applied sciences

MDPI

Article

Enhancing Agile Story Point Estimation: Integrating Deep Learning, Machine Learning, and Natural Language Processing with SBERT and Gradient Boosted Trees

Burcu Yalçıner [1,*], Kıvanç Dinçer [2], Adil Gürsel Karaçor [3] and Mehmet Önder Efe [1]

[1] Department of Computer Engineering, Hacettepe University, 06800 Ankara, Türkiye; onderefe@hacettepe.edu.tr
[2] Department of Software Engineering, Ostim Technical University, 06374 Ankara, Türkiye; kivanc.dincer@outlook.com
[3] HORTIAI PTY LTD. 5/1 Moore St, Canberra, ACT 2601, Australia; gursel.karacor@hortiai.com
* Correspondence: burcuyalciner@cs.hacettepe.edu.tr; Tel.: +90-532-605-9393

Abstract: Advances in software engineering, particularly in Agile software development (ASD), demand innovative approaches to effort estimation due to the volatility in Agile environments. Recent trends have made the automation of story point (SP) estimation increasingly relevant, with significant potential for enhancing accuracy. This study introduces a novel model for software effort estimation (SEE) utilizing a deep learning (DL)-based sentence-BERT (SBERT) model for feature extraction combined with advanced gradient-boosted tree (GBT) algorithms. A comprehensive evaluation shows that the proposed model outperforms standard SEE and state-of-the-art models, demonstrating a mean absolute error (MAE) of 2.15 and a median absolute error (MdAE) of 1.85, representing a 12% improvement over the baseline model and an 18% improvement over the best-performing state-of-the-art model. The standardized accuracy (SA) is 93%, which is 7% higher than the next best model, across a dataset of 31,960 issues from 26 open-source Agile projects. This study contributes to software engineering by offering a more accurate and reliable decision support system for estimating project efforts.

Keywords: Agile software development; software effort estimation; natural language processing; sentence-BERT; gradient-boosted trees; ensemble learning; machine learning; deep learning

Citation: Yalçıner, B.; Dinçer, K.; Karaçor, A.G.; Efe, M.Ö. Enhancing Agile Story Point Estimation: Integrating Deep Learning, Machine Learning, and Natural Language Processing with SBERT and Gradient Boosted Trees. *Appl. Sci.* **2024**, *14*, 7305. https://doi.org/10.3390/app14167305

Academic Editor: David Ruano Ordás

Received: 20 July 2024
Revised: 11 August 2024
Accepted: 15 August 2024
Published: 19 August 2024

1. Introduction

When the "Agile Manifesto" was published on 11–13 February 2001 [1], it marked a pivotal moment in the software development landscape. Traditional software development methods, including the waterfall model, v model, incremental model, and iterative model, were prevalent. These traditional methods, while prevalent, often fell short in accommodating the rapid changes and collaborative needs of modern software projects. Software organizations recognized the limitations of traditional methods, such as long development cycles, limited customer involvement, and difficulty in managing large projects. This recognition led them to embark on a paradigm shift towards Agile methodologies. Agile methodologies offered a more adaptable and responsive approach, meeting the evolving needs of software development. The aforementioned challenges, along with late defect detection, limited adaptability to change, rigidity, and stakeholder dissatisfaction, further fueled the shift towards Agile methodologies [2,3]. The transition to Agile, characterized by its flexibility, iterative development, and customer-centricity, emerged as a solution to address the aforementioned challenges facing traditional software development methods and improve overall software development efficiency. Shortcomings like long development cycles often led to project failures. Meanwhile, the limited adaptability to change hindered innovation and competitive advantages. High costs were incurred due to the

late detection of defects and the difficulty in managing large projects. Stakeholder dissatisfaction and risk management challenges underscored the need for a more dynamic and responsive approach. Agile methodologies, with their iterative development and focus on customer involvement, aimed to significantly reduce project failures and costs, enhance adaptability to change, foster innovation, and boost stakeholder satisfaction. As a result, many software organizations gradually shifted their development methodologies to embrace Agile, recognizing its potential to mitigate the damages associated with traditional approaches [4,5].

Despite transitioning to Agile as a solution to overcome the challenges and shortcomings of traditional software development methods, software organizations still face significant delays in project timelines and cost overruns. They also face critical issues that need to be addressed. Numerous studies in the literature highlight that most software projects struggle with timeline overruns and frequently exceed budgetary constraints. A McKinsey and University of Oxford study covering 5400 large-scale IT projects revealed alarming statistics: On average, projects ran 66% over budget and 33% overtime [6]. In a parallel vein, another study [7], this time scrutinizing 1471 software projects, indicated that one in six experienced a staggering budget overrun of 200% and a schedule overrun of nearly 70%. Additionally, the project cost management statistics for 2023 [8] revealed that 47% of Agile projects experienced delays or budget overruns or lead to customer dissatisfaction. Moreover, 11% of Agile projects faced outright failure, ultimately delivering no software product. These findings underscore the pervasive challenges in project management, accentuating the pivotal role of accurate effort estimation in ensuring timely completion and adherence to budget constraints in software projects.

In ASD projects, particularly during the inception of each sprint in sprint planning, the Scrum team employs expert-based estimation methods such as planning poker, T-shirt sizing, the Fibonacci sequence, the bucket system, and more to assess the complexity and effort associated with user stories or issues slated for development. However, these estimations are inherently susceptible to inaccuracies, as they heavily rely on individual knowledge and past experiences in evaluating the intricacies of similar stories or issues from prior projects. This introduces a notable challenge, as it opens the door to biases, hindering the team's ability to provide precise estimations. Consequently, this challenge of inaccurate estimations contributes to instances in which the entirety of the identified user stories or issues is not completed within each sprint, leading to budget and schedule overruns in the project. Given these challenges, there is a significant opportunity for research into automated techniques to estimate SPs with the aim of reducing the inaccuracies tied to expert judgment. Moreover, this pursuit is crucial, particularly for an Agile team, as it aims to foster consistent estimations across the entire life cycle of the project.

