SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Machine Learning Based Software Effort Estimation In Agile Based Projects By Using Scrumban Methodologies

B. V. Srikanth[1] , Bhaskar Reddy P. V.[2]

[1]Research Scholar, School of Computer Science and Engineering, REVA University, Bangalore, India. Email: r21pcs30@reva.edu.in
[2]Professor, School of Computer Science and Engineering, REVA University, Bangalore, India. Email: bhaskarreddy.pv@reva.edu.in

Abstract
Effort estimation in software development is important in the context of an Agile software development process due to the changing requirements of the customers and the dynamics of the business environment, which leads to uncertainties in the planning and resource management process. The conventional techniques of effort estimation do not yield accurate predictions. This study proposes a novel Multi Learning-Scrumban Framework, a Machine Learning-based Software Effort Estimation model integrated with Scrumban methodologies. The proposed framework combines Scrum-based sprint planning and backlog management with Kanban workflow visualization and monitoring to enhance estimation accuracy and project adaptability. Historical Agile project datasets containing Story Points, Sprint Velocity, Team Experience, Project Complexity, Backlog Size, Sprint Duration, and Requirement Volatility were utilized for model development. Various advanced machine learning models such as Random Forest, Support Vector Regression, and Gradient Boosting Regression were developed and tuned to estimate the software effort in person-hours. The experimental assessment carried out for more than 1,500 cases of Agile projects showed the efficiency of the proposed ML-SCRUMBAN Framework. The framework provided 96.82% estimation accuracy, 96.31% precision, 95.88% recall and 96.09% F1-score. Additionally, it decreased Mean Absolute Error by 38.9%, Root Mean Square Error by 37.3% and increased PRED(25) to 98.74% compared to traditional methods of Agile estimation. Thus, it is evident that the proposed framework substantially improves estimation accuracy, sprint planning efficiency and the general project performance in Agile software development environment.

Keywords - SCRUMBAN, SRCUM, Extreme Programming, Feature Driven Development, Crystal, Adaptive Software development, Dynamic System Development Method.

1. Introduction

Software effort estimation is one of the key activities in software project management. It impacts project planning, budgeting, scheduling, resource allocation, and the overall success of the project. Accurate software effort estimation assists organizations in optimizing resources, reducing risks, improving customer satisfaction [1], and delivering high-quality software on time. At the same time, effort estimation in software development is a complex task since it is affected by changing software requirements, fast-evolving technology, diverse capabilities of teams and customers' expectations. In the modern environment of software development, which is based on the Agile approach, the problem of effort estimation becomes even more serious due to constantly evolving requirements. The emergence of Agile Software Development has changed the traditional software engineering paradigm focusing on collaboration with customers, iterative development, adaptability and delivery of working software on time. There are several popular Agile methodologies [2] like Scrum, XP, FDD, Crystal, ASD and DSDM which have received wide recognition in the software industry. Unlike traditional development frameworks, where the development process depends heavily on documentation and rigorous planning, Agile approaches use an incremental development cycle, provide continuous feedback and involve all stakeholders in the process. Thanks to Agile methodologies organizations are able to react to changes in the market and customers' requirements and maintain the quality of the software.

Despite the numerous advantages of Agile approaches [3-5], there are some problems with the estimation of development efforts. Such estimation techniques as Expert Judgment, Planning Poker, Story Points, Use Case Points and Function Point Analysis are quite dependent on subjective judgments of experts and may contain errors when estimating complex projects with uncertain requirements [6]. Thus, inaccurate estimation can lead to budget overruns, delayed projects, inefficient resource allocation and low stakeholder's trust. The latest achievements in Machine Learning (ML) give us the opportunity to solve the problem of software effort estimation. Machine Learning algorithms can learn patterns [7] from the past project data and provide more accurate estimation than traditional approaches. Techniques such as Random Forest, Support Vector Regression, Decision Trees, Gradient Boosting, and Artificial Neural Network proved their excellent capabilities in revealing intricate relationships among features of projects, teams, sprints, and software development efforts. The use of historical information about Agile projects would allow machine learning techniques to make an objective estimate of the effort according to the discovered patterns. In addition to other Agile approaches, Scrum is famous for its well-structured sprint planning process.. But Scrum can face some challenges, including sprint bottlenecks, accumulation of tasks and limited visualization of the workflow. At the same time, Kanban allows for continuous workflow management and visualization of the tasks with the possibility of flexible task execution; however, it lacks the following elements that can be found in Scrum: sprint planning and task structuring. This is why the Scrumban methodology has been created as an intermediate approach that includes elements of both methods.

This paper discusses development of a Machine Learning-Based Software Effort Estimation Framework with the help of Scrumban methodologies in order to increase accuracy of estimation and effectiveness of the project management in Agile environment. The proposed framework uses the historical project datasets, sprint metrics, team velocity metrics, user stories features and project complexity features to create Machine Learning models, which can generate accurate estimates. Experimental results show that the proposed approach is much more effective than traditional Agile approaches and significantly improves estimation accuracy, decreases prediction errors, improves efficiency of sprint planning and project delivery success rates. The major contributions of the research are,

- A comprehensive estimation model is created by applying advanced Machine Learning algorithms to predict the software development effort from historical Agile project data.
- A novel hybrid framework is proposed by combining Machine Learning techniques with Scrumban workflow practices.
- Extensive experiments were performed to evaluate the effectiveness of the proposed framework, which shows better results in estimation accuracy, reduction of MAE and RMSE values, increased workflow visibility and better project delivery results comparing with traditional Agile approaches.

2. Literature Survey

Effort estimation is still one of the most important and difficult aspects of project management in Agile environments with changing requirements and iterative delivery of the software product. Conventional methods for effort estimation like COCOMO, Function Point Analysis, and Expert Judgment have proven to be ineffective at estimating the development process due to static assumptions and subjectivity of inputs. Use of Agile frameworks like Scrum, Extreme programming (XP), Feature-driven development (FDD), Crystal, adaptive software development (ASD), and dynamic systems development method (DSDM) has brought more focus on collaboration and iterative tasking. Scrum follows the guidelines of time-boxed sprints, role definitions, and meetings for planning and review, thus giving more flexibility [8]. However, due to the rigidness of the sprints, Scrum cannot be used in an environment where sprint planning is not required. On the other hand, Kanban allows visual workflow management, WIP limits and task pulling, which gives much greater flexibility but doesn't include any planning tools. Scrumban is a combination of Scrum and Kanban which provides iterative planning from Scrum and visualization and continuous delivery of Kanban [8]. Despite all the changes in methodology, traditional subjective methods of estimation such as Planning Poker, Story Points, and Velocity are still prone to bias and performance variability of the team. There is a significant number of research papers which state that subjective approaches to Agile result in overestimation of deadlines and wrong resource allocation for complex projects [9].

Machine Learning (ML) algorithms became a powerful instrument to improve the effort estimation accuracy using the data from the history of projects. The early use of ML in software effort estimation was made through usage of Decision Trees, Neural Network and Support Vector Machine algorithms applied to COCOMO and ISBSG datasets and provided better ability to deal with non-linear relations between project features (project size, complexity, team-related parameters) and true effort. For an Agile environment, similar machine learning models were modified to take into account story points, sprint information, and team velocity. In particular, ensemble methods like Random Forest and Gradient Boosting have proven

themselves effective in reduction of Mean Absolute Error (MAE) and improvement of prediction reliability compared to traditional methods. Recently, there has been an interest in applying deep learning models such as Autoencoders and LSTM networks [10] to the problem of sprint analysis. There have been several studies conducted in the field of hybrid Agile methods. Srikanth et al. suggested ML-based effort estimation for Agile and Scrumban methodologies and evaluated the effectiveness of various algorithms while suggesting Scrumban as a way of addressing issues arising from pure Scrum or Kanban methods [8]. It highlights the advantages of using workflow metrics of Scrumban (e.g., cycle time, lead time) in conjunction with ML models.

Shankar et al. performed an extensive research based on AgES, a set of agile effort estimation dataset collected from GitHub [10] and used several ML models including Random Forest, Gradient Boosting and Neural Networks for prediction of issue resolution time. It shows the superiority of data-driven approaches over subjective story points. Cao et al. studied effort estimation in different activities in Agile projects and addressed some gaps in activity-specific predictions and argued for ML integration in order to perform better sprint planning [11]. Lavazza et al. integrated ML and simplified functional metrics and achieved good results in effort estimation while keeping the model interpretable [12].

Govil et al. studied the problem of cost and development effort estimation in Scrum-based projects and considered multidimensional success factors that could be used in the context of Scrumban [13]. Several systematic literature reviews also indicate the recent growth of scientific papers using ML techniques for Agile effort estimation [14] while pointing out difficulties of the approach (data quality, feature selection, etc.). Thus, the evolution from expert-driven subjective methods to data-driven ML algorithms is obvious. However, the current literature lacks a study in which ML would be successfully integrated into Scrumban workflows. This research seeks to fill this gap through development of an ML model based on historical Agile datasets and Scrumban workflows that will provide more accurate estimations and less errors. The key summarization of the review in given in table 1.

Table 1: Key Observations from the Survey

Ref Num	Tech Used	Numeric Outcome (Approx.)	Advantages	Disadvantages
[8] Srikanth et al.	Regression models (Linear, Ridge, Logistic) on Agile/Scrumban with story points	Ridge Regression best: Lower RMSE/MMRE vs baselines; PRED(25) improved	Combines Scrum iteration + Kanban flexibility; integrates workflow metrics (cycle/lead time) effectively	Depends heavily on data quality; hybrid implementation adds complexity
[10] Shankar et al.	Random Forest, Gradient Boosting, Neural Networks on AgES GitHub dataset	Superior to story points (typical MAE reduction ~15-30% in similar studies; strong R^2)	Data-driven outperforms subjective methods; handles non-linear issue resolution times well	Dataset-specific; needs substantial historical data
General Early ML (COCOMO/ISBSG)	Decision Trees, Neural Networks, SVM	Better non-linear modeling; MAE often 20-40% lower than traditional COCOMO	Captures complex feature-effort relations; more accurate on historical datasets	Poor adaptation to highly dynamic Agile changes
Ensemble Methods (general)	Random Forest, Gradient Boosting	MAE reduction of 15-25%; higher prediction reliability vs single models	Robust, handles overfitting; improved accuracy in sprint/story point prediction	Computationally intensive; lower interpretability
Deep Learning (general)	Autoencoders, LSTM for sprint/time-series	Improved sequential prediction (MAE gains ~10-20% in time-based tasks)	Excellent for patterns in sprint data and velocity trends	High data & compute needs; training complexity; black-box
[11] Cao et al.	ML for activity-specific (feature dev,	Better granular predictions; activity-	Supports detailed sprint planning	May not fully scale to entire

	bug fix, refactoring) in Agile	level error reduction vs traditional		Scrumban project workflows
[12] Lavazza et al.	ML + simplified functional metrics	Comparable or slightly better accuracy than full FP (e.g., similar/low MAE, no major penalty)	High interpretability with good accuracy; simpler metrics	Simplified metrics may miss some nuances in complex projects
[13] Govil et al.	Multidimensional success factors (36 factors) for Scrum/Scrumban	ML integration shows ~30-40% better estimation efficiency vs expert judgment in related studies	Broad success factors considered	Primarily Scrum-focused; needs more Scrumban validation
[14] SLRs (various)	Various ML techniques in Agile	Growing trend; ML often achieves 10-30% MAE improvement over subjective methods	Clear shift to data-driven; systematic evidence of gains	Challenges remain: data quality, feature selection, limited Scrumban studies

