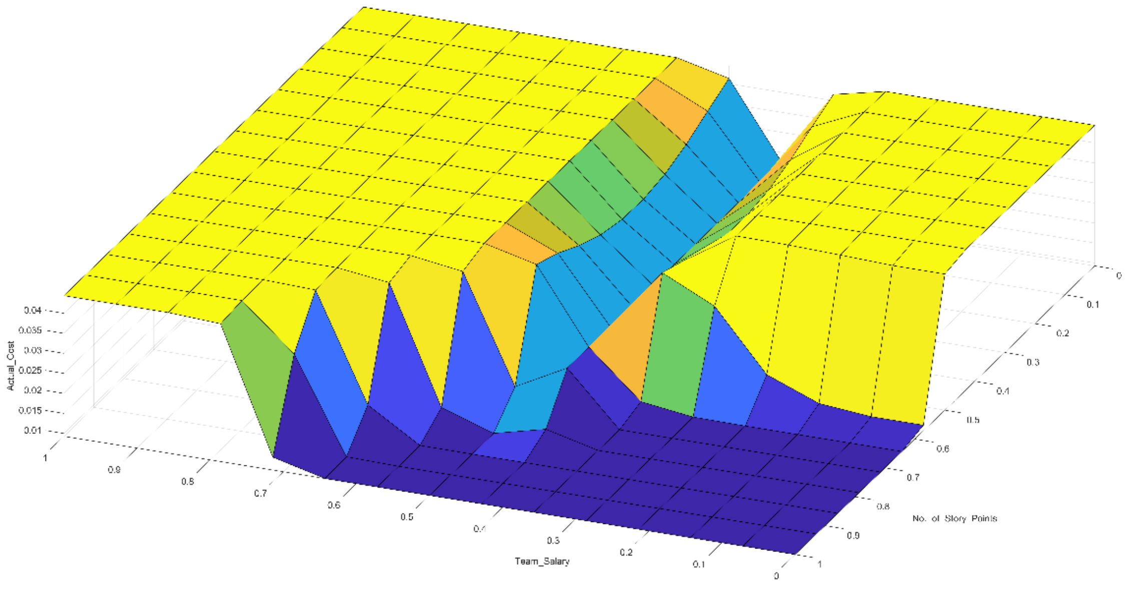


Figure 5.12(b) Surface Plot for No. of Story Points & Team Salary vs Actual Cost

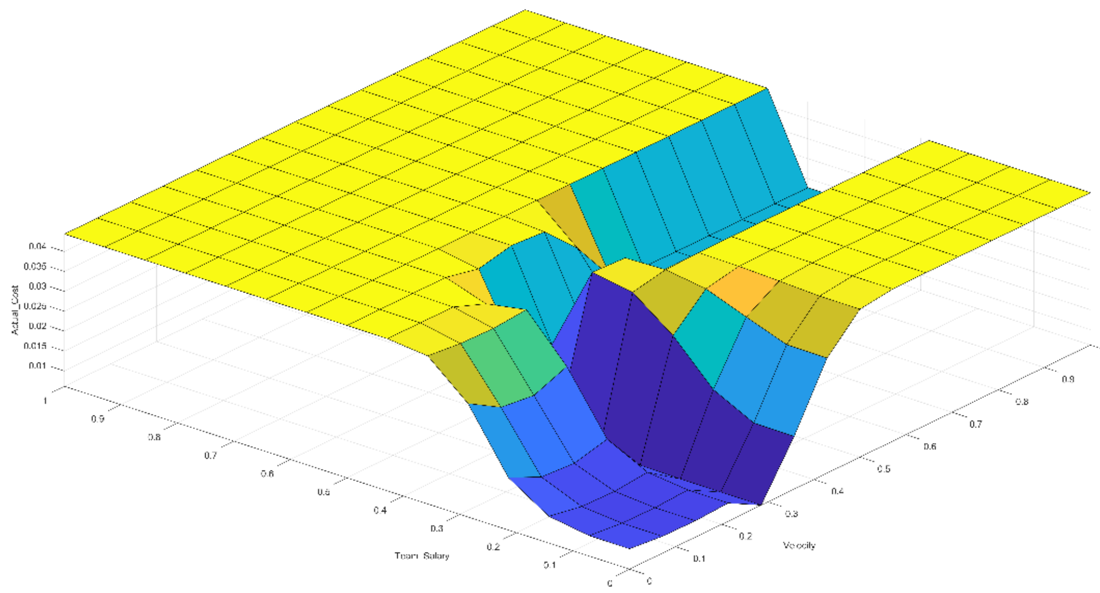


Figure 5.12(c) Surface Plot for Velocity & Team Salary vs Actual Cost

5.3.5 ANFIS-CEBAT performance evaluation and comparative analysis

We have used the Zia dataset for both performance and comparative analysis. Several renowned performance metrics have been used to analyze the performance of the ANFIS-CEBAT algorithm. We have usedR^2, MSE, RMSE, MAE, MAPE, MMRE and, PRED as shown in Table 5.7. Our analysis shows that ANFIS-CEBAT is 99.47% accurate when properly tuned and beats Zia's algorithm, standing at 99%, by a clear margin. Hence, The IT industry stakeholders thus can be assured of precise cost estimation in a Scrum project. It also provides 100% estimation accuracy up to 5.6% for PRED.

Table 5.7 ANFIS-CEBAT performance metric evaluation

Model	R^2	RMSE	MAE	MAPE	MMRE	PRED (25%)	PRED (5.6%)
ANFIS-CEBAT	0.994	24251.79212	3798.415	0.524	2.370974364	100	100
Standard ANFIS	0.899	102060.9803	10204.790	1.812	8.18773613	80	60

ANFIS-PSO (custom code)	0.976	102166.6061	70796.157	7.196	7.312754751	100	40
ANFIS-GA (custom code)	0.971	165060.4348	126420.476	12.488	12.48840386	60	20
ANFIS-BAT	0.932	131855.0466	33097.541	6.282	12.45133916	60	20
Zia [6]	0.990	39417.73196	8482.414	0.122	4.111956902	100	80

In figure 5.13, it is implied that ANFIS-CEBAT is most robust to outliers and has the least MRE for all values in the Zia dataset as compared to other algorithms, including the Ensemble algorithms like RF and Regression-based algorithms like SGD and SVR.

The values of real and predicted cost for ANFIS-CEBAT are compared against ANFIS, ANFIS-BAT, ANFIS-GA, ANFIS-PSO, and Zia as shown in Figure 5.14, which demonstrates its capability to accurately estimate the cost of substantially smaller projects.