There are studies proposing model-based SP estimation using ML techniques, utilizing historical data with features designed to characterize a software project, but excluding SPs, issues, or user stories. These models solely predict the SP required to complete the entire software project rather than individual user stories or issues [9–24]. Few studies in the literature [25–31] specifically concentrate on estimating the SP of user stories or issues based on the similarities between their definitions, written in natural language, and those of previously completed user stories or issues.

In addition, the literature shows that software engineering data for effort estimation has substantial variability. This is due to the inherent complexity and dynamic nature of software development processes [32,33]. This variability is influenced by diverse factors, including a wide range of project characteristics such as size, scope, and technologies used, as well as the dynamic nature of team composition, changing requirements, technological advancements, human factors, and external influences. Additionally, variations in data collection methods contribute to the observed variability. Managing and comprehending this variability is crucial for precise SEE [34].

To mitigate this variability, a publicly available dataset [35] from 26 Agile projects, crafted by Tawosi et al. for SP estimation, was utilized in the proposed model. A decision-

making system is introduced by the proposed model to empower Scrum teams, enabling them to predict the SPs of user stories or the issues they define. The distinctive innovation of the proposed method lies in transforming unstructured data into a structured form using robust and powerful transformers. To achieve this, the potent DL technique SBERT was employed. Furthermore, GBTs, known for their efficacy in handling such data, were utilized to forecast the SPs of user stories or issues.

To assess the efficiency of the proposed model, a rigorous 10-fold cross-validation technique and the validation data were employed. MAE and MdAE were chosen as the primary evaluation metrics, reflecting the average magnitude of errors in predictions without considering their direction. Additionally, SA, which is derived from MAE and MdAE, was used. It provides a normalized measure of the model's precision across diverse project scales. This comprehensive evaluation framework aimed to provide a robust understanding of the model's performance across various dimensions.

In a rigorous evaluation across 26 Agile projects, the model utilized SBERT's advanced transformers to convert unstructured data written in natural language from user story titles and descriptions into structured form. It yielded the lowest MAE and MdAE values, ranging between 2.10 and 2.25 for MAE and 1.80 and 2.00 for MdAE across 10-fold cross-validation.

This process further enriched the dataset by numerically encoding 'issue types' with label encoding and 'components' with multiple one-hot encoding techniques. The prepared dataset was then used to train the light gradient-boosting machine (LightGBM) algorithm, a variant of GBTs, which is named as **SBERT-LGBM$_{TC}$**. **SBERT-LGBM$_{TC}$** demonstrated exceptional prowess by outshining both conventional estimation methodologies and four cutting-edge models. Notably, in 17 distinct projects, the model surpassed all compared models. Furthermore, a substantial enhancement with a significant effect size was observed in three projects, showcasing the model's robustness and ability to markedly improve estimation accuracy.

This comprehensive analysis confirms the **SBERT-LGBM$_{TC}$** model's quantitative superiority in estimation performance, demonstrated by significant improvements in MAE (2.15) and MdAE (1.85) metrics, resulting in a 12% improvement over the baseline model. The SA was 93%, setting a new benchmark in SP estimation and providing more reliable project management and planning.

The structure of this paper unfolds as follows: Section 2 offers an in-depth review of studies conducted on effort estimation. Section 3 provides a concise overview of the methodologies employed in developing the proposed model. The intricacies of the proposed model are elucidated in Section 4. The empirical study design is delineated in Section 5, while Section 6 presents the results of the proposed model. An exhaustive experimental evaluation is expounded in Section 7. Potential threats to validity are meticulously addressed in Section 8, and concluding remarks and paving the way for future work are encapsulated in Section 9.

2. Related Work

Research on effort and size estimation in ASD has a rich history spanning several decades. These endeavors have been meticulously identified and categorized using a classification framework proposed by various studies [26] and with visual representations serving as guidelines [36]. Leveraging insights from these comprehensive studies, techniques for estimating effort or size in ASD can be broadly classified into three distinctive groups: expert-based estimation methods, data-based estimation methods, and combination-based estimation methods. The latter category integrates both expert-based and data-based estimation methods for a more nuanced and comprehensive estimation process.

Several studies in the literature have employed expert-based estimation methods for forecasting effort or size in ASD. These methods heavily depend upon expert judgment, drawing upon the wealth of experience and insights possessed by knowledgeable individuals. Among the widely used expert-based estimation methods are planning poker,

expert judgment, and wideband Delphi. To estimate effort or size in the context of ASD, some studies [27,37–52] utilized planning poker, while expert judgment was preferred in others [39,43,53–58]. Additionally, wideband Delphi was applied in studies [37,48,49,59] to estimate effort or size in ASD. While expert-based models harness the wealth of experience from seasoned professionals, they pose inherent challenges marked by subjectivity and potential biases.

To counter these limitations, data-based estimation methods have emerged, presenting notable advantages in ASD. Leveraging quantitative patterns from historical data, these methods introduce objectivity by minimizing individual subjectivity and biases. Their consistency in delivering results, scalability for handling large datasets, and adaptability to evolving project conditions make them invaluable tools. The automation potential, especially in ML and neural network (NN)-based approaches, reduces the reliance on constant expert involvement—a contrast to expert-based methods, which are often dependent upon ongoing expert input for accurate estimations. Data-based approaches further provide the benefit of reduced bias, being less influenced by individual opinions and experiences. The incorporation of quantitative metrics enhances the precision and thoroughness of data-based estimation methods. However, the choice between data-based and expert-based methods relies on contextual factors, available data, and the specific requirements of the estimation task, with some scenarios favoring a hybrid approach.

