Software Quality Journal (2025) 33:35
https://doi.org/10.1007/s11219-025-09731-6

RESEARCH

Improving story points estimation using ensemble machine learning

Zuhaimi Ahmad[1] · **Matthew M. Y. Kuo**[1]

Received: 4 December 2024 / Accepted: 22 September 2025

Abstract
Agile software development (ASD) emphasizes iterative development, continuous feedback, and team collaboration, addressing the limitations of traditional methodologies. This research explores the application of machine learning (ML) to improve story point estimation in ASD, a critical practice for planning and prioritization. Traditional methods like Planning Poker often suffer from human biases and inconsistencies, leading to unreliable estimates. This study introduces an innovative ML-based ensemble stacking technique, combining RoBERTa, a transformer model for natural language processing, with BiLSTM, a neural network adept at handling sequential data. The research involves reviewing existing ML methodologies, developing the proposed model, and evaluating its effectiveness using 21,064 data points from 14 open-source projects. The model's performance was assessed through Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE). Results show that the proposed ensemble model achieved lower MAE and MAPE, with performance improvements ranging from 4% to 32% over state-of-the-art models. While promising, the study suggests there is still room for further refinement, indicating the potential for ongoing advancements. This research contributes to the integration of ML in software engineering, offering a path toward more accurate and efficient project management.

Keywords Agile Software Development · Story Point Estimation · Machine Learning · Ensemble Stacking · RoBERTa · BiLSTM

Zuhaimi Ahmad and Matthew M.Y. Kuo contributed equally to this work.

✉ Zuhaimi Ahmad
zuhaimi@netbytesecurity.com

✉ Matthew M. Y. Kuo
matthew.kuo@aut.ac.nz

[1] Computer and Information Sciences Department, Auckland University of Technology, 6 St Paul St, 1010 Auckland, New Zealand

Published online: 13 November 2025

1 Introduction

Agile software development (ASD) methodology is a modern approach that aims to address the limitations of traditional software development methods by offering a more flexible and adaptive framework (Alsaqqa et al., 2020). Unlike traditional methodologies, which focus on upfront planning and extensive documentation, ASD encourages iterative development, continuous feedback, and collaboration between cross-functional teams and stakeholders. This iterative and collaborative approach allows team members to respond quickly to changing requirements, deliver high-quality software, and prioritise work based on customer needs and business value (Gheorghe et al., 2020).

Several frameworks are available for ASD, such as Scrum, Extreme Programming (XP), Dynamic Systems Development Method (DSDM), and Feature Driven Development (FDD) (Gheorghe et al., 2020). These frameworks provide a structure for organising and managing the development process, defining roles and responsibilities, and facilitating communication and coordination among team members (Alsaqqa et al., 2020). One of the key practices of all these frameworks is the use of user stories, which contain user requirements and feedback in the form of short and concise descriptions (Schön et al., 2017). They can vary in detail, often starting as epics that are broken down into smaller user stories during the planning phase. Before implementing these user stories, it is necessary to prioritise and estimate them (Soukaina et al., 2022). Prioritisation allows the team to identify which user stories should be implemented first based on their importance and value to the customer or end user. Similarly, estimation is necessary for the team to determine the work required to implement each prioritised user story (Schön et al., 2017; Soukaina et al., 2022).

Estimating the effort required to implement user stories is crucial for effective planning in ASD projects. It can be done using various techniques, such as story points or ideal time (Satpathy et al., 2016). Story points are values used to represent the relative size or complexity of tasks or user stories. Whereas ideal time refers to the time it would take an individual or team to complete a task or user story without any interruptions or external factors. Between both techniques, story point estimation is widely used in ASD due to its ability to accommodate uncertainty and complexity (Canedo et al., 2018). Instead of estimating in hours or days, which can be affected by factors such as individual capabilities and external dependencies, story points involves assigning a numerical values to each user story or task based on the complexity and effort required for its implementation, allowing for more accurate and meaningful estimates (Canedo et al., 2018; Satpathy et al., 2016).

The current practice of story point estimation in ASD projects often relies on expert judgment and team consensus. Techniques such as Planning Poker, Expert Opinion and Analogy are some of the commonly used methods (Usman et al., 2014). However, these methods are subjective and can be influenced by several factors, including bias, individual expertise, and group dynamics, leading to inconsistencies and inaccuracies in the estimation process (Usman et al., 2014). To address these issues, the emergence of machine learning (ML) and artificial intelligence (AI) offer new opportunities. By using historical data and ML algorithms, ML models can be developed to estimate story points. This can help reduce human bias and improve the estimation process.

Therefore, this study aims to investigate further a machine learning-based approach for estimating story points in ASD by utilising the ensemble stacking technique, which combines RoBERTa, a robust transformer-based model for natural language processing, with

BiLSTM, a bidirectional long short-term memory network known for its effectiveness in handling sequential data. This combined approach is expected to leverage the strengths of both models, enhancing the accuracy and reliability of story point estimations.

1.1 Objectives

The main objectives of this research are:

1. To conduct a comprehensive review of the latest techniques in story point estimation in Agile software development, with a specific focus on machine learning methodologies.
2. To develop and implement a machine learning-based model for story point estimation using the ensemble stacking technique, combining RoBERTa and BiLSTM models.
3. To evaluate the effectiveness of the proposed ensemble stacking model in estimating story points and compare its performance against the state-of-the-art model.
4. To analyse the proposed model's performance, validating its accuracy and reliability using historical data from ASD projects.

1.2 Research questions

This research aims to answer the following research questions:

- RQ-1: What is the current state-of-the-art story point estimation in Agile software development using machine learning?
- RQ1.1: What are the metrics for evaluating its accuracy and performance?
- RQ-2: How can we improve the story point estimation using machine learning?
- RQ-3: How effective is the proposed model in estimating story points compared to the other machine learning model?

2 Systematic literature review

This study uses a systematic literature review (SLR) methodology, guided by steps from Kitchenham et al. (2009), to thoroughly examine and synthesise existing studies on story point estimation in ASD using machine learning techniques. The SLR approach ensures a rigorous and transparent process for identifying, selecting, and evaluating relevant literature. This is essential for providing an objective and reproducible way to address specific research questions and enhance the reliability of the findings. To guide the SLR, this section focuses on addressing the following research questions:

- **RQ-1: What is the current state-of-the-art (SOTA) story point estimation technique in Agile software development using machine learning?** This question aims to uncover the latest advancements in machine learning for story point estimation and provide a comprehensive overview of the techniques, algorithms, and frameworks proposed and evaluated in the literature.
- **RQ-1.1: What are the metrics for evaluating its accuracy and performance?** This sub-question explores the specific metrics and evaluation criteria used to assess the ef-

fectiveness of machine learning models in story point estimation. Understanding these metrics is crucial for comparing different approaches and determining their practical applicability for accuracy and performance evaluation.

The subsequent sections will elaborate on the SLR process, including the search strategy, inclusion and exclusion criteria, and data extraction methods.

2.1 Search strategy

To thoroughly review the relevant literature, this research employs a detailed search strategy across several electronic databases. The main databases used for this study are IEEE Xplore, ACM Digital Library, Scopus, and Google Scholar. Searches are conducted individually on each database to ensure the highest likelihood of retrieving all relevant studies from each source. As illustrated in Fig. 1, the search strategy follows a systematic approach that includes:

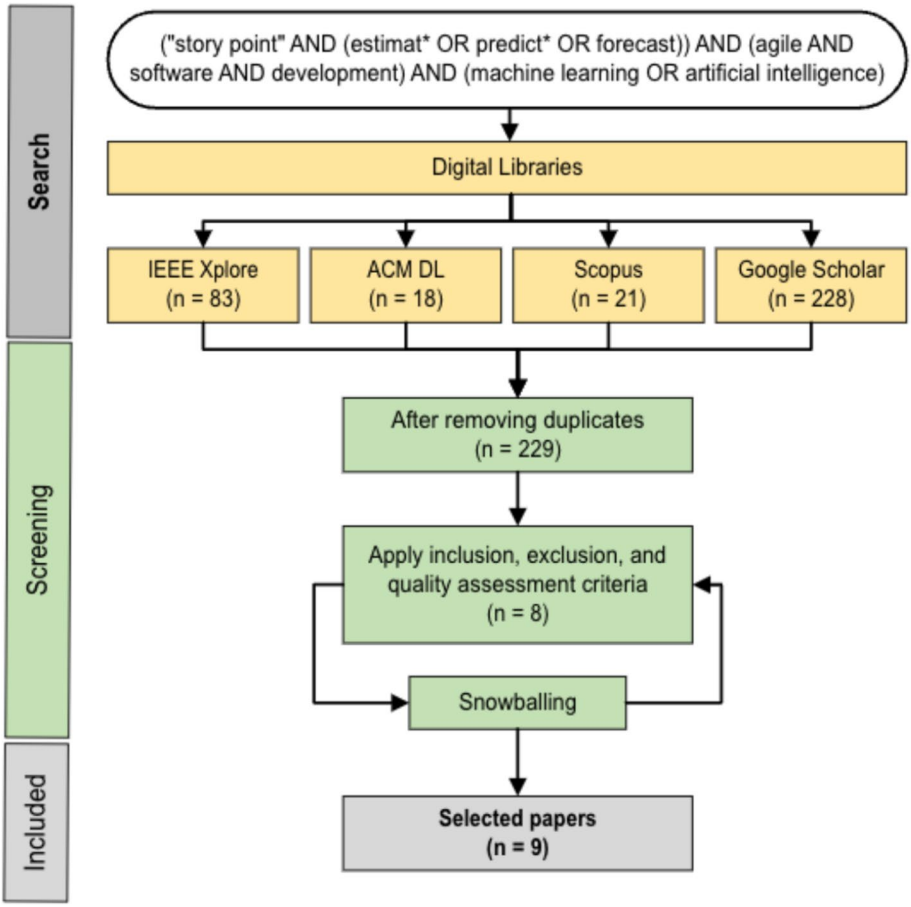


Fig. 1 Systematic Literature Review Process

i. Search keyword identification: Identify search strings and keywords based on research scope.
ii. Search: Execute a search on each database to retrieve relevant literature.
iii. Screening: Applying inclusion and exclusion criteria to filter out irrelevant studies.
iv. Documentation: Documenting the search process, including the number of results retrieved and screened at each stage.

The search strategy involves the systematic use of identified keywords combined with logical operators (AND, OR) to refine and optimise search results. Search strings are constructed as: ("story point" AND (estimate* OR predict* OR forecast)) AND (agile AND software AND development) AND (machine learning OR artificial intelligence).

From the search executed on each database, a total of 350 research papers were found. Results were exported into the Mendeley reference manager application, where studies were merged and duplicates were removed. Once duplicates were removed, the remaining 229 research papers were screened based on inclusion and exclusion criteria defined in Table 1 by manually reviewing each selected paper's titles, abstracts, and keywords. Following the screening process, 27 papers were selected for full-text review and evaluated based on quality assessment criteria to determine their relevance and quality. Subsequently, 8 papers were filtered before employing the snowballing technique to identify any additional relevant studies from the reference lists of the selected papers, which identified an additional 1 relevant study, resulting in a total of 9 studies for this systematic literature review.