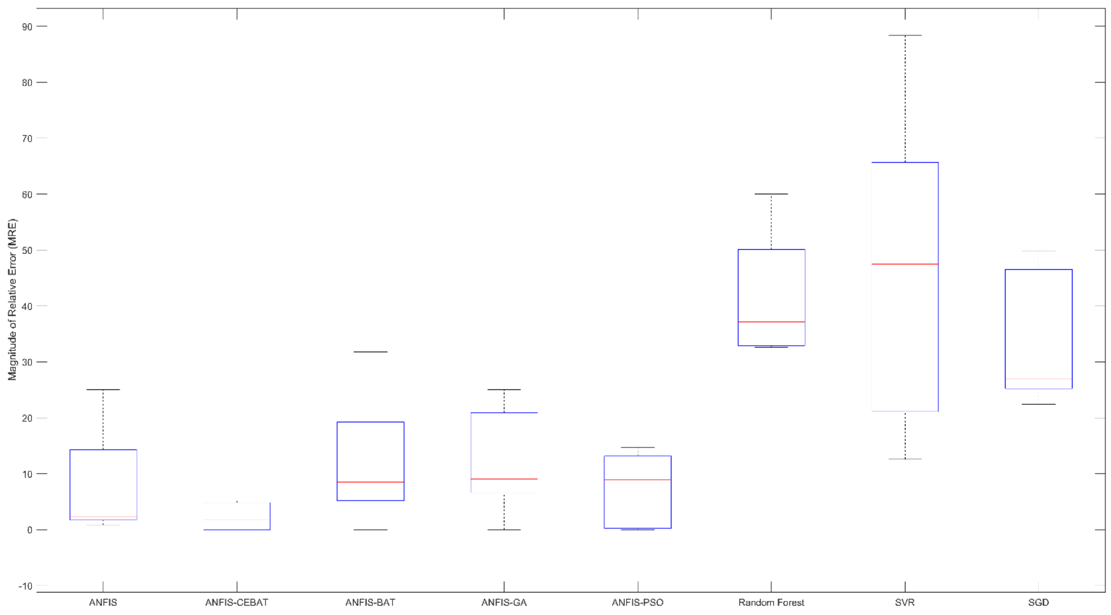


Figure 5.13 Box Plot of ANFIS-CEBAT with other models on Zia dataset

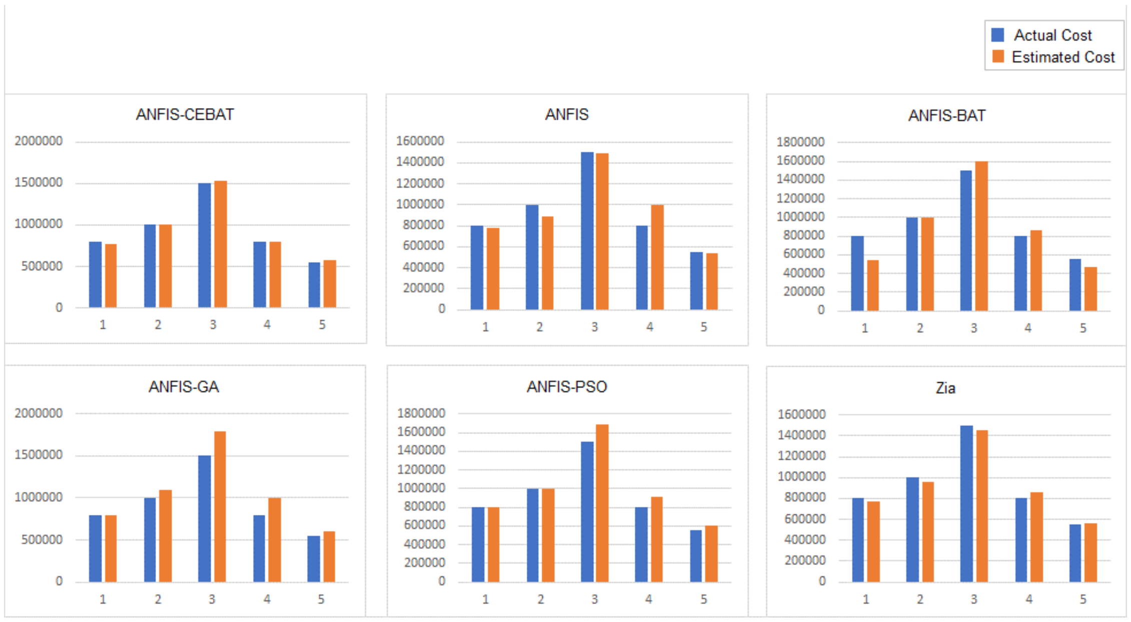


Figure 5.14 Actual vs Estimated values of Cost in Scrum Projects

5.4 Summary

The problem of cost estimation in Scrum projects is rampant as it requires meticulous details about the project, in the form of different components, directly or indirectly affecting it. Our rationale to develop an intelligent cost prediction algorithm is to assist the managers in software companies in estimating the cost of a project without delving into the details of the components that accrued the expenses for the software project. We have thus developed a meta-heuristic optimization algorithm called Cost-Efficient BAT that is having three primary factors, Energy factor, Memory Space, and Cost Multiplier. The Energy factor is crucial to indulge each bat of the population in searching for the solution according to their eagerness. The fitness of a bat is determined by this Energy factor, where the stronger bats are given the task of exploration while the weaker bats are made to exploit each of the explored spaces, to find the most optimal solution amongst them. This means that once the best cost of a project has been estimated among a pool of similar estimates, the bats shift to the next area to search the same. In the end, the final cost from the set of best costs is selected, which according to our analysis is 99.47% accurate. With the introduction of Memory Space, the bats remember the explored spaces, thus they are not subjected to repeat the search. This improves the rate of converges and prevents converging to local optima. An added benefit is the improvement in the time complexity of the algorithm. The introduction of Cost Multiplier is valuable for covering the non-linearity in cost estimation in a Scrum project. The cost will be affected by direct and indirect components, which will allow the bias to ploy in its estimation, hampering the development as a result. Hence, it is unacceptable to let human bias sway the cost estimation of a software project. Cost Multiplier takes this into account by bridging the gap between direct and indirect components. It is effective in reducing manual mode of estimation and prejudice as shown by the results.

The experimental results and discussion section gives insight into the dataset profiling, selection of features and labels, ANFIS-CEBAT model generation and structure, and the performance and comparative analysis against several other algorithms. The dataset has been described on several characteristics like count, median, standard deviation, minimum and maximum value, etc. We have used ANFIS based exhaustive search functionality in MATLAB and found the best set of features based on the lowest error. Furthermore, we have validated our feature selection with the use of a correlogram. It is also effective in quantitatively explaining the Cost Multiplier factor. Detailed profiling of these features is provided. Using the Quantile Transformer, we have described its effectiveness in transforming the data into Gaussian-like distribution, with the help of several plots.

