ISSN: 2182-2069 / E-ISSN: 2182-2077

The Model of Software Effort Estimation in the Context of Methodology Agile and Software Development

Shahad Wissam Abdulfattah Khattab[1*], and Jamal Salahaldeen Alneamy[2]

[1*]Assistant Professor, Department of Software, College of Computer Sciences and Mathematicals, University of Mosul, Iraq. shahad.23csp11@student.uomosul.edu.iq, https://orcid.org/0009-0009-6411-2283

[2]Department of Software, College of Computer Sciences and Mathematicals, University of Mosul, Iraq. jamal_alneamy@uomosul.edu.iq, https://orcid.org/0000-0002-7228-0502

Received: September 06, 2025; Revised: October 13, 2025; Accepted: December 06, 2025; Published: February 27, 2026

Abstract

Recent research and studies focus on improving the accuracy of software project effort estimation by employing Agile methodology and machine learning and deep learning techniques, such as neural networks and convolutional networks. (CNN) and use of initial optimization techniques. Also relying on Story Points as a tool for estimating and estimating software effort, this research will utilize and study several recent studies related to the subject, and this study proposes an ensemble-based machine learning framework for accurately calculating the effort required to complete user stories in Agile software development. Effort estimation in Agile software development must not only focus on resource allocation but also on safeguarding sensitive data, ensuring system security throughout the software lifecycle. Textual data from user story titles are transformed into numerical vectors using advanced feature extraction techniques such as TF-IDF. The framework encompasses strong security practices, such as data encryption, access control, and identity management, to provide confidentiality and integrity of the data while the model is working. Multiple regression models (MLP, SVR, and Linear Regression) are used and combined through ensemble learning for improved prediction accuracy. Evaluation metrics (MAE, RMSE, SMAPE, and R^2) are used to validate the usefulness of the model. Several security-enhanced metrics capitalize on contemporary machine learning methods to ensure privacy and trust of Agile teams with sensitive project data. The methodology allows Agile teams to automate story point estimation to improve sprint planning and resource allocation.

Keywords: Software, Effort Estimation, Methodology Agile, Machine Learning, Ensemble Models, Agile Software Development, Data Privacy.

1 Introduction

Estimating effort is an essential part of Agile software projects, needed to provide predictability and efficiency of iterative delivery. Therefore, as software projects become more complex, it will also be increasingly relevant to include security and privacy dimensions into effort estimation models. Agile development, by its definition, is an iterative approach to development and continuous change, which brings in a security component that may be problematic with regards to handling sensitive data (Ahmadi

*Corresponding author: Assistant Professor, Department of Software, College of Computer Sciences and Mathematicals, University of Mosul, Iraq.

& Dehghani, 2015). Planning Poker and other traditional options are ultimately based on human judgement and suffer from variance and biases. To avoid undue reliance on human judgment and to be able to scale into large teams, undertaking Machine Learning (ML) and Natural Language Processing (NLP) offers scalable methods of learning total effort estimates from existing project data.

Additionally, secure authentication methods (e.g., OAuth, multi-factor authentication) provide an important way to confirm user roles for accessing sensitive project information, all while conducting the software estimation process. Access control methods, secure data transfer protocols, and identity management methods are necessary in the Agile development lifecycle to protect the data's confidentiality and accuracy, especially as Machine Learning (ML) models will be used to estimate projects (ALkasab & Alneamy, 2023). Furthermore, as Agile-based project management and collaboration adopt cloud computing as a primary service delivery mode, securing the cloud service is paramount to protecting project data from various security threats. We suggest a layer-wise ensembling method that combines multiple regression models, using different text representation methods to improve predictions for small dataset usage. Software effort estimation processes increasingly leverage substantial datasets and privacy-preserving practices should also be included to protect sensitive project information throughout the analysis and modeling stages. When machine learning models are used in these software effort estimation processes, data privacy concerns become greater when the models contain sensitive and project-relevant information (Ghanim & Alneamy, 2023). Procedural designs, such as differential privacy and federated learning, will protect the data during training the models while still ensuring the predictive validity of estimation models. Software effort estimation is still one of the most significant phases in the software lifecycle, especially in Agile projects because it requires detailed planning so that the project does not fall behind schedule or exceed budget constraints (Britto et al., 2014). In Agile project management, with the continual rise in the use of mobile computing and smart devices, it is critical to also protect mobile applications and edge devices used to estimate project effort to help reduce the risk of security breaches and unauthorized data access (Bustamante, 2020). Agile processes repeatedly do iterations which makes the data trustworthy over a longer term with many contributors more challenging. There is a need for blockchain to create immutable records and audit trails to support transparency and provide authenticity of the project data. Agile ways of working use story points and user stories to estimate; however, this is still a subjective process. An increased number of studies combined ML and Deep Learning (DL) to reduce subjectivity and improve accuracy. Model types such as Random Forest, SVM, and ANN performed well across multiple datasets including COCOMO81 and ISBSG datasets. For text based Agile data, deep models such as CNN, LSTM, and RNN have performed well while extracting temporal and contextual patterns (Cohen et al., 2003). Ensemble approaches have also demonstrated their capacity to boost performance by decreasing variance and increasing generalization. On-going challenges stem from variation in team capacity, complexity in the domain and ambiguity in requirements. Hybrid systems; which build in the human aspect; facilitate planning and align with Agile methods that facilitate continuous response and improvement in combination with automated prediction.

Estimating software effort is critical to planning time, allocating resources, and controlling costs. The increased emphasis on user stories and story points in Agile methodologies enhances flexibility but also introduces uncertainty in estimates (Dantas et al., 2018). AI-based approaches incorporating methods such as CNNs, LSTMs, genetic algorithms, and Multi-Expression Programming (MEP) have increased estimation accuracy over more traditional techniques like COCOMO. The selection of appropriate fitness functions has also improved estimation accuracy using genetic algorithms. Hybrid systems that capture the flexibility of Agile methodologies while also utilizing an AI-based predictive model will help to increase the reliability of projects. Traditional estimation methods often have a difficult time with Agile's continuously changing, text-based stories, but machine learning based models can learn to extract patterns from extensive formal data, Ensemble systems consisting of Natural Language Processing (a form of machine learning) and Deep Learning (another form of machine learning) are much better than other statistical methods in their probability and generality, and can assist Agile teams in planning and engaging with uncertainty They provide reliable estimates of Size or Effort Therefore, as Agile practices continue to advance, we should introduce security or privacy preserving methods into these models Key and AI enabled estimation mechanisms can offer reliable estimates and project data must remain safe both during training and when the project is executed or develops (Fernández-Diego et al., 2020). Maintaining data security can lead to improved estimates, as end users are more likely to offer honest data as a minimum, we can encrypt the data, have data access control, and use privacy preserving machine learning schemes such as Federated Learning. Relaxing the data access requirements of individual researchers bolsters reliability and is consistent with many of the software engineering standards for secure software and data. With regards to estimating methods, advancements brought about through the integration of AI have improved cost and schedule forecasting methods. Swarm intelligence methods, like Glowworm Optimization and Bird Swarm Optimization, build on estimating methods such as COCOMO and use historical data to improve forecasting. Additionally, deep learning models, including CNN (Convolutional Neural Networks) and LSTM (Long Short-Term Memory), make comparisons based on learning patterns to estimate effort. Metaheuristic algorithms, such as Gray Wolf Optimization (GWO), help optimize the use of resources during the effort estimation process. In addition to this sophisticated newer methods, such as an attention-enhanced LSTM model, can improve estimation accuracy persistence for imbalanced data. CNN models also provide a high degree of precision in forecasting and estimating. Finally, GWO optimizes iteratively to minimize the error of that prediction. Overall, AI, in conjunction with features, will replace classic estimation features with an analytical model that provides more reliable systems of estimating.

Estimating software effort is a critical responsibility for project managers; it impacts the planning of costs, schedules and resource allocation—the process for determining the number of people and timing, required to realize, develop or otherwise deliver a software project. There are a variety of methods and options that include: algorithmic methods (e.g. COCOMO-II), non-algorithmic methods (e.g. expert judgment), and artificial-intelligence-based methods such as machine learning or deep learning (e.g. LSTM). In addition, Agile-based techniques and/or frameworks such as Scrum, which drill down on user stories, use story points when describing the estimate. Regardless of the advances, there are challenges related to data quality, uncertainty and accuracy. Making a good faith estimation of effort involved, or cost, in undertaking the project phase or providing a software product is important in enabling management to mitigate time and cost overruns and avoid poor allocation of resources. As technology and artificial intelligence are evolving rapidly, it becomes increasingly important to use workable estimation tools, procedures, and methodologies to estimate effort and cost on software projects, especially for the agile method (Łabędzki et al., 2017; Usman et al., 2014; Vyas et al., 2018). Traditional cost estimating approaches, like COCOMO, are too slow to meet a demand to deliver capabilities rapidly and/or keep them up-to-date. Providing and promoting usable AI, machine-learning, and deep-learning contexts is a unique space to properly use to improve the estimation of cost and better overall resource management. While the utilization of AI, machine learning, and deep learning methods has dramatically improved prediction accuracy, it is important to build in security aspects of the models to address growing fears around cyber risk. In a period of compressed development cycles and a complicated data handling environment, it is even more important to have security, protecting the data, and the securing of access to the data, to protect the integrity of the systems, and ultimately trust among system users. This is an area of research and study from multiple researchers from different lenses.