Table 1 Screening Criteria

Type	Criteria
Inclusion	i. Research articles that focus on story point estimation in Agile software development.
	ii. Studies that use machine learning techniques to improve story point estimation.
	iii. Studies published in peer-reviewed conferences or journals.
	iv. Articles written in English.
Exclusion	i. Studies not related to Agile methodologies or story point estimation.
	ii. Research that does not relate to machine learning techniques.
	iii. Studies before the year 2019.
	iv. Non-peer-reviewed articles.
Quality assessment	i. Were the research purposes and aims clearly stated?
	ii. Were the research methods and the proposed techniques adequately described?
	iii. Was the dataset used appropriately described?
	iv. Were the evaluation metrics used to assess the accuracy and performance of the story point estimation models clearly stated?
	v. Were the results presented clearly and comprehensively?
	vi. Did the authors discuss the limitations and potential implications of their findings?

2.2 Findings from the literature review

To answer **RQ-1**, nine research papers which are closely related to story point estimation using machine learning techniques from 2019 onwards were identified. This section will review each paper to investigate their techniques and contributions towards improving story point estimation.

Choetkiertikul et al. (2019) proposed a new approach that combined two deep learning architectures: long short-term memory (LSTM) and recurrent highway network (RHN). They named this model Deep-SE (Deep Learning Model for Story Point Estimation). To train and assess their model, they created a new dataset consisting of 23,313 issues with actual story point values assigned by humans. They gathered from 16 JIRA open-source projects across 9 repositories. These datasets were made publicly available. To evaluate the performance of their model, the authors used three common baseline estimators: random guessing, mean, and median. They also compared their model's performance with the previous study from Porru et al. (2016). Their findings revealed that the proposed approach consistently outperformed the baselines. Specifically, when compared to the mean and median estimators, the proposed approach improved Mean Absolute Error (MAE) by 34.06% and 26.77%, respectively, across each project. Deep-SE also surpassed the model from Porru et al. (2016) in all cases, with improvements ranging from 18.18% to 56.48% in terms of MAE.

Marapelli et al. (2020) explored the use of BiLSTM, a type of Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN) to predict story points. This study utilised FastText to convert the text description into a format that can be accepted by the neural network. Similar to Choetkiertikul et al. (2019), this study also approached story point estimation as a regression problem and utilised datasets prepared by Choetkiertikul et al. (2019) to evaluate their model performance using MAE and R^2 scores. Although no detailed explanation of the comparison with Deep-SE was made, their study mentioned that their model outperformed the baseline model and achieved the highest R^2 score, which is about 74.2% on one of the datasets.

Tawosi et al. (2022b) focused on a subset of the TAWOS dataset (Tawosi et al., 2022a), analysing 31,960 issues from 26 open-source projects. They employed different approaches for story point estimation using unsupervised learning with Latent Dirichlet Allocation (LDA) and Hierarchical Clustering (HC). LDA was used to analyse textual features of software issues, while HC was used to cluster similar issues based on topic similarity. Their approach, called LHC-SE (LDA-based Hierarchical Clustering for Story Point Estimation), was evaluated using three estimation models: Cluster Mean-based, Cluster Median-based, and Closest Point-based. The issues' title and description were combined and went through basic text cleaning and pre-processing steps, such as removing URLs, code snippets, and all non-alphanumeric characters, converting the text to lowercase, removing English stopwords, and removing words with less than two characters. Evaluation metrics used include MAE, Median Absolute Error in Metrics (MdAE), and Standard Accuracy (SA), and the study was benchmarked against Deep-SE (Choetkiertikul et al., 2019), TF-IDF-SE (Porru et al., 2016), and the Mean, Median, and Random Guessing baselines. The results indicated that the LHC-SE performed similarly to the Deep-SE. However, the researchers observed that none of the methods showed significant improvement over simpler estimators, such as the Median baseline, in all instances. This suggests that the additional complexity of the

approaches may not always be justified. The study also publicly made their data and scripts available for further study and extension.

Fu and Tantithamthavorn (2023) introduced a transformative approach known as GPT2SP, a Transformer-based method, to predict story points. The primary objective was to mitigate the limitations of Deep-SE, specifically the project-specific pre-training and accuracy concerns in cross-project estimations. This study employed the Byte-Pair Encoding (BPE) algorithm and the GPT-2 (small) language model to construct their model, evaluating its performance using datasets from open-source projects (Choetkiertikul et al., 2019). The data was divided into training (60%), validation (20%), and testing (20%) subsets, like Deep-SE, to facilitate a fair comparison. Performance evaluation was based on MAE in both within-project and cross-project scenarios. Their findings indicated that GPT2SP achieved an MAE of 6% to 47% relative to Deep-SE for the within-project scenario and 3% to 46% for cross-project scenarios. Furthermore, the study created a web-based application, integrating the GPT2SP model, and conducted a survey to delve deeper into the challenges of story point estimation.

Phan and Jannesari (2022) conducted a study on using Graph Neural Networks (GNN) for text classification in story point estimation. They built upon previous work on Text Graph Neural Networks (Huang et al., 2019) and named their solution TextLevelGNN. The TextLevelGNN architecture encompasses the Graph Construction process, Graph Neural Network training using the Message Passing Mechanism, and using the Softmax and Relu functions for label prediction. The research utilised a dataset from Choetkiertikul et al. (2019). It assessed their performance for story point level classification and regression against traditional approaches, specifically Term Frequency-Inverse Document Frequency (TFIDF) vectorisation combined with Random Forest for classification. Their results showed that the TextLevelGNN model achieved competitive accuracy levels compared to the traditional TFIDF-RF approach for story point level classification. However, the model's performance in regression indicated a need for improvement, as it yielded a higher Mean Absolute Error (MAE) compared to the traditional random forest regression.

Arachchi and Amalraj (2023) investigated the utilisation of LSTM, Support Vector Machine (SVM), and Word2Vec in their proposed solution to enhance story point estimation using machine learning techniques. The study approached story point estimation as a classification problem rather than regression and leveraged 6801 issues extracted from six different open-source projects. Within their report, they referenced the use of MAE and SA to evaluate the model's performance but only provided MAE data for four projects. The study reported an average MAE of 1.9902 for these four projects and indicated that their approach outperformed three common baselines (Mean, Median, and Random Guessing). However, no additional information, such as comparison statistics, was provided on this finding.

Kassem et al. (2023) developed a machine learning regression model for estimating story points using hierarchical attention networks (HAN). The model consisted of five layers: input layer, embedding layer, encoding layer, attention layer, and output layer. This research leveraged GloVe for converting input text into dense word vectors and compared the model's performance using MAE and MdAE with Deep-SE. The study employed the dataset from Choetkiertikul et al. (2019), allocating 60% for training, 20% for validation, and 20% for testing. The findings demonstrated significant performance improvements of the proposed model over Deep-SE, with MAE enhancements ranging from 0.7% to 28% and MdAE improvements ranging from 18% to 68%.

Amasaki (2023) focused on investigating the efficacy of further pre-trained BERT models for estimating story points. The study entailed a comparison between off-the-shelf BERT models (i.e. books corpus and English Wikipedia), domain-specific (i.e. Stack Overflow, App review) and repository-specific models (i.e. Apache, MuleSoft). It included Deep-SE as a reference for comparative analysis. Utilising datasets from open-source projects (Choetkiertikul et al., 2019), the study evaluated the performance of the various BERT models using the MAE metric. The results indicated that while further pre-training with both domain-specific and repository-specific datasets revealed some improvements over off-the-shelf models, there were limitations to their overall effectiveness. Domain-specific models demonstrated superior performance compared to off-the-shelf ones. In contrast, repository-specific models exhibited mixed results, outperforming in certain aspects but also presenting limitations in comparison to Deep-SE.

Tawosi et al. (2023) conducted a replication study on Deep-SE to assess its effectiveness for within-project and cross-project story point estimation. The study used the same code created for Deep-SE, similar data from the original study, and an additional 31,960 datasets from TAWOS (Tawosi et al., 2022a). During the evaluation, they found that Deep-SE does not preprocess the title and description of the issues. Additionally, it was discovered that Deep-SE performs a transformation on the dataset before splitting it to reduce the range of the story point distribution across the entire dataset. This led to discrepancies in the replication results. The study found that the results were generally worse when the transformation was not applied. It was also noted from the Deep-SE source code that the preprocessing step is applied to the entire dataset before splitting, which could undermine the validation process and is not considered a good practice (Tawosi et al., 2023). Their replication result showed variations in the results between the replication and the original study, thus, they were unable to confirm all the conclusions made by the original study.

The systematic literature review conducted in this study has provided a comprehensive overview of the state-of-the-art (SOTA) techniques and methodologies employed in story point estimation within ASD using machine learning. All these findings are summarised in Table 2, which highlights the most relevant technical details of each study. A recurring theme throughout these studies is the application of various deep learning and machine learning models, such as LSTM, BiLSTM, CNN, GPT-2, GNN, and HAN. These models have been evaluated using diverse datasets, predominantly sourced from the publicly available data compiled by Choetkiertikul et al. (2019). Metrics such as MAE and R^2 have been consistently used to estimate model performance, with MAE being the most common metric across all studies, thereby addressing **RQ-1.1**.

3 Proposed machine learning technique

3.1 Objective

For RQ-2, the aim is to explore methods for improving story point estimation using machine learning techniques. This experiment proposes developing and evaluating an ensemble stacking model that combines RoBERTa and BiLSTM architectures. The objective is to investigate how this ensemble approach can enhance the accuracy and reliability of story point estimation in ASD projects. Furthermore, for RQ-3, the objective is to evaluate the

Table 2 Comparison of Relevant Story Points Estimation Studies

Title	ML	Preprocessing	Hyperparameter	Evaluation	Baseline
A Deep Learning Model for Estimating Story Points Choetkiertikul et al. (2019)	LSTM, RHN	Train: 60%, Validation: 20%; Test: 20%	Epoch: 1000, Batch Size: 100, Learning Rate (LR): 0.01, Adaption Rate: 0.9, Smoothing Factor: 10^{-6}, Max Sequence: 100	MAE, MdAE, SA	Random Guessing, Mean, & Median
RNN-CNN model: A bi-directional long short-term memory deep learning network for story point estimation Marapelli et al. (2020)	BiLSTM, CNN, FastText	Training: 80% Validation: 20%	Default	MAE, R^2	Deep-SE
Investigating the effectiveness of clustering for story point estimation Tawosi et al. (2022b)	LDA, HC	Train: 60%, Validation: 20%; Test: 20%	Not specified	MAE, MdAE, SA	TF-IDF-SE, Deep-SE, Random Guessing, Mean, & Median
Story Point Level Classification by Text Level Graph Neural Network Phan and Jannesari (2022)	GNN, GloVe	Training: 80% Validation: 20%	Batch size: 32, drop out: 0.5, window offset: 20	MAE	TFIDF-RF
GPT2SP: A Transformer-Based Agile Story Point Estimation Approach Fu and Tantithamthavorn (2023)	GPT-2, BPE	Train: 60%, Validation: 20%; Test: 20%	LR: 5e-4, vocab: 768	MAE	Deep-SE
An Agile Project Management Supporting Approach for Estimating Story Points in User Stories Arachchi and Amalraj (2023)	LSTM, RHN, Word2Vec, SVM	NA	Not specified	MAE, SA	Random Guessing, Mean, & Median
Story Point Estimation Using Issue Reports with Deep Attention Neural Network Kassem et al. (2023)	HAN, GloVe	Train: 60%, Validation: 20%; Test: 20%	Not specified	MAE, MdAE	Deep-SE
On Effectiveness of Further Pre-training on BERT Models for Story Point Estimation Amasaki (2023)	BERT	Train: 60%, Validation: 20%; Test: 20%	Batch size: 32; epoch: {3,5,10,20}; learning rate: {1e-4, 1e-6, 2e-5, 5e-5, and 5e-6}.	MAE	Deep-SE
Agile Effort Estimation: Have We Solved the Problem Yet? Insights From a Replication Study Tawosi et al. (2023)	LSTM, RHN (replication study)	Train: 60%, Validation: 20%; Test: 20%	Epoch: 1000, Batch Size: 100, Learning Rate (LR): 0.01, Adaption Rate: 0.9, Smoothing Factor: 10^{-6}, Max Sequence: 100	MAE, MdAE, SA	Random Guessing, Mean, & Median

performance and effectiveness of the proposed ensemble stacking model compared to the existing SOTA model. By conducting a comprehensive comparative analysis, this experiment aims to determine whether the proposed model outperforms the SOTA approach in terms of estimation accuracy and predictive performance.