The tuning of ANFIS is critical to its delivery of exceptional performance on the provided set of inputs. Such tuning requires resolute selection of the type and number of membership functions. To reflect the real-world nature of the dataset, we used the Gaussian function as the choice of our membership function. With the hit and trial method, we select the number of rules for each membership function as ten. Hence, we use ten sets of rules against four sets of inputs to determine the Scrum project cost.

In the performance and comparative analysis of ANFIS-CEBAT, its effectiveness against several popular algorithms, ANFIS, ANFIS-BAT, ANFIS-PSO, ANFIS-GA, Random

Forest, SVR, SGD, and Zia, its closest competitor has been summarized. From the analysis, it can be assured that ANFIS-CEBAT is 99.47% accurate, with a 100% value of PRED up to 5.6%. It is also worth noting that ANFIS-CEBAT can be utilized for small projects made by small teams and thus, is not limited to enterprise software projects.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

Estimation is an indispensable requisite that assist project managers to take firm decisions and fulfilling client commitments. When the requirements are discussed between the client and manager, they comprehend some estimated cost of the project as per their experience. As stated by the current literature, during the start of any typical IT project, managers primarily depend upon empirical estimation. Due to the complex nature of projects, estimation based on an educated guess does not yield fruitful results. Effort estimation is entirely human-based and directly proportional to the human experience in one or several domains thus threatening to delay the tasks if not properly strategized. However, cost estimation is more an arduous task because of its dependency on several direct and indirect components, which sways the actual cost of estimation, resulting in discrepancies. Human bias leads to opinionated planning and erroneous estimation.

ML-assisted estimation narrows down this gap by using logical estimation based on statistics and fuzzy logic. It helps in decreasing the difference between the actual and estimated effort to a substantial level. We have discussed the works of several authors that use several machine learning techniques for this exact purpose. Though they are profound in performance when compared against traditional techniques, yet there are challenges in adopting them in the industries. These techniques require heavy computational resources that should be scalable over time to minimize performance degradation. Hyperparameter tuning is necessary to achieve peak-level performance from the algorithm. Such challenges require using optimization algorithms that can tune the hyperparameters easily. We have presented such hybrid techniques in an attempt to understand the challenges associated with them. We again infer the same challenge of powerful performance against increased algorithmic complexity and failure to accurately estimate the cost of a Scrum project. We thereby proposed using NF systems, a hybrid system of neural networks and fuzzy logic.

We have attempted to accommodate the challenge of effort and cost estimation using the most popular NF system called ANFIS. ANFIS is capable of modeling complex human tasks, thus it’s a worthy candidate to solve the challenges in Scrum project estimation. But standard ANFIS has inherent complications that make it unsuitable for our domain. Replacing the original optimization algorithms in ANFIS with the standard Bat algorithm does not yield significant results. We attempt to solve this challenge by designing two different optimization algorithms based on standard Bat called Energy Efficient Bat Algorithm and Cost Estimating Bat Algorithm.

The ANFIS-EEBAT approach makes use of the three capabilities neural networks, fuzzy, and novel Bat hybrid EEBAT. The complexity of the proposed algorithm is managed by our novel energy equation and memory space concept. We have provided the detailed architecture of our algorithm, from preparing the data to providing values of estimated effort. The dataset we used is provided by Zia which contains substantially fewer values to train and test our model. Hence, we applied K-means SMOTE to synthesize the data, which gave us wonderful results without any repetitive values. The structure of ANFIS, its MFs, and Rules has been described using figures. We have provided the pseudo-code of the algorithm to help understand it’s working. We used MATLAB to program our algorithm and collect estimated effort data. From the comparative analysis, we conclude that our algorithm performs the best amongst other state-of-the-art algorithms viz. ANFIS, ANFIS-GA, ANFIS-PSO, RF, SGD, and SVR, against MMRE and PRED performance metrics.

For cost estimation, we have used the ANFIS-CEBAT approach that uses the novel Cost Multiplier concept to solve the non-linearity in cost estimation in software engineering, focusing on the Scrum project. We have introduced this novel feature to bridge the gap between direct and indirect components that determine the cost of a project, using a single variable. We use the original Zia dataset and program the algorithm in MATLAB.

The pseudo code of the algorithm has been provided for better understanding. We have compiled the data for estimated cost value and compared it against other similar algorithms viz. ANFIS, ANFIS-GA, ANFIS-PSO, RF, SGD, and SVR against MMRE and PRED performance metrics. The comparative analysis places our developed algorithm at the top.

Additionally, we have analyzed the time and space complexity of our developed algorithms using the Big O notation. These algorithms are ANFIS, ANFIS-GA, ANFIS-PSO, ANFIS-BAT, ANFIS-EEBAT, and ANFIS-CEBAT.

Table 6.1 Time and Space Complexity of several algorithms

Algorithm	Time Complexity	Space Complexity
ANFIS-EEBAT	O(m.n)	O(m.n)
ANFIS-CEBAT	O(m.n)	O(m.(n+p))
ANFIS	O(n)	O(m.n)
ANFIS-PSO	O(m.n)	O(m.n)
ANFIS-BAT	O(m.n)	O(m.n)
ANFIS-GA	O(g(m.n+n))	O(m.n)

Table 6.1 shows the comparison of time and space complexities of various algorithms. Here m is the number of iterations, n is the number of inputs and g is the number of generations.

We have calculated the time complexity of the entire algorithm, including the cost function used in optimization. Most algorithms require quadratic computation time on a system to execute for a certain set of values. ANFIS-GA is the slowest algorithm due to repeatedly computing the values of estimation for a set of inputs for each set of elements in the generation.

In terms of space complexity, ANFIS-CEBAT takes the maximum memory. This is due to the inherent structure of the algorithm, where we store the global solutions to keep a track of the explored solutions. The other algorithms avail quadratic units of memory to store their data.

It is worth noting that the time-space trade-off affects the feasibility of the algorithms. Some algorithms are extremely space-efficient while some are extremely time-efficient but not both.

ANFIS-EEBAT and ANFIS-CEBAT provide the best time-space tradeoff for each of their purposes, effort, and cost estimation respectively. ANFIS-EEBAT is most effective when the inputs increase considerably. ANFIS-CEBAT models the non-linearity nature of cost estimation with added space complexity, which is beneficial in the long term as memory becomes cheaper. Hence, these algorithms provide the perfect blend of performance and accuracy.