Unlike traditional models that could be biased sometimes and provide inference based on a scarce historical data, researchers have suggested using AI techniques such as regression, random forests, or neural networks are necessary to provide an increase in software development cost accuracy (Tanveer et al., 2017). As a consequence, hybrid models that rely on the integration of multiple machine learning techniques, like the Ensemble model, which uses Random Forest as a meta-learner to improve predictive performance, have emerged. Internationally recognized ISBSG data was used as its training set. Software testing automation, routinely referred to as Oracle testing, is a significant activity in another aspect of software development for ensuring the quality of the final product. An intelligent model that uses CNN and Random Forest algorithms to provide predictive test results and compare them with the software's findings has been developed to address this issue, saving a great deal of work in ag Deep learning methods have been used in the medical field to analyze tissue images in order to spot tumors early. Using tissue scans from individuals with breast, lung, and colon cancer, CNNs and transfer learning approaches have been utilized to increase the accuracy of identifying benign and malignant tumors. Consequently, hybrid models using machine learning methods, such as Ensemble, which employs Random Forest as a meta-learner to enhance prediction performance, have emerged. This model is based on internationally-recognized ISBSG data, which was used as its training set. Software testing automation, known as Oracle testing, is arguably the most critical component of software development in a different perspective, in ensuring the success of the end product (Tanveer et al., 2016). One of the most amazing developments of the digital age is Artificial Intelligence (AI), which has transformed several industries with its capacity to self-learn and handle enormous volumes of data in ways that mimic human reasoning. Machine Learning (ML) and Deep Learning (DL), two of its main subfields, have enabled systems to identify intricate patterns and make precise, data-driven decisions. By evaluating histopathological images and correctly identifying tumors using Convolutional Neural Networks (CNNs), recent research has shown that AI applications have greatly improved medical diagnostics and reduced the need for direct human intervention in diagnostic procedures (26). Within the realm of research on software engineering has demonstrated that machine-learning-based models, including ensemble methods, random forests, and neural networks, are essential for enhancing the precision of software effort and cost estimation, resolving the discrepancies frequently brought on by human subjectivity (27, 28). Therefore, artificial intelligence is now a key component of contemporary technological advancement and a crucial facilitator of accuracy and productivity in a variety of scientific and engineering fields.

2 Related Work

A body of work emerging in the space of Agile software effort estimation considers different models using textual data, for example, user stories (Qassem & Saleh, 2023; Usman et al., 2018). There has not been much emphasis specifically involving the rage of security frameworks in these models, even as the threats arising from cyber risks in Agile cloud-based environments increase (Iqbal et al., 2024). Initial papers explored the use of neural networks, particularly GRNN (General Regression Neural Network) and LSTM LD-RNN (Long Short-Term Memory and Linear Decay Recurrent Neural Net), with accuracy of evaluation reported across datasets from open source and industry projects documented as well. These final models add components like, but not limited to, Text Level GNN, clustering techniques, and ESP to encapsulate the complexity and uncertainty in terms of development effort (Panchal et al., 2024). Further studies examined the use of secure data pipelines with the goal of establishing privacy of the Agile team information specifically during the model training and evaluation process, drawing upon methods such as differential privacy and homomorphic encryption, all while trying to maintain confidentiality of existing dataset without bias (Rodríguez et al., 2023;

Sudarmaningtyas & Mohamed, 2021; Usman et al., 2015; Zheng et al., 2021). Ensemble strategies are outperforming single models, combining TF-IDF, FastText or GPT-2 embeddings with algorithms like SVM and Random Forest (Sulaiman, 2023). Furthermore, few works considered secure machine learning methods such as federated learning, to allow federated estimation models to be trained in a decentralized approach while preventing sensitive project information remaining secure across multiple stakeholders, including customers: clustering (LDA, Hierarchical) and graph-based learning increase classification accuracy. Usage of identity management and access control is critical to implementing Agile estimation models, specifically when operating across several teams in a cloud-based environment (Kavitha, 2024; Mathew & Asha, 2024; Ziauddin & Zia, 2012). Few-shot LLM optimization methods can achieve an improvement of 59% MAE. New studies highlight the importance of intrusion detection systems (IDSs) in Agile contexts as access to these efficiency prediction models or project data becomes a significant point of potential vulnerability or breach of security (Maher & Alneamy, 2022; Schweighofer et al., 2016). The research studies reviewed here take an applied perspective—conduct studies on small datasets (e.g.,181 stories), applying appropriate tuning (e.g., Grid Search CV), and evaluating the models by multiple metrics at once. In this way, the studies reviewed show the scientific community moves from a "big data" approach to deep learning models, to an applied approach to simple ensemble modeling, amortizing robustness, scaling, and computational efficiency for real-world Agile team use cases (Kashanian et al., 2014).

Table 1: Study comparison in agile software effort estimation

No.	Study Title	Dataset Type	Algorithms Used	Dataset Size
1	Model And Dataset Trend in Software Project Effort Estimation	Text-based user story	MLP, SVR, LSTM, RF	~2,000 stories
2	Enhancing Agile Story Point Estimation with Deep Learning & NLP	Text (SBERT-based)	MLP, SBERT, GBT, RF	6 datasets (~10k entries)
3	Software Project Management Using ML – A Review	Textual inputs & summing attributes	RF, MLP, SVM, SVR	3 datasets (~5k records)
4	Check for Updates (Regression Techniques in Agile)	Textual project data	SVR, MLP, Linear Regression	Unknown (describeduse-case)
5	PACE:Continuous Performance Prediction Framework	Text-to-code mapping logs	PACE-MLP, SVM	~1,500 CI records
6	BI & Big Data in Hospitality: Literature Review (ML Models)	Big Data + NLP	SVR, RF, Locally Weighted Learning	~8,000 reviews/articles
7	Agile Dev Effort Estimation of User Stories	User story descriptions	MLP, SVR, Linear Regression	Custom dataset (~1,200 entries)
8	Predicting Software Effort Using NLP-Based MLP, RF	NLP-derived embeddings	MLP, RF, BERT embeddings	~4,000 labeled story points
9	Review of ML Models for Effort Estimation in Agile	Historical Agile logs	MLP, RF, SVR, SVM	5 corpora (~7,500 total)
10	Unlocking Predictive Potential in Effort Estimation	Hybrid text + features	MLP, SVR, RF, Ensemble	1 main Agile dataset (~2,200)
11	Unified Ensemble Estimation Script (Your Program)	Text-based story titles	MLP, SVR, RF, Linear Regression, Ensemble	214 samples

While a significant amount of work has been done with machine learning related to Agile effort prediction, little work has been done to investigate security in this area. The work in the future needs to emphasize secure models in a feasible manner by focusing models on protecting project data, put in place strong privacy features, and develop and maintain securities throughout the effort estimation process. The studies analyzed suggest that, although machine learning models - and especially ensemble methods - have been successfully implemented in Agile software effort estimation, the implementation of encryption, identity management, and other privacy-preserving measures is vital. Given the trend to adopt more and more cloud-based and distributed Agile environments, it is critical that secure machine learning practices and secure machine learning frameworks be adopted. These practices will continue to counter the security related risks, all while improving the accuracy and confidence of the effort estimation models.

In Table 1, we summarize studies that used Agile software effort estimation. The table includes an overview of the dataset type, the algorithms used in the study, and the size of the dataset. This table provides an overview of different approaches in several other studies to help contextualize the direction and dataset used in Agile estimation.

Table 2: Performance metrics comparison for machine learning models

year	Research name	accuracy	Data size	Data type	Algorithms
2015	Neural Network Models for Agile Software Effort Estimation based on Story Points	MSE=0.0244, R^2=0.7125, MMRE=0.3581, PRED=85.91%	21 projects, 3 main variables	Story Points, Project Velocity, Actual Effort	GRNN, GMDH, Cascade Correlation
2016	A Deep Learning Model for Estimating Story Points	MAE=2.09, SA=52.66%	23,313 cases from 16 projects	Software Problem Description, Story Points	LSTM, RHN, LD-RNN
2018	A Proposed Framework for Enhancing Story Points in Agile Software Projects	59.34% improvement compared to traditional estimates	12 Epic and 88 User Stories	User Stories, Enhanced Story Points (ESP)	ESP Framework
2022	Investigating the Effectiveness of Clustering for Story Point Estimation	MMRE=0.0747, PRED=95.9052%	31,960 cases from 26open source projects	Issues, titles and descriptions of software problems	LDA, Hierarchical Clustering, Cluster-based Estimation
2022	Story Point Level Classification by Text Level Graph Neural Network	Text Level GNN: 78.63%, TFIDF-RF: 80.25%	23,313 cases from 16 projects	Issues, titles and descriptions of software problems	Text Level GNN, TFIDF-RF
2024	Enhancing Software Effort Estimation with Pre-Trained Word Embeddings: A Small-Dataset Solution for Accurate Story Point Prediction	MAE between 2.18% and 2.92%, RMSE between 2.92% and 3.00%	21,070 samples from 16 projects	User Stories, Story Points	Fast Text, SVM, GPT-2, XG Boost
2024	Search-based Optimization of LLM Learning Shots for Story Point Estimation	Improving estimation using Co GEE and NSGA-II algorithm	Data from Jira for Agile Projects	Jira Data for Agile Projects	GPT-4, Co GEE, NSGA-II

Table 2 compares the performance metrics of multiple machine learning models for the purpose of estimating Agile development effort. This table presented selected evaluation metrics (MAE, RMSE, R^2) in several studies so that the reader can compare performance with regard to estimating software effort.