Among data-based estimation methods, ML techniques such as k-nearest neighbor (KNN), random forest (RF), decision tree (DT), support vector machine (SVM), stochastic gradient boosting (SGB), and naive Bayes (NB) have been extensively explored [27–30,39,40,42,54,60–66]. NNs, including artificial neural network (ANN), deep belief network (DBN), fuzzy neural network (FNN), long–deep neural network, feed-forward, multilayer perceptron (MLP), etc., have been widely studied as well [30,41,60,67–76]. Other studies exploring data-based methods include functional size measurement (FSM), which was applied in [41,44,46,53,61,77–83]; regression, which was utilized in [41,58,60,73,80,84]; algorithmic methods, which were used in [78,85–88]; fuzzy logic (FL), which was applied in [89–91]; swarm intelligence (SI), which was used in [92–94]; and Bayesian network (BN), which was employed in [60,95,96]. Additionally, specific studies [57,74,97] have utilized data-based estimation techniques, namely COCOMO II, Monte Carlo, and principal component analysis (PCA), respectively, and there are some studies [48–50,76,98–101] which utilized models based on both expert opinion and data, namely combination-based estimation methods.

Combination-based estimation methods offer a hybrid approach, incorporating the strengths of both expert-based and data-based methods. By integrating expert judgment with quantitative insights from historical data, these methods aim to mitigate the limitations of individual approaches. They leverage the complementary aspects of expert knowledge and empirical patterns, thus providing a more robust foundation for software effort and size estimation. This hybridization allows for increased flexibility, adaptability, and potentially enhanced accuracy in addressing the challenges posed by the inherent subjectivity of expert-based methods and the variability found in data-based approaches. The combination-based methods explore synergies to achieve a comprehensive and balanced estimation strategy, offering a nuanced and context-aware approach to project planning and management.

The studies discussed so far have predominantly centered around estimating the effort needed to complete an entire software project or have relied upon historical data derived from completed projects. In contrast, the proposed model places a primary emphasis on predicting the effort required for each user story or issue to be developed iteratively, specifically within a sprint. This involves exploring semantic similarities between each user story or issue and historical data containing natural language descriptions of user stories and issues, along with SPs estimated by human expertise. Studies within this specific focus have been thoroughly investigated in order to obtain guidelines and propose an innovative model that has better performance than others.

In the pursuit of predicting the SPs of a user story, the initial automated method was introduced by Abrahamsson et al. [26]. They extracted features, including the number

of characters, presence of keywords (the 15 most frequent terms with their probabilities), and priorities from user stories collected from two industrial Agile projects. The first project comprised 1325 user stories in a customized version of extreme programming (XP), while the second project included 13 user stories in XP. These features were used to train algorithmic models, such as SVM, linear regression (LR), and radial basis function (RBF). For model evaluation, they employed leave-one-out (LOOV) cross-validation [36], and various metrics, including magnitude of relative error (MRE), prediction at level k (PRED(k)), magnitude of error relative (MER), and standard deviation (SD) [102]. Comparisons were made with their previous study [103], which focused on estimation based on design metrics and developers' subjective estimates. The results indicate that the extracted feature set is effective at achieving good accuracy and that their model performs as well as, if not better than, developers' subjective estimates, especially for large datasets.

Porru et al. [27] introduced an ML classifier designed to estimate the SPs necessary for addressing an issue. They conducted feature extraction from 4908 issues sourced from one industrial project and eight open-source projects. The features included the issue type, related components, and term-frequency inverse document frequency (TF-IDF). Issues from the industrial project were obtained through JIRA, a widely used issue-tracking system, with the TF-IDF being derived from the summary and description fields. These features were employed in SVM, KNN, DT, and NB classifiers, chosen for their suitability in text classification applications. To validate their ML classifiers, a 10-fold cross-validation was applied, and performance metrics such as mean magnitude of relative error (MMRE) and accuracy were assessed. Their results highlight that fields containing the summary, description, names of related components, and issue type in an issue report provide relevant features for accurate SP estimation. Furthermore, their findings suggest that training the classifiers with more than 200 issues is preferable in order to ensure a stable model with satisfactory accuracy before deploying it for SP estimation.

Scott et al. [28] embarked upon another classification task to estimate SPs. The authors undertook feature extraction encompassing both developer-based and text-based features from 4142 user stories sourced from eight open-source projects. Developer-based features included developers' reputation, current workload, total number of issues, number of developer comments, and total SPs. Text-based features comprised context, number of characters, and n-grams essential for calculating TF-IDF. The features were utilized to train an SVM for classification. The performance of their predictive model was evaluated using a 10-fold cross-validation and metrics such as accuracy, MAE, and SA. Their findings highlighted a slight out-performance of models trained with developer-based features compared to those trained solely with text-based features. Additionally, the results indicated that using an SVM classifier and exclusively relying upon developer features for predicting SPs surpass the performance achieved by using textual features alone or a combination of textual and developer features.

In a distinctive classification pursuit, Soares [29] conducted a study using four diverse variants of auto-encoder networks in order to estimate the SPs of user stories by relying upon their semantic similarities obtained from textual data contained in issue summaries. The author also fine-tuned these four variants through an optimization process facilitated by randomized search. Intriguingly, the study's outcomes revealed no discernible performance disparity among these four variants, which was attributed to the inherent semantic simplicity of user stories. Evaluation of the study encompassed a dataset featuring 3439 issues extracted from six open-source Agile projects and employing rigorous 10-fold cross-validation for each model variant.

Choetkiertikul et al. [30] introduced an end-to-end prediction system, named DeepSE, designed to estimate SPs for user stories. The system leverages the synergy of two potent DL architectures: long short-term memory (LSTM) and recurrent highway network (RHWN). The textual information from user story titles and descriptions was amalgamated into a unified text representation, with each text segment being transformed into a vector representation through the LSTM layer. Subsequently, a differentiable regressor

was employed for the immediate prediction. The RHWN assisted in mapping the deep representation of the vector extracted from the LSTM layer. Finally, a linear activation function in a feed-forward neural network served as the ultimate regressor for producing the SP estimation. The proposed system underwent fine-tuning for crucial hyperparameters, including the number of word-embedding dimensions and the number of hidden layers in the RHWN component. Evaluation utilized a dataset comprising 23,313 issues sourced from nine major open-source repositories through the JIRA issue tracking system. Deep-SE was compared with baseline methods (mean and median), as well as other models like RF, SVM, automatically transformed linear model (ATLM), and linear regression model (LRM) coupled with LSTM, based on metrics such as MAE, MdAE, and SA. The results demonstrated that Deep-SE significantly outperformed both baseline methods and the model proposed by Porru et al. [27] in terms of MAE.