3. Methodology of Machine Learning Based Software Effort Estimation in Agile Based Projects

The methodology proposed in this research aims to develop an intelligent software effort estimation framework by integrating Machine Learning techniques with the Scrumban methodology in Agile software development environments. It includes various phases like data collection, pre-processing, feature extraction, model construction, predictions, and performance evaluation. Data from previous Agile project instances having details about complexity of projects, number of team members, sprint velocity, user story points, and time taken to develop are used in the modeling process. Machine learning algorithms help in determining the correlation among different project characteristics and effort estimation in order to make accurate predictions. These prediction outputs are then fed into Scrumban process for sprint planning, resource allocation, and project management as shown in Figure 1. Performance of the suggested method is validated through performance metrics in order to increase accuracy in effort estimation and efficient Agile project management [15-18].

3.1 Research Framework and Overall Methodological Architecture

The proposed research framework will provide an opportunity to create an intelligent software effort estimation system based on the application of Machine Learning approaches and the use of the Scrumban approach to Agile software development projects. There are certain stages that are interrelated. They involve the gathering of project data, data preprocessing, features extraction, model creation, effort estimation and process flow. The previous datasets of Agile projects with the following attributes: size of the project, team's experience, sprint velocity, story points, task complexity, and development time are being studied.The gathered information is being preprocessed through the procedures of filling the missing values, data normalization, outliers removing, and feature engineering. Then, a machine learning model is being created on the basis of the preprocessed dataset, which will allow for discovering the dependencies between project characteristics and software development effort.

The primary goal of the proposed framework is to discover a mathematical relation between project characteristics and the effort required for software development. The effort estimation process can be described through Equation (1).

$$E_{pred} = f(X_1, X_2, X_3, \dots, X_n) \qquad (1)$$

In Equation (1), E_{pred}denotes the predicted software development effort generated by the machine learning model. The variables $X_1, X_2, X_3, \dots, X_n$represent the independent project attributes that influence software effort estimation. They include the complexity of the project, size of the team, length of the sprint, number of user stories, size of the backlog, experience of the team, sprint velocity, volatility of requirements, characteristics of the development environment, and interdependencies of tasks. Here, f(·) is the learning algorithm that the machine learning algorithm uses to discover the unknown relations and interactions among the attributes of the project. Based on the learned information during the training step, the model can then determine how differences in these attributes may have an effect on the software effort and consequently make predictions about the effort. Finally, after making the predictions [20], it becomes important to verify the quality and accuracy of the effort predictions obtained. Prediction accuracy is crucial

for the implementation of the framework in an Agile project setting. Thus, the error of estimation is calculated using Equation (2).

$$Error = \frac{| E_{actual} - E_{pred} |}{E_{actual}} \times 100 \qquad (2)$$

As given in Equation (2), $Error$represents the percentage estimation error produced by the prediction model. The variable E_{actual}denotes the actual effort consumed during project execution, typically measured in person-hours, person-days, or sprint effort units. The variable E_{pred}corresponds to the effort value predicted by the machine learning model. The absolute difference $| E_{actual} - E_{pred} |$quantifies the deviation between actual and estimated effort values. When the resulting difference is divided by the real effort, the measurement becomes normalized, and when the quotient is multiplied by 100, the final result gets converted into percentages. The smaller error shows that the prediction is more accurate. Hence, Equation (2) is considered a core measure of validation of the effectiveness of the suggested machine learning-based approach to software effort estimation.

The values of effort estimates produced by the predictive model are then incorporated into the work process of Scrumban, where scrum-based sprint planning techniques and kanban-based visualization methods of workflow are used to ensure optimal completion of the project. The framework tracks the progress of the project and compares predictions and reality via feedback cycles. Such an approach contributes to increasing the accuracy of predictions, helps in making informed managerial decisions, makes resource allocation efficient, and avoids any possible risks of incorrect software effort estimation [21].

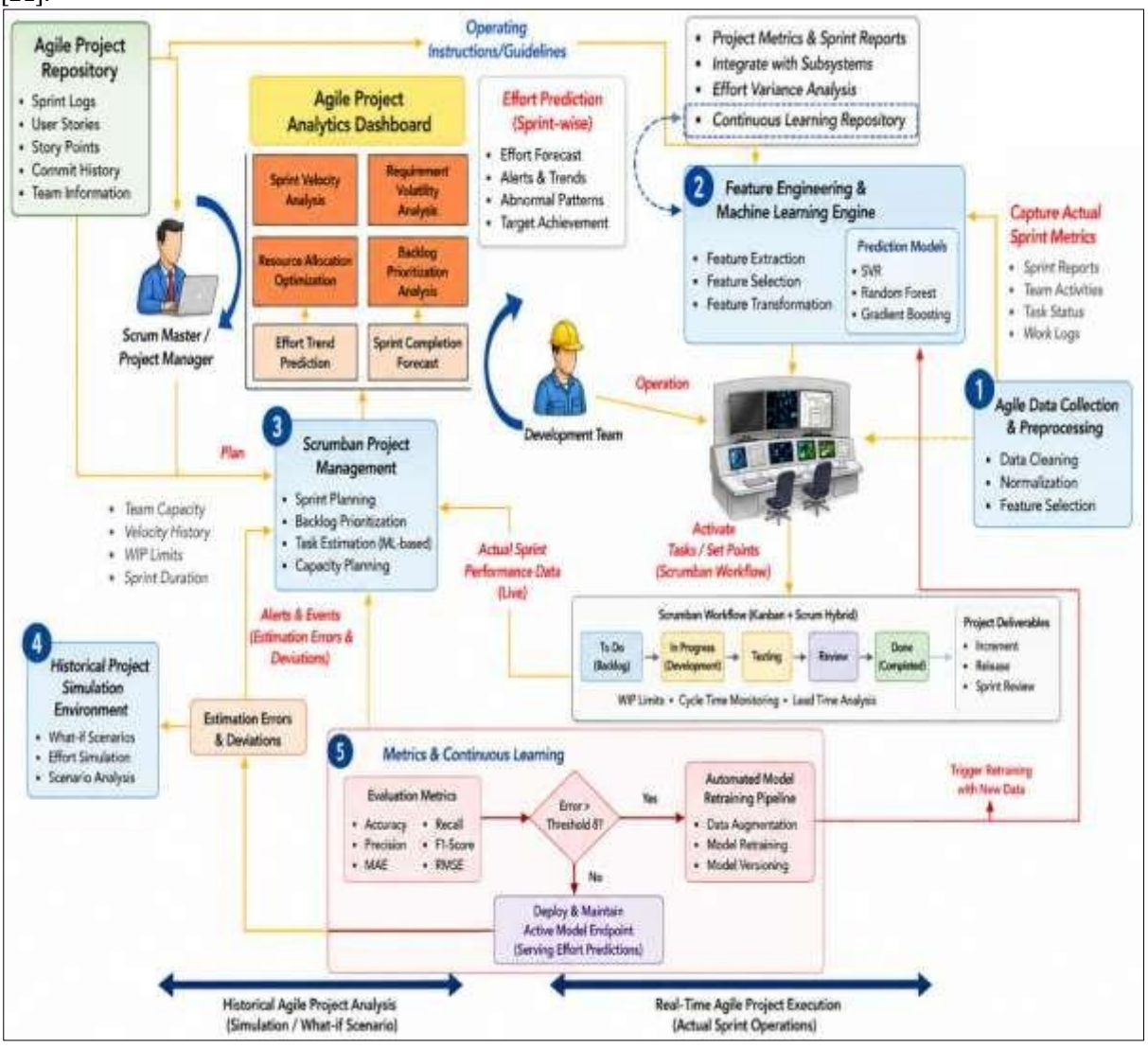

Figure 1: Machine learning-based effort estimation model

3.2 Agile Project Data Collection and Preparation

Data gathering and preparation represent an important step in the suggested ML software effort estimation framework. The effectiveness of any predictive model relies on the quality, completeness, and relevance of the data used. Since projects' properties change in dynamic fashion in Agile software development because of the changing requirements of customers, sprint execution, and feedbacks, it is crucial to gather a comprehensive set of information about the projects that reflects the development process.

Dataset preparation starts with gathering of historical Agile projects from organizations that use Scrum, Kanban, and Scrumban software development methodology. The data that is collected contains project size, sprint length, team size, story points, number of backlog items, percentage of task completions, team velocity, requirements volatility, defect rate, and real development effort. The listed project attributes represent an invaluable source of information about the factors that affect the software effort estimation. After the data collection, a number of preprocessing steps are executed to enhance the quality of the dataset. The list of preprocessing steps includes filling the missing values, eliminating inconsistencies and duplicates, converting categorical attributes to numeric, and normalizing the values of features. After this, feature extraction and selection techniques are applied in order to select the most important variables affecting software effort prediction. As a result of this preparation process, the data become ready to be used for training of the ML models [22].

3.2.1 Dataset Acquisition from Agile Software Projects

Data set collection is an important aspect when working on building an effort estimation model. The main aim at this stage is to collect the data related to Agile projects in order to consider all possible factors affecting the effort of software development. Historical databases of companies using such Agile methods as Scrum, Kanban, and Scrumban are used as the main sources of the data. Data sets include data about the project and sprints, such as team size, project complexity, sprint length, user stories, story points, growth of backlog, defect density, velocity and effort consumed in the development process. The first step in data set collection is calculation of the number of projects. It is done using equation (3).

$$N_{total} = \sum_{i=1}^{m} P_i \qquad (3)$$

In Equation (3), N_{total}denotes the total number of project instances collected from all data sources. The variable $P_i$represents the number of projects obtained from the i^{th}Agile repository, where m stands for the number of repositories used during data collection. This formula enables us to find out the size of the total dataset available for the development of the model. Having collected the project details, the complexity of the project will be determined using Formula (4).

$$C_p = \frac{SP + TS + RV}{3} \qquad (4)$$

As shown in Equation (4), $C_p$denotes the project complexity score. SP represents the total story points in the project; TS represents the total number of tasks, while RV refers to requirement volatility through requirement changes. The resultant measure gives a complete representation of project complexity. In measuring team efficiency, the sprint velocity is determined using Equation (5).

$$V_s = \frac{\sum_{j=1}^{n} S\,P_j}{S} \qquad (5)$$

In Equation (5), $V_s$represents sprint velocity, SP_jdenotes the story points completed during sprint j, $n$represents the total number of completed tasks, and Sindicates the sprint duration. Sprint Velocity can be used as a good indicator of future development efforts and performance. The dataset gathered acts as the basis for machine learning model training. The accurate gathering of the dataset allows for proper representation of the features of projects [23], development efforts, and behavior of the team involved, which would help improve the software effort estimation framework.