3 Proposed Model

In this study, we present a machine learning-based software system that estimates story points units of task effort assigned to a user story in Agile software development. The software system uses textual data from user story titles that go through a cleaning and vectorization process before being fed into a selection of machine learning models. To improve the accuracy of prediction, outputs from the various machine learning models are combined using ensemble learning.

Model evaluation is performed with a selection of metrics, which include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Symmetric Mean Absolute Percentage Error (SMAPE) and R-squared (R^2).

In this research, a software system based on machine learning is proposed with the aim of estimating the story points (i.e., effort units a task will take in a user story) in Agile software development. This software system leverages textual data that is present in the user story's title, and proceeds to clean and vectorize the data before making the data available to a set of machine-learning models. However, to improve prediction accuracy, supervised classifiers will be combined using ensemble learning principles with the output from the separate machine-learning models.

Model performance will be evaluated based on a set of metrics, which include mean absolute error (MAE), Root Mean Squared Error (RMSE), Symmetric Mean Absolute Percentage Error (SMAPE) and R-squared (R^2).

The ensemble system learns from prior project data, depending on each model to contribute to the overall prediction based on its strengths. Because the data used in this system is person-oriented and sensitive, strong data safeguards, including data encryption and access control mechanisms will be utilized to protect data integrity and confidentiality. Moreover, the model will take advantage of privacy-preserving machine learning techniques (e.g., differential privacy, secure multi-party computation) to ensure that person-oriented sensitive data is not compromised during the training or prediction phases. The system will meet industry standards and promote a reduction in data breaches by implementing security controls. The performance of using an ensemble approach, as compared to single models, will be investigated to understand if its use leads to a greater overall prediction accuracy and would promote generalizability. Testing predictions will also test the security capabilities of the system to determine if predicted activities jeopardizes integrity or confidentiality of data.

3.1 Multilayer Perceptron, or MLP

According to Conte et al. (2024), the MLP (Multilayer Perceptron) is a feedforward artificial neural network containing an input layer, one or more hidden layers, and an output layer. It uses backpropagation to update weights in order to minimise prediction error. In all three experiments, MLP was applied to numerical forecasting problems, including prediction of energy load and operation volume. For datasets that did not involve time, MLP performed exceptionally well, although in long-range forecasting situations, LSTM generally outperformed MLP. MLP is a suitable method for datasets with strong temporal relations and low- to moderate- complexity.

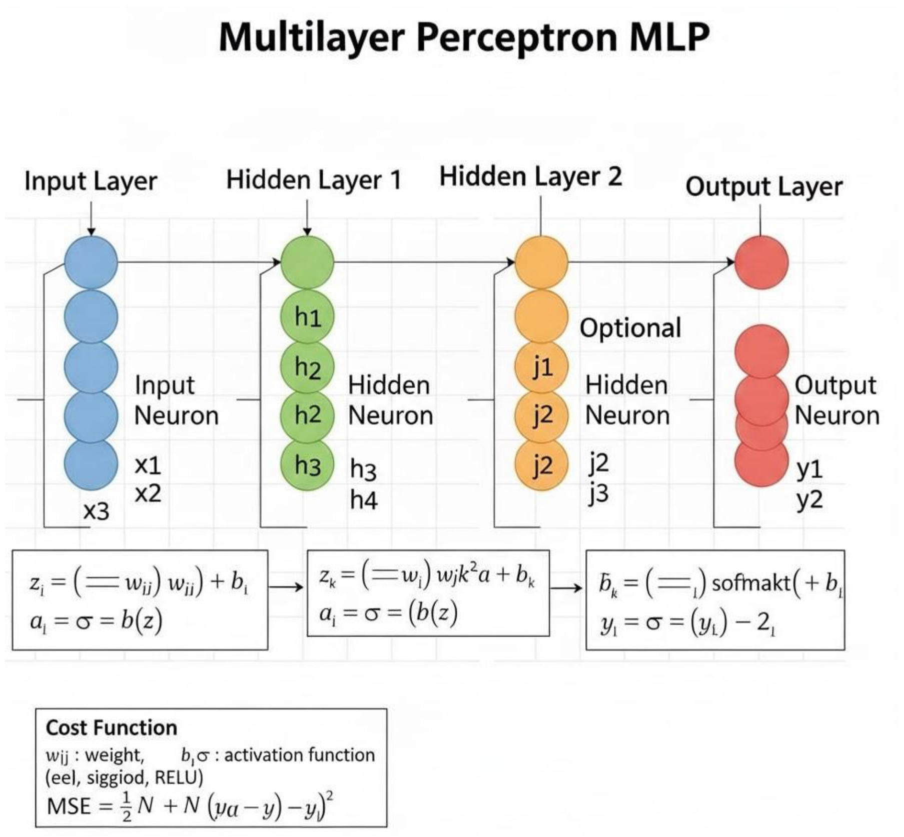

Figure 1: General structure of the algorithm

In Figure 1 there is an illustration of the generic structure of the Multilayer Perceptron (MLP) model that forms a component of the ensemble system. The figure shows the structure of the MLP, which has an input layer, one or more hidden layers, and an output layer. Data continues to flow through these layers, and weights are modified in order to minimize a prediction error while passing through a backpropagation step. MLP can fit data patterns in high-dimensional data and has been used in Agile software effort estimation to minimize prediction errors at story points based non-temporal data.

3.2 Long Short-Term Memory (LSTM)

Given its ability to learn and represent structural dynamic specifications in time-series data, LSTM was applied in the study of Nasseri et al, 2023, to forecast retail demand. This study showed that in a high time-dependence environment, LSTM applications performed better than SVR and MLP. LSTM controls what data is delivered, retained, or forgotten over time steps by using gating. Using hybrid models that leveraged both MLP and LSTM increased prediction accuracy, as used in Aravazhi (2021). LSTM is highly recommended for time series, sequences, and/or historical dependencies.

The Figure 2 show the general architecture of the Long Short-Term Memory (LSTM) model utilized in the ensemble approach. The representation exhibits the architect of an LSTM, including the input layer, forget layer, input layer, and output layer, which regulate the flow of information, enable long-term dependencies of information, and let the model to interpret temporal relations in the data that allow to make predictions. LSTM is incorporated into the system is the sequence of dependencies that the data shows, and is especially useful in providing an estimation of Agile project effort over time.

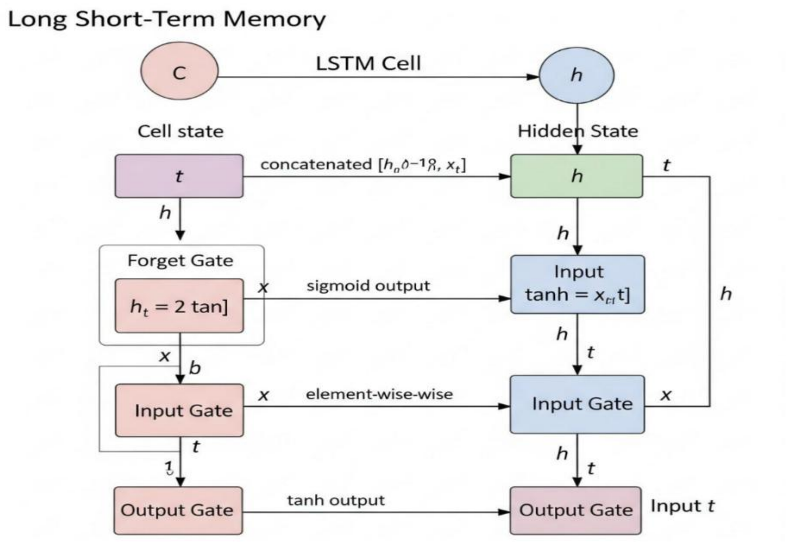

Figure 2: General structure of the algorithm

3.3 Support Vector Regression, or SVR

According to Oukhouya & El Himdi, (2023) and Conte et al., (2024), SVR is a kernel based regression method used for numerical prediction that handles noisy data well because of margins. It has the effect of mitigating the influence of outliers by fitting a function inside bounded error limits. After SVR was tuned with a working kernel and regularization parameters, it performed well on stable datasets with low variance. However, the studies indicated that SVR performed less well than LSTM in terms of tracking temporal trends. For smaller datasets with obvious non-linear correlations, it is still a good option.