6.2 Future Work

Project estimation is a critical step in the software development process. The current state of the IT industry still lacks the accurate estimation ability hence the transition to Agile methodologies is still a challenge. In our contributions to developing a system of effort and cost estimation to assist project managers, whether experienced or new to the role, there is a lot of research potential that we have identified in this field. Many modern machine learning techniques and optimization algorithms remain to be tested for estimation. They can further add to the literature of the current work. The factors that contribute to estimation are plenty but in the current scenario only. Potential accelerating and decelerating factors are yet to be gauged due to changes in requirements over time. It is vital to assess and take these factors into account. We have estimated the effort based on the number of story points and velocity in a sprint which we have assumed are validated. If we consider estimation sprint-wise, the descriptions of user stories can be used to analyze and estimate the effort, using Natural Language Processing techniques. Cost estimation can be improved as well. In an extension of this idea, we can add another component that is unique to the industry and team. There currently exists no generic scale for story size and story complexity. A team that works on a project is liable to work with some set standard that influences their strategy over the process of development and estimation. We thereby have identified that this unique component when inserted into the estimation process, will be responsible for adapting to the needs of every team based on their individuals, suitable to a particular feature of the project of industry.

REFERENCES

[1] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, “Neural network models for software development effort estimation: a comparative study,” *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2369–2381, 2016, doi: 10.1007/s00521-015-2127-1.

[2] B. Prakash and V. Viswanathan, “A survey on software estimation techniques in traditional and agile development models,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 7, no. 3, pp. 867–876, 2017, doi: 10.11591/ijeecs.v7.i3.pp867-876.

[3] S. Bilgaiyan, S. Sagnika, S. Mishra, and M. Das, “A systematic review on software cost estimation in Agile Software Development,” *J. Eng. Sci. Technol. Rev.*, vol. 10, no. 4, pp. 51–64, 2017, doi: 10.25103/jestr.104.08.

[4] L. R. Nerkar, “Software Cost Estimation using Algorithmic Model and Non-Algorithmic Model a Review,” *Int. J. Comput. Appl.*, vol. 0975–8887, pp. 4–7, 2014.

[5] J. M. Alostad, L. R. A. Abdullah, and L. S. Aali, “A Fuzzy based Model for Effort Estimation in Scrum Projects,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 9, pp. 270–277, 2017, [Online]. Available: www.ijacsa.thesai.org.

[6] Ziauddin, S. K. Tipu, and S. Zia, “An Effort Estimation Model for Agile Software Development,” *Adv. Comput. Sci. its Appl.*, vol. 2, no. 1, pp. 314–324, 2012.

[7] M. Usman, J. Börstler, and K. Petersen, *An Effort Estimation Taxonomy for Agile Software Development*, vol. 27, no. 4. 2017.

[8] M. Choetkiertikul, H. K. Dam, T. Tran, T. T. M. Pham, A. Ghose, and T. Menzies, “A deep learning model for estimating story points,” *IEEE Trans. Softw. Eng.*, vol. 45, no. 7, pp. 637–656, 2019, doi: 10.1109/TSE.2018.2792473.

[9] J. S. R. Jang, “ANFIS: Adaptive-Network-Based Fuzzy Inference System,” *IEEE Trans. Syst. Man Cybern.*, vol. 23, no. 3, pp. 665–685, 1993, doi: 10.1109/21.256541.

[10] T. Chen and C. Guestrin, “XGBoost : A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data*, 2016, pp. 785–794.

[11] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, “CatBoost : unbiased boosting with categorical features,” in *32nd Conference on Neural Information Processing Systems*, 2018, no. Section 4, pp. 1–11.

[12] A. Kaushik, D. K. Tayal, and K. Yadav, “The Role of Neural Networks and Metaheuristics in Agile Software Development Effort Estimation,” *Int. J. Inf. Technol. Proj. Manag.*, vol. 11, no. 2, pp. 50–71, 2020, doi: 10.4018/IJITPM.2020040104.

[13] A. Kaushik, D. K. Tayal, and K. Yadav, “A Comparative Analysis on Effort Estimation for Agile and Non-agile Software Projects Using DBN-ALO,” *Arab. J. Sci. Eng.*, vol. 45, pp. 2605–2618, 2020, doi: 10.1007/s13369-019-04250-6.

[14] A. L. I. Oliveira, “Estimation of software project effort with support vector regression,” *Neurocomputing*, vol. 69, pp. 1749–1753, 2006, doi: 10.1016/j.neucom.2005.12.119.

[15] T. T. Khuat and M. H. Le, “A Novel Hybrid ABC-PSO Algorithm for Effort Estimation of Software Projects Using Agile Methodologies,” *J. Intell. Syst.*, vol. 27, no. 3, pp. 489–506, 2018, doi: 10.1515/jisys-2016-0294.

[16] X. S. Yang, “A new metaheuristic Bat-inspired Algorithm,” *Stud. Comput. Intell.*, vol. 284, pp. 65–74, 2010, doi: 10.1007/978-3-642-12538-6_6.

[17] M. Pant, K. Deep, J. C. Bansal, K. N. Das, and A. K. Nagar, *Soft computing for problem solving*, vol. 9, no. 1. 2018.

[18] L. Jun, L. Liheng, and W. Xianyi, “A double-subpopulation variant of the bat algorithm,” *Appl. Math. Comput.*, vol. 263, pp. 361–377, 2015, doi: 10.1016/j.amc.2015.04.034.

[19] R. Y. M. Nakamura, L. A. M. Pereira, D. Rodrigues, K. A. P. Costa, J. P. Papa, and X. S. Yang, “Binary Bat Algorithm for Feature Selection,” *Swarm Intell. Bio-Inspired Comput.*, no. 2010, pp. 225–237, 2013, doi: 10.1016/B978-0-12-405163-8.00009-0.

[20] S. Eskandari and M. M. Javidi, “A novel hybrid bat algorithm with a fast clustering-based hybridization,” *Evol. Intell.*, pp. 1–16, 2019, doi: 10.1007/s12065-019-00307-5.

[21] X. Shan, K. Liu, and P. L. Sun, “Modified Bat Algorithm Based on Lévy Flight and Opposition Based Learning,” *Sci. Program.*, pp. 1–13, 2016, doi: 10.1155/2016/8031560.

[22] A. Rekaby, “Directed Artificial Bat Algorithm (DABA) - A new bio-inspired algorithm,” in *International Conference on Advances in Computing, Communications and Informatics, ICACCI*, 2013, pp. 1241–1246, doi: 10.1109/ICACCI.2013.6637355.