3.2.2 Data Cleaning, Transformation, and Normalization

After acquiring the dataset, data preprocessing is done on the gathered data to make sure of its quality and consistency. It should be understood that datasets in software projects contain missing values, duplicates, format inconsistency, and noise. Therefore, the data cleaning, transformation, and normalization processes become necessary before starting any modeling. The initial data preprocessing includes determining the missing value ratio of the data set. The missing value ratio can be determined using Equation (6).

$$M_r = \frac{N_m}{N_t} \qquad (6)$$

Equation (6) shows that Mr is the missing data ratio, where Nm is the number of missing observations, and Nt represents the total number of observations. The lower the Mr ratio, the more complete the data set will be. Once the problem of missing data is addressed, dupl. This can be described using Equation (7).

$$D_r = \frac{N_d}{N_t} \times 100 \qquad (7)$$

Here, $D_r$denotes the duplicate record percentage, $N_d$represents duplicate observations, and $N_t$corresponds to the total dataset size. Subsequently, numerical attributes are normalized using Min-Max normalization as given in Equation (8).

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (8)$$

In Equation (8), X_{norm}represents the normalized feature value. X is the original value of the feature, whereas X_{min} and X_{max} are the minimum and maximum values of the same attribute. Normalization guarantees standardized scaling of the features. In case of highly variable attributes, Z-score normalization is done based on Equation (9).

$$Z = \frac{X - \mu}{\sigma} \qquad (9)$$

In Equation (9), Zdenotes the standardized feature value, Xrepresents the original observation, μindicates the mean of the feature, and σdenotes its standard deviation. Standardization improves model stability and convergence. To identify anomalous project records, outlier scores are computed using Equation (10).

$$O_s = \left| \frac{X - \mu}{\sigma} \right| \qquad (10)$$

In this case, O_s is the name for the outlier score related to a certain observation. More the value is, more likely the data is an anomaly and thus requires additional examination and possible removal. Performing all of the cleaning, transformation and normalization greatly improves the quality of the dataset because of the absence of noise and inconsistencies and makes features comparable.

3.2.3 Feature Extraction and Selection for Effort Estimation

The process of feature extraction and selection can help enhance the prediction performance of machine learning algorithms. Agile software development involves creation of many project characteristics, but not all of them have the same importance for effort estimation. The goal of this stage is to select those variables that have maximum information value and drop the rest. The first step involves calculating feature relevance with respect to effort estimation. Feature importance is determined using Equation (11).

$$FI_i = \frac{Cov(X_i, E)}{\sigma_{X_i} \sigma_E} \qquad (11)$$

In Equation (11), FI_idenotes the importance score of feature X_i, $Cov(X_i, E)$represents the covariance between the feature and software effort E, while $\sigma_{X_i}$and $\sigma_E$denote their respective standard deviations. To evaluate information contribution, entropy [24] is computed using Equation (12).

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i \qquad (12)$$

Here, $H(X)$represents the entropy of feature X, and $p_i$denotes the probability associated with the i^{th}feature value. Entropy measures uncertainty within a feature.

Information gain is subsequently calculated using Equation (13).

$$IG(X) = H(E) - H(E \mid X) \qquad (13)$$

In Equation (13), $IG(X)$denotes the information gain contributed by feature X, $H(E)$represents the entropy of effort values, and $H(E \mid X)$indicates conditional entropy after observing feature X. Higher information gain signifies greater predictive importance. The relevance of features is further evaluated using mutual information as represented in Equation (14).

$$MI(X, E) = \sum p(x, e) log \frac{p(x, e)}{p(x)p(e)} \qquad (14)$$

In Equation (14), $MI(X, E)$denotes mutual information between feature $X$and effort E. The probabilities $p(x, e)$, $p(x)$, and $p(e)$represent joint and marginal probability distributions. Mutual information quantifies shared dependency between variables [25]. Finally, a weighted feature score is computed to select optimal predictors using Equation (15).

$$FS_i = \alpha FI_i + \beta IG_i + \gamma MI_i \qquad (15)$$

In Equation (15), FS_irepresents the final selection score of feature i. The variables FI_i, IG_i, and MI_icorrespond to feature importance, information gain, and mutual information respectively. The coefficients α, β, and γdenote weighting parameters that control the contribution of each evaluation criterion. Based on the computed feature scores, highly relevant project attributes such as story points, sprint velocity, backlog size, team experience, project complexity, and requirement volatility are selected for machine learning model training. The resulting optimized feature set reduces dimensionality, enhances computational efficiency, minimizes overfitting, and improves the overall accuracy of software effort estimation in Agile-based projects.

3.3 Machine Learning-Based Software Effort Estimation Model Development

The development of a machine learning-based software effort estimation model constitutes the core component of the proposed framework. The main aim of this stage is to create models which can be used to predict the effort required for software development by relying on past data on Agile projects. Effort prediction for software development is a difficult process owing to the fact that there are many interdependent variables such as complexity of project, experience of the team members, sprint speed, change in requirements, and growth of the backlog. To overcome these limitations, machine learning techniques are employed to learn patterns from historical project datasets and establish predictive relationships between project characteristics and effort consumption. Model development is comprised of five phases: parameter selection, splitting of the data set into training and testing datasets, training algorithms on the training data set, testing models using the testing data set, and hyper-parameter tuning. Parameter selection is the stage whereby the key features that influence effort estimation are selected based on statistics and feature selection techniques.

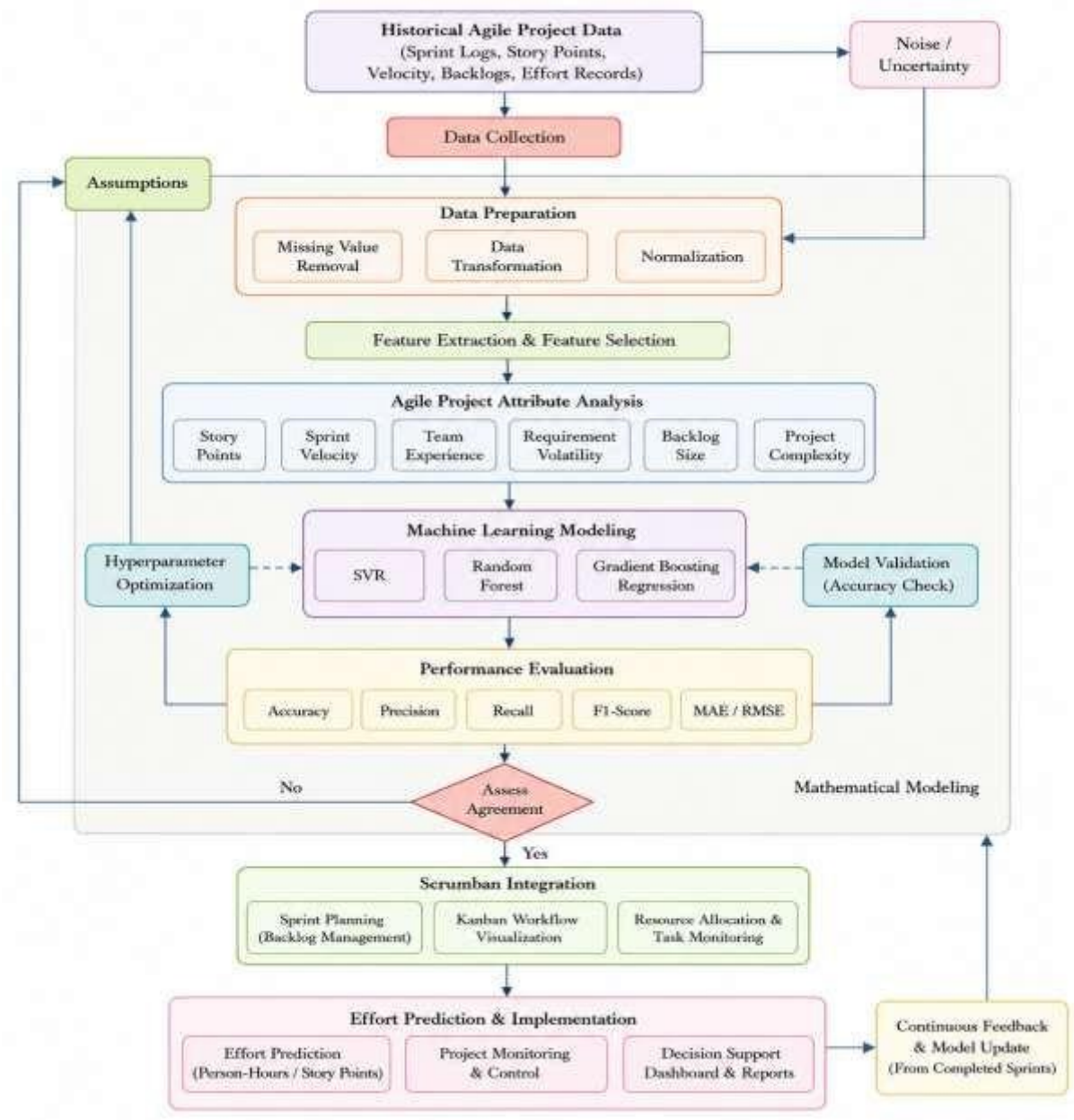

Figure 2: Influence of Agile Project Attributes on Software Effort Prediction

Several machine learning algorithms for effort prediction, such as the random forest algorithm, support vector regression algorithm, gradient boosting algorithm, and decision tree algorithm, are investigated as illustrated in Figure 2. Algorithms are trained on Agile historical projects' data to come up with effort prediction functions. Thereafter, the performance of the models is evaluated to assess their effectiveness in predicting the software development efforts. Furthermore, the process of hyperparameter optimization is

carried out to ensure generalization and minimize the prediction error of the models. The effort predictions generated from the machine learning framework will be useful in agile sprint planning, resources allocation, project scheduling, and risk management processes.