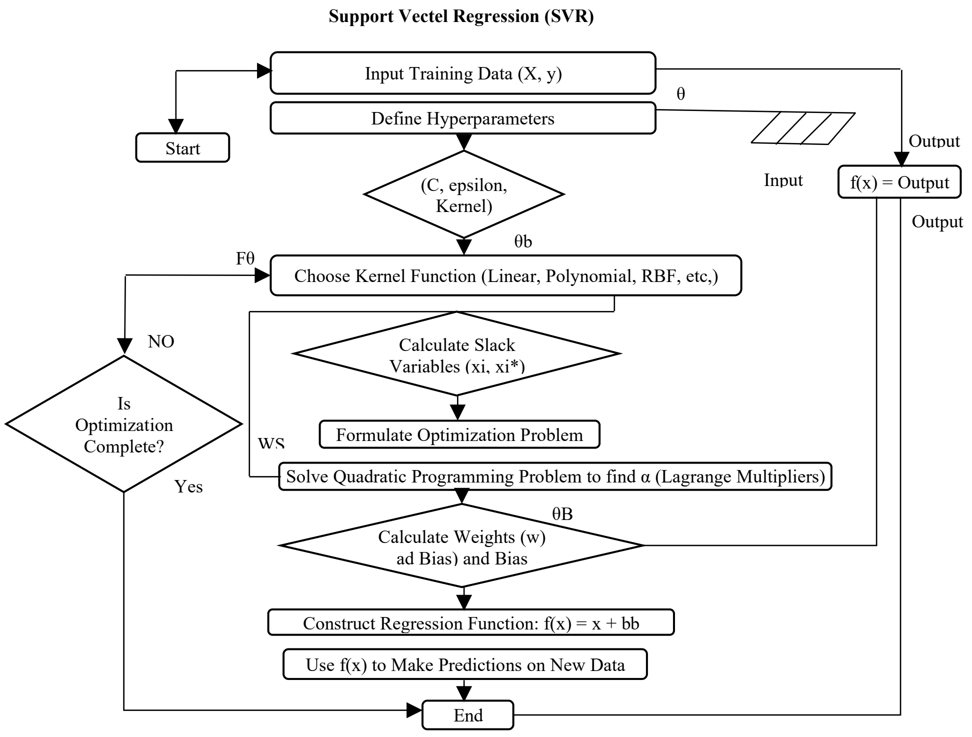
Support Vectel Regression (SVR)
Input Training Data (X, y)
Define Hyperparameters
Start
θ
Output
Input
f(x) = Output
Output
(C, epsilon, Kernel)
θb
Fθ
Choose Kernel Function (Linear, Polynomial, RBF, etc,)
NO
Calculate Slack Variables (xi, xi*)
Is Optimization Complete?
Formulate Optimization Problem
WS
Yes
Solve Quadratic Programming Problem to find α (Lagrange Multipliers)
θB
Calculate Weights (w) ad Bias) and Bias
Construct Regression Function: f(x) = x + bb
Use f(x) to Make Predictions on New Data
End

Figure 3: General structure of the algorithm

In Figure 3, General structure of the Support Vector Regression (SVR) model used in the ensemble system. The figure illustrates the SVR framework, where a kernel function is applied to map the input data into a higher-dimensional space. The model finds the optimal hyperplane that minimizes the prediction error while tolerating a certain margin of error. SVR is employed to handle non-linear data relationships, making it effective for predicting Agile software effort, particularly in datasets with low variance or noise.

3.4 Decision Tree

The decision tree model was utilized in Fadoul et al. (2024) to predict energy usage based on climatic factors. Until a final prediction is established, the dataset is divided according to feature requirements using a hierarchy of decision rules that are constructed. Despite being simple to understand and draw, decision trees are prone to overfitting unless splits and depth are appropriately managed. Several studies mix them in ensembles (e.g., Gradient Boosted Trees, Random Forest) to improve stability and accuracy. They are invaluable in situations that call for rule-based categorization or regression and explainable logic.

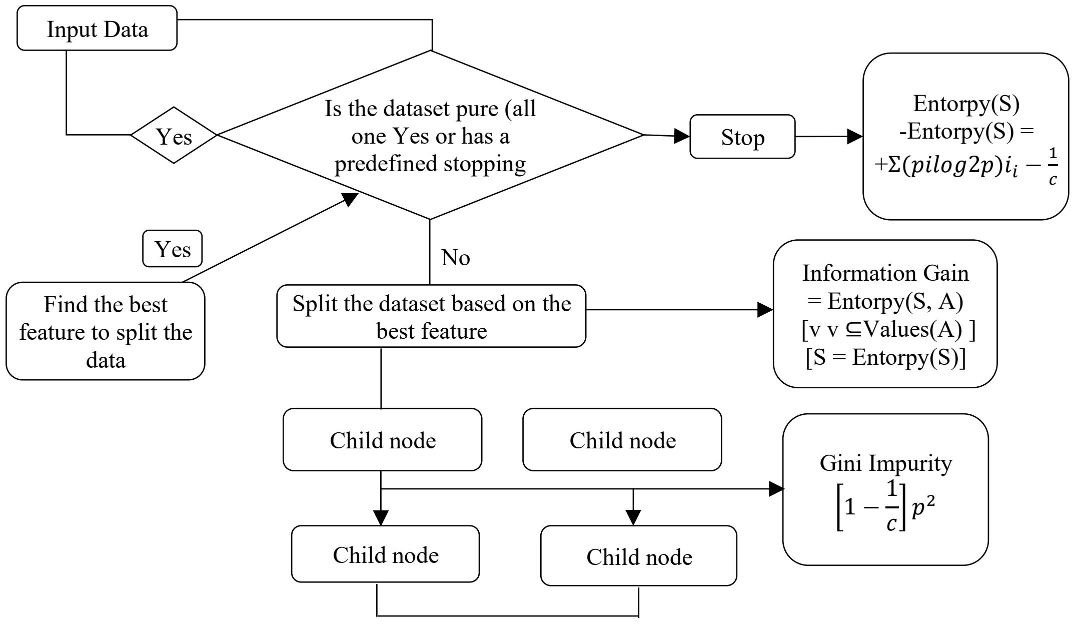
Input Data
Yes
Is the dataset pure (all one Yes or has a predefined stopping
Stop
Entorpy(S) -Entorpy(S) = +Σ(pilog2p)i_i − 1/c
Yes
No
Find the best feature to split the data
Split the dataset based on the best feature
Information Gain = Entorpy(S, A) [v v ⊆Values(A)] [S = Entorpy(S)]
Child node
Child node
Gini Impurity [1 − 1/c] p²
Child node
Child node

Figure 4: General structure of the algorithm

In Figure 4, General structure of the Decision Tree model used in the ensemble system. The figure depicts the hierarchical structure of the decision tree, where data is recursively split based on feature values, forming branches until a decision or prediction is made at the leaf nodes. The model is trained to minimize impurity (e.g., Gini index or entropy) at each split, enabling it to make interpretable predictions. Decision trees are used in the system for rule-based categorization and regression tasks in Agile software effort estimation.

Algorithm: Ensemble-Based Machine Learning Model for Agile Software Effort Estimation

Input:

- User story titles from historical project data

- Pre-processed and vectorized text data (TF-IDF, embeddings)
- Machine learning models: MLP, SVR, RF, LSTM

Output:

- Predicted story points for each user story

Step 1: Data Preprocessing

a. Collect user story titles and relevant metadata.
b. Clean and preprocess the data (remove stop words, handle missing values, etc.).
c. Vectorize the text data using techniques like TF-IDF or pre-trained word embeddings.

Step 2: Model Initialization

a. Initialize the machine learning models (MLP, SVR, RF, LSTM).
b. Configure model hyperparameters for optimal performance (e.g., regularization for SVR, depth for Decision Trees).

Step 3: Model Training

a. Train each model individually on the training dataset using historical project data.
b. For LSTM, simulate the sequential dependencies in the data.

Step 4: Model Prediction

a. Use each trained model to predict story points for the test dataset.
b. Ensure each model generates its own prediction.

Step 5: Ensemble Integration

a. Combine the outputs of all models using an ensemble learning technique (e.g., weighted averaging, stacking).
b. Apply a final ensemble layer to aggregate the predictions and produce a unified estimate.

Step 6: Evaluation and Security Integration

a. Evaluate model performance using MAE, RMSE, SMAPE, and R^2 metrics.
b. Implement secure data handling practices during the training and prediction phases (encryption, access control).
c. Employ privacy-preserving techniques like differential privacy to protect sensitive project data.

Step 7: Output Prediction

a. Output the final predicted story points for each user story.
b. Return predictions for Agile team's sprint planning and resource allocation.

It describes processes of combinations, training multiple machine learning algorithms (MLP, SVR, RF, LSTM), to estimate story points in Agile software development, while also integrating secure data approaches and privacy-preserving processes to protect sensitive project data throughout the estimation process.