Abadeer and Sabetzadeh [104] conducted a comprehensive evaluation study of Deep-SE [30] using a dataset comprising 4727 user stories extracted from a commercial project. The project involved a 27-member Agile team working on a data anonymization product at IQVIA, a healthcare data science company. The study aimed to assess the generalizability of Deep-SE in non-open-source environments. Comparing its results with baseline models revealed that Deep-SE significantly outperformed these models, showcasing its effectiveness with proprietary data. Importantly, the findings suggested that Deep-SE was versatile and applicable not only within the confines of a single company, but also in cross-company scenarios.

In their replication study, Tawosi et al. [105] sought to investigate the efficacy of Deep-SE for both within-project and cross-project effort estimation. The study utilized the Deep-SE model along with an additional dataset consisting of 31,960 issues obtained from TAWOS [35]. Deep-SE was compared with three baseline estimators (random, mean, and median) and a previously proposed method for ASD effort estimation named TF/IDF-SVM [27]. Contrary to previous findings on the effectiveness of Deep-SE, the replication results indicated that Deep-SE only outperformed the median baseline estimator and TF/IDF-SVM in very few cases with statistical significance. This divergence in results added complexity to the understanding of Deep-SE's performance across different scenarios.

Tawosi et al. [31] introduced an innovative approach for SP estimation by leveraging the textual features of issues. This method utilized both topic modeling and clustering techniques to enhance prediction accuracy. Latent Dirichlet allocation (LDA) was employed to extract topic similarities from the textual content found in titles and descriptions. Additionally, hierarchical clustering was applied to group issues based on their topic similarities. Subsequently, three estimation models were investigated within each cluster: cluster-mean-based estimation, which provides the mean SP of all issues in the cluster; cluster-median-based estimation, which offers the median SP of all issues in the cluster; and closest point-based estimation, returning the SP of the closest issue to the new incoming issue. Evaluation of the proposed approach involved a dataset comprising 31,960 issues from 26 projects across 13 different open-source repositories. Comparative analysis against baselines (random guessing [106] and mean and median method [19,107,108]) and two state-of-the-art models, Deep-SE [30], and TF-IDF-SE [27], was performed using metrics such as MAE, MdAE, and SA. The results demonstrated the superiority of the proposed approach over both baselines and state-of-the-art models.

Fu et al. [109] proposed GPT2SP, a transformer-based Agile SP estimation approach that leveraged a transformer-based architecture to enhance SP estimation. The model used a GPT-2 pre-trained language model, enabling it to capture the relationships among words while considering the context surrounding each word and its position in the sequence. Comparative analysis using MAE demonstrated significant improvements in estimation accuracy for both within-project and cross-project scenarios, outperforming other baseline approaches, including Deep-SE. Moreover, GPT2SP includes an explainability component, which has been shown to increase the perceived usefulness and trustworthiness of AI-based SP estimations among practitioners.

The study proposed by Sousa et al. [110] explored the application of ML techniques for task effort and duration estimation in software projects. The authors conducted experiments using datasets from three project management tools, training regression models with eight different ML algorithms. They found that ensemble algorithms like RF, extra trees regressor, and XGBoost consistently outperformed non-ensemble ones. The study highlighted the potential of ML-based estimation methods to provide accurate and efficient estimates for individual software tasks, although the quality of the estimates varied significantly depending on the characteristics of the datasets and projects. This research contributed to the growing body of literature on ML applications in software engineering, particularly in the context of task-level estimation.

3. Background

3.1. Scrum

Scrum, a renowned Agile framework, orchestrates projects through fixed-length iterations known as sprints, typically spanning two to four weeks. Each sprint is dedicated to delivering a potentially shippable product increment, promoting constant integration and responsiveness to evolving requirements [111].

In order to efficiently manage and develop software projects, user stories (which serve as the presentation format for software requirements including features, key issues, changes, enhancements, and fixes) are identified, prioritized, and then placed into a dynamic, prioritized list known as the product backlog within the Scrum framework [112,113]. Each sprint involves the selection of user stories by developers, aiming to complete them within a maximum of two weeks. Before the sprint commences, the complexity and effort required for the chosen user stories are estimated using the SP metric during sprint planning [112,113]. This estimation aids teams in assessing the workload, planning sprints more effectively, and allocating resources more efficiently.

3.2. Sentence Transformers

Sentence transformers represent a transformative paradigm in NLP specifically tailored to generate meaningful sentence embeddings. These embeddings serve as dense numerical representations of sentences, capturing their semantic meaning and facilitating various downstream tasks such as text classification, similarity analysis, and information retrieval. There are different types of sentence transformers available, namely SBERT, universal sentence encoder (USE), InferSent, MobileBERT, DistillBERT, language-agnostic BERT sentence embedding (LaBSE), etc. [114]. While BERT (bidirectional encoder representations from transformers) excels in contextualized word embeddings, SBERT refines this capability to the sentence level. This specialization enables SBERT to outperform generic BERT embeddings in tasks that demand a nuanced understanding of sentence semantics [115].