3.2 Hypothesis

This research will formulate the following hypotheses to guide this controlled experiment successfully and meet the objectives. These hypotheses will help investigate the potential improvements and impacts of the proposed technique.

i. Primary Hypothesis: The ensemble stacking model combining RoBERTa and BiLSTM will provide a more accurate estimate of story points than individual models and the SOTA model. Combining RoBERTa's contextual understanding and BiLSTM's sequential processing capabilities is expected to improve estimation accuracy and performance compared to individual and SOTA models.
ii. Alternative hypothesis: Different preprocessing techniques will impact model accuracy and performance. From the findings during the SLR, the SOTA model (Choetkiertikul et al., 2019) uses a transformation technique to manage outliers in datasets during preprocessing. Thus, this hypothesis aims to investigate the proposed model's performance and accuracy when employing different preprocessing styles, particularly for managing outliers. Specifically, two techniques will be used to manage outliers: one by removing outliers and another by replicating the transformation method used by the SOTA model. By exploring these approaches, the study aims to compare the effectiveness of both preprocessing approaches and fairly evaluate the model's accuracy in relation to the SOTA model.

3.3 Datasets

This study uses the dataset Choetkiertikul et al. (2019) collected to train and evaluate the proposed model's performance and compare it with the SOTA Deep-SE approach. The dataset contains 21,064 issues (i.e. user story, bug, improvement) extracted from 14 open-source projects utilising the JIRA issue tracking and Agile project management system. Initially, 23,313 issues were collected from 16 projects, but one of the repositories comprising two projects has been removed from the public domain (Choetkiertikul et al., 2019) and, therefore, will be excluded from this study. Each dataset contains columns for the issue key, title, description, and story point. Each project's statistics are summarised in Table 3. Only five projects use the Fibonacci scale in the story point field, while others use the custom values (Choetkiertikul et al., 2019). As a result of this inconsistency, this research will approach the story point estimation effort as a regression rather than a classification problem. This is because classification requires distinct and consistent categories, which is not feasible given the diverse scales used (Tawosi et al., 2022b).

Table 3 Statistic of Datasets

Repository	Project	Abb.	Fib. Scale	Total Issues	SP ^{min}	SP ^{max}	SP ^{mean}	SP ^{median}
Apache	Mesos	MS	N	1680	1	40	3.09	3
	Usergrid	UG	Y	482	1	8	2.85	3
Appcelerator	Aptana Studio	AS	Y	829	1	40	8.02	8
	Appcelerator Studio	AP	Y	2919	1	40	5.64	5
	Titanium SDK / CLI	TS	N	2251	1	34	6.32	5
DuraSpace	DuraCloud	DC	N	666	1	16	2.13	1
Atlasian	Bamboo	BB	N	521	1	20	2.42	2
	Clover	CV	N	384	1	40	4.59	2
	JIRA Software	JS	N	352	1	20	4.43	3
Moodle	Moodle	MD	N	1166	1	100	15.54	8
Lsstcorp	Data Management	DM	N	4667	1	100	9.57	4
Mulesoft	Mule	MU	Y	889	1	21	5.08	5
	Mule Studio	MS	Y	732	1	34	6.40	5
Spring	Spring XD	XD	N	3526	1	40	3.70	3
	Total			21,064				

Fib: Fibonacci; SP: story points; Y: yes; N: no

3.4 Experiment setting

The experiment consists of two main parts: Experiment 1, known as **Case N**, and Experiment 2, known as **Case M**. In Case N, outliers are managed by removing them from the dataset, while Case M addresses them by replicating a method based on the SOTA study, which involves transforming the story point values using a threshold value. To assess the performance of the proposed models against the SOTA model, Deep-SE, this study will use the MAE measurement to compare. For a more current comparison, this research will use replication results conducted on the same Deep-SE implementation, which is more recent than the original study (Tawosi et al., 2023).

The proposed models will be implemented using Python with Jupyter Notebook, leveraging open-source libraries such as Keras, TensorFlow, and Hugging Face's Transformers. The experiments will be executed on Kaggle, utilising the GPU 100 accelerator with the following hardware specifications: CPU Intel ® Xeon ® CPU @ 2.00GHz, GPU Tesla P100-PCIE-16GB, and a maximum RAM of 29GB.

The hyperparameters for all models will be set as follows: a learning rate of 1e-4, a batch size of 16, and 20 epochs, with the early stopping feature enabled to halt training when no training improvement is observed. The maximum token length will be 256. The Adam optimiser will be adopted, and the loss will be measured using MAE. For the RoBERTa model, text data will be converted into a format the model can understand using the RoBERTa tokenizer, instantiated with a pre-trained model, specifically 'roberta-base'. A complete summary of these hyperparameters is provided in Table 4.

3.5 Evaluation metrics

This research evaluates the performance of both BiLSTM and RoBERTa models at Level 0 and the Meta-learner at Level 1 using MAE. Other evaluation metrics, such as SA and R^2 scores, could also be considered; however, MAE is chosen for its straightforward inter-

Table 4 Hyperparameters Used in Experiments

Hyperparameter	Value / Setting
Learning rate	1e-4
Batch size	16
Number of epochs	20 (with early stopping)
Maximum token length	256

pretation, consistency with regression problems (Wahid et al., 2021), and comparability with the SOTA study in story point estimation. Besides, it clearly indicates how close the model's predictions are to the actual values without being sensitive to outliers (Wahid et al., 2021). At Level 1, the Meta-learner model is also assessed using MAPE to gauge prediction accuracy in percentage terms, providing a comprehensive performance evaluation. MAPE is particularly valuable for understanding the relative error size in relation to actual values, offering an additional dimension to assess model effectiveness alongside MAE (Mora et al., 2021).

To further evaluate whether the proposed stacking ensemble provides statistically significant improvements over its base learners, the Wilcoxon signed-rank test is employed on paired absolute errors for each project dataset. For each case, comparisons are restricted to two planned tests per project (Ensemble vs. RoBERTa and Ensemble vs. BiLSTM). To control the family-wise error rate, the Bonferroni correction is applied, which is conservative and reduces the risk of reporting spurious significance. Bonferroni-adjusted *p*-values are reported alongside Vargha–Delaney's $\hat{A}_{12}$ effect size to quantify the magnitude of differences.

3.5.1 Mean Absolute Error (MAE)

MAE is a metric that measures the average magnitude of errors in a set of predictions without considering their direction. It calculates the average absolute difference between predicted and actual values. The formula for MAE is (1):

$$\mathbf{MAE} = \frac{1}{\mathrm{n}} \sum_{\mathbf{i=1}}^{\mathrm{n}} \mid \mathrm{actual}_{\mathbf{i}} - \mathrm{predicted}_{\mathbf{i}} \mid \tag{1}$$

Where:

- n is the number of data points
- $\mathrm{actual}_{\mathbf{i}}$ is the observed actual value for the issue
- $\mathrm{predicted}_{\mathbf{i}}$ is the predicted value for the issue
- $\sum$ represents the summation of the absolute errors over all data points

MAE is an easy-to-understand metric that gives an idea of the average error magnitude. A lower MAE value indicates better model performance. However, MAE treats all errors equally, regardless of the magnitude of the actual values.

3.5.2 Mean Absolute Percentage Error (MAPE)

MAPE is a statistical measure used to evaluate the accuracy of prediction models. It expresses the average absolute deviation between the predicted and actual values as a percentage of the actual values. MAPE is calculated using the formula (2) and is expressed as a percentage. A lower MAPE value indicates higher accuracy, while a higher one indicates lower accuracy.

$$\text{MAPE} = \frac{\sum \frac{|\text{A}-\text{F}|}{\text{A}} * \mathbf{100}}{\text{N}} \tag{2}$$

Where:

- N is the number of data points
- A is the actual observed value
- F is the prediction value
- $\sum$ represents the summation of the absolute percentage errors over all data points

The calculation involves taking the absolute difference between the actual and predicted values, dividing it by the actual value to get the percentage error, and then averaging these percentage errors across all data points.

3.6 Data preprocessing

The data preprocessing process involves different steps for Case N and Case M before the dataset is split into training, validation, and test sets, as shown in Fig. 2. The dataset is divided into three sets: 60% for training, 20% for validation, and 20% for testing. This ensures that the models can be trained, tuned, and evaluated on separate data, reducing the risk of overfitting and improving generalizability. This split ratio is also chosen to allow for a fair comparison with the SOTA technique, as their study uses the same split ratio. For Case M, outliers' transformation is applied before the dataset is split. This involves modifying extreme values to fit within a threshold by adjusting the input labels (story points column).

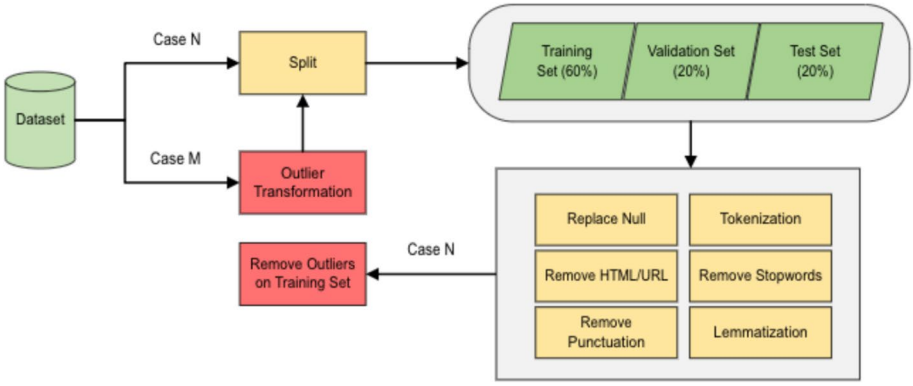


Fig. 2 Data Preprocessing Process

90% of the input labels are adjusted to be less than or equal to a calculated threshold. Any label greater than the threshold is replaced by the threshold value itself. Meanwhile, for Case N, outliers are removed only from the training set to ensure that the model's performance is evaluated under conditions resembling real-world scenarios, thus enabling fair evaluation of the model performance (Tawosi et al., 2023). Outliers are removed using the Interquartile Range (IQR) method (Perez & Tah, 2020), and the Natural Language Toolkit (NLTK) is used to remove stopwords, eliminate common but uninformative words, lemmatize words to their base forms, and tokenize text into individual tokens.