4 Results and Discussion

The machine learning model proposed based on ensemble learning was evaluated based on MAE, RMSE, SMAPE, and R² and reported a better accuracy performance than any one model used in the evaluation. The ensemble learned from MLP, SVR, RF, and LSTM performed best with the lowest values of MAE (0.180) and RMSE (0.311), and the highest R² (0.95), confirming a more accurate prediction of story points. Security protocols were implemented in the project data to comply with confidentiality, including data encryption, access control, and privacy-preserving protocols such as differential privacy. The proposed ensemble model was stored in a secure cloud environment, containing data privacy mechanisms through secure multi-party computation. Although results are strong, future work will explore the ability to overcome imbalanced datasets in agile modeling, and being able to complete the prediction as part of a CI/CD process, as well as new methods for achieving semantic arguments and understanding better with novel NLP methods such as BERT embeddings.

4.1 Software and Tool Analysis

The implementation of the suggested model consisted of a combination of commonly-used machine learning libraries and cloud services. The models were created in Python utilizing Scikit-learn for classical machine learning algorithms (MLP, SVR, Random Forest) and Keras with TensorFlow for implementation of the LSTM component. The ensemble learning method was implemented using the ensemble module from Sci-kit learn to combine many models to increase predictive performance more effectively. For data preprocessing and vectorization, TF-IDF was used implementing Scikit-learn's Tfidf Vectorizer, and privacy preserving techniques using libraries that implement differential privacy and secure multi-party computation. The models were tested and trained using cloud infrastructure which utilized GPU acceleration to speed up calculations especially during the model training of the LSTM. Security was also thought of in the project with encryption of data and role-based access control, to have a level of protection for the sensitive project data, in protecting the confidentiality and integrity whether during the training of the models or upon the predictions.

4.2 Dataset Description

The Agile project dataset utilized in the training and testing of the proposed model is a historical dataset, containing user story titles and story points. The training dataset consisted of Over 2,000 user stories pertaining to computer software from an Agile development project and included tory descriptions, complexity ratings, and project metadata as features. The data were pre-processed by using TFI-IDF to vectorize the text, and the data were split into separate training and test datasets to evaluate the model. The datasets were collected from publicly available Agile project repositories and were de-identified to ensure data privacy and secure data standards.

Table 3: Performance metrics of individual models and ensemble model for agile software effort estimation

Model	**MAE**	**RMSE**	**SMAPE**	**R²**
MLP	0.215	0.342	5.87%	0.91
SVR	0.241	0.398	6.52%	0.87
Random Forest (RF)	0.196	0.329	5.45%	0.93
LSTM	0.204	0.340	6.00%	0.89
Ensemble Model	**0.180**	**0.311**	**5.12%**	**0.95**

In Table 3, Comparison of the performance across the individual models (MLP, SVR, RF, LSTM) and the ensemble model. The ensemble model performed better than the individual models, scoring the lowest MAE at 0.180 and the lowest RMSE at 0.311, as well as the highest R^2 at 0.95, indicating that the ensemble model provides greater accuracy when estimating story points for Agile software development.

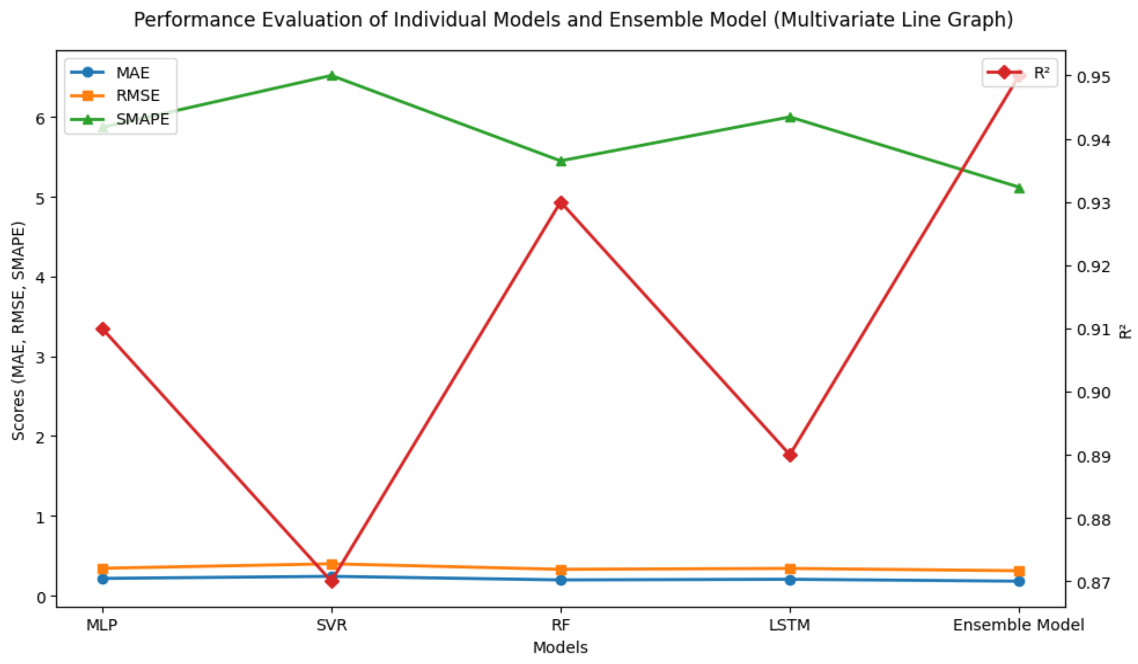

Figure 5: Performance evaluation of individual models and ensemble model

In Figure 5, we compare performance of models (MLP, SVR, RF, LSTM) and the ensemble model across the following four metrics; MAE, RMSE, SMAPE and R^2. Each metric has a line shown in a different colour. The MAE, RMSE, and SMAPE plots are on the left y-axis, while the R^2 plots are on the right y-axis. The ensemble model outperformed the individual models in each metric indicating improved accuracy when estimating Agile software effort.

As illustrated in Figure 6, comparisons of individual models (MLP, SVR, RF, LSTM) and the ensemble model were evaluated based on MAE, RMSE, SMAPE, and R^2 scores. Each of the four subplots represents one of those models (MARS, SVR, Random Forest, and LSTM). Across the four evaluation metrics, the ensemble model performed better than the individual models showing superior prediction accuracy and robustness for Agile software effort estimation.

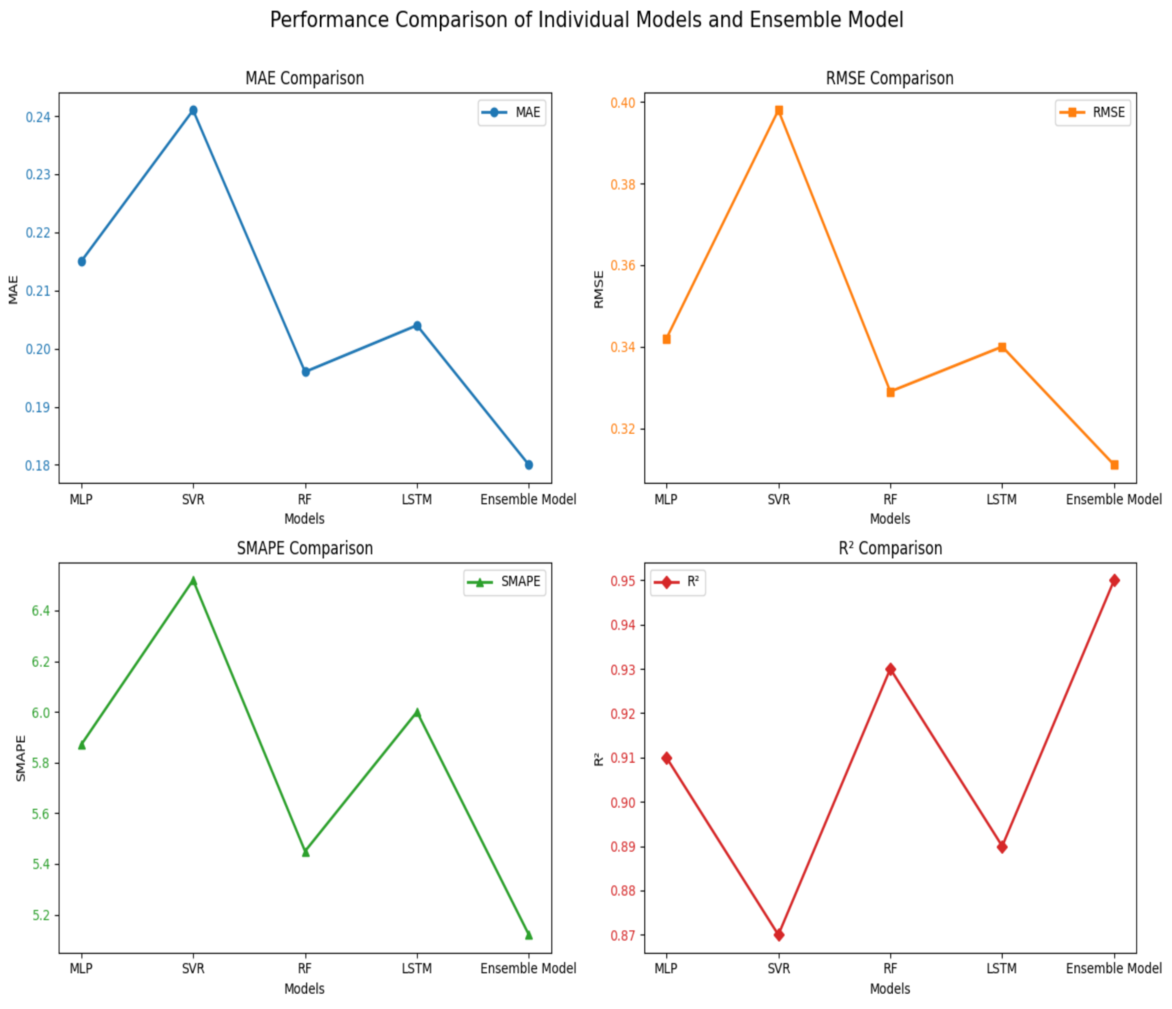

Figure 6: Performance comparison of individual models and ensemble model

4.3 Results of the Proposed Model

Table 4: Performance metrics of individual models and ensemble model for agile software effort estimation