SBERT: This model is designed to capture sentence-level representations and goes beyond traditional word embeddings by considering the contextual nuances of entire sentences. Developed by researchers at the UKP Lab, SBERT leverages a Siamese or triplet network architecture to learn powerful sentence embeddings. SBERT uses a Siamese network structure, in which two identical NNs share weights and learn to map sentences into a high-dimensional space. This architecture encourages the model to understand the semantic relationships between sentences. Additionally, SBERT introduces a novel training objective, employing contrastive learning to enhance the discrimination between similar and dissimilar sentence pairs. The significance of SBERT lies in its ability to generate semantically meaningful sentence embeddings. Unlike traditional embedding methods that treat sentences as mere collections of words, SBERT considers the contextual interactions between words, resulting in embeddings that encapsulate the entire sentence's overall meaning. This is particularly valuable in tasks n which understanding the semantic similarity or dissimilarity between sentences is crucial. All of these factors make SBERT a powerful tool in the realm of NLP, contributing to enhanced performance in various applications [115].

3.3. Estimation Model

GBTs: GBTs represent a powerful ensemble learning technique that has gained prominence in various ML applications. GBTs are particularly effective in regression and classification tasks, leveraging the strength of DTs to create a robust predictive model [116]. GBTs are an ensemble of DTs in which each tree is constructed sequentially in order to correct the errors of the preceding ones. The model starts with a single, relatively simple tree and subsequently adds more trees, each focusing on the mistakes made by the ensemble so far [116]. The combination of these weak learners results in a strong predictive model. In each iteration, a new tree is added to the ensemble, and its predictions are combined with the predictions of the existing trees [116]. The key innovation lies in adjusting the weights of the observations, giving more emphasis to the instances where the model has performed poorly. This iterative process continues until a predefined number of trees is reached or until further tree additions cease to improve performance.

Recent studies have reinforced the preeminence of tree-based models like GBTs in handling structured data. Extensive benchmarks on various datasets demonstrate that tree-based models, including GBTs, continue to be the state-of-the-art for medium-sized data, outperforming standard and novel DL methods, even without considering their superior speed [117].

Some specific strengths of GBTs make them a suitable choice to be used in SP estimation models. GBTs are versatile and can handle both regression and classification tasks effectively. GBTs inherently capture non-linear relationships in the data, making them suitable for complex, real-world problems. The sequential construction of trees allows GBTs to adapt and learn from errors, leading to robust and accurate predictions. GBTs provide a measure of feature importance, allowing users to identify which variables contribute most significantly to the model's predictive power [118,119].

The other reason why we chose GBTs as SP estimation models is that GBTs exhibit exceptional performance with structured data, making them a popular choice for text-based regression and estimation problems. Their ability to handle a mix of numerical and categorical features, capture non-linear relationships, and effectively manage large datasets aligns well with the characteristics of structured data. GBTs provide:

- **Categorical Feature Handling**: They can naturally handle categorical features without the need for one-hot encoding, simplifying the preprocessing steps for structured data and outperforming other algorithms [117];
- **Robust Performance**: They often outperform other algorithms when faced with structured data due to their adaptability and ability to capture intricate patterns [117];
- **Predictive Accuracy**: They excel at achieving high predictive accuracy, making them particularly valuable in scenarios in which precise predictions are crucial [117]; and
- **Interpretability**: While not as inherently interpretable as simpler models, GBTs provide insights into feature importance, aiding in understanding the underlying patterns in tabular datasets [117].

In conclusion, GBTs represent a formidable ML tool, especially when dealing with structured data, like in the problem of SP estimation based on user story or issue texts.

4. The Proposed Model

The proposed model hinges on feature extraction from project issues, estimating unknown SP values through comparison with issues that share semantic similarities. This involves transforming issue text into semantically meaningful vectors, facilitating semantic similarity comparisons using cosine similarity.

The process incorporates advanced NLP techniques to examine word contexts and relationships, determining semantic similarities. The SBERT model, known for its sentence-level embeddings, was selected for its ability to capture sentence nuances more effectively than traditional BERT models that focus on token-level embeddings. This feature is crucial for accurately representing the complex nature of user story titles and descriptions.

SBERT's design, optimized for similarity tasks, makes it suitable for this model, aiding in semantic comparisons crucial for SP estimation. By calculating cosine similarity between sentence embeddings, semantic proximity among textual data from different issues is quantified, grounding SP estimations in a deep understanding of textual semantics.

For regression tasks, the model is prepared to work with gradient-boosted models. Therefore, three types of GBTs—extreme gradient boosting (XGBoost), LightGBM, and CatBoost—are selected for their proven effectiveness in similar tasks. These algorithms are trained on structured data derived from SBERT to predict SPs from Scrum software project text entries.

During training, challenges like balancing the dataset to avoid bias and capturing the semantic essence of diverse terminologies are encountered. Overcoming these requires innovative strategies and advanced techniques to ensure model robustness.

After training, the most suitable model is selected based on its accuracy using a validation subset from the TAWOS [35] dataset. This selection is guided by the need for a model that not only demonstrates high analytical accuracy but also aligns with practical project management needs.

4.1. Text Preprocessing

To analyze the issues, the 'title' and 'description' sections of each issue report are combined to form a unified text document for each issue within a project. Basic text preprocessing, such as removing embedded URLs and special characters while retaining punctuation, is applied to these documents. Subsequently, each text document is converted into a word-embedding vector featuring low-dimensionality, real-valued attributes and continuous features suitable for complex analytical tasks. The primary goal is to create a structured model using transformers, which are known for their prowess in managing complex, unstructured data. Initially, English stop words, code snippets, punctuation, and words shorter than two letters are retained. Tests are conducted both with and without these textual elements in order to assess this decision's impact. The findings indicate that removing these elements does not significantly alter the accuracy. Consequently, the decision is made not to exclude these elements to preserve the texts' original meaning. Moreover, stemming and lemmatization are excluded in order to avoid oversimplification and reduce computational costs, ensuring the preservation of the original terms and contextual integrity. Guided by research [120] indicating potential accuracy reduction due to over-stemming and semantic alteration from stemming and lemmatization, these techniques are omitted to ensure result precision. This cautious approach to linguistic preprocessing aims to maintain the textual data's integrity and enhance the subsequent analyses' reliability.