3.7 Model architecture

The model architecture proposed in Fig. 3 utilises an ensemble stacking technique, which combines BiLSTM and RoBERTa as base models at **Level 0** and a Dense Neural Network (DNN) acting as a meta-learner at **Level 1**. This ensemble method aims to harness the strengths of each model while mitigating its weaknesses to achieve higher prediction accuracy.

Ensemble machine learning techniques involve combining predictions from multiple base models. These methods are used in various machine learning applications because they can exploit different models' strengths and reduce individual models' weaknesses. The main goal of these techniques is to enhance prediction accuracy, generalizability, and robustness by combining the outputs of different base models (Mienye & Sun, 2022). Ensemble learning algorithms can be categorised into three primary types: bagging, boosting, and stacking. The bagging method requires training multiple independent models in parallel and combining their predictions through averaging or a voting technique. The boosting technique involves building models sequentially, with each subsequent model correcting the errors of its predecessor. Meanwhile, stacking combines the predictions of multiple models using another model, often referred to as a meta-learner (Mienye & Sun, 2022). The proposed model employs stacking, which trains a meta-learner to make predictions based on the outputs of multiple base models (BiLSTM and RoBERTa).

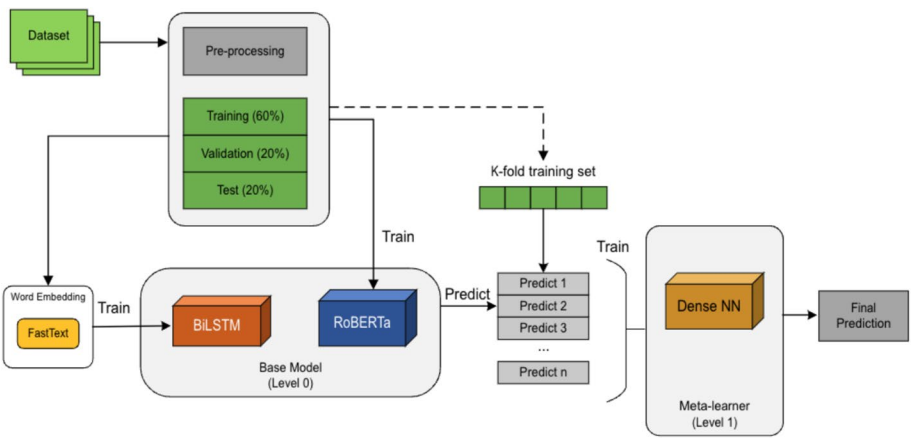


Fig. 3 High-level Proposed Architecture

At **Level 0**, the BiLSTM model processes sequential data, capturing both past and future contexts in the input sequence by using a bidirectional LSTM architecture. The input to the BiLSTM model is the preprocessed text, as explained in Section 3.6. For this model, an additional step is introduced before model training. FastText is used to convert the cleaned text into word embeddings, capturing the semantic meaning of words and providing a dense vector representation. This representation is essential for the BiLSTM model to process sequential data and capture long-term dependencies effectively. The architecture of the BiLSTM model includes an embedding layer to handle the word vectors, bidirectional LSTM layers that process the input sequence in both forward and backwards directions, dropout layers to prevent overfitting, and a dense layer to generate predictions.

The RoBERTa model, also at **Level 0**, is a transformer-based architecture that generates contextualized word embeddings directly from the input text. RoBERTa tokenizes the input text, converting it into tokens that are passed through multiple transformer encoder layers. These layers, equipped with self-attention mechanisms and feed-forward networks, capture complex relationships and dependencies in the data. Unlike the BiLSTM model, RoBERTa does not require pre-trained word embeddings since it creates its contextual representations. After processing the input through several transformer layers, the RoBERTa model outputs a dense representation of the text, which is used as predictions for the meta-learner.

At **Level 1**, the meta-learner is a DNN that combines the predictions from the BiLSTM and RoBERTa models to generate the final output. The input to the DNN consists of the predictions made by the base models for each data point. These predictions serve as input features for the DNN, which consists of fully connected layers. The DNN learns to assign appropriate weights to the outputs from BiLSTM and RoBERTa, determining how to best combine them to make the final prediction. The architecture of the DNN includes an input layer to handle the predictions from the base models, multiple dense layers with activation functions (ReLU) to capture non-linear relationships, and an output layer to produce the final predictions. The training process for this ensemble model uses K-fold cross-validation technique. This approach is employed to fully use the limited training data and provide a more reliable estimate of unseen data (Wong & Yeh, 2020). The training data is divided into five-folds, and each fold is used once as validation data, while the remaining folds form the training data. After the base models generate predictions from these divided training data, the meta-learner is trained on the combined outputs from the generated predictions. After the training process is complete, the performance of the meta-learner is evaluated using MAE and MAPE on the validation set.

3.7.1 FastText

FastText is a word embedding and text classification library developed by Facebook's AI Research lab, designed to enhance the capabilities of traditional word embedding models like Word2Vec. Word embeddings represent words as vectors, enabling machines to understand and process language by capturing both semantic and syntactic relationships between words. In addition to word embeddings, FastText is widely used for text classification, a process that involves assigning predefined labels to textual data, analyzing the content and context of text (such as sentences or documents), and categorizing them based on their meaning.

FastText was created to address the limitations of earlier models like Word2Vec, which generate vectors only for words that appear in the training data, ignoring the morphological characteristics of words. This limitation prevents Word2Vec from effectively handling out-of-vocabulary (OOV) words, as no vectors can be produced for new words. FastText solves this problem by generating word embeddings using subword units, such as character n-grams. Each word is represented as a sum of these subwords, allowing FastText to create vectors even for words that were not encountered during training, capturing richer morphological information in the process (Choi & Lee, 2020; Amalia et al., 2020).

Like Word2Vec, FastText employs Continuous Bag of Words (CBOW) and Skip-Gram models to calculate vectors. CBOW predicts a target word based on the context of surrounding words, while Skip-Gram predicts the context words surrounding a given target word. The use of subwords and these modeling techniques enhances FastText's ability to generate high-quality word embeddings and makes it robust for languages with rich morphological structures (Choi & Lee, 2020).

In addition to its effectiveness in word representation, FastText has proven valuable for text classification tasks. As shown in studies like Amalia et al. (2020), FastText demonstrates high classification accuracy across multiple languages, including Bahasa Indonesia, and is particularly useful for large-scale datasets (Amalia et al., 2020). Facebook also provides pre-trained FastText models, including a model trained on Common Crawl data with 300-dimensional word vectors for English, available for download and use in various natural language processing (NLP) tasks (Grave et al., 2018).

3.7.2 BiLSTM

Bidirectional Long Short-Term Memory (BiLSTM) is an extension of the traditional Long Short-Term Memory (LSTM) model, specifically designed to capture long-term dependencies and context in sequential data more effectively. LSTM is a type of Recurrent Neural Network (RNN) known for its ability to handle long-term dependencies by addressing the vanishing gradient problem, which is common in standard RNNs. While LSTM processes data in one direction, typically forward in time, BiLSTM improves upon this by processing data in both directions: forward and backward. This bidirectional approach enables the model to utilize information from both past and future sequences, providing a more comprehensive understanding of the input data (Li et al., 2020; David & Renjith, 2021; Kadu et al., 2022).

In BiLSTM, two LSTM layers are connected: one processes the input sequence in the forward direction, while the other processes it in the reverse direction. This allows the model to learn the context from both preceding and succeeding elements in the sequence. This dual-layer architecture makes BiLSTM highly effective in tasks like sentiment analysis, machine translation, and sequence labeling, where understanding context from both directions is essential (David & Renjith, 2021; Kadu et al., 2022).

The core components of BiLSTM are similar to LSTM, including memory cells and three main gates: the input gate, forget gate, and output gate. These gates regulate the flow of information through the memory cell and manage how much of the previous information is retained or discarded. BiLSTM uses two hidden layers (forward and backward), allowing it to capture dependencies from both directions in the input sequence. By learning

bidirectional relationships, BiLSTM outperforms unidirectional models in various natural language processing (NLP) tasks, such as sentiment classification and speech recognition (Karakaya & Kilimci, 2024; Li et al., 2020).

The model's parameters include the number of LSTM units, the number of layers, the input and output dimensions, and activation functions. Training is done through backpropagation through time (BPTT), a method that calculates the gradients and updates the weights accordingly. This training approach ensures that the model efficiently learns long-term dependencies in the input data (Karakaya & Kilimci, 2024).

3.7.3 RoBERTa

RoBERTa (Robustly Optimized BERT Approach) is an advanced model based on the Bidirectional Encoder Representations from Transformers (BERT) architecture. Both RoBERTa and BERT belong to the family of Transformer models, which were designed to handle sequence-to-sequence modeling tasks, addressing long-range dependencies more effectively than traditional models like RNNs and LSTMs. The Transformer model consists of three main components: a tokenizer, transformers, and heads. The tokenizer encodes raw text into sparse index encodings, which are then transformed into contextual embeddings by the transformers. Finally, the heads are responsible for applying these embeddings to various downstream tasks, such as text classification, sentiment analysis, and language translation (Kadu et al., 2022; Tan et al., 2022).

RoBERTa improves upon the BERT model by modifying both the training method and data. In contrast to BERT, which utilizes the Next Sentence Prediction (NSP) task, RoBERTa removes this component and focuses solely on the Masked Language Model (MLM) task. Additionally, RoBERTa replaces BERT's static masking mechanism with dynamic masking, allowing the model to see different tokens masked during each epoch. This dynamic masking strategy enables RoBERTa to generalize better across different tasks (Liu et al., 2022).

One of the key distinctions of RoBERTa is the use of byte-level Byte-Pair Encoding (BPE) for tokenization, which supports a larger vocabulary set (50K subword units) than BERT's 30K subword units. RoBERTa is trained on significantly more data than BERT, leveraging large datasets, including books, web pages, and Wikipedia articles. This extensive training allows RoBERTa to capture long-range dependencies more effectively and handle diverse text data from multiple sources (Tan et al., 2022).

The architecture of the RoBERTa-base model comprises 12 transformer layers with 768 hidden units and 12 attention heads, totaling 125 million parameters. This robust architecture enables RoBERTa to excel in various NLP tasks, including sentiment analysis, question answering, and text classification. In comparative studies, RoBERTa consistently outperforms its predecessor BERT, achieving higher accuracy and better overall performance across multiple datasets (Adoma et al., 2020).