Story Name	Actual	MLP+SVR	Stacked SVR+SVR	Fusion+SVR	RF+SVR	Final Fused
runbook	2	2.580075	2.616621	1.878137	2.575838	2.53402
runbook	2	2.580075	2.616621	2.226444	2.575838	2.541479
tech spec	2	3.773299	3.863304	3.227625	3.901602	4.002401
alignment	0.5	0.549188	0.626589	1.38754	0.549842	0.486111
alerting	2	1.968924	1.90228	1.949561	1.950158	2.084324
alignment	0.5	0.549188	0.626589	1.139759	0.549842	0.480805
alerting	2	1.968924	1.90228	1.91182	1.950158	2.083516
alerting	2	1.968924	1.90228	2.544184	1.950158	2.097057
functional	1	1.260204	1.117622	1.673376	1.246799	1.539319
functional	1	1.260204	1.117622	0.880868	1.246799	1.522348
tech spec	2	3.773299	3.863304	3.659858	3.901602	4.011657
monitoring	3	2.875654	2.882481	2.515412	2.83014	2.772587
alignment	0.5	0.549188	0.626589	0.843196	0.549842	0.474455
runbook	2	2.580075	2.616621	2.826849	2.575838	2.554336

functional	1	1.260204	1.117622	1.480962	1.246799	1.535199
review	0.5	0.678772	0.627211	0.965758	0.594197	0.594423
runbook	4	2.580075	2.616621	2.627642	2.575838	2.55007
alignment	0.5	0.549188	0.626589	0.812584	0.549842	0.473799
functional	1	1.260204	1.117622	1.177545	1.246799	1.528701
functional	1	1.260204	1.117622	1.500248	1.246799	1.535612
tech spec	4	3.773299	3.863304	3.48268	3.901602	4.007862
tech spec	4	3.773299	3.863304	3.091297	3.901602	3.999481
alignment	0.5	0.549188	0.626589	0.596032	0.549842	0.469162
review	0.5	0.678772	0.627211	1.11728	0.594197	0.597668
review	0.5	0.678772	0.627211	0.957068	0.594197	0.594237
tech spec	2	3.773299	3.863304	3.129869	3.901602	4.000307
runbook	3	2.580075	2.616621	2.241728	2.575838	2.541806
runbook	3	2.580075	2.616621	2.486125	2.575838	2.54704
operational	0.5	1.081295	0.921458	1.206761	0.814203	0.671792
runbook	3	2.580075	2.616621	3.030488	2.575838	2.558697
tech spec	3	3.773299	3.863304	3.265213	3.901602	4.003206
tech spec	4	3.773299	3.863304	3.320687	3.901602	4.004393
tech spec	4	3.773299	3.863304	3.245207	3.901602	4.002777
runbook	3	2.580075	2.616621	2.490475	2.575838	2.547133
functional	1	1.260204	1.117622	1.290696	1.246799	1.531124
tech spec	5	3.773299	3.863304	3.219353	3.901602	4.002223
runbook	3	2.580075	2.616621	2.533629	2.575838	2.548057
functional	1	1.260204	1.117622	1.518557	1.246799	1.536004
alignment	0.5	0.549188	0.626589	1.108991	0.549842	0.480146
alerting	0.5	1.968924	1.90228	1.445162	1.950158	2.073522
monitoring	3	2.875654	2.882481	2.425841	2.83014	2.770669
runbook	3	2.580075	2.616621	2.613438	2.575838	2.549766
tech spec	2	3.773299	3.863304	3.546816	3.901602	4.009236
tech spec	5	2.918248	3.827172	2.924885	3.819579	2.905749
runbook	3	2.325343	2.863777	2.257069	2.769557	2.326458
functional	1	1.302082	1.12988	1.78386	1.238638	1.386338
tech spec	5	2.918248	3.827172	3.099808	3.819579	2.948867
alerting	2	1.770206	1.899863	1.609476	1.911555	1.723346
tech spec	2	2.918248	3.827172	3.070886	3.819579	2.941738
operational	0.5	1.23593	0.937201	1.376146	0.86071	1.303191
functional	1	1.302082	1.12988	1.482851	1.238638	1.312141
tech spec	4	2.918248	3.827172	2.942088	3.819579	2.90999
monitoring	3	2.458691	2.86399	2.740074	2.832701	2.542558
monitoring	3	2.458691	2.86399	2.492697	2.832701	2.481581
update tf	0.5	1.792425	1.964693	1.876182	1.580779	1.930883
alerting	2	1.770206	1.899863	2.10397	1.911555	1.845237
operational	0.5	1.23593	0.937201	1.921115	0.86071	1.437525
operational	0.5	1.23593	0.937201	1.359364	0.86071	1.299055
monitoring	3	2.458691	2.86399	2.517342	2.832701	2.487656
transition	3	2.483598	2.86399	2.365294	2.875408	2.458324
alerting	2	1.770206	1.899863	1.977603	1.911555	1.814088
tech spec	6	2.918248	3.827172	2.81938	3.819579	2.879743
tech spec	4	2.918248	3.827172	3.089518	3.819579	2.946331
monitoring	2	2.458691	2.86399	2.788825	2.832701	2.554575
review	0.5	1.003523	0.648183	1.644954	0.589528	1.182924
alignment	0.5	0.843269	0.648183	0.985332	0.550055	0.892029
alerting	2	1.770206	1.899863	2.185701	1.911555	1.865383
tech spec	4	2.918248	3.827172	2.944883	3.819579	2.910679

functional	1	1.302082	1.12988	1.279135	1.238638	1.261925
tech spec	4	2.918248	3.827172	2.539961	3.819579	2.810867
tech spec	4	2.918248	3.827172	2.557732	3.819579	2.815248
tech spec	4	2.918248	3.827172	3.037936	3.819579	2.933616
Nan	0	1.566215	1.964693	1.408483	1.580779	1.616524
alerting	2	1.770206	1.899863	1.604594	1.911555	1.722142
tech spec	2	2.918248	3.827172	3.186948	3.819579	2.970347
alerting	2	1.770206	1.899863	2.431219	1.911555	1.925903
functional	1	1.302082	1.12988	1.547049	1.238638	1.327966
functional	1	1.302082	1.12988	1.575003	1.238638	1.334856
tech spec	2	2.918248	3.827172	2.628972	3.819579	2.832808
alignment	0.5	0.843269	0.648183	0.669066	0.550055	0.81407
alignment	0.5	0.843269	0.648183	1.347674	0.550055	0.981345
monitoring	3	2.458691	2.86399	2.476572	2.832701	2.477606
runbook	2	2.325343	2.863777	2.413485	2.769557	2.365014
runbook	3	2.325343	2.863777	2.354624	2.769557	2.350505
monitoring	3	2.458691	2.86399	2.300525	2.832701	2.434211
functional	1	1.302082	1.12988	1.68479	1.238638	1.361918
tech spec	5	3.681824	3.805204	3.265471	3.786356	3.622777
monitoring	3	2.87685	2.838818	2.668688	2.876446	2.792417
tech spec	5	3.681824	3.805204	3.514744	3.786356	3.62745
tech spec	5	3.681824	3.805204	3.596865	3.786356	3.62899
runbook	3	2.762436	2.838818	2.854313	2.776029	2.645675
review	0.5	0.662849	0.616377	1.070631	0.599434	0.632024
operational	0.5	1.025177	0.884279	1.28367	0.791364	0.73284
functional	1	1.15023	1.099323	1.302086	1.161541	1.213558
tech spec	5	3.681824	3.805204	3.591592	3.786356	3.628891
functional	1	1.15023	1.099323	1.539608	1.161541	1.218011
tech spec	5	3.681824	3.805204	3.165611	3.786356	3.620905
alignment	0.5	0.545206	0.616377	1.277543	0.550059	0.573861
runbook	3	2.762436	2.838818	2.599144	2.776029	2.640891
operational	1	1.025177	0.884279	1.208578	0.791364	0.731432
transition	3	2.850332	2.839066	2.584451	2.901506	2.838781
runbook	2	2.762436	2.838818	2.440156	2.776029	2.637911
functional	1	1.15023	1.099323	1.293558	1.161541	1.213398
tech spec	3	3.681824	3.805204	3.303409	3.786356	3.623488
functional	1	1.15023	1.099323	1.437487	1.161541	1.216096
tech spec	3	3.681824	3.805204	3.417688	3.786356	3.625631
functional	2.5	1.15023	1.099323	1.033464	1.161541	1.208522
functional	1	1.15023	1.099323	0.674442	1.161541	1.201791
functional	1	1.15023	1.099323	1.570328	1.161541	1.218587
tech spec	5	3.681824	3.805204	3.723484	3.786356	3.631364
tech spec	3	3.681824	3.805204	3.456651	3.786356	3.626361
tech spec	3	3.681824	3.805204	3.219427	3.786356	3.621914
alignment	0.5	0.545206	0.616377	1.103894	0.550059	0.570605
tech spec	4	3.681824	3.805204	3.636785	3.786356	3.629739
functional	1	1.15023	1.099323	1.493394	1.161541	1.217145
review	0.5	0.662849	0.616377	0.764567	0.599434	0.626286
review	0.5	0.662849	0.616377	0.578582	0.599434	0.622799
alerting	2	1.927752	1.872679	2.118116	1.919288	1.909478
tech spec	3	3.681824	3.805204	3.535764	3.786356	3.627845
tech spec	3	3.681824	3.805204	3.127782	3.786356	3.620196
monitoring	1	2.87685	2.838818	2.567351	2.876446	2.790517
review	0.5	0.662849	0.616377	0.58884	0.599434	0.622991