[23] A. O. Topal and O. Altun, “A novel meta-heuristic algorithm: Dynamic Virtual Bats Algorithm,” *Inf. Sci. (Ny).*, vol. 354, pp. 222–235, 2016, doi: 10.1016/j.ins.2016.03.025.

[24] A. Alihodzic and M. Tuba, “Improved bat algorithm applied to multilevel image thresholding,” *Sci. World J.*, vol. 2014, no. 176718, pp. 1–16, 2014, doi: 10.1155/2014/176718.

[25] S. S. Guo, J. S. Wang, and X. X. Ma, “Improved Bat Algorithm Based on Multipopulation Strategy of Island Model for Solving Global Function Optimization Problem,” *Comput. Intell. Neurosci.*, pp. 1–12, 2019, doi: 10.1155/2019/6068743.

[26] N. S. Jaddi, S. Abdullah, and A. R. Hamdan, “Optimization of neural network model using modified bat-inspired algorithm,” *Appl. Soft Comput. J.*, vol. 37, pp. 71–86, 2015, doi: 10.1016/j.asoc.2015.08.002.

[27] M. Fozuni Shirjini, A. Nikanjam, and M. Aliyari Shoorehdeli, “Stability analysis of the particle dynamics in bat algorithm: standard and modified versions,” *Eng. Comput.*, no. 0123456789, 2020, doi: 10.1007/s00366-020-00979-z.

[28] X. S. Yang, “Bat algorithm for multi-objective optimisation,” *Int. J. Bio-Inspired Comput.*, vol. 3, no. 5, pp. 267–274, 2011, doi: 10.1504/IJBIC.2011.042259.

[29] Y. Wang *et al.*, “A novel bat algorithm with multiple strategies coupling for numerical optimization,” *Mathematics*, vol. 7, no. 2, pp. 1–17, 2019, doi: 10.3390/math7020135.

[30] Q. Liu, L. Wu, W. Xiao, F. Wang, and L. Zhang, “A novel hybrid bat algorithm for solving continuous optimization problems,” *Appl. Soft Comput. J.*, vol. 73, pp. 67–82, 2018, doi: 10.1016/j.asoc.2018.08.012.

[31] M. Chawla and M. Duhan, “Bat algorithm: A survey of the state-of-the-art,” *Appl. Artif. Intell.*, vol. 29, no. 6, pp. 617–634, 2015, doi: 10.1080/08839514.2015.1038434.

[32] R. K. Mallidi and M. Sharma, “Study on Agile Story Point Estimation Techniques and Challenges,” *Int. J. Comput. Appl.*, vol. 174, no. 13, pp. 9–14, 2021, doi: 10.5120/ijca2021921014.

[33] A. Karapantelakis, “Estimating costs for adopting and using model-based testing in agile SCRUM teams,” *Proc. - 2021 IEEE 14th Int. Conf. Softw. Testing, Verif. Valid. Work. ICSTW 2021*, pp. 199–204, 2021, doi: 10.1109/ICSTW52544.2021.00042.

[34] N. Gupta and R. P. Mahapatra, “An effective agile development process by a hybrid intelligent effort estimation protocol,” *J. Ambient Intell. Humaniz. Comput.*, no. 0123456789, 2021, doi: 10.1007/s12652-021-03088-x.

[35] R. A. de Morais, “DEEP LEARNING BASED MODELS FOR SOFTWARE EFFORT ESTIMATION USING STORY POINTS IN AGILE ENVIRONMENTS,” 2021.

[36] Z. R. Mohsin, “Application of Artificial Neural Networks in Prediction of Software Development Effort,” *Turkish J. Comput. Math.*, vol. 12, no. 14, pp. 4186–4202, 2021.

[37] S. Briatore and A. Golkar, “Estimating Task Efforts in Hardware Development Projects in a Scrum Context,” *IEEE Syst. J.*, pp. 1–7, 2021, doi: 10.1109/JSYST.2021.3049737.

[38] K. Periyasamy and J. Chianelli, “A project tracking tool for scrum projects with machine learning support for cost estimation,” in *29th International Conference on Software Engineering and Data Engineering*, 2021, vol. 76, pp. 86–94, doi: 10.29007/6vwh.

[39] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, “An effective approach for software project effort and duration estimation with machine learning algorithms,” *J. Syst. Softw.*, vol. 137, pp. 184–196, 2018, doi: 10.1016/j.jss.2017.11.066.

[40] H. H. Arifin, J. Daengdej, and N. T. Khanh, “An empirical study of effort-size and effort-time in expert-based estimations,” in *Proceedings - 8th IEEE International Workshop on Empirical Software Engineering in Practice*, 2017, pp. 35–40, doi: 10.1109/IWESEP.2017.21.

[41] M. Salmanoglu, T. Hacaloglu, and O. Demirörs, “Effort Estimation for Agile Software Development : Comparative Case Studies Using COSMIC Functional Size Measurement and Story Points,” in *IWSM/Mensura*, 2017, pp. 1–9.

[42] R. de A. Araújo, A. L. I. Oliveira, and S. Meira, “A class of hybrid multilayer perceptrons for software development effort estimation problems,” *Expert Syst. Appl.*, vol. 90, pp. 1–12, 2017, doi: 10.1016/j.eswa.2017.07.050.

[43] S. Dragicevic, S. Celar, and M. Turic, “Bayesian network model for task effort estimation in agile software development,” *J. Syst. Softw.*, vol. 127, pp. 109–119, 2017, doi: 10.1016/j.jss.2017.01.027.

[44] V.-S. Ionescu, H. Demian, and I.-G. Czibula, “Natural Language Processing and Machine Learning Methods for Software Development Effort Estimation,” *Stud. Informatics Control*, vol. 26, no. 2, pp. 219–228, 2017, doi: 10.24846/v26i2y201710.

[45] M. Łabędzki, P. P. Promiński, A. Rybicki, and M. Wolski, “Agile effort estimation in software development projects – case study,” *Cent. Eur. Rev. Econ. Manag.*, vol. 1, no. 3, pp. 135–152, 2017.

[46] L. Lavazza and S. Morasca, “On the Evaluation of Effort Estimation Models,” in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering - EASE’17*, 2017, pp. 41–50, doi: 10.1145/3084226.3084260.

[47] J. López-Martínez, A. Ramírez-Noriega, R. Juárez-Ramírez, G. Licea, and S. Jiménez, "User stories complexity estimation using Bayesian networks for inexperienced developers," *Cluster Comput.*, pp. 1–14, 2017, doi: 10.1007/s10586-017-0996-z.