4.2. Feature Extraction

After completing the preprocessing process outlined above, the SBERT model is applied to the cleaned and preprocessed texts in order to obtain word embeddings, or in other words, vector representations of the texts. These word embeddings are derived from large transformer models within the SBERT model. These larger transformer models contribute significantly to the accuracy of word embeddings by leveraging vast amounts of data and sophisticated architectures to understand the context and nuances of language. They are trained on diverse corpora, enabling them to grasp subtle semantic relationships and the contextual use of words in various linguistic settings. By processing text through layers of attention mechanisms and NNs, they generate embeddings that capture the meaning of individual words and how those meanings change in different sentences. Using these large transformer models, the BERT model first obtains contextual embeddings for each token in each text. Then, these contextual embeddings from BERT are pooled using pooling strategies to form a single fixed-size sentence embedding. This pooling process condenses complex data into a manageable form, ensuring that critical semantic

insights are retained within a compact representation and thereby balancing efficiency with depth of understanding.

Ultimately, the process of generating structured data, or more precisely, probabilities of semantic similarities, is facilitated by these word embeddings. This is achieved through a comparison of the angle between two embedding vectors, a method known as cosine similarity. This measure provides a quantifiable metric of how closely related the meanings of two sentences are, even when their surface structures differ. In essence, the similarity score reflects the semantic similarity between the two sentences, indicating that high similarity scores suggest similar meanings, even if different words are used. This feature is particularly beneficial for applications with the objective of finding semantic similarity, such as semantic search, text clustering, and information retrieval.

The proposed model leverages SBERT to predict the SPs based on the similarities of semantically similar sentences. Since similar user stories often require tasks with similar sizes and complexities, it is assumed that the amounts of effort needed for their implementation should be close.

4.3. Estimation Models

Once the features are extracted from the structured data, SP estimation models are constructed using these extracted features. Given that it is a regression problem, the GBT models are deemed suitable for the dataset. Three powerful GBT models were applied:

1. **Catboost**: Known for efficiently handling categorical features, CatBoost excels in automatic categorical feature conversion and provides robust handling without the need for manual preprocessing [121]. Given the dataset's inclusion of categorical variables, CatBoost is a fitting choice.
2. **Xgboost**: Another GBT model with improved speed and performance, XGBoost incorporates regularization techniques to prevent overfitting [122]. Its flexibility in defining custom optimization objectives makes it a valuable addition to SP estimation models.
3. **LightGBM**: Utilizing a histogram-based learning approach, LightGBM proved effective in handling large datasets. Its capability to handle categorical features, coupled with outstanding parallelization, significantly improved performance and reduced memory usage [123].

   These models were implemented and trained in Python 3.8 using the CatBoostRegressor 1.2.5 (https://catboost.ai/, accessed on 18 August 2024), XGBRegressor 2.1.1 (https://xgboost.readthedocs.io/en/stable/, accessed on 18 August 2024), and LGBMRegressor 4.5.0 (https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html, accessed on 18 August 2024) libraries, respectively.

For the optimization of these GBT models, various techniques are employed, including **manual tuning** [124] and **automated hyperparameter tunings** [124]. For automated hyperparameter tuning, operations such as **random search** [124] and **Bayesian optimization** [124] were implemented using libraries such as RandomizedSearchCV 1.5.1 (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html accessed on 18 August 2024), optuna 3.6.1 (https://optuna.org/ accessed on 18 August 2024), and hyperopt 0.2.7 (https://pypi.org/project/hyperopt/ accessed on 18 August 2024). In all these approaches, the root mean square error (RMSE) was utilized as the objective function to assess and guide the optimization of the GBT models.

After evaluating each model's performance and optimizing models using different hyperparameter tuning techniques with a focus on their SA in estimating SP values, the model that demonstrates the best performance is selected as the primary model. After the completion of experiments, the **SBERT-LGBM$_{\mathbf{TC}}$** model with manual tuning is selected as the primary model.

This comprehensive model ensures a thorough exploration of the capabilities of each model, taking into account both manual and automated hyperparameter tuning strategies to enhance the accuracy of SP estimation in the context of Scrum software projects.

Figure 1 illustrates the flowchart of our proposed model. A detailed diagram of the proposed model is provided in Appendix A.

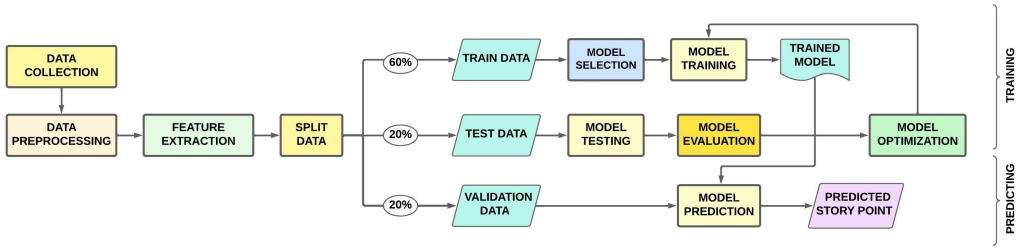

Figure 1. The flowchart of the proposed model.

5. Empirical Study Design

The primary objective of this study is to develop a decision-support system that takes user stories written in natural language as input and provides estimated SP values, allowing for the identification of complexity and effort required for implementation. In order to achieve the primary objective of this study, the identification of semantic similarity between categorical features is evaluated, and estimation techniques conducive to the goal and dataset are determined. After that, the accuracies of SP estimation models are thoroughly investigated based on the estimated SP values to assess the model's performance. Furthermore, it is explored whether incorporating additional features related to user story definition in either the feature extraction or estimation process enhances the overall performance of the proposed model. Additionally, optimizing models through various hyperparameter tuning techniques plays a crucial role in refining the estimation process. A comparative analysis of different model variations aimed at improving estimation accuracy is conducted. Upon the identification of the best-performing model, it is compared with Random guessing, mean, median estimators, and four other state-of-the-art techniques (TF-IDF-SE, DEEP-SE, LHC-SE, and LHC_{TC}-SE) in the literature within the domain of SP effort estimation using NLP techniques. The detailed empirical study design aspect is described below.