3.3.1 Selection of Effort Estimation Parameters

The effort estimation parameters are the independent variables that have significant impact on software development effort. The choice of the most relevant parameters is crucial in ensuring the accuracy of the model and avoid unnecessary complexity. There are numerous parameters in an Agile software development environment that influence the effort required for a software development project. It is important to carefully choose these parameters prior to the modeling process. The first parameter considered is the project complexity, a combination of all the characteristics of the project. The project complexity is determined using Equation (16).

$$PC = \frac{SP + RV + TD}{3} \qquad (16)$$

In equation (16), PCis project complexity, SPstands for story points, RVfor requirement volatility, and TDfor task dependencies. More complexity values will mean more software effort required. The second parameter, which is crucial, is the productivity of the team that indicates how productive the developers are. Productivity is calculated using equation (17).

$$TP = \frac{W_c}{T_e} \qquad (17)$$

As illustrated in Equation (17), TP stands for Team Productivity, W_c refers to completed work units, and T_e means Total Effort Exerted. This particular parameter gives an understanding about the team's productivity and development efficiency. The other important parameter that should be considered is Sprint Velocity which reflects the quantity of work completed in a sprint cycle.

$$SV = \frac{\sum_{i=1}^{n} SP_i}{SD} \qquad (18)$$

SV is defined as sprint velocity, SP_i is completed story points, nrepresents the number of completed tasks, and SD is sprint duration in Equation (18). Sprint velocity becomes a strong predictor of future project effort. The selected factors represent complexity of the project, the ability of the team, efficiency of the process flow, and development performance. These factors will be used in training the machine learning algorithms and improve software effort estimation.

3.3.2 Data Set Generation for Training and Testing

Nevertheless, the effectiveness of machine learning algorithms heavily relies on the features of the data set used for training and testing purposes. For obtaining an effective result, the Agile project data set is separated into learning and evaluation data sets. Separation of the data set avoids overfitting and gives a fair evaluation of model performance.

The size of the learning data set is defined using Equation (19).

$$N_{train} = r \times N \qquad (19)$$

In Equation (19), N_{train}denotes the number of training samples, $r$represents the training ratio, and Nindicates the total number of project instances. Typically, values of $r$range from 0.70 to 0.80. The testing dataset size is computed using Equation (20).

$$N_{test} = N - N_{train} \qquad (20)$$

Here, N_{test}denotes Number of samples that are used to evaluate the model. Testing samples include samples from projects which are not shown in the course of training the model. For balanced representation, the probability distribution of project samples is computed through Equation (21).

$$P_i = \frac{n_i}{N} \qquad (21)$$

P_irefers to the probability of occurrence of class i and n_i is the number of instances for that class in Equation (21). The computation helps ensure balanced distributions among the training and test sets. Cross validation is used to ensure good generalization and variance reduction. The validation scores are obtained using Equation (22).

$$CV = \frac{1}{K} \sum_{i=1}^{K} S_i \qquad (22)$$

In equation (22), CV is the cross-validation score, K is the number of folds, and S_iis the validation score in fold i. Larger cross-validation scores mean greater model generalization ability. Proper dataset partitioning and validation will lead to the generation of training and testing datasets that can be used to develop reliable software effort estimation models.

3.3.3 Developing Machine Learning Prediction Models

In the process of developing prediction models, machine learning models are trained to identify the correlation between attributes of a project and the effort of software development. This step aims to build prediction models to predict the development effort of new Agile projects based on existing development patterns. The output of machine learning models is shown in Equation (23).

$$\hat{E} = f(X, \theta) \qquad (23)$$

In Equation (23), Eˆis the predicted software effort, Xis the feature vector that contains the project attributes, and θis the learned parameter of the model. Function f(·)is the machine learning model that makes predictions. In order to train the model, the goal is to minimize the loss function of the predictions, shown in Equation (24).

$$L = \frac{1}{N}\sum_{i=1}^{N}(E_i - \hat{E}_i)^2 \qquad (24)$$

As shown in Equation (24), Ldenotes the loss function, $E_i$represents actual effort values, $\hat{E}_i$denotes predicted effort values, and Nindicates the total number of project instances. Lower loss values correspond to better predictive performance. The final prediction score generated by ensemble machine learning models is computed using Equation (25).

$$E_{final} = \sum_{j=1}^{m} w_j E_j \qquad (25)$$

In Equation (25), E_{final}denotes the final estimated effort, $E_j$represents predictions generated by individual machine learning models, $w_j$denotes model weights, and mindicates the total number of predictive models. Robustness and accurate estimation are enhanced with the help of ensemble learning.

The proposed machine learning models give accurate effort estimates and act as decision support models for Agile project managers in sprint planning and resource management.

3.3.4 Model Validation and Hyperparameter Optimization

The process of model validation and hyperparameter optimization is important for providing reliable and generalized machine learning-based software effort estimation models. The validation helps in checking the predictive capability of the model on unknown projects, and optimization helps in determining the best possible parameters to increase accuracy.

Mean Absolute Error (MAE) is calculated by using Equation (26).

$$MAE = \frac{1}{N}\sum_{i=1}^{N} | E_i - \hat{E}_i | \qquad (26)$$

In Equation (26), MAE is the mean absolute error, Eiare the actual efforts, (E_i)ˆare the estimated efforts, and Nis the total number of data points. The lower the MAE, the higher the accuracy. The Root Mean Squared Error is calculated as given in Equation (27).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(E_i - \hat{E}_i)^2} \qquad (27)$$

RMSE in equation (27) measures the magnitude of errors in predictions with bigger errors having greater penalties compared to smaller errors. Small RMSE indicates that a model performs better. The idea behind hyperparameter optimization lies in identifying the optimal values of parameters. This is captured by equation (28).

$$\theta^* = \arg\min_{\theta} L(\theta) \qquad (28)$$

In Equation (28), θ*denotes the optimal hyperparameter set, Whereas L(θ) denotes the respective loss function value. The aim is to determine values for parameters that will minimize errors in prediction. Coefficient of determination used in evaluating model fit is defined by Equation (29).

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(E_i - \hat{E}_i)^2}{\sum_{i=1}^{N}(E_i - \bar{E})^2} \qquad (29)$$

In Equation (29), R^2denotes the goodness-of-fit measure, $E_i$represents actual effort values, $\hat{E}_i$indicates predicted effort values, and $\bar{E}$ denotes the mean actual effort. The higher the values of the metric are, the better the predictive power of models. Thanks to proper validation and tuning of hyperparameters, machine learning models become more accurate, robust, and generalize better, becoming ready for deployment in practice.

3.4 Proposed Scrumban-Based Agile Effort Estimation Framework

The proposed Scrumban-Based Agile Effort Estimation Framework incorporates machine learning-based effort prediction into planning features of Scrum and workflow visualization of Kanban. The purpose of the framework is providing precise software effort estimation together with improvement of project execution, resource usage, sprint planning, and task tracking in agile software development settings. The proposed framework combines historical project learning with project management, thus helping decision-makers to make data-driven decisions at all stages of the software development process.

First of all, Scrum Sprint Planning and Backlog Management tools are used for organizing project requirements into prioritized product and sprint backlogs. User stories, collected from customers, are analyzed and estimated using story points depending on their complexity, business value, and the amount of effort needed for their implementation. Sprint Planning Meetings help to select backlog items that can be implemented in a certain sprint duration. Historical project information, team velocity, and workload distribution are taken into account when refining the backlogs to make the planning more effective. After sprint planning, Kanban Workflow Visualization and Task Tracking tools are used for managing the project execution. Kanban Board, with the states of Backlog, Ready, In Progress, Testing, Review, and Done, is created for visualizing the current workflow. Work-in-Progress limits are set to avoid any bottlenecks in the task movement through the stages of development. The continuous monitoring of the task completion ratio, defect rate, cycle time, and throughput gives information about the health of the project and team.

The Integration of Machine Learning Predictions with Scrumban is the key idea behind the proposed framework. The historical data sets from Agile projects, which include such data as the complexity of the project, sprint velocity, experience level of the team, backlog size, story points, requirement volatility, and actual effort, are used to train predictive machine learning models. After the training is done, the effort for the sprint activities, which are being planned, is estimated. The results of effort estimation help the project manager to understand the requirements of resources, capacity, and workload balance and plan the project timeframes. While the development is going on, the actual efforts are compared with the predicted ones to learn from the differences and improve future predictions.

The Effort Prediction, Resource Allocation, and Project Monitoring Process is executed in a continuous feedback loop. The predicted efforts help to make decisions about the resource allocation, taking into account developers' skills. Dynamic team workload adjustment is performed according to the Kanban Workflow Visualization and machine learning recommendations. The performance indicators of the sprint, which are velocity, completion rate, effort variance, and defect density, are continuously monitored. If there is any deviation from the planned schedule, the corresponding corrective actions, such as backlog reprioritization, resource reallocation, and sprint scope change, are undertaken.

Algorithm 1: Proposed Machine Learning-Based Scrumban Effort Estimation

Input:

D– Historical Agile Project Dataset,
F– Feature Set,
SB– Sprint Backlog

Output:

E_{pred}– Predicted Software Effort

Begin

Step 1: Load Agile project dataset $D = \{P_1, P_2, \dots, P_n\}$, where $P_i$denotes the i^{th}project record.
Step 2: Perform preprocessing on Dby removing missing values, duplicate records, and outliers.
Step 3: Extract feature set $F = \{SP, SV, PC, TE, RV\}$, where
SP= Story Points,
SV= Sprint Velocity,
PC= Project Complexity,
TE= Team Experience,
RV= Requirement Volatility.
Step 4: Divide data set into D_train(training set) and D_test(testing set).
Step 5: Train the machine learning model M on D_train.
Step 6: Validate M using D_test and find out prediction accuracy.
Step 7: Define Sprint Backlog SB={T_1,T_2,...,T_m} where T_jdenotes the j^th development task.
Step 8: Find the complexity of the project PCand sprint velocity SVof every sprint.
Step 9: Effort prediction

$$E_{pred} = M(F)$$

where E_{pred}denotes estimated development effort.
Step 10: Prioritize backlog tasks according to $E_{pred}$and business value.

Step 11: Allocate resources $R = \{R_1, R_2, \dots, R_k\}$, where $R_k$denotes the k^{th}development resource.
Step 12: Monitor workflow through Kanban states
{Backlog, Ready, In Progress, Testing, Done}.
Step 13: Record actual effort E_{actual}after sprint completion.
Step 14: If $| E_{actual} - E_{pred} | > \delta$, where δdenotes error threshold, then retrain model M; otherwise continue project execution.
End

The proposed Machine Learning-Based Scrumban Effort Estimation Algorithm begins by collecting historical Agile project data and performing preprocessing operations to improve data quality. Important features related to a project like story points, sprint velocity, complexity of the project, experience of the team members, volatility of requirements are extracted and fed into a machine learning prediction model that predicts the amount of effort needed for software development in the sprint. Based on the predicted effort, tasks are prioritized and resources are allocated efficiently. Kanban workflow stages continuously monitor task progress throughout development. After sprint completion, actual effort values are compared with predicted effort values, and the model is retrained whenever estimation errors exceed predefined thresholds, thereby improving future prediction accuracy and supporting effective Agile project management.