3.7.4 Dense neural network

Dense Neural Networks (DNN), also known as fully connected networks, are among the simplest yet most foundational neural network architectures. Inspired by the biological

processes of the human brain, DNNs consist of interconnected layers of neurons, where each neuron in a given layer is connected to every neuron in the subsequent layer. This fully connected nature enables DNNs to learn complex representations of data, making them powerful tools for tasks that involve pattern recognition and prediction (Nazari & Yan, 2021).

In DNNs, each neuron receives inputs from all neurons in the previous layer, processes this information by applying learned weights, and then passes the result to the neurons in the next layer. The dense connections allow DNNs to perform well in capturing intricate relationships in data. The architecture of a DNN typically includes an input layer, one or more hidden layers, and an output layer. Through the process of backpropagation, the network adjusts its weights during training to minimize errors, enabling it to optimize its performance over time.

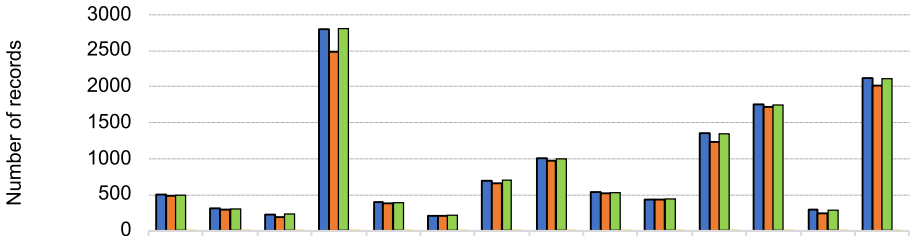
DNNs are widely used in various domains for tasks such as regression analysis, classification, and even unsupervised clustering. Their versatility makes them applicable to a wide range of fields, including image recognition, natural language processing, and predictive modeling. The ability of DNNs to analyze complex data patterns and self-optimize through learning makes them essential in the development of advanced machine learning and artificial intelligence systems (Nazari & Yan, 2021).

4 Result and analysis

4.1 Dataset analysis

This section examines the distribution of story points in the training set across the 14 projects by comparing Case N and Case M, before and after executing preprocessing techniques, in order to understand the impact of different preprocessing styles on model training accuracy.

Before preprocessing, the distribution of story points was inconsistent, particularly for the DM project (Refer to Appendix A - Fig. 9). Due to time constraints, this research will treat the inconsistent data as outliers. Therefore, Case N will use the IQR method to remove the inconsistent data, preventing it from impacting the model's performance during evaluation. A total of 766 records were removed from the 12,619 records in the training dataset, as shown in Fig. 4. The CV project had the highest record removal rate at 14%, while the DM project had 319 records, accounting for 11% of its total. In contrast, the AS and MU projects had the lowest outlier removal rates, with only about 1% of their records identified as outliers. The significant reduction of outliers in some projects indicates a substantial improvement in data consistency, while the minimal removal of outliers in other projects suggests more stable data distributions. Meanwhile, as highlighted in the previous section, Case M does not remove the outliers like Case N but only replaces the story point values with the threshold value (Refer to Appendix A - Figs. 10 and 11 for story points distribution after preprocessing).



	AS	BB	CV	DM	DC	JS	MD	ME	MU	MS	AP	TS	UG	XD
Before PP	496	311	229	2799	398	210	698	1007	532	438	1349	1750	288	2114
Case N	491	294	197	2480	378	204	657	978	525	431	1238	1719	250	2011
Case M	496	311	229	2799	398	210	698	1007	532	438	1349	1750	288	2114
% Removed	1%	5%	14%	11%	5%	3%	6%	3%	1%	2%	8%	2%	13%	5%

Fig. 4 Number of Records in Training Set

All datasets have a minimum story point value of 1. When looking at the maximum values from Fig. 5, the dataset before preprocessing shows the highest story point values compared to Case N and Case M. When comparing Case N and Case M, Case N generally shows lower maximum values for certain projects. For example, the CV and UG projects have maximum story point values of 8 and 3, respectively. Meanwhile, three projects, BB, DM, and MS, maintain the same maximum values in both cases, at 5, 21, and 13, respectively. However, the remaining nine projects show higher maximum story point values in Case N compared to Case M. After outlier removal, the AS project shows the most significant difference, with a maximum story point value of 21 in Case N compared to just 13 in Case M. This suggests that the preprocessing steps in Case N, particularly outlier removal, significantly impacted the maximum story point values across various projects.

Before preprocessing, the mean value of story points for all projects was higher, with the DM and MD projects exhibiting the highest values, each exceeding 10. After preprocessing, the mean value of the story point distribution reveals that nine projects in Case M have higher mean values than in Case N. However, the differences between Case M and Case N are generally not significant, except for the DM, MD, and CV projects, where the average difference is approximately 1.6 of the mean value.

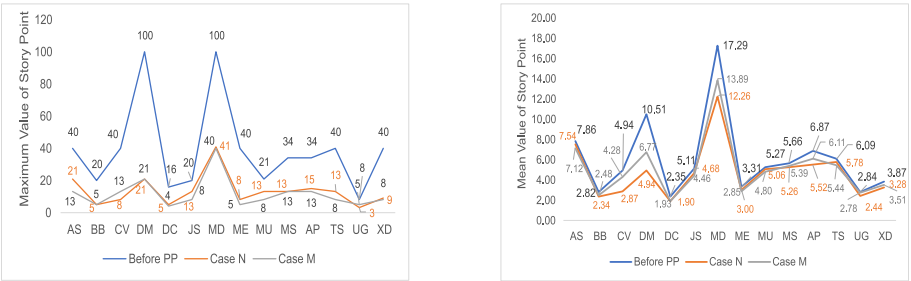


Fig. 5 Maximum and Mean Value of Story Point in Training Set After Preprocessing

4.2 Experimental results

This section conducts a comparative analysis of Case N and Case M, and specifically compares the stacking model with the SOTA model using results from Case M to ensure a fair comparison, as different preprocessing techniques were applied in each case. The performance of each model across both datasets is recorded in Table 5, which enables comparison between preprocessing strategies (Case N and Case M) and against the SOTA model, allowing direct observation of the individual contributions of each base model to the ensemble's performance.

4.2.1 Experiment 1: Case N

Based on the evaluation results shown in Table 5, RoBERTa shows varying performance in Case N. It excels in simpler projects like BB, with a low MAE of 0.67, but struggles with complex ones like MD, where the MAE is 13.59. BiLSTM shows similar trends, with a slight improvement over RoBERTa in some cases, such as an MAE of 13.27 for MD. The stacking model generally performs competitively by leveraging the strengths of both models. For example, it achieves the lowest MAE of 1.42 for the JS project, often landing between the individual performances of RoBERTa and BiLSTM, demonstrating a balanced and effective approach.

After comparing the results in Fig. 6, it is evident that for Case N, the ensemble stacking model performs competitively alongside RoBERTa and BiLSTM. Specifically, the stacking model achieves the lowest MAPE for the CV (0.58), DM (0.91), and JS (0.62) datasets.

Table 5 MAE Comparison of Evaluation Results for Case N, Case M, and SOTA (Deep-SE)

Project	Case N			Case M			Deep SE
	RoBERTa	BiLSTM	Stacking	RoBERTa	BiLSTM	Stacking	
AS	**3.50**	3.94	3.60	**2.69**	2.95	2.84	3.57
BB	**0.67**	0.76	**0.67**	**0.68**	0.94	**0.68**	0.80
CV	**1.98**	2.11	2.01	**1.64**	1.82	1.68	2.46
DM	6.21	**5.92**	6.24	4.03	3.72	4.04	**3.70**
DC	**0.77**	0.83	0.78	0.67	0.76	0.66	**0.64**
JS	1.48	1.93	**1.42**	1.49	1.92	**1.43**	1.57
MD	13.59	**13.27**	13.65	9.73	8.58	9.58	**6.89**
ME	**1.31**	1.33	**1.31**	1.13	1.20	1.12	**1.05**
MU	2.32	**2.24**	2.38	**2.03**	2.13	2.04	2.26
MS	**4.56**	4.67	4.61	3.19	3.17	3.23	**3.12**
AP	**2.48**	2.67	**2.48**	2.31	2.38	2.31	**2.10**
TS	1.39	1.58	**1.38**	**1.35**	2.03	1.36	1.41
UG	**0.79**	0.97	0.82	**0.75**	0.82	0.76	1.06
XD	**1.80**	1.86	**1.80**	1.79	1.88	1.81	**1.66**

Note: The best results are highlighted in bold.

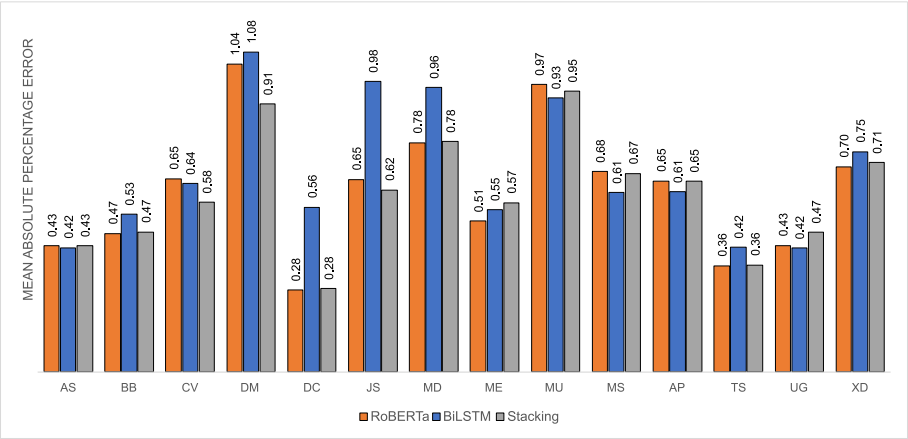


Fig. 6 Comparison of MAPE for Each Model for Case N

RoBERTa demonstrates strength in the XD (0.70) and ME (0.51) datasets, while BiLSTM excels in the AS (0.42), AP (0.61), MU (0.93), MS (0.61), and UG (0.42) projects. For AS, BB, DC, MD, AP, and TS datasets, the stacking model's performance is comparable to RoBERTa, indicating closely aligned results. Overall, the stacking model often matches or slightly improves upon the individual models' results, which illustrates the advantage of leveraging both RoBERTa's contextual understanding and BiLSTM's sequential processing capabilities.

4.2.2 Experiment 2: Case M

From the results shown in Table 5, for Case M, both RoBERTa and BiLSTM show significant improvements. MAE on RoBERTa for the MD project decreases from 13.59 in Case N to 9.73, and BiLSTM's MAE for the DM project drops from 5.92 to 3.72. This indicates that the transformation technique greatly enhances model accuracy. The stacking model also performs well in this scenario, achieving an MAE of 9.58 for MD, which is better than RoBERTa but slightly below BiLSTM. These results show that while the transformation technique improves the performance of all models, the stacking model remains robust and competitive, effectively leveraging the benefits of both RoBERTa and BiLSTM.