5.1. Research Questions

While refining the proposed model, a comprehensive roadmap was devised, elucidating detailed research questions. Crucial observations, implementations, experiments, and enhancements have been executed to address these questions.

RQ1: *Does the proposed model, which involves both feature extraction and estimation techniques, contribute to the accurate estimation of SPs for the identified issues?*

In order to examine this inquiry and to assess the viability of the suggested model combining feature extraction and estimation techniques, a thorough analysis has been carried out comparing it to conventional estimation methods (such as random guessing, mean, and median estimators) as well as four cutting-edge models (LHC-SE, LHC_{TC}-SE, Deep-SE, and TF-IDF-SE). These models attempt to estimate the SPs of the issues from unstructured data using different techniques and approaches. This comparative analysis aims to provide a clearer understanding of the model's impact on the accuracy of SP estimation.

RQ2: *Does adding other features consisting of a single word or a few words, namely 'issue type' and 'components', to the feature extraction process improve the estimation accuracy of the proposed model?*

Additional features, including textual data represented as single words or short phrases, are integrated into the feature extraction process for further investigation. Three different variations are explored:

1. In the first variation, only 'issue type', such as bugs, tasks, enhancements, or user stories, is added to the feature extraction process. 'Issue type' serves as the categorization or classification of a task within a system or project management framework. It

aids in organizing and prioritizing work, providing a means to distinguish between various types of tasks within a project or system.
2. In the second variation, only 'components', such as user interface (UI), runtime, etc., referring to modular and reusable units of code or software that perform specific functions, are added to the feature extraction process.
3. In the last variation, both 'issue type' and 'components' are included in the feature extraction process.

These variations are applied to investigate whether adding 'issue type' and/or 'components' to the feature extraction process enhances the accuracy of estimation. The goal is to understand the impact of these additional features on the semantic similarities of the issues and on the model's estimation performance.

RQ3: *Does adding 'issue type' and 'components'—which consist of a single word and a few words, respectively—to the estimation process instead of the feature extraction process improve the estimation accuracy of the proposed model?*

During the investigation, instead of integrating additional features—'issue type' and 'components'—into the feature extraction process, they are incorporated into the features extracted from 'title' and 'description_text'. Since 'issue type' is a categorical variable with unique and meaningful categories or labels that forms a predefined set of distinct categories, 'label encoding' is employed to convert it into a unique integer or numerical value. In contrast, 'components', a categorical variable with multiple categories separated by the '|' sign, is handled by utilizing 'one-hot encoding' to convert these categories into numerical values. This approach allows for the integration of the categorical information from 'issue type' and 'components' into the feature set derived from the 'title' and 'description_text' fields, enhancing the representation of the textual data in the analysis.

5.2. Dataset

In this study, a subset of the TAWOS [35] dataset has been utilized, encompassing 31,960 issues extracted from 26 projects developed using Scrum as the software development methodology. These projects originated from 13 diverse open-source repositories. Derived from a reputable source, the TAWOS dataset ensures a broad representation of scenarios across these projects, enhancing the reliability of the data. The dataset includes crucial information such as 'title', 'description', 'issue type', 'components', and 'SPs' for each issue within the projects. The first four features pertain to textual data, while the last one consists of numeric values representing SPs. Notably, the SPs in the dataset are estimated by software experts, adding to the dataset's quality and reliability. Table 1 provides descriptive statistics of the data. Figure 2 visually illustrates the comparative average SPs across various projects, clearly highlighting the differences in workload or complexity as represented by SPs. It offers a succinct overview of how SPs vary from one project to another, potentially reflecting the diverse nature of tasks or challenges inherent in each project. Figure 3 visually illustrates the distribution of SPs across all projects, showing a clear concentration of SPs in the lower range, with a significant drop-off as the SPs increase. This suggests that most tasks are of lower complexity, with fewer tasks requiring a higher number of SPs, which could indicate more complexity or effort required.

For each project, the issues are arranged in ascending order based on their creation time and are subsequently divided into two subsets–training and validation—with an 80% and 20% ratio, respectively. This standard 80–20% split is a widely accepted practice in ML, ensuring that models are trained on a representative sample and validated on a separate set to assess their generalization ability. The textual data of issues for each project in both the training and the validation subsets are amalgamated, resulting in a single text dataset for each project. This amalgamation ensures consistency in model training and validation, highlighting the meticulous approach to the analytical process. These are recorded as two datasets, namely 'train_text' and 'validation_text'. The 'train_text' dataset is utilized for the training phase. During this phase, the dataset is further divided into two subsets—train and text—with an 80% and 20% ratio, respectively. Meanwhile, the 'validation_text' dataset

(comprising 20% of the initial TAWOS subset) serves the purpose of validating the trained model's performance. This comprehensive approach, considering both textual and numeric features, results in a well-defined training process followed by a robust evaluation and validation. By leveraging the comprehensive nature of this approach, the methodology's robustness is underscored, and alignment with established data science practices is ensured, guaranteeing the accuracy of SP estimation across multiple projects without the need for separate models for each project.

Table 1. Story point statistics for various projects.