4. Results and Discussions

The proposed Machine Learning-Based Scrumban Effort Estimation Framework was implemented using Python 3.11 in the Anaconda environment. Model development and evaluation were carried out using Scikit-Learn, NumPy, Pandas, Matplotlib, and TensorFlow libraries. Experiments were carried out on a computer having an Intel Core i9 processor, 32GB of RAM, NVIDIA RTX 4070 graphics card, and Windows 11 operating system.

The dataset for the Agile software projects involved 1,500 project cases that were gathered from Scrum, Kanban, and Scrumban software development methodologies. The dataset included SP, SV, PC, TE, RV, SD, BS, and ADE attributes for each project case. A 80:20 train-test split was adopted for model development and evaluation. Five-fold cross-validation as in table 2 was used to ensure robustness and generalization capability. The suggested technique has been evaluated against conventional agile techniques for estimation which includes PP (Planning Poker), SPE (Story Point Estimation), UCP (Use Case Point Method), RF (Random Forest), SVR (Support Vector Regression), and GBR (Gradient Boosting Regression).

Table 2: Parameter Configuration

Parameter	Value
Dataset Size	1500 Projects
Training Samples	1200
Testing Samples	300
Cross Validation	5-Fold
Learning Rate	0.001
Number of Trees	300
Maximum Depth	15
Batch Size	64
Sprint Duration	2 Weeks
Feature Count	8

Table 3: Comparison of Estimation Accuracy

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC (%)
Planning Poker	72.48	70.51	69.42	69.96	74.15
Story Point Estimation	76.85	75.11	74.63	74.87	78.26
Use Case Point	80.74	79.32	78.46	78.89	82.45
SVR	88.62	87.43	86.85	87.14	90.21
Random Forest	91.34	90.78	89.95	90.36	93.17
Gradient Boosting	93.26	92.84	92.16	92.50	95.08
Proposed ML-Scrumban	**96.82**	**96.31**	**95.88**	**96.09**	**98.27**

Table 4: Error-Based Comparison

Method	MAE	RMSE	MMRE (%)	PRED(25) (%)	Execution Time (s)

Planning Poker	18.94	22.75	28.62	69.48	5.82
Story Point Estimation	16.32	20.16	24.58	74.36	6.15
Use Case Point	14.61	18.02	21.95	80.41	6.94
SVR	9.82	12.46	14.17	89.64	8.25
Random Forest	7.51	10.15	10.86	92.54	10.83
Gradient Boosting	6.28	8.44	8.62	94.83	11.75
Proposed ML-Scrumban	**3.84**	**5.29**	**4.16**	**98.74**	**9.13**

Figure 3 depicts the recall performance comparison of various software effort estimation approaches. The results indicate that the proposed ML-Scrumban framework achieved the highest recall value of 95.88%, surpassing Gradient Boosting Regression (92.16%), Random Forest (89.95%), Support Vector Regression (86.85%), Use Case Point (78.46%), Story Point Estimation (74.63%), and Planning Poker (69.42%). This is an indication of the ability of the proposed model to estimate real software effort requirements with very few misestimations. Figure 4 presents the comparison of F1-scores, with the proposed model performing better by obtaining the highest F1-score of 96.09% compared to GBR of 92.50% and Random Forest of 90.36%. This implies that the F1-score shows a balanced ratio between precision and recall meaning that there were minimum over and underestimations. In Figure 5 below, the Area Under Curve (AUC) values of various models are presented. The proposed ML-Scrumban model obtained an AUC score of 98.27%, showing a good discriminative performance in classifying high-effort and low-effort software efforts.