In an analysis of MAPE across various projects for Case M (Fig. 7), the ensemble stacking model demonstrates competitive performance alongside RoBERTa and BiLSTM. The stacking model achieves the lowest MAPE for the CV dataset (0.60), highlighting its effectiveness. Furthermore, it shows similar performance to RoBERTa in projects such as JS, MU, MS, TS, and XD, while BiLSTM outperforms the other models in the ME (0.56), MU (0.89), and MS (0.63) projects. On the other hand, RoBERTa performs better than BiLSTM in several projects, including BB, DM, DC, MD, and AP, with improvements ranging from

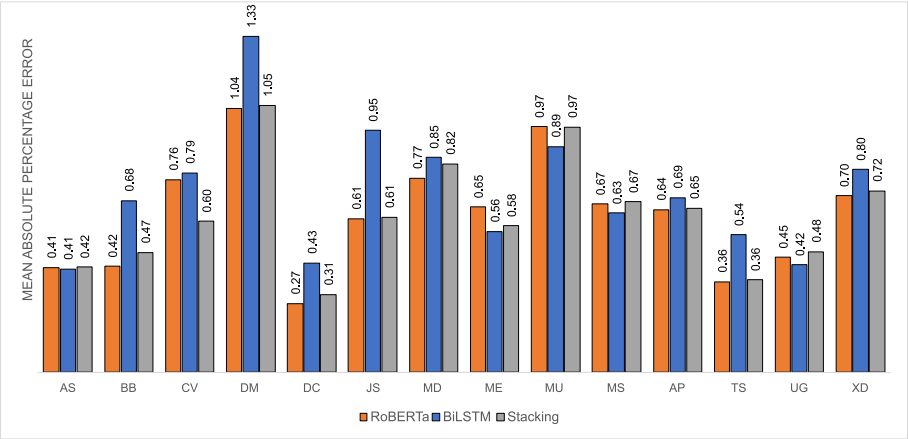


Fig. 7 Comparison of MAPE for Each Model for Case M

3% to 38%. This suggests that while BiLSTM excels in specific datasets, RoBERTa generally provides better accuracy across a wider range of projects.

4.2.3 Case N vs Case M

After comparing the MAPE in Fig. 8 for stacking Case N and stacking Case M across multiple projects, the preprocessing techniques impact model performance. Case N demonstrates better accuracy for the DM (13% improvement) and DC (10% improvement) datasets. On the other hand, Case M shows a 2% improvement for the AS and JS datasets. For the remaining projects, the performance differences are minimal, ranging from 0% to 5%, indicating similar accuracy between the two cases. Specifically, the projects BB, MS, AP, and TS show no performance difference (0%), while other datasets like CV, MD, ME, MU,

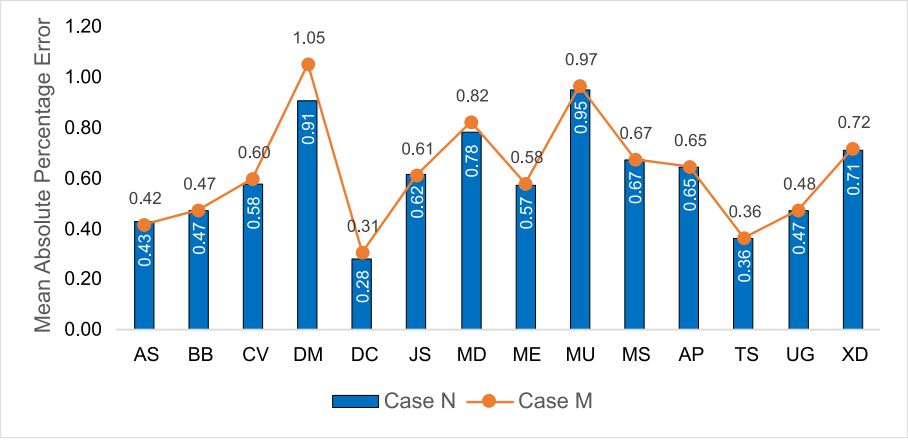


Fig. 8 Comparison of MAPE for Ensemble Stacking Model for Case N and Case M

UG, and XD have slight variations, with Case N and Case M performing comparably. This analysis highlights that while certain preprocessing techniques can significantly improve specific datasets, the overall performance across most projects remains relatively consistent between the two cases.

The statistical results from the Wilcoxon signed-rank test with Bonferroni correction provide further insights into these patterns (Table 6). Across many projects, the stacking ensemble significantly outperformed at least one base learner, with $\hat{A}_{12}$ values often above 0.6, indicating moderate to large effect sizes. In several projects, however, the corrected *p*-values were close to 1.000, suggesting that the ensemble and the corresponding base model achieved comparable performance. These non-significant findings highlight that the benefits of ensembling are not uniform across all datasets and may depend on the characteristics of each project.

When comparing the statistical outcomes between Case N and Case M, a similar pattern emerges. Projects such as BB, JS, and TS consistently show strong evidence in favor of the ensemble across both cases, reflected in $p < 0.001$ and large $\hat{A}_{12}$ values. In contrast, projects like AS, CV, and ME show significance in one case but not the other, suggesting that the ensemble's relative advantage can be sensitive to preprocessing choices and evaluation conditions. Finally, projects such as MD, MU, and MS yield non-significant results in both cases ($p \approx 1.000$), reinforcing that ensembling does not always guarantee improve-

Table 6 Wilcoxon test results with Vargha-Delaney $\hat{A}_{12}$ effect sizes (in brackets) for absolute errors in Case N and Case M, highlighting comparisons between Ensemble vs. RoBERTa and Ensemble vs. BiLSTM.

Project	Case N - Ensemble vs.		Case M - Ensemble vs.	
	RoBERTa	BiLSTM	RoBERTa	BiLSTM
AS	0.078 [0.45]	1.000 [0.55]	1.000 [0.45]	< 0.001 [0.47]
BB	1.000 [0.45]	0.169 [0.46]	1.000 [0.51]	< 0.001 [0.39]
CV	1.000 [0.48]	1.000 [0.52]	1.000 [0.50]	1.000 [0.48]
DM	1.000 [0.49]	< 0.001 [0.47]	1.000 [0.51]	< 0.001 [0.41]
DC	1.000 [0.45]	1.000 [0.43]	1.000 [0.51]	0.224 [0.43]
JS	1.000 [0.47]	1.000 [0.42]	1.000 [0.42]	1.000 [0.46]
MD	< 0.001 [0.44]	1.000 [0.55]	1.000 [0.53]	1.000 [0.56]
ME	1.000 [0.49]	< 0.001 [0.47]	1.000 [0.51]	< 0.001 [0.46]
MU	1.000 [0.52]	0.003 [0.46]	1.000 [0.48]	0.025 [0.45]
MS	1.000 [0.51]	< 0.001 [0.47]	1.000 [0.49]	< 0.001 [0.43]
AP	1.000 [0.44]	< 0.001 [0.39]	1.000 [0.55]	< 0.001 [0.41]
TS	1.000 [0.63]	< 0.001 [0.36]	1.000 [0.61]	< 0.001 [0.38]
UG	1.000 [0.56]	1.000 [0.51]	1.000 [0.54]	1.000 [0.45]
XD	1.000 [0.51]	< 0.001 [0.48]	< 0.001 [0.47]	< 0.001 [0.47]

ment when a single base learner already dominates. Taken together, this case-wise analysis illustrates that while stacking generally improves robustness, its effectiveness can vary with dataset-specific characteristics and preprocessing configurations.

4.2.4 Comparison with Deep-SE

After comparing the stacking model in Case M with the SOTA Deep-SE model, it is evident that it often matches or surpasses Deep-SE's performance in several cases. Specifically, the stacking model achieves better MAE in seven out of fourteen projects: AS (2.84 vs 3.57, a 20% improvement), BB (0.68 vs 0.80, a 15% improvement), CV (1.68 vs 2.46, a 32% improvement), JS (1.43 vs 1.57, an 9% improvement), MU (2.04 vs 2.26, a 10% improvement), TS (1.36 vs 1.41, a 4% improvement), and UG (0.76 vs 1.06, a 28% improvement). As indicated in Table 5, these results show that the stacking model outperforms Deep-SE in half of the projects, demonstrating its robustness and effectiveness.

However, Deep-SE consistently achieves a lower MAE in the remaining projects, highlighting its improved accuracy in those cases. For instance, in the MD project, Deep-SE achieves an MAE of 6.89, compared to the stacking model's 9.58, representing a 28% improvement. Deep-SE achieves an MAE of 2.10 in the AP project, while the stacking model's MAE is 2.31, indicating a 9% improvement. The performance comparison shows that the stacking model performs better than Deep-SE in a range of 4% to 32%, while Deep-SE outperforms the stacking model by 3% to 9%, except for the MD project.

5 Discussion

This section presents the study's main findings, which aim to address **RQ-3** by evaluating the effectiveness of the proposed model in estimating story points. The discussion centres on the impact of preprocessing techniques on model performance, the comparative analysis of various machine learning models, and the comparison of the proposed stacking model with the SOTA Deep-SE model.

5.1 Impact of preprocessing techniques

The dataset analysis from Section 4.1 highlights that inconsistent records were removed to prevent them from affecting model performance, specifically for Case N. The IQR method used during preprocessing considers any data as outliers if it falls below the lower bound (Q1 – 1.5*IQR) or above the upper bound (Q3 + 1.5*IQR), regardless of the number of such points. In the DM project, 319 records were removed because the records fell within this range. However, upon closer inspection, these data points should not be disregarded as they contain valid information about real tasks with actual story points. Removing them can result in the loss of potentially valuable information that could have contributed to the models' accuracy. Additionally, this could lead to biased estimates and an overly optimistic assessment of model performance (Ur Rehman & Belhaouari, 2021). When comparing Case N and Case M, the performance of Case M is better than that of Case N. All models have the same architecture but use different training data. This difference in performance could be attributed to the size, inconsistency, and quality of data. While Case N removes outli-

ers, potentially discarding important data, Case M transforms these outliers, retaining more information within the dataset. This suggests that the approach to handling outliers significantly impacts the model's effectiveness. While preprocessing steps like outlier removal are crucial for ensuring data quality and model accuracy (Ur Rehman & Belhaouari, 2021), it is important to consider the potential loss of valuable information and its impact on model performance. Case M's superior performance suggests that alternative preprocessing techniques that retain more data, even if transformed, could lead to better and more reliable models.

5.2 Interpretation of results

The experimental results in Section 4.2 provide a comprehensive comparison of model performances across Cases N and M. In Case N, where outliers are removed during preprocessing steps, the RoBERTa and BiLSTM models showed varied performance, with RoBERTa excelling in some datasets. The stacking model, which combines RoBERTa and BiLSTM, demonstrated competitive performance and often achieved the lowest MAPE for datasets like CV, DM, and JS. On the other hand, Case M, which employed a transformation technique like the SOTA, saw significant improvements in both RoBERTa and BiLSTM performances. The stacking model showed robust performance, leveraging the strengths of both underlying models and maintaining competitiveness across different datasets. These findings suggest that while preprocessing techniques significantly impact model accuracy, the stacking model's ability to adapt and perform well in both cases highlights its robustness and flexibility.