[48] M. Owais and R. Ramakishore, "Effort, duration and cost estimation in agile software development," in *9th International Conference on Contemporary Computing*, 2017, pp. 1–5, doi: 10.1109/IC3.2016.7880216.

[49] S. H. Samareh Moosavi and V. Khatibi Bardsiri, "Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation," *Eng. Appl. Artif. Intell.*, vol. 60, pp. 1–15, 2017, doi: 10.1016/j.engappai.2017.01.006.

[50] S. M. Satapathy and S. K. Rath, "Empirical assessment of machine learning models for agile software development effort estimation using story points," *Innov. Syst. Softw. Eng.*, vol. 13, no. 2–3, pp. 191–200, 2017, doi: 10.1007/s11334-017-0288-z.

[51] A. Sharma and R. Ranjan, "Software Effort Estimation using Neuro Fuzzy Inference System : Past and Present," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 5, no. 8, pp. 78–83, 2017.

[52] B. Tanveer, "Guidelines for utilizing change impact analysis when estimating effort in agile software development," in *EASE*, 2017, pp. 1–6, doi: 10.1145/3084226.3084284.

[53] S. Basri, N. Kama, F. Haneem, and S. A. Ismail, "Predicting effort for requirement changes during software development," in *Proceedings of the Seventh Symposium on Information and Communication Technology - SoICT '16*, 2016, pp. 380–387, doi: 10.1145/3011077.3011096.

[54] S. Bilgaiyan, S. Mishra, and M. Das, "A Review of Software Cost Estimation in Agile Software Development Using Soft Computing Techniques," in *2nd International Conference on Computational Intelligence and Networks (CINE)*, 2016, pp. 112–117, doi: 10.1109/CINE.2016.27.

[55] A. Sharma and Karambir, "Experimental Recognition of Random Forest for Agile Software Effort Estimation," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 6, no. 7, pp. 520–524, 2016.

[56] K. Z. Khan, S. K. Tipu, and S. Zia, "An Intelligent Software Effort Estimation System," *J. Expert Syst.*, vol. 1, no. 4, pp. 91–98, 2012.

[57] K. Moharreri, A. V. Sapre, J. Ramanathan, and R. Ramnath, "Cost-Effective Supervised Learning Models for Software Effort Estimation in Agile Environments," in *IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, 2016, pp. 135–140, doi: 10.1109/COMPSAC.2016.85.

[58] B. Tanveer, L. Guzmán, and U. M. Engel, "Understanding and improving effort estimation in Agile software development- an industrial case study," in *Proceedings of the International Workshop on Software and Systems Process*, 2016, pp. 41–50, doi: 10.1145/2904354.2904373.

[59] A. Panda, S. M. Satapathy, and S. K. Rath, "Empirical Validation of Neural Network models for Agile Sooftware Effort Estimation based on Story Points," in *3rd International Conference on Recent Trends in Computing*, 2015, pp. 772–781.

[60] M. Usman, E. Mendes, and J. Börstler, "Effort estimation in Agile software development: A survey on the state of the practice," in *ACM International Conference Proceeding Series*, 2015, pp. 1–10, doi: 10.1145/2745802.2745813.

[61] H. Zahraoui and M. A. Janati Idrissi, "Adjusting story points calculation in scrum effort & time estimation," in *10th International Conference on Intelligent Systems: Theories and Applications, SITA*, 2015, pp. 1–8, doi: 10.1109/SITA.2015.7358400.

[62] V. S. Dave and K. Dutta, "Neural network based models for software effort estimation: A review," *Artif. Intell. Rev.*, vol. 42, no. 2, pp. 295–307, 2014, doi: 10.1007/s10462-012-9339-x.

[63] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, "Neural network models for software development effort estimation: a comparative study," *Neural Comput. Appl.*, pp. 1–15, 2015, doi: 10.1007/s00521-015-2127-1.

[64] I. Manga and N. V. Blamah, "A particle Swarm Optimization-based Framework for Agile Software Effort Estimation," *Int. J. Eng. Sci.*, vol. 3, no. 6, pp. 30–36, 2014, [Online]. Available: http://www.theijes.com/papers/v3-i6/Version-5/D0365030036.pdf.

[65] R. Popli and N. Chauhan, "Agile estimation using people and project related factors," in *International Conference on Computing for Sustainable Global Development, INDIACom 2*, 2014, pp. 564–569, doi: 10.1109/IndiaCom.2014.6828023.

[66] G. S. Rajput and R. Litoriya, "Corad Agile Method for Agile Software Cost Estimation," *Open Access Libr. J. Corad*, vol. 1, pp. 1–13, 2014, doi: 10.4236/oalib.1100579.

[67] R. Popli and N. Chauhan, "Cost and effort estimation in agile software development," in *Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on*, 2014, pp. 57–61, doi: 10.1109/ICROIT.2014.6798284.

[68] S. M. Satapathy, A. Panda, and S. K. Rath, "Story Point Approach based Agile Software Effort Estimation using Various SVR Kernel Methods," in *The 26th International Conference on Software Engineering and Knowledge Engineering,*

2014, pp. 304–307, [Online]. Available: https://ksiresearchorg.ipage.com/seke/seke14paper/seke14paper_150.pdf.

[69] M. Usman, E. Mendes, F. Weidt, and R. Britto, “Effort estimation in Agile Software Development: A systematic literature review,” in *ACM International Conference Proceeding Series*, 2014, pp. 82–91, doi: 10.1145/2639490.2639503.

[70] Z. C. Ani and S. Basri, “A case study of effort estimation in Agile software development using Use Case Points,” in *Agile Symposium, Malaysia*, 2013, vol. 25, no. 4, pp. 1111–1115.

[71] Abhilasha and A. Sharma, “Test effort estimation in regression testing,” in *Innovation and Technology in Education (MITE)*, 2013, pp. 343–348, doi: 10.1109/MITE.2013.6756364.

[72] A. Coelho, Evita, Basu, “Effort Estimation in Agile Software Development using Story Points,” *Int. J. Appl. Inf. Syst.*, vol. 3, no. 7, pp. 7–10, 2012.

[73] R. Litoriya and A. Kothari, “An Efficient Approach for Agile Web Based Project Estimation : AgileMOW,” *J. Softw. Eng. Appl.*, vol. 6, pp. 297–303, 2013.

[74] R. Tamrakar and M. Jørgensen, “Does the Use of Fibonacci Numbers in Planning Poker Affect Effort Estimates ?,” in *Proceedings of the EASE 2012*, 2012, pp. 228–232.

[75] E. E. Hassanein and S. A. Hassanien, “Cost Efficient Scrum Process Methodology to Improve Agile Software Development,” *Int. J. Comput. Sci. Inf. Secur.*, vol. 18, no. 4, pp. 123–131, 2020.