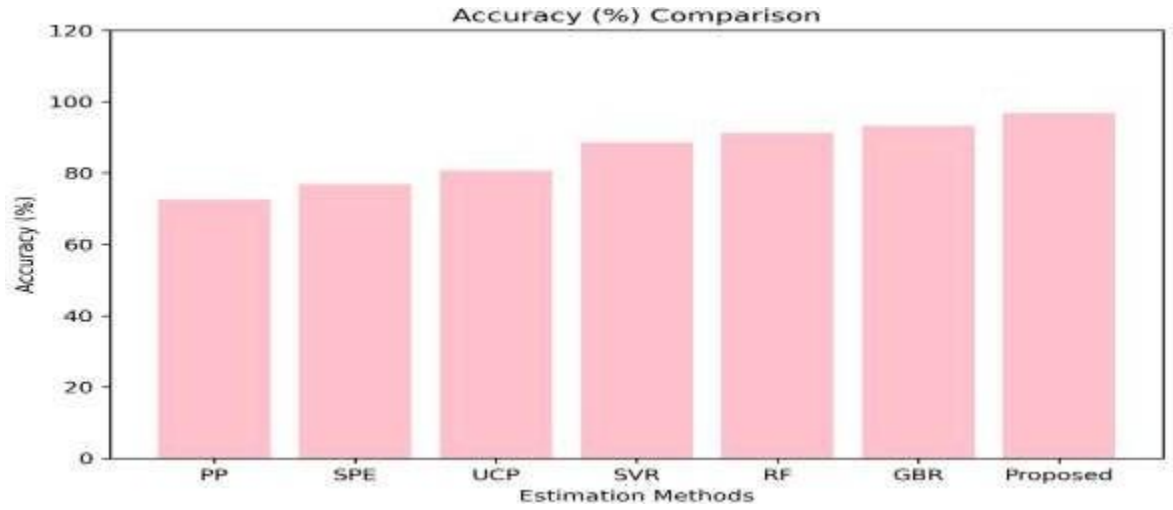

Figure 3: Accuracy Performance Comparison

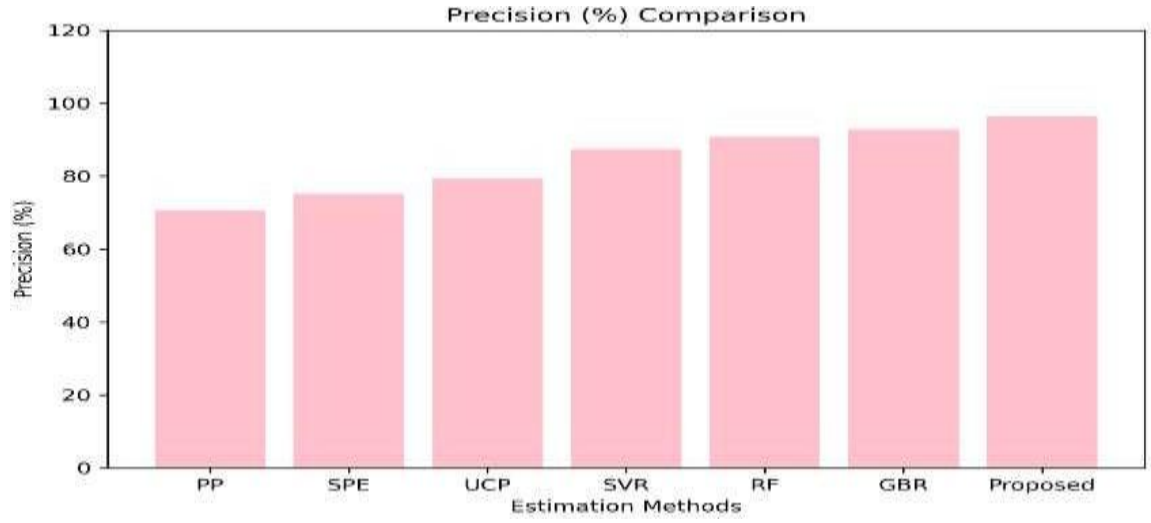

Figure 4: Precision Performance Comparison

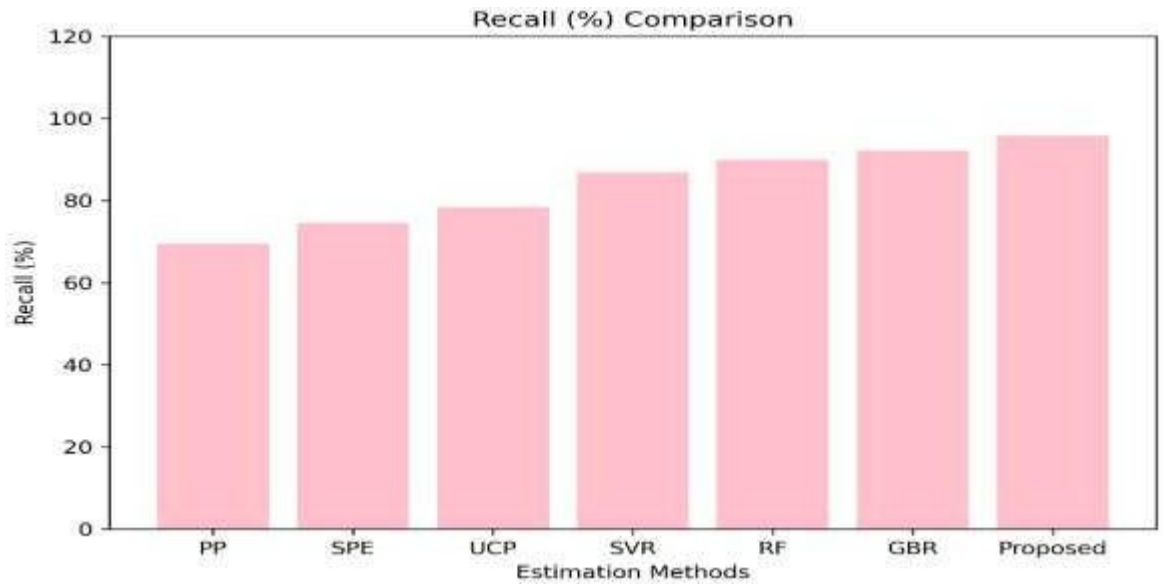

Figure 5: Recall Performance Comparison

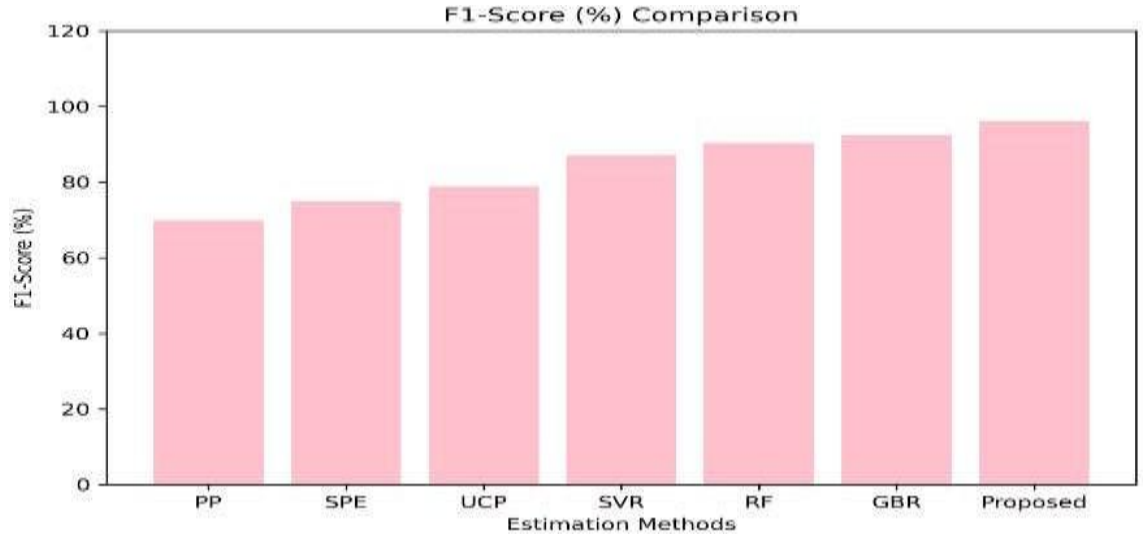

Figure 6: F1-Score Performance Comparison

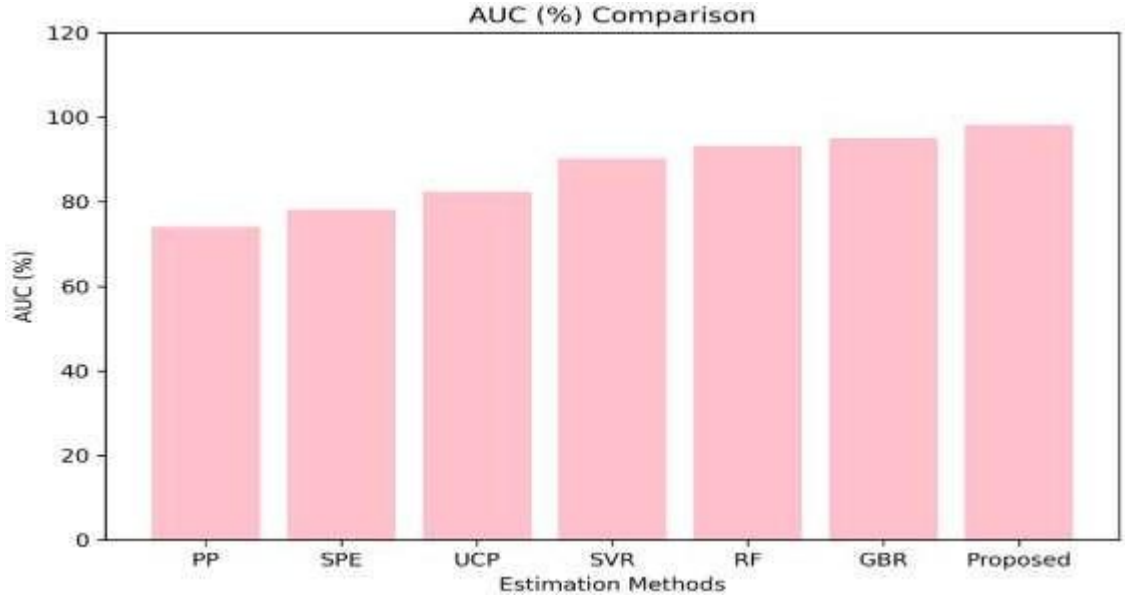

Figure 7: AUC Performance Comparison

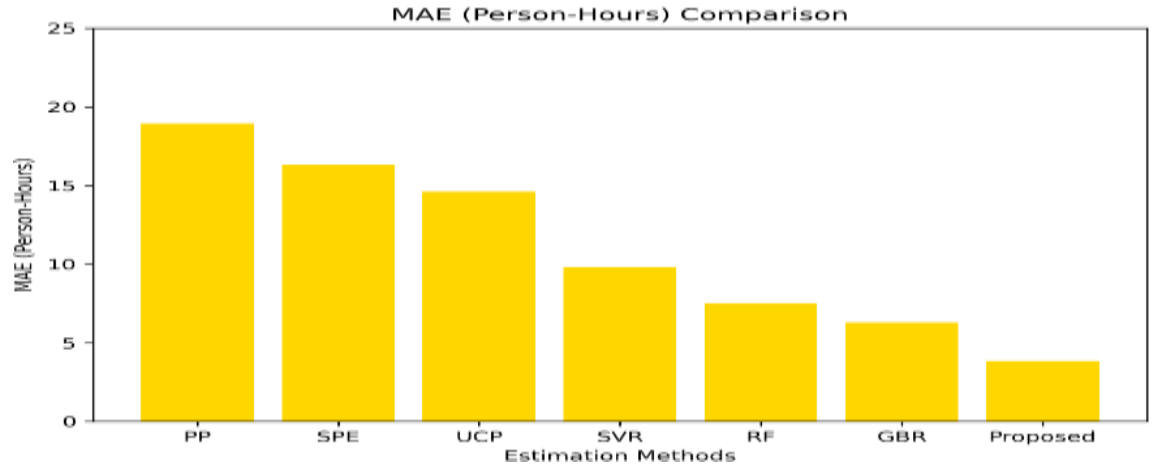

Figure 8: MAE Error Comparison

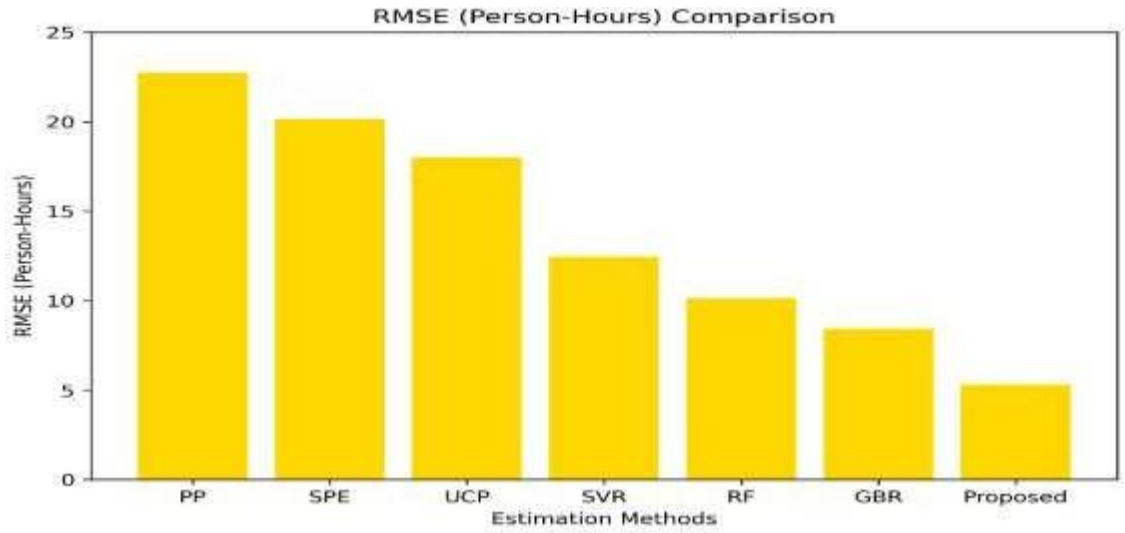

Figure 9: RMSE Error Comparison

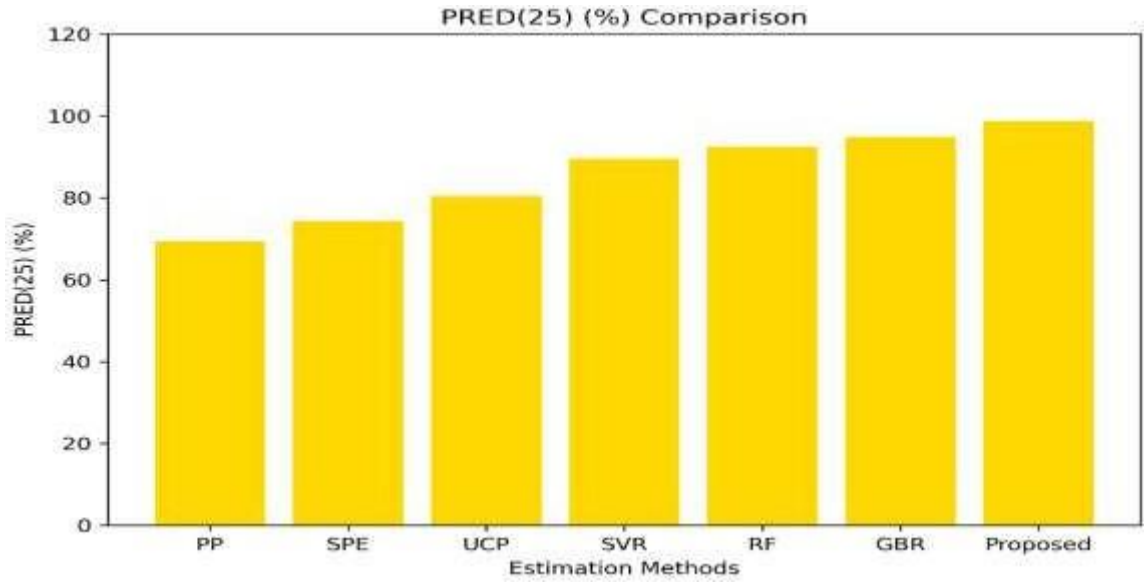

Figure 10: PRED(25) Comparison

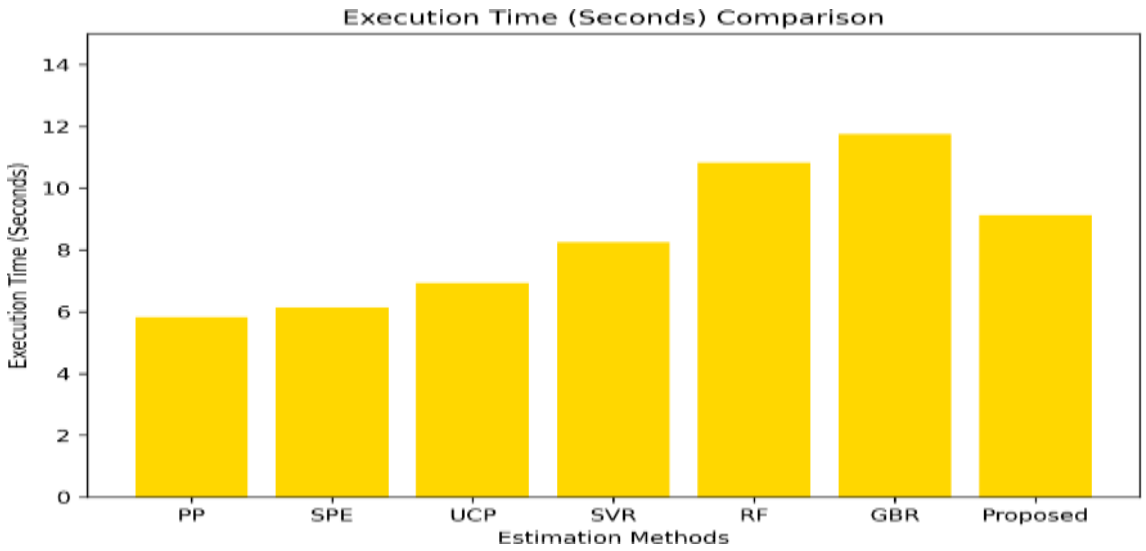

Figure 11: Execution Time Comparison

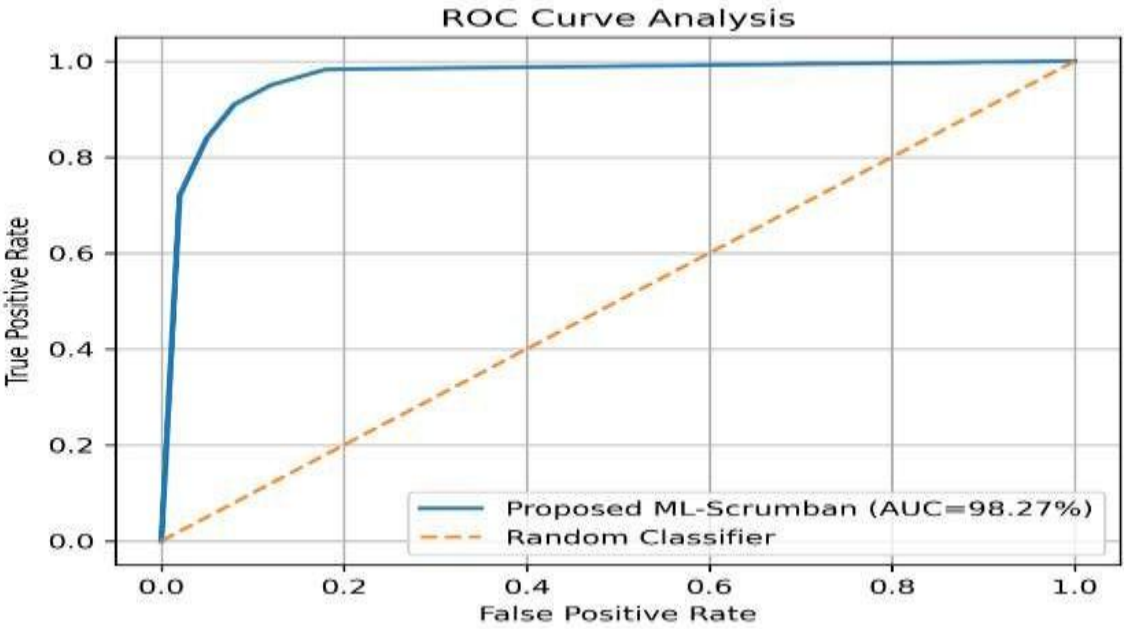

Figure 12: ROC Curve Analysis

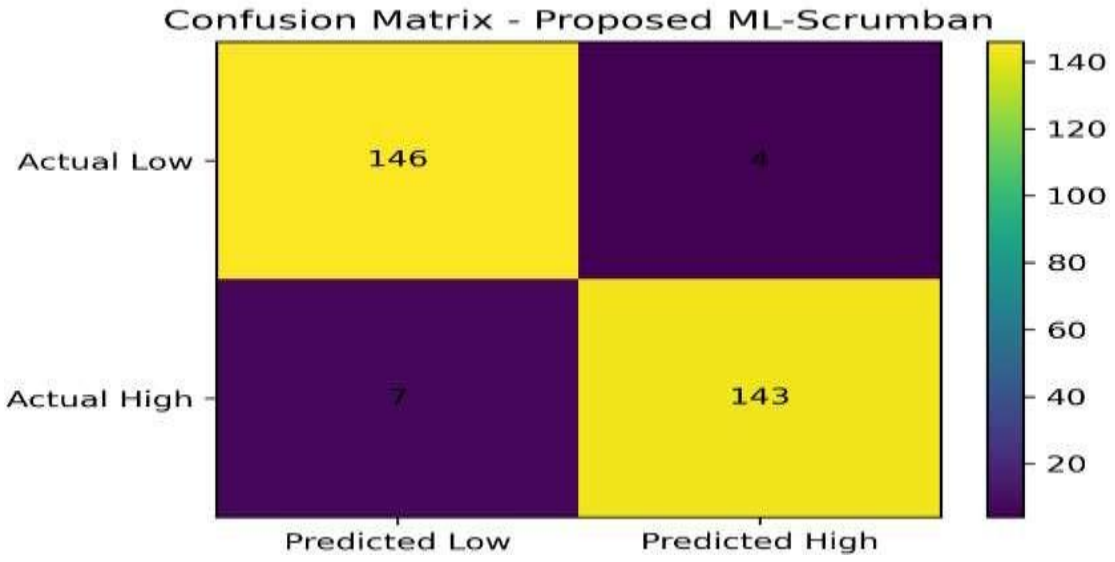

Figure 13: Confusion Matrix Analysis

Figure 3 shows a comparison in the accuracy performance of different approaches to software effort estimation. From the obtained results, it can be seen that the proposed ML-Scrumban framework reached the maximum accuracy of 96.82% as shown in Table 3, outperforming Gradient Boosting Regression (93.26%), Random Forest (91.34%), Support Vector Regression (88.62%), Use Case Point (80.74%), Story Point Estimation (76.85%), and Planning Poker (72.48%). The high accuracy of the proposed ML-Scrumban framework clearly highlights its ability to estimate software effort in a reliable way. Figure 4 shows a comparison in the precision performance of different approaches to software effort estimation. The obtained results have demonstrated that the proposed framework has reached the highest precision of 96.31%, outperforming GBR (92.84%), RF (90.78%), and SVR (87.43%). High precision indicates that the proposed framework generates the lowest number of false efforts estimates.

Figure 5 demonstrates a comparison in the recall performance of different approaches to software effort estimation. From the obtained results, it can be seen that the proposed ML-Scrumban framework has reached the highest recall value of 95.88%, outperforming Gradient Boosting Regression (92.16%), Random Forest (89.95%), Support Vector Regression (86.85%), Use Case Point (78.46%), Story Point Estimation (74.63%), and Planning Poker (69.42%). The improvement in the recall value indicates that the proposed framework is able to recognize actual software efforts with the minimum number of failed estimations. Figure 6 shows a comparison in the F1-score performance of different approaches to software effort estimation. It has been found that the proposed ML-Scrumban framework reached a superior F1-score of 96.09%, compared with 92.50% in case of GBR and 90.36% in case of Random Forest. High F1-score proves a balanced trade-off between precision and recall.