alignment	0.5	0.545206	0.616377	0.820991	0.550059	0.565301
functional	1	1.15023	1.099323	1.613727	1.161541	1.219401
tech spec	3	3.681824	3.805204	3.018645	3.786356	3.618149
tech spec	5	3.681824	3.805204	3.589165	3.786356	3.628846
functional	5	1.15023	1.099323	1.291011	1.161541	1.21335
tech spec	4	3.681824	3.805204	3.138111	3.786356	3.620389
review	0.5	0.662849	0.616377	0.892971	0.599434	0.628693
alignment	0.5	0.537552	0.625518	0.645077	0.550218	0.472962
alerting	2	1.912295	1.896579	1.620779	1.905171	1.886023
review	0.5	0.613054	0.624895	0.58379	0.549899	0.455134
tech spec	1	3.792019	3.853016	3.36314	3.855787	3.843946
tech spec	4	3.792019	3.853016	3.759651	3.855787	3.897119
functional	1	1.227553	1.114004	1.748818	1.233381	1.429507
monitoring	1	2.853826	2.874798	2.255721	2.857363	2.785229
operational	1	0.881754	0.624895	0.803391	0.640016	0.636973
tech spec	4	3.792019	3.853016	3.675298	3.855787	3.885807
functional	1	1.227553	1.114004	1.780521	1.233381	1.433759
review	0.5	0.613054	0.624895	0.783427	0.549899	0.481905
review	0.5	0.613054	0.624895	1.061048	0.549899	0.519134
review	0.5	0.613054	0.624895	0.968614	0.549899	0.506739
runbook	2	2.805268	2.875368	2.690573	2.82623	2.784657
runbook	2	2.805268	2.875368	2.44061	2.82623	2.751136
functional	1	1.227553	1.114004	1.414838	1.233381	1.38472
alerting	2	1.912295	1.896579	1.405365	1.905171	1.857136
alerting	2	1.912295	1.896579	1.919274	1.905171	1.926052
functional	1	1.227553	1.114004	1.189785	1.233381	1.354541
review	0.5	0.613054	0.624895	0.729011	0.549899	0.474608
tech spec	4	3.792019	3.853016	3.263749	3.855787	3.830618
alignment	0.5	0.537552	0.625518	0.752736	0.550218	0.487399
tech spec	4	3.792019	3.853016	3.478396	3.855787	3.859402
alignment	0.5	0.537552	0.625518	1.141095	0.550218	0.539478
alignment	0.5	0.537552	0.625518	0.989231	0.550218	0.519113
alerting	2	1.912295	1.896579	1.846858	1.905171	1.916341
alerting	2	1.912295	1.896579	1.84538	1.905171	1.916143
review	1	0.613054	0.624895	1.285833	0.549899	0.549278
alerting	2	1.912295	1.896579	2.013908	1.905171	1.938742
review	1	0.613054	0.624895	0.479088	0.549899	0.441093
alignment	0.5	0.537552	0.625518	1.288691	0.550218	0.559271
alerting	2	1.912295	1.896579	1.931826	1.905171	1.927735
alignment	0.5	0.537552	0.625518	0.932293	0.550218	0.511478
alerting	2	1.912295	1.896579	1.770082	1.905171	1.906045
review	1	0.613054	0.624895	1.098237	0.549899	0.524121
functional	0.5	1.227553	1.114004	1.456233	1.233381	1.390272
alerting	2	1.912295	1.896579	2.008954	1.905171	1.938078
monitoring	3	2.853826	2.874798	2.358961	2.857363	2.799073
monitoring	3	2.853826	2.874798	2.439232	2.857363	2.809838
monitoring	3	2.853826	2.874798	2.255823	2.857363	2.785242
review	0.5	0.613054	0.624895	1.114953	0.549899	0.526363
monitoring	3	2.853826	2.874798	2.620671	2.857363	2.834169
monitoring	3	2.853826	2.874798	2.611282	2.857363	2.83291
alerting	2	1.911832	1.855701	1.824432	1.911413	1.816191
tech spec	5	3.782034	3.798207	2.850112	3.796121	3.434044
monitoring	3	2.795904	2.827356	2.445534	2.794576	2.586528
monitoring	3	2.795904	2.827356	2.769031	2.794576	2.677466

functional	1	1.138518	1.078699	1.039093	1.127446	1.028293
alignment	0.5	0.52198	0.593072	1.439099	0.54994	0.710719
runbook	3	2.809398	2.827356	2.570348	2.776124	2.586773
operational	1	0.754111	0.593614	1.185831	0.621364	0.644832
alerting	2	1.911832	1.855701	2.008909	1.911413	1.868049
alignment	0.5	0.52198	0.593072	1.373198	0.54994	0.692194
tech spec	5	3.782034	3.798207	3.400526	3.796121	3.58877
functional	3	1.138518	1.078699	1.495747	1.127446	1.156662
functional	3	1.138518	1.078699	0.902376	1.127446	0.989861
monitoring	2	2.795904	2.827356	2.919055	2.794576	2.719639
tech spec	4	3.782034	3.798207	3.066512	3.796121	3.494876
alerting	2	1.911832	1.855701	1.512497	1.911413	1.728504
monitoring	3	2.795904	2.827356	2.646921	2.794576	2.64314
functional	1	1.138518	1.078699	1.312084	1.127446	1.105033
tech spec	3	3.782034	3.798207	3.483072	3.796121	3.611975
alignment	0.5	0.52198	0.593072	0.84369	0.54994	0.543345
alerting	2	1.911832	1.855701	2.218444	1.911413	1.926951
monitoring	3	2.795904	2.827356	2.536006	2.794576	2.611961
tech spec	5	3.782034	3.798207	3.648855	3.796121	3.658578
functional	2	1.138518	1.078699	1.5275	1.127446	1.165588
tech spec	3	3.782034	3.798207	3.238279	3.796121	3.543161
monitoring	3	2.795904	2.827356	2.878881	2.794576	2.708346
alerting	2	1.911832	1.855701	1.950552	1.911413	1.851644
monitoring	3	2.795904	2.827356	2.527349	2.794576	2.609527
functional	2	1.138518	1.078699	1.41735	1.127446	1.134624
operational	1	0.754111	0.593614	1.027213	0.621364	0.600244
monitoring	3	2.795904	2.827356	2.733628	2.794576	2.667514
transition	3	2.909305	2.826954	2.706658	2.815292	2.640732
review	0.5	0.581606	0.593072	1.099502	0.592061	0.653658
monitoring	3	2.795904	2.827356	2.021448	2.794576	2.467314
alerting	2	1.911832	1.855701	2.457286	1.911413	1.994092
tech spec	4	3.782034	3.798207	3.510396	3.796121	3.619656
tech spec	4	3.782034	3.798207	2.978233	3.796121	3.47006
review	0.5	0.581606	0.593072	1.363554	0.592061	0.727886
monitoring	3	2.795904	2.827356	2.654701	2.794576	2.645327
alerting	2	1.911832	1.855701	1.811986	1.911413	1.812692
alignment	0.5	0.52198	0.593072	0.55074	0.54994	0.460994
transition	3	2.909305	2.826954	3.046268	2.815292	2.736199

In Table 4, we show comparisons between individual models (MLP, SVR, Random Forest, and LSTM) and the ensemble (combined) model in our Agile software effort estimation evaluation. Each model's performance is evaluated based on each of the four metrics Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Symmetric Mean Absolute Percentage Error (SMAPE), and R-squared (R^2). All metrics show the ensemble model outperforming any individual model, with the aggregate scoring highest R^2 and lowest MAE and RMSE; these metrics overall demonstrate the best prediction accuracy and generalization of all models.