Comparing the stacking model in Case M with the SOTA Deep-SE model shows valuable competitive dynamics. The stacking model outperformed Deep-SE in seven of fourteen datasets, demonstrating improvements ranging from 4% to 32%. Notable examples include datasets such as AS (20% improvement) and CV (32% improvement), where the stacking model showed substantial superiority. However, Deep-SE maintained a lower MAE in the remaining projects, underscoring its overall consistency accuracy. Particularly in the MD project, Deep-SE achieved a 28.06% better than the stacking model. The stacking model performed poorly on this MD dataset compared to Deep-SE. Based on the dataset analysis in Section 4.1, this could be due to the small size of the dataset and the spread of story point values, which are inconsistent compared to other datasets, highlighting areas for further investigation and potential optimisation. However, the individual performance measurement for the stacking model showed a lower MAPE, indicating good accuracy and performance. In addition, the performance differences vary from 4% to 32%, indicating that while the stacking model is highly competitive and often superior in specific datasets, Deep-SE retains a slight edge in overall performance consistency. This comparison highlights the strengths of both models. The proposed Ensemble Stacking model can effectively complement or surpass the existing SOTA model, making it a valuable addition to the toolkit for story point estimation.

The Wilcoxon signed-rank analysis further strengthens these findings. Across several datasets, the ensemble's advantage over its base learners was confirmed with significant p-values and moderate-to-large $\hat{A}_{12}$ effect sizes, underscoring that the observed improvements are not due to random variation. At the same time, a subset of projects yielded non-significant results ($p \approx 1.000$), showing that in some cases, RoBERTa or BiLSTM already

performed competitively, leaving little room for the ensemble to improve. This dual outcome highlights the nuanced role of ensembling: it provides consistent gains in many settings, but its effectiveness can depend strongly on dataset size, variability, and preprocessing strategy.

Answer to RQ-3: The proposed ensemble stacking model is highly effective in estimating story points, outperforming the state-of-the-art Deep-SE model in seven out of fourteen datasets and demonstrating signi cant improvements ranging from 4% to 32%.

5.3 Agile methodology integration

The present work focuses primarily on evaluating predictive performance rather than deployment pipelines. In practice, new projects or teams often face challenges in determining appropriate story point assignments. The proposed approach, along with similar machine learning models, can be integrated into Agile workflows to streamline estimation and provide a consistent baseline for new projects or project teams.

This model is particularly well-suited for generating initial story point estimates in early Agile sprints. In subsequent sprints, these estimates can be refined either through manual methods such as planning poker or by retraining the model with updated data that incorporates discrepancies observed during development. One method could be to interleave sprints with manual story point estimation with model provided estimation. Manual story point estimations can be used as additional datasets after each sprint to retrain and tailor the model for the project. This will in turn improve model accuracy as well as save developer estimation time.

A key future research direction will be to examine and evaluate the practicality of machine learning to be used to assist in Agile workflows. This could include developing an incremental retraining strategy, integrating the model into commonly used Agile tools (e.g., Jira), and exploring continuous learning approaches to maintain accuracy in dynamic environments.

5.4 Limitations

Despite the promising results, this research encountered several limitations due to time and resource (processing power) constraints, which could affect the generalizability and accuracy of the findings. The limitations are outlined below:

i. This study evaluates the model performance using validation data from the same domain and repository-specific dataset, specifically relying on publicly available open-source datasets (Choetkiertikul et al., 2019). While this approach ensures reproducibility, it may limit generalizability to other domains, industrial datasets, or completely unseen environments, where issue descriptions and estimation patterns could differ significantly.
ii. In this study, the same hyperparameter setting is used for each model. Initially, the study attempted to find the optimal hyperparameter setting for each model. However,

resource constraints prevented this. It is important to note that these constraints could affect the accuracy of the models.

iii. Inconsistent data were managed only using the IQR method to treat them as outliers. Due to the limited time available for this research, other techniques could not be tested, which might have provided better handling of inconsistencies and model performance.

iv. The proposed model currently operates as a black box without integrated explainability or interpretability, which may hinder understanding of its predictions and limit adoption in real-work Agile settings.

6 Conclusion

In conclusion, this research has achieved its objective by introducing an ensemble stacking technique for estimating story points, a key area in ASD. The unique aspect of this research is the use of an ensemble machine learning technique, specifically the stacking technique, which combines the strengths of RoBERTa and BiLSTM. RoBERTa provides deep contextualised representation, and BiLSTM can capture sequential information. This approach has demonstrated promising results, outperforming the state-of-the-art Deep-SE model in seven of fourteen datasets with improvements ranging from 4% to 32%. On the other hand, Deep-SE outperforms the proposed ensemble stacking technique in the remaining datasets with differences of less than 10%, excluding the MD dataset. The proposed ensemble stacking technique also achieved competitive performance with less than 1% MAPE across all datasets, indicating its robustness and accuracy in estimating story points.

Despite these results, there is room for improvement and further validation of the proposed model based on the achieved performance. Future research should focus on training and evaluating models with larger datasets, such as using the TAWOS version 1.1 (Tawosi et al., 2022a). This dataset contains 458,232 issues from 39 open-source projects extracted from 12 public JIRA repositories. It would be interesting to assess how well the proposed ensemble stacking technique performs with this larger dataset and whether it can maintain its competitive performance. Additionally, to address the limitation of relying solely on open-source datasets, future studies should incorporate industrial datasets from Agile teams operating in diverse domains (e.g. finance, healthcare, e-commerce). This will allow evaluation of model robustness in environments with varying documentation styles, estimation practices, and domain-specific terminology. Domain adaptation techniques or transfer learning could be explored to improve cross-domain performance. Finding an alternative to manage data inconsistency and experimenting with different hyperparameter settings to find the optimal configuration for the ensemble stacking technique could lead to even better performance and accuracy than the current results. Moreover, future research could involve extending the base models with additional machine learning models using different paradigms or architectures to improve the performance of the proposed ensemble stacking model. Integrating explainability tools such as SHAP or attention visualizations could improve transparency and foster adoption in real-world Agile environments. Finally, practical deployment aspects should also be explored, including integration into Agile workflow platforms (e.g. Jira), incremental retraining strategies to reflect evolving data, and continuous learning methods to maintain model accuracy over time.