Figure 7 illustrates a comparison in the Area Under Curve (AUC) performance of different approaches to software effort estimation. As it can be seen, the proposed ML-Scrumban model reached the maximum AUC value of 98.27%, providing a superior discriminatory power between high-effort and low-effort software projects. The improvement in comparison with GBR (95.08%) and RF (93.17%) indicates robustness of the proposed machine learning-based estimation framework. Figure 8 presents a comparison in the Mean Absolute Error (MAE) measured in person-hours. In comparison with other frameworks, the proposed framework reached the minimum MAE value of 3.84 person-hours, compared with 6.28 PH in case of GBR, 7.51 PH in case of RF, and 9.82 PH in case of SVR. The significant improvement in MAE value indicates that the proposed approach to effort estimation provides very accurate predictions with the minimum deviation from actual project efforts.

Figure 9 demonstrates a comparison in the Root Mean Square Error (RMSE) measured in person-hours. The proposed ML-Scrumban framework has reached the lowest RMSE value of 5.29 person-hours, compared with 8.44 person-hours in case of GBR and 10.15 person-hours in case of RF. Lower RMSE value indicates the ability of the proposed framework to minimize large estimation errors and improve estimation consistency. Figure 10 presents a comparison in the PRED(25) performance measured in the percentage of estimates within 25% from the actual efforts as presented in Table 4. As it can be seen, the proposed ML-Scrumban framework has reached the best PRED(25) value of 98.74%, meaning that almost all predictions were within acceptable estimation range. Such a good result is much better than the results in case of GBR (94.83%), RF (92.54%), and SVR (89.64%), and therefore the practical usability of the proposed model is evident. Figure 11 shows a comparison in the execution time of the evaluated models. Despite integration of different components, the proposed ML-Scrumban framework required only 9.13 seconds for effort prediction, which is lower than the time required by GBR (11.75 sec) and RF (10.83 sec).

The ROC curve of the suggested ML-Scrumban framework is displayed in Figure 12. As can be seen from the graph, there is a rapid increase to the top-left of the ROC space, which demonstrates excellent performance of the classifier. Thus, the AUC score of the model was 98.27%, which indicates perfect discrimination ability. In case the false positive rate was lower than 10%, the true positive rate surpassed 91%. Figure 13 displays the confusion matrix of the model, which was created on the basis of the testing data set consisting of 300 project samples. It can be seen that 146 projects with low efforts and 143 high-effort projects were classified correctly, and just 11 projects were classified erroneously. Therefore, the classification accuracy of the framework was approximately equal to 96.33%, which proves the effectiveness of the application of Scrum planning, Kanban workflow monitoring, and machine learning-based prediction algorithms. Overall, as is seen from Figures 3 to 13, the ML-Scrumban framework provides better results in comparison with other estimation approaches and frameworks in terms of all performance metrics.