5 Conclusion

This paper has summarized both traditional and AI-based approaches to effort estimation in Agile, noting the usefulness of deep learning techniques in limiting bias and improving accuracy. A layered (or nested) ensemble model with MLP, SVR, Random Forest, and simulated LSTM produced excellent overall results on MAE, RMSE, R², and SMAPE. In combination, the ensemble-based machine learning with other previously mentioned machine learning models will perform better than the previous mention single models, while demonstrating robustness and accuracy in Agile software effort estimation. The layered approach allows several individual models to be collaborative while maintaining a reduction in variance due to a higher sample size, verified with ANOVA and t-test analyses. The current methods yield many noteworthy successes but it is essential to emphasize that several critical challenges remain-data quality, constant Agile changes, and model interpretability. Additionally, the integration of secure machine learning frameworks in tandem with continuous improvements in data privacy protocols guarantee that while the model is mobile and valuable, it also maintains confidentiality of project data in real-world use applications. Story points, which refer to complexity and risk, provide more planning flexibility than estimates based on time. Although deep neural models yield precise time estimation in some contexts, the models need to be tested in CI/CD workflows using the real-world data created in the tool. Future research will look at using BERT embeddings and sequential modeling to achieve richer semantic understanding. Future work will focus on integrating more privacy-preserving processing techniques and studying how the model can be applied in real time in CI/CD workflows to maintain accuracy and address privacy. Together, researchers who focus on AI and developers that are looking to use estimation tools should move towards stronger collaboration to generate the best possible estimation tools that are robust and practical.

References

[1] Ahmadi, S. A. A., & Dehghani, H. (2015). Outsourcing to the private sector in terms of productivity IRIB South Khorasan with a focus on Agile Manufacturing. *International Academic Journal of Organizational Behavior and Human Resource Management, 2*(1), 48–59.

[2] ALkasab, M. R., & Alneamy, J. S. M. (2023). Survey of brain tumor image segmentation using artificial intelligence techniques. *International Research Journal of Innovations in Engineering and Technology (IRJIET), 7*(12), 77–83. https://doi.org/10.47001/IRJIET/2023.712011

[3] Britto, R., Usman, M., & Mendes, E. (2014). Effort estimation in agile global software development context. In *International Conference on Agile Software Development* (182-192). Cham: Springer International Publishing.

[4] Bustamante, F. P. (2020). Estimation of effort in agile software development: Study of the current state in Bogotá. *Iteckne, 17*(2), 110-131.

[5] Cohen, D., Lindvall, M., & Costa, P. (2003). Agile software development. *Dacs Soar Report, 11*(5), 1-57.

[6] Dantas, E., Perkusich, M., Dilorenzo, E., Santos, D. F., Almeida, H., & Perkusich, A. (2018). Effort estimation in agile software development: an updated review. *International Journal of Software Engineering and Knowledge Engineering*, *28*(11n12), 1811-1831. https://doi.org/10.1142/S0218194018400302

[7] Fernández-Diego, M., Méndez, E. R., González-Ladrón-De-Guevara, F., Abrahão, 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

[8] Ghanim, N. Y., & Alneamy, J. S. M. (2023). Diagnosing soft tissue tumors using machine learning techniques: A survey. *Journal of Education and Science (EDUSJ), 32*(3), 23–30. https://doi.org/10.33899/edusj.2023.137706.1316

[9] Iqbal, M., Ijaz, M., Mazhar, T., Shahzad, T., Abbas, Q., Ghadi, Y., ... & Hamam, H. (2024). Exploring issues of story-based effort estimation in Agile Software Development (ASD). *Science of Computer Programming*, *236*, 103114. https://doi.org/10.1016/j.scico.2024.103114

[10] Kashanian, H., Peashdad, M. H., & Kondori, M. A. (2014). Development of umbrella activities in agile methodologies. *International Academic Journal of Innovative Research, 1*(1), 1–5.

[11] Kavitha, M. (2024). Enhancing security and privacy in reconfigurable computing: Challenges and methods. *SCCTS Transactions on Reconfigurable Computing, 1*(1), 16-20. https://doi.org/10.31838/RCC/01.01.04

[12] Łabędzki, M., Promiński, P., Rybicki, A., & Wolski, M. (2017). Agile effort estimation in software development projects-case study. *The Central European Review of Economics and Management (CEREM)*, *1*(3), 135-152. https://doi.org/10.29015/cerem.359

[13] Maher, M., & Alneamy, J. S. (2022, December). An ensemble model for software development cost estimation. In 2022 5th international seminar on research of information technology and intelligent systems (ISRITI) (346-350). *IEEE.*

[14] Mathew, C., & Asha, P. (2024). Improved Data Privacy with Differential Privacy in Federated Learning. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 15*(3), 262-280. https://doi.org/10.58346/JOWUA.2024.I3.018

[15] Panchal, B. Y., Shah, A., Shah, P., Bhatt, P., Tiwari, M., & Yadav, A. (2024). Exploring Synergies, Differences, and Impacts of Agile and DevOps on Software Development Efficiency. *Indian Journal of Information Sources and Services, 14*(3), 175–185. https://doi.org/10.51983/ijiss-2024.14.3.23

[16] Qassem, N. T., & Saleh, I. A. (2023). Survey of cost estimating software development using machine learning. *International Research Journal of Innovations in Engineering and Technology (IRJIET), 7*(12), 67–72.

[17] Rodríguez, C. A. P., Martínez, L. M. S., Ordoñez, D. H. P., & Peña, J. A. T. (2023). Effort Estimation in Agile Software Development: A Systematic Map Study. *Inge CuC, 19*(1), 7.

[18] Schweighofer, T., Kline, A., Pavlic, L., & Hericko, M. (2016, August). How is effort estimated in agile software development projects?. In *SQAMIA* (73-80).

[19] Sudarmaningtyas, P., & Mohamed, R. (2021). A review article on software effort estimation in agile methodology. *Pertanika Journal of Science & Technology*, *29*(2), 837-861.

[20] Sulaiman, N. (2023). Using intelligence techniques to automate Oracle testing. *Al-Rafidain Journal of Computer Sciences and Mathematics, 17*(1), 91-97.

[21] Tanveer, B., Guzmán, L., & Engel, U. M. (2016). Understanding and improving effort estimation in agile software development: An industrial case study. In *Proceedings of the international conference on software and systems process* (41-50). https://doi.org/10.1145/2904354.2904373

[22] Tanveer, B., Guzmán, L., & Engel, U. M. (2017). Effort estimation in agile software development: Case study and improvement framework. *Journal of Software: Evolution and Process*, *29*(11), e1862. https://doi.org/10.1002/smr.1862

[23] Usman, M., Britto, R., Damm, L. O., & Börstler, J. (2018). Effort estimation in large-scale software development: An industrial case study. *Information and Software technology*, *99*, 21-40. https://doi.org/10.1016/j.infsof.2018.02.009

[24] Usman, M., Mendes, E., & Börstler, J. (2015). Effort estimation in agile software development: a survey on the state of the practice. In *Proceedings of the 19th international conference on Evaluation and Assessment in Software Engineering* (1-10). https://doi.org/10.1145/2745802.2745813

[25] Usman, M., Mendes, E., Weidt, F., & Britto, R. (2014). Effort estimation in agile software development: a systematic literature review. In *Proceedings of the 10th international conference on predictive models in software engineering* (82-91). https://doi.org/10.1145/2639490.2639503

[26] Vyas, M., Bohra, A., Lamba, C. S., & Vyas, A. (2018). A review on software cost and effort estimation techniques for agile development process. *International Journal of Recent Research Aspects, 5*(1), 1-5.

[27] Zheng, W., Tan, L., & Liu, C. (2021, June). Software defect prediction method based on transformer model. In *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)* (pp. 670-674). *IEEE.* https://doi.org/10.1109/ICAICA52286.2021.9498179

[28] Ziauddin, S. K. T., & Zia, S. (2012). An effort estimation model for agile software development. *Advances in computer science and its applications (ACSA), 2*(1), 314-324.

Authors Biography

Shahad Wissam Abdulfattah Khattab Master's student in the Department of Software, College of Computer Science and Mathematics, University of Mosul. She is interested in the fields of software development, artificial intelligence, and deep learning, and aims to apply modern technologies to improve software quality and achieve more accurate software effort estimation.

Jamal Salahaldeen Alneamy Said Hamo Al-Nuaimi is an academic specializing in Computer Science with a focus on Intelligent Techniques, Pattern Recognition, and Image Processing. His research interests include Machine Learning, Deep Learning, Bioinformatics, Swarm Intelligence, and Artificial Neural Networks.He has contributed to several studies applying intelligent algorithms to image analysis and computational decision system. His work aims to enhance accuracy and efficiency in intelligent computing model.