[76] A. K. Nazir, I. Zafar, and M. Abbas, “The impact of agile methods on software project management,” in *International Conference on Engineering, Computing & Information Technology*, 2017, pp. 1–6, doi: 10.1109/ecbs.2005.68.

[77] B. Croix, “The Role of the Agile Manifesto in Partial and Tailored Agile Manifesto in Partial and Tailored Agile Methods Adoption : A Systematic Literature Review,” 2018.

[78] P. S. Kumar, H. S. Behera, A. K. K, J. Nayak, and B. Naik, “Advancement from neural networks to deep learning in software effort estimation : Perspective of two decades,” *Comput. Sci. Rev.*, vol. 38, p. 100288, 2020, doi: 10.1016/j.cosrev.2020.100288.

[79] M. Y. Maarif, S. M. Shahar, M. F. H. Yuso, and N. S. M. Satar, “The Challenges of Implementing Agile Scrum in Information System’s Project,” *Jour Adv Res. Dyn. Control Syst.*, vol. 10, no. 9, pp. 2357–2363, 2018.

[80] F. Hayat, A. U. Rehman, K. S. Arif, K. Wahab, and M. Abbas, “The influence of Agile Methodology (Scrum) on Software Project Management,” in *20th*

IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2019, pp. 145–149, doi: 10.1109/SNPD.2019.8935813.

[81] A. E. Akgün, "Team wisdom in software development projects and its impact on project performance," *Int. J. Inf. Manage.*, vol. 50, pp. 228–243, 2020, doi: 10.1016/j.ijinfomgt.2019.05.019.

[82] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods," *J. Softw. Evol. Process*, vol. 31, no. 10, pp. 1–25, 2019, doi: 10.1002/smr.2211.

[83] A. Altaleb, M. Altherwi, and A. Gravell, "An Industrial Investigation into Effort Estimation Predictors for Mobile App Development in Agile Processes," in *9th International Conference on Industrial Technology and Management*, 2020, pp. 291–296.

[84] E. Dantas, M. Perkusich, E. Dilorenzo, D. F. S. Santos, H. Almeida, and A. Perkusich, "Effort Estimation in Agile Software Development: an Updated Review," *Softw. Eng. Knowl. Eng.*, no. June, pp. 1–7, 2018, doi: 10.18293/SEKE2018-003.

[85] M. Jørgensen and T. Halkjelsvik, "Sequence effects in the estimation of software development effort," *J. Syst. Softw.*, vol. 159, pp. 1–11, 2020, doi: 10.1016/j.jss.2019.110448.

[86] O. Malgonde and K. Chari, "An ensemble-based model for predicting agile software development effort," *Empir. Softw. Eng.*, pp. 1017–1055, 2019.

[87] and A. A. Ali Bou Nassif, Mohammad Azzeh , Ali Idri, "Software Development Effort Estimation Using Regression Fuzzy Models," *Comput. Intell. Neurosci.*, vol. 2019, no. 8367214, pp. 1–17, 2019, doi: 10.1109/9780471683179.ch11.

[88] S. Ezghari and A. Zahi, "Uncertainty management in Software effort estimation using a consistent fuzzy analogy-based method," *Appl. Soft Comput.*, vol. 67, pp. 540–557, 2018, doi: 10.1016/j.asoc.2018.03.022.

[89] C. Premalatha, Hosahalli Srikrishna, "Effort Estimation in Agile Software Development using Evolutionary CostSensitive Deep Belief Network," *Int. J. Intell. Eng. Syst.*, vol. 12, no. 2, pp. 261–269, 2019.

[90] C. Ratke, H. H. Hoffmann, T. Gaspar, and P. E. Floriani, "Effort Estimation using Bayesian Networks for Agile Development," in *2nd International Conference on Computer Applications and Information Security, ICCAIS 2019*, 2019, pp. 1–4, doi: 10.1109/CAIS.2019.8769455.

[91] G. Sharma, "Evidence-Based Software Cost Effort Estimation of Verification , Validation and Testing in Nepal," *J. Comput. Sci. Inf. Technol.*, vol. 1, no. 1, pp.

29–40, 2020.

[92] M. Jorgensen and M. Shepperd, “A Systematic Review of Software Development Cost Estimation Studies,” *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 33–53, 2007, doi: 10.1109/TSE.2007.256943.

[93] B. Singh and S. Gautam, “Situational Factors Affecting the Software Process : A Systematic Literature Situational Factors Affecting the Software Process : A Systematic Literature Review,” in *International Conference on Advanced Computing and Software Engineering (ICACSE-16)*, 2016, no. February, pp. 1–9.

[94] F. Ugalde, A. Quesada-López, Christian Martínez, and J. Marcelo, “A comparative study on measuring software functional size to support effort estimation in agile,” in *CIbSE*, 2020, pp. 1–10.

[95] L. Radu, “Effort Prediction in Agile Software Development with Bayesian Networks,” in *Proceedings ofthe 14th International Conference on Software Technologies (ICSOFT2019)*, 2019, no. Icsoft, pp. 238–245, doi: 10.5220/0007842802380245.

[96] H. Karna and S. Gotovac, “Application of data mining methods for effort estimation of software projects,” *Softw. Pract. Exp.*, no. May, pp. 1–21, 2018, doi: 10.1002/spe.2651.

[97] A. Kaur and K. Kaur, “Investigation on test effort estimation of mobile applications: Systematic literature review and survey,” *Inf. Softw. Technol.*, vol. 110, no. February, pp. 56–77, 2019, doi: 10.1016/j.infsof.2019.02.003.

[98] A. Alhaddad, I. Albaltah, A. Abualkishik, M. Abdellatief, and A. A. Al Kharusi, “A systematic mapping study on software effort estimation,” *J. theoratical Appl. Inf. Technol.*, vol. 98, no. 17, pp. 3620–3625, 2005.

[99] M. Jha and R. Jha, “Comparing the Effort Estimated By Different Models,” in *6th International Conference on Advanced Computing & Communication Systems (ICACCS)*, 2020, pp. 1148–1154.

[100] P. Efe and O. Demirors, “A change management model and its application in software development projects,” *Comput. Stand. Interfaces*, vol. 66, no. April, pp. 1–12, 2019, doi: 10.1016/j.csi.2019.04.012.

[101] W. H. de C. Almeida, “A Model using Agile Methodologies for Defining Metrics to be used by the Public Sector in Brazil to set Remuneration for Outsourced Software Development,” *Comput. Sci.*, pp. 272–274, 2021, doi: 10.1109/icse-companion52605.2021.00125.