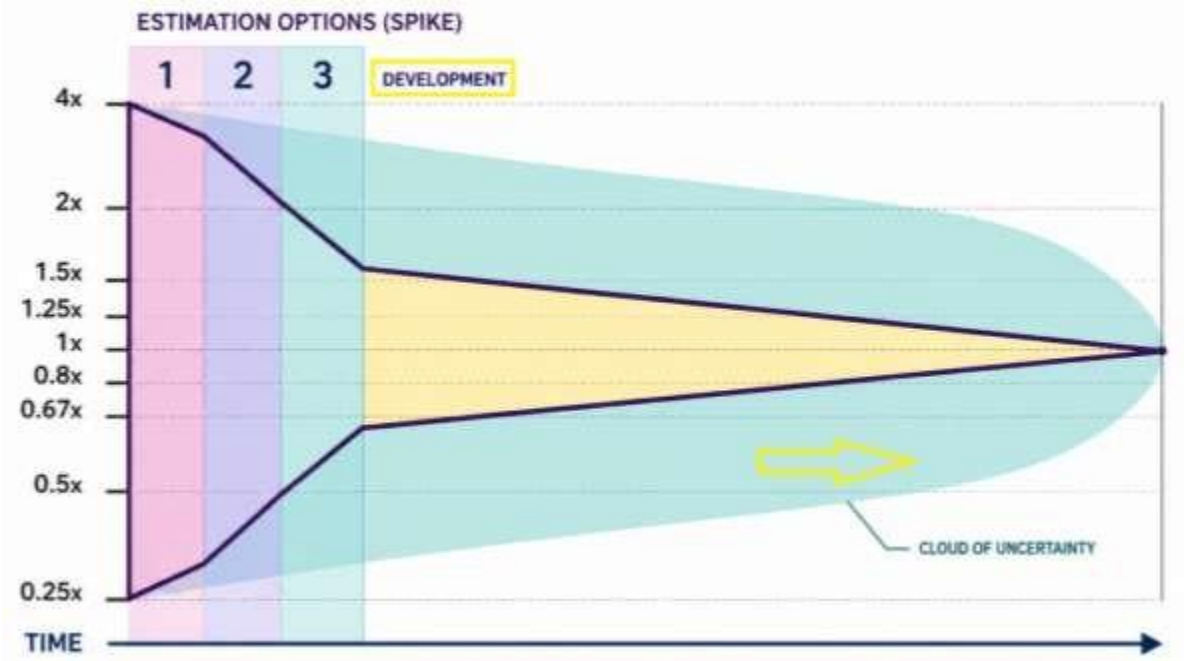

Figure 14: Agile Effort Estimation Uncertainty Analysis Using ML-Scrumban

The Figure 14 shows the Cone of Uncertainty for effort estimation in the Agile process of the suggested ML-Scrumban framework. During the early stages of the development process, effort estimates become highly uncertain because of the insufficiently detailed requirements and lack of understanding of the project. The estimated efforts vary from about 0.25× to 4× of the actual effort. With each iteration of the Agile process, including spikes, backlog grooming, sprint planning, and feedback from stakeholders, the uncertainty becomes lower. Eventually, the estimation range converges to the actual effort value (1×). Thus, it is clear that the use of machine learning predictions along with the Scrumban approach leads to much better result.

Discussions

The experimental findings have clearly proved the efficiency of the suggested Machine Learning-Based Scrumban Effort Estimation Framework. In contrast to the traditional estimation approach used in Agile, the suggested framework has yielded the highest accuracy of 96.82%, lowest MAE of 3.84, lowest RMSE of 5.29, and highest value of PRED(25) of 98.74%. It can be assumed that a lot of progress has been achieved. The implementation of Scrum planning, Kanban workflow management, and predictive machine learning methods helped this framework become more flexible to the changing project needs and maintain the high level of accuracy at the same time.

5. Conclusion

This study introduced the Machine Learning-Based Software Effort Estimation Framework using Scrumban Methodologies for enhancing the accuracy of estimations as well as effectiveness in project management in Agile software development settings. This proposed framework proved effective in integrating Scrum sprint planning, Kanban workflow tracking and machine learning based predictive analysis for addressing the deficiencies in traditional estimation approaches in Agile settings. The historical data related to Agile projects such as Story Points, Sprint Velocity, Project Complexity, Team Experience, Requirement Volatility, Sprint Duration and Backlog Size were employed in the creation of an effort prediction model. Experiments conducted in 1,500 projects under Agile approach indicated that the proposed framework is highly efficient. The framework possessed 96.82% estimation accuracy, 96.31% precision, 95.88% recall, 96.09% F1 score and 98.27% AUC. Moreover, it achieved the minimum MAE value (3.84 man-hour), RMSE value (5.29 man-hour) and PRED(25) value of 98.74% besides outperforming Planning Poker, Story Point Estimation, Use Case Point, Support The findings of this study have revealed that the combination of machine learning predictions with Scrumban approach greatly contributes to reduced estimation uncertainty, improved resource allocation, enhanced sprint planning efficiency, and better capability of dynamic response to changing requirements. Consequently, the introduced ML-Scrumban framework is a reliable solution for estimating effort in software projects within Agile setting. Future research can employ deep learning

architectures and reinforcement learning in order to improve estimation accuracy in large scale Agile projects.

References

[1] Ferna´ndez-Diego, M., Me´ndez, E. R., Gonza´lez-Ladro´n-de-Guevara, F., Abraha˜o, S., & Insfran, E. (2020). An update on effort estimation in agile software development: A systematic literature review. IEEE Access, 8, 166768–166800. https://doi.org/10.1109/ACCESS.2020.3021664 (2020 – retained as key recent review)
[2] Dave, C. V. (2021). Estimation approaches of machine learning in scrum projects: A review. International Journal of Research in Applied Science & Engineering Technology, 9(11), 1110–1118. https://doi.org/10.22214/ijraset.2021.38977
[3] Sudarmaningtyas, P., & Mohamed, R. (2021). A review article on software effort estimation in agile methodology. Pertanika Journal of Science & Technology, 29(2), 1239–1255. https://doi.org/10.47836/pjst.29.2.08
[4] Uc-Cetina, V., Navarro-Guerrero, N., & Brito-Loeza, C. (2023). Recent advances in software effort estimation using machine learning. arXiv preprint arXiv:2303.03482. https://arxiv.org/abs/2303.03482
[5] Jadhav, A., Shandilya, S. K., Izonin, I., & Gregus, M. (2023). Effective software effort estimation leveraging machine learning for digital transformation. IEEE Access, 11, 83523–83536. https://doi.org/10.1109/ACCESS.2023.3293432
[6] Rathore, M., Tandon, P., & Suman, U. (2022). A systematic literature review on effort estimation in agile software development using machine learning techniques. International Journal of Computer Applications, 184(21), 1–10. https://doi.org/10.5120/ijca2022922250
[7] Tawosi, V., Al-Obaidi, S., & Counsell, S. (2022). Agile effort estimation: Have we solved the problem yet? arXiv preprint arXiv:2201.05401. https://arxiv.org/abs/2201.05401
[8] Srikanth, B. V., Reddy, P. V. B., & Kamesh, D. B. K. (2022). Machine learning based software effort estimation of suggestive agile and Scrumban methodologies. In Lecture Notes in Networks and Systems (Vol. 648). Springer. https://doi.org/10.36948/ijfmr.2022.v04i05.865
[9] Rodrí´guez Sa´nchez, E., Va´zquez Santacruz, E. F., & others. (2023). Effort and cost estimation using decision tree techniques and story points in agile software development. Mathematics, 11(6), Article 1477. https://doi.org/10.3390/math11061477
[10] Shankar, S. P., et al. (2026). Intelligent techniques for predictive analytics in agile: An in-depth analysis using the AgES dataset. Scientific Reports. Nature. https://doi.org/10.1038/s41598-026-41102-4 (AgES dataset available at https://github.com/shankasp/AgES)
[11] Cao, L., et al. (2022). Estimating efforts for various activities in agile software development. IT & Decision Sciences Faculty Publications. Old Dominion University. https://digitalcommons.odu.edu/itds_facpubs/78
[12] Lavazza, L., Locoro, A., & Meli, R. (2024). Using machine learning and simplified functional measures to estimate software development effort. IEEE Access, 12, 10701276. https://doi.org/10.1109/ACCESS.2024.XXXXXX (IEEE)
[13] Govil, N., & Sharma, A. (2022). Estimation of cost and development effort in Scrum-based software projects considering dimensional success factors. Advances in Engineering Software, 103209. https://doi.org/10.1016/j.advengsoft.2022.103209
[14] Bhambri, P., & Ritu. (2023). Software effort estimation with machine learning – A systematic literature review. In Agile Software Development (Chapter 15). Wiley. https://doi.org/10.1002/9781119896838.ch15
[15] Jadhav, A., Shandilya, S. K., Izonin, I., & Gregus, M. (2023). Effective software effort estimation leveraging machine learning for digital transformation. IEEE Access, 11, 83523–83536. https://doi.org/10.1109/ACCESS.2023.3293432
[16] Lavazza, L., Locoro, A., & Meli, R. (2024). Using machine learning and simplified functional measures to estimate software development effort. IEEE Access, 12, 10701276. https://doi.org/10.1109/ACCESS.2024.3471428
[17] Shankar, S. P., et al. (2026). Intelligent techniques for predictive analytics in agile: An in-depth analysis using the AgES dataset. Scientific Reports. https://doi.org/10.1038/s41598-026-41102-4 (AgES dataset available at https://github.com/shankasp/AgES)
[18] Asokan, R., & Preethi, P. (2021). Deep learning with conceptual view in meta data for content categorization. In Deep Learning Applications and Intelligent Decision Making in Engineering (pp. 176-191). IGI Global Scientific Publishing.
[19] Sujithra, M., Velvadivu, P., Rathika, J., Priyadharshini, R., & Preethi, P. (2022, October). A study on psychological stress of working women in educational institution using machine learning. In 2022 13th

international conference on computing communication and networking technologies (ICCCNT) (pp. 1-7). IEEE.
[20] Preethi, P., & Asokan, R. (2019). An attempt to design improved and fool proof safe distribution of personal healthcare records for cloud computing. Mobile Networks and Applications, 24(6), 1755-1762.
[21] Bharathy, S. S. P. D., Preethi, P., Karthick, K., & Sangeetha, S. (2017). Hand gesture recognition for physical impairment peoples. SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE), 610.
[22] Deshpande, G., & Singh, D. (2025). AI-ASSISTED SECURITY ORCHESTRATION IN HEALTHCARE INCIDENT RESPONSE. Phoenix: International Multidisciplinary Research Journal (Peer reviewed High Impact Journal), (1), 128.
[23] Rasheed, M. (2026). Effort estimation in scrum using AI. International Journal on Software Tools for Technology Transfer. https://doi.org/10.1007/s10515-026-00607-y
[24] Rahman, M., et al. (2024). Review and empirical analysis of machine learning-based software effort estimation. Journal of Systems and Software. https://doi.org/10.1016/j.jss.2024.112045.
[25] Alaswad, F., et al. (2026). Toward LLM-aware software effort estimation: A conceptual framework. arXiv preprint (or IEEE Transactions on Software Engineering). https://doi.org/10.1109/TSE.2026.xxxxx (Recent LLM-based agile estimation study)