Appendix A Story points distribution

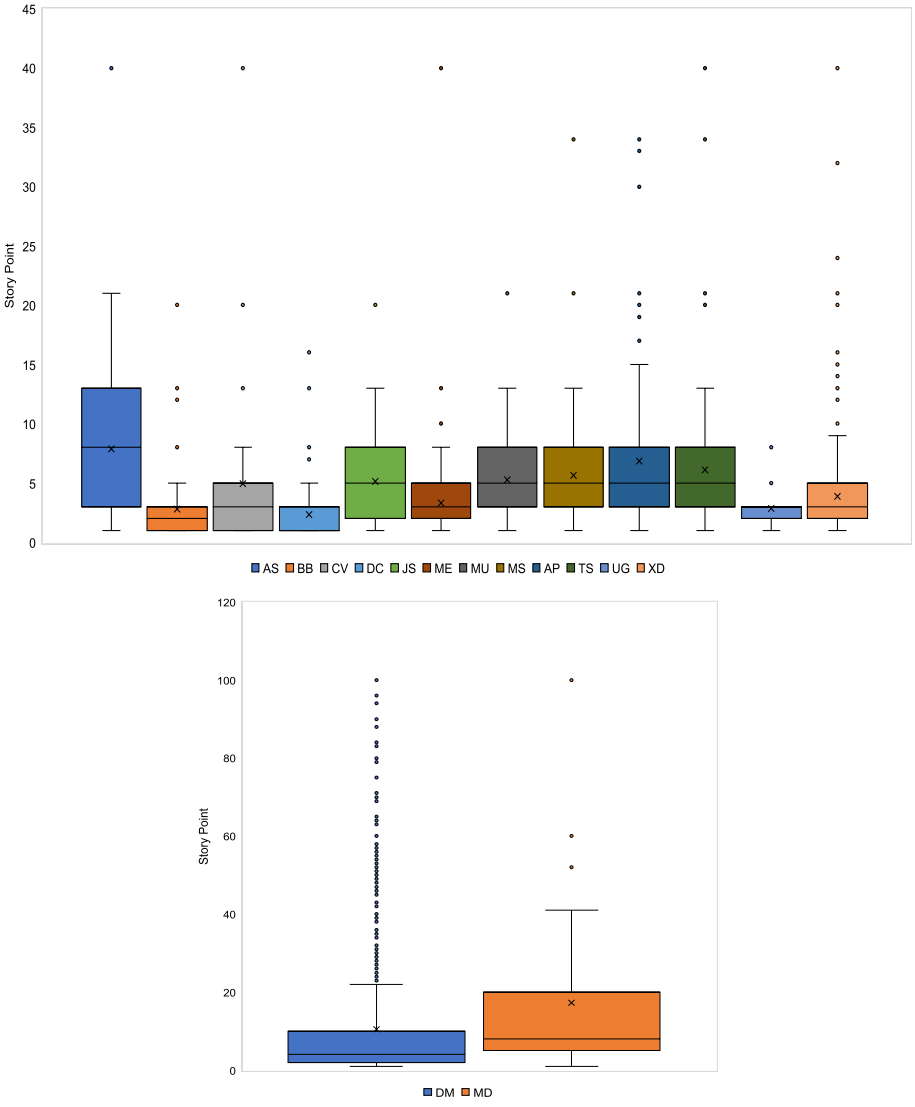


Fig. 9 Datasets Condition Before Preprocessing

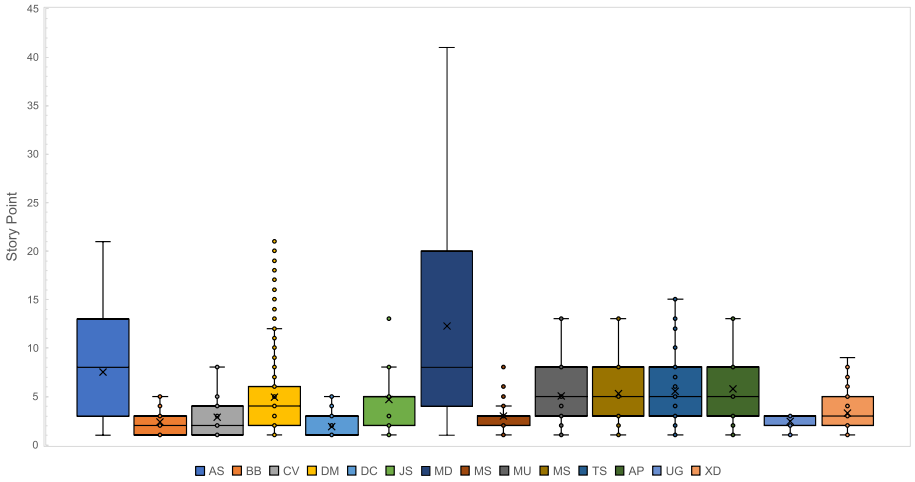


Fig. 10 Story Point Distribution for Case N After Preprocessing

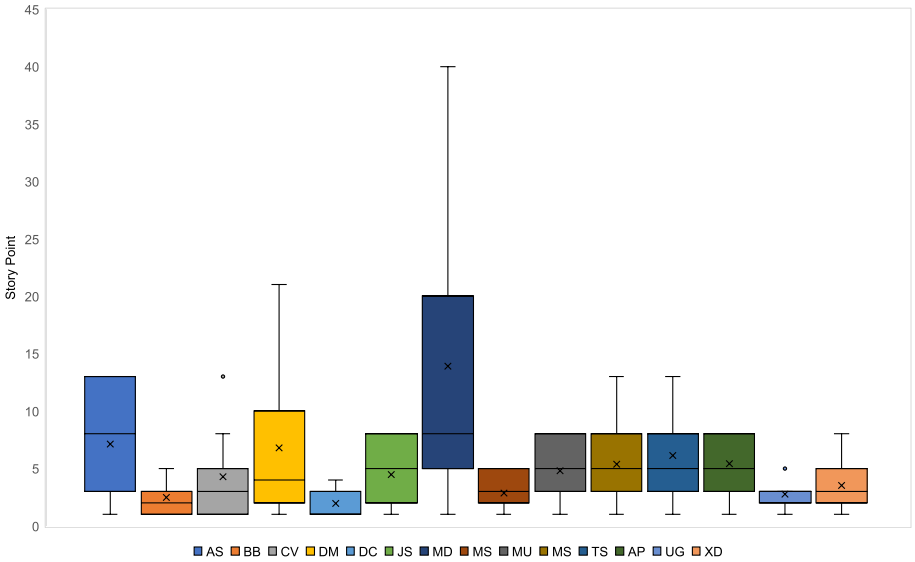


Fig. 11 Story Point Distribution for Case M After Preprocessing

Author Contributions Z.A. did most of the experiments, collected the data, and wrote the main manuscript. M.K. supported the research, discussions, and manuscript writing.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. This study was conducted without external funding.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

References

Adoma, A. F., Henry, N. M., & Chen, W. (2020). Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition. In: 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP). IEEE, pp 117–121, https://doi.org/10.1109/ICCWAMTIP51612.2020.9317379

Alsaqqa, S., Sawalha, S., & Abdel-Nabi, H. (2020). Agile software development: methodologies and trends. *International Journal of Interactive Mobile Technologies (iJIM), 14*(11), 246–270. https://doi.org/10.3991/ijim.v14i11.13269

Amalia, A., Sitompul, O. S., Nababan, E. B., et al. (2020). An efficient text classification using fastText for bahasa indonesia documents classification. In: 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA), pp 69–75, https://doi.org/10.1109/DATABIA50434.2020.9190447

Amasaki, S. (2023). On effectiveness of further pre-training on BERT models for story point estimation. In: Proceedings of the 19th International Conference on Predictive Models and Data Analytics in Software Engineering. Association for Computing Machinery, New York, NY, USA, pp 49–53, https://doi.org/10.1145/3617555.3617877

Arachchi, K. J. W., & Amalraj, C. R. J. (2023). An agile project management supporting approach for estimating story points in user stories. In: 2023 8th International Conference on Information Technology Research (ICITR), pp 1–6, https://doi.org/10.1109/ICITR61062.2023.10382930

Canedo, E. D., Aranha, D. P., Cardoso, M. D. O., et al. (2018). Methods for estimating agile software projects: Systematic literature review. In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE), pp 34–39

Choetkiertikul, M., Dam, H. K., Tran, T., et al. (2019). A deep learning model for estimating story points. *IEEE Transactions on Software Engineering, 45*(7), 637–656. https://doi.org/10.1109/TSE.2018.2792473

Choi, J., & Lee, S. W. (2020). Improving fasttext with inverse document frequency of subwords. *Pattern Recognition Letters, 133*, 165–172.

David, M. S., & Renjith, S. (2021). Comparison of word embeddings in text classification based on rnn and cnn. *IOP Conference Series: Materials Science and Engineering,1187*(1), Article 012029. https://doi.org/10.1088/1757-899X/1187/1/012029

Fu, M., & Tantithamthavorn, C. (2023). Gpt2sp: a transformer-based agile story point estimation approach. *IEEE Transactions on Software Engineering, 49*(2), 611–625. https://doi.org/10.1109/TSE.2022.3158252

Gheorghe, A. M., Gheorghe, I. D., & Iatan, I. L. (2020). Agile software development. *Informatica Economica*, *24*(2). https://doi.org/10.24818/issn14531305/24.2.2020.08

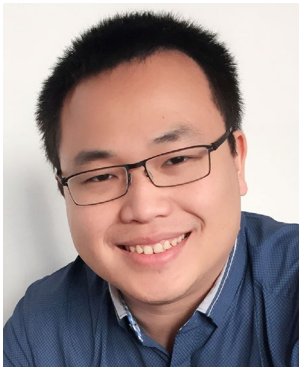
Grave E, Bojanowski P, Gupta P, et al (2018) Learning word vectors for 157 languages. arXiv:1802.06893
Huang L, Ma D, Li S, et al (2019) Text level graph neural network for text classification. arXiv:1910.02356
Kadu, M. L., Deshpande, D., & Pawar, D. (2022). Survey of deep learning approaches for twitter text classification. *International Journal of Advanced Engineering Research and Science9*(12), 106–112. https://doi.org/10.22161/ijaers.912.12
Karakaya, O., & Kilimci, Z. H. (2024). An efficient consolidation of word embedding and deep learning techniques for classifying anticancer peptides: fasttext+ bilstm. *PeerJ Computer Science,10*, Article e1831. https://doi.org/10.7717/peerj-cs.1831
Kassem, H., Mahar, K., & Saad, A. A. (2023). Story point estimation using issue reports with deep attention neural network. *e-Informatica Software Engineering Journal17*(1), 230104. https://doi.org/10.34808/230104
Kitchenham, B., Brereton, O. P., Budgen, D., et al. (2009). Systematic literature reviews in software engineering - a systematic literature review. *Information and Software Technology, 51*(1), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009
Li, Y. H., Harfiya, L. N., Purwandari, K., et al. (2020). Real-time cuffless continuous blood pressure estimation using deep learning model. *Sensors, 20*(19), 5606. https://doi.org/10.3390/s20195606
Liu, X., Zhao, W., & Ma, H. (2022). Research on domain-specific knowledge graph based on the roberta-wwm-ext pretraining model. *Computational Intelligence and Neuroscience, 2022*, 1–11. https://doi.org/10.1155/2022/8656013
Marapelli B, Carie A, Islam SMN (2020) Rnn-cnn model: A bi-directional long short-term memory deep learning network for story point estimation. In: 2020 5th International Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA), pp 1–7, https://doi.org/10.1109/CITISIA50690.2020.9371770
Mienye, I. D., & Sun, Y. (2022). A survey of ensemble learning: concepts, algorithms, applications, and prospects. *IEEE Access, 10*, 99129–99149. https://doi.org/10.1109/ACCESS.2022.3207287
Mora, E., Cifuentes, J., & Marulanda, G. (2021). Short-term forecasting of wind energy: a comparison of deep learning frameworks. *Energies, 14*(23), 7943. https://doi.org/10.3390/en14237943
Nazari F, Yan W (2021) Convolutional versus dense neural networks: comparing the two neural networks' performance in predicting building operational energy use based on the building shape. In: Proceedings of Building Simulation 2021: 17th Conference of IBPSA. KU Leuven, https://doi.org/10.26868/25222708.2021.30735
Perez, H., & Tah, J. H. M. (2020). Improving the accuracy of convolutional neural networks by identifying and removing outlier images in datasets using t-sne. *Mathematics, 8*(5), 662. https://doi.org/10.3390/math8050662
Phan, H., & Jannesari, A. (2022). Story point effort estimation by text level graph neural network. arXiv:2203.03062.
Porru, S., Murgia, A., Demeyer, S., et al. (2016). Estimating story points from issue reports. Association for Computing Machinery, New York, NY, USA, https://doi.org/10.1145/2972958.2972959
Satpathy T, et al. (2016). *A guide to the scrum body of knowledge (sbok™ guide)*. Scrumstudy™, a brand of VMEdu, Inc p 58.
Schön, E. M., Thomaschewski, J., & Escalona, M. J. (2017). Agile requirements engineering: a systematic literature review. *Computer Standards & Interfaces, 49*, 79–91. https://doi.org/10.1016/j.csi.2016.08.011
Soukaina, N., Soukaina, M., & Abdelaziz, M. (2022). Sqrum: An improved method of scrum. *International Journal of Advanced Computer Science and Applications13*(9). https://doi.org/10.14569/IJACSA.2022.0130928
Tan, K. L., Lee, C. P., Anbananthen, K. S. M., et al. (2022). Roberta-lstm: a hybrid model for sentiment analysis with transformer and recurrent neural network. *IEEE Access, 10*, 21517–21525. https://doi.org/10.1109/ACCESS.2022.3152828
Tawosi V, Al-Subaihin A, Moussa R, et al (2022a) A versatile dataset of agile open source software projects. Association for Computing Machinery, New York, NY, USA, MSR '22, p 707–711, https://doi.org/10.1145/3524842.3528029
Tawosi, V., Al-Subaihin, A., & Sarro, F. (2022b). Investigating the effectiveness of clustering for story point estimation. In: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp 827–838, https://doi.org/10.1109/SANER53432.2022.00101
Tawosi, V., Moussa, R., & Sarro, F. (2023). Agile effort estimation: have we solved the problem yet? insights from a replication study. *IEEE Transactions on Software Engineering, 49*(4), 2677–2697. https://doi.org/10.1109/TSE.2022.3228739
Ur Rehman, A., & Belhaouari, S. B. (2021). Unsupervised outlier detection in multidimensional data. *Journal of Big Data, 8*(1), 80.

Usman, M., Mendes, E., Weidt, F., et al. (2014). Effort estimation in agile software development: a systematic literature review. In: Proceedings of the 10th International Conference on Predictive Models in Software Engineering. Association for Computing Machinery, New York, NY, USA, p 82–91, https://doi.org/10.1145/2639490.2639503

Wahid, N. A., Bae, H., Adi, T. N., et al. (2021). Parallel-structure deep learning for prediction of remaining time of process instances. *Applied Sciences, 11*(21), 9848. https://doi.org/10.3390/app11219848

Wong, T. T., & Yeh, P. Y. (2020). Reliable accuracy estimates from k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering, 32*(8), 1586–1594. https://doi.org/10.1109/TKDE.2019.2912815

Zuhaimi Ahmad is a master's student at the School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand. His research interests include the use of artificial intelligence and machine learning to improve software engineering processes, software quality, and Agile software development methodologies.



Matthew M.Y. Kuo (Member, IEEE) received the B.E. (Hons.) and Ph.D. degrees in electrical and computer systems engineering from the University of Auckland, Auckland, New Zealand, in 2008 and 2015, respectively. He is currently a Senior Lecturer with the School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand. His current research interests include artificial intelligence applications, cyber-physical embedded systems, Internetof- Things, precision timed systems, industrial automation, and safety-critical systems.