[102] M. I. Khan, Z. U. Din, M. A. Abid, and T. Naeem, “User Story Characteristics Affecting Software Cost in Agile Software Development: A Systematic Literature Review,” *Int. J. Comput. Sci. Netw. Secur.*, vol. 19, no. 12, pp. 13–18, 2019.

[103] R. Popli and N. Chauhan, “Cost and effort estimation in agile software development,” in *International Conference on Reliability, Optimization and Information Technology*, 2014, pp. 57–61, doi: 10.1109/ICROIT.2014.6798284.

[104] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, “Systematic literature review of machine learning based software development effort estimation models,” *Inf. Softw. Technol.*, vol. 54, no. 1, pp. 41–59, 2012, doi: 10.1016/j.infsof.2011.09.002.

[105] E. Mendes, I. Watson, C. Triggs, N. Mosley, and S. Counsell, “A comparative study of cost estimation models for web hypermedia applications,” *Empir. Softw. Eng.*, vol. 8, no. 2, pp. 163–196, 2003, doi: 10.1023/A:1023062629183.

[106] L. Radlinski, “A survey of bayesian net models for software development effort prediction,” *Int. J. Softw. Eng. Comput.*, vol. 2, no. 2, pp. 95–109, 2010.

[107] M. Azzeh, A. B. Nassif, and S. Banitaan, “Comparative analysis of soft computing techniques for predicting software effort based use case points,” *IET Softw.*, vol. 12, no. 1, pp. 19–29, 2018, doi: 10.1049/iet-sen.2016.0322.

[108] Q. M. Yousef and Y. A. Alshaer, “Dragonfly Estimator : A Hybrid Software Projects’ Efforts Estimation Model using Artificial Neural Network and Dragonfly Algorithm,” *Int. J. Comput. Sci. Networ Secur.*, vol. 17, no. 9, pp. 108–120, 2017.

[109] T. Menzies, Y. Yang, G. Mathew, B. Boehm, and J. Hihn, “Negative results for software effort estimation,” *Empir. Softw. Eng.*, vol. 22, no. 5, pp. 2658–2683, 2017, doi: 10.1007/s10664-016-9472-2.

[110] A. Idri, M. Hosni, and A. Abran, “Systematic Literature Review of Ensemble Effort Estimation,” *J. Syst. Softw.*, vol. 1, pp. 1–35, 2016, doi: 10.1016/j.jss.2016.05.016.

[111] M. Padmaja and D. Haritha, “Software Effort Estimation using Meta Heuristic Algorithm,” *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 5, pp. 196–201, 2017, [Online]. Available: https://search.proquest.com/openview/dc9da6e8d9857b23c8c88c47eba7c3ed/1?pq-origsite=gscholar&cbl=1606379.

[112] J. Murillo-Morera, C. Quesada-López, C. Castro-Herrera, and M. Jenkins, “A genetic algorithm based framework for software effort prediction,” *J. Softw. Eng. Res. Dev.*, vol. 5, no. 1, pp. 1–33, 2017, doi: 10.1186/s40411-017-0037-x.

[113] Tung Khuat and Hanh Le, “An Effort Estimation Approach for Agile Software Development using Fireworks Algorithm Optimized Neural Network,” *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 7, pp. 122–130, 2018, doi: 10.1162/neco.2008.20.1.65.

[114] M. Adnan and M. Afzal, “Ontology Based Multiagent Effort Estimation System for Scrum Agile Method,” *IEEE Access*, vol. 5, pp. 25993–26005, 2017.

[115] T. T. Khuat and M. H. Le, "A Novel Hybrid ABC-PSO Algorithm for Effort Estimation of Software Projects Using Agile Methodologies," *J. Intell. Syst.*, vol. 27, no. 3, pp. 489–506, 2017, doi: 10.1515/jisys-2016-0294.

[116] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, "Estimating Story Points from Issue Reports," in *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, 2016, pp. 1–10, doi: 10.1145/2972958.2972959.

[117] A. Panda, S. M. Satapathy, and S. K. Rath, "Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points," *Procedia Comput. Sci.*, vol. 57, pp. 772–781, 2015, doi: 10.1016/j.procs.2015.07.474.

[118] S. M. Satapathy, "Effort Estimation Methods in Software Development Department of Computer Science and Engineering National Institute of Technology Rourkela Effort Estimation Methods in Software Development using," National Institute of Technology Rourkela, 2016.

[119] K. Anjali Sharma, "Empirical Validation of Random Forest for Agile Software," *Int. J. Eng. Sci. Res. Technol.*, vol. 5, no. 7, pp. 1437–1446, 2016.

[120] M. Choetkiertikul, H. K. Dam, T. Tran, T. T. M. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," *IEEE Trans. Softw. Eng.*, vol. 14, no. 8, pp. 1–12, 2018, doi: 10.1109/TSE.2018.2792473.

[121] M. Adnan and M. Afzal, "Ontology based Multiagent Effort Estimation System for Scrum Agile Method," *IEEE Access*, vol. 5, pp. 25993–26005, 2017, doi: 10.1109/ACCESS.2017.2771257.

[122] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *J. Artif. Intell. Res.*, vol. 16, no. Sept. 28, pp. 321–357, 2002, doi: 10.1613/jair.953.

[123] F. Last, G. Douzas, and F. Bacao, "Oversampling for Imbalanced Learning Based on K-Means and SMOTE," *Inf. Sci. (Ny).*, vol. 465, pp. 1–19, 2018, doi: 10.1016/j.ins.2018.06.056.

[124] J. L. Hodges and E. L. Lehmann, "Rank Methods for Combination of Independent Experiments in Analysis of Variance," *Ann. Math. Stat.*, vol. 33, no. 2, pp. 482–497, 1962, doi: 10.1214/aoms/1177704575.

LIST OF PUBLICATIONS

1. Arora M., Verma S., Kavita, Chopra S. (2020) **A Systematic Literature Review of Machine Learning Estimation Approaches in Scrum Projects**. In: Mallick P., Balas V., Bhoi A., Chae GS. (eds) Cognitive Informatics and Soft Computing. Advances in Intelligent Systems and Computing, vol 1040. Springer, Singapore. https://doi.org/10.1007/978-981-15-1451-7_59.
2. M. Arora, S. Verma, and Kavita, "**An efficient effort and cost estimation framework for Scrum Based Projects**," International Journal of Engineering and Technology (UAE), vol. 7, no. 4.12 Special Issue 12, pp. 52–57, 2018
3. Arora, M., Chopra, S., Gupta, P.: **Estimation of regression test effort in Agile projects**. Far East J. Electron. Commun. 3(II), 741–753, 2016