Repository	Project Key	Project	#Issues	Mean	Std	Min	25%	50%	75%	Max
Apache	MESOS	Mesos	1513	3.15	2.14	0	2	3	5	13
Appcelerator	ALOY	Alloy	251	3.71	2.32	0	3	3	5	13
Appcelerator	TISTUD	Appcelerator Studio	2794	5.48	3.23	0	3	5	8	40
Appcelerator	APSTUD	Aptana Studio	476	7.93	7.19	0	5	8	8	100
Appcelerator	CLI	Command–Line Interface	293	3.18	2.30	0	1	3	5	13
Appcelerator	DAEMON	Daemon	205	5.58	3.76	1	3	5	8	13
Appcelerator	TIDOC	Documentation	1005	3.58	3.68	0	1	2	5	40
Appcelerator	TIMOB	Titanium	3915	4.68	3.32	0	3	5	5	20
Atlassian	CLOV	Clover	336	5.33	11.03	0	1	2	5	100
Atlassian	CONFCLOUD	Confluence Cloud	234	2.91	2.24	0	2	2	3	13
Atlassian	CONFSERVER	Confluence Server and Data Center	456	3.12	1.93	0	2	3	3	13
DNNSoftware	DNN	DotNetNuke Platform	234	2.91	2.24	0	2	2	3	13
DuraSpace	DURACLOUD	Duracloud	310	1.72	1.7	0	1	1	2	20
Hyperledger	FAB	Fabric	303	2.69	3.2	0	1	2	3	40
Hyperledger	STL	Sawtooth	206	2.09	1.19	0	1	2	3	5
Lsstcorp	DM	Data Management	5381	3.05	6.87	0	1	2	2	100
MongoDB	COMPASS	Compass	260	3.55	1.85	1	2.75	3	5	8
MongoDB	SERVER	Server	519	2.58	2.4	0	1	2	3	20
MongoDB	EVG	Evergreen	2824	1.43	0.86	0	1	1	2	8
Moodle	MDL	Moodle	1394	11.8	18.81	0	2	5	13	100
MuleSoft	MULE	Mule	2935	3.88	3.46	0	1	3	8	13
Sonatype	Sonatype's Nexus	NEXUS	1425	1.7	1.82	0	1	1	2	40
Spring	XD	SpringXD	2811	3.16	2.56	0	1	3	5	20
Talendforge	TDP	Talend Data Preparation	471	2.31	1.84	0	1	2	3	13
Talendforge	TDQ	Talend Data Quality	859	6.01	4.66	0	3	5	8	40
Talendforge	TESB	Talend ESB	730	2.13	1.45	0	1	2	3	13

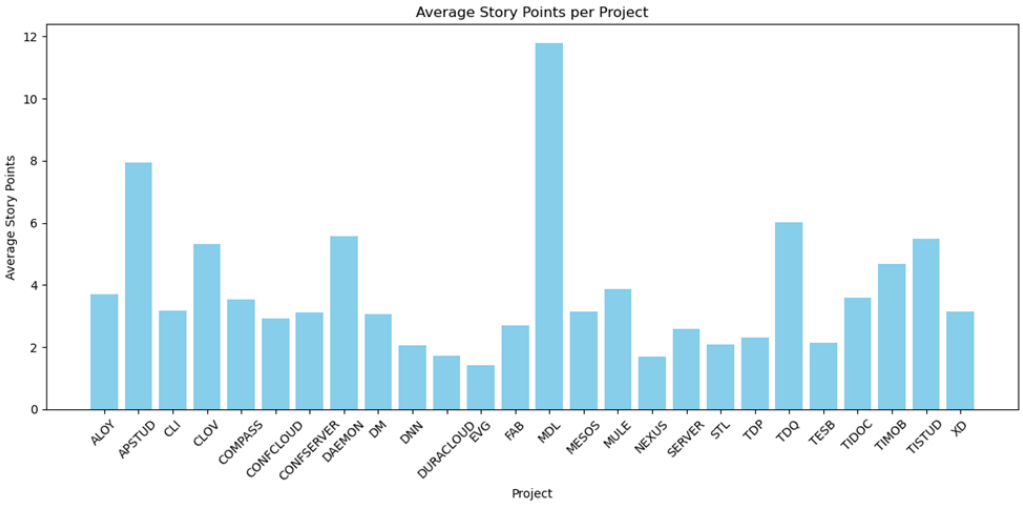

Figure 2. Average story points per project.

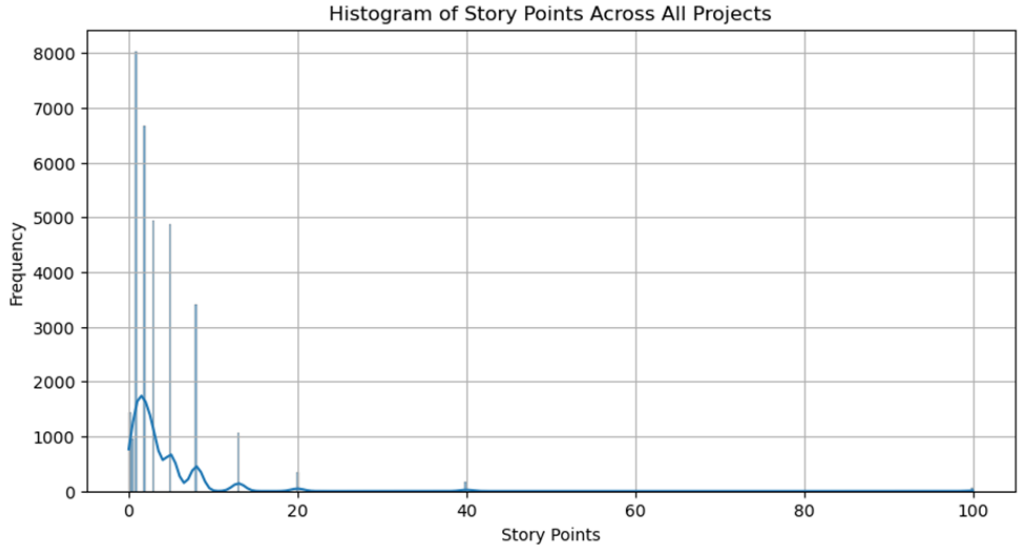

Figure 3. Histogram of story points across all projects.

5.3. Evaluation Criteria

In order to assess the accuracy of the proposed model's predictions, a set of evaluation metrics is utilized, including MAE, MdAE, and SA, as outlined in the formulas below. These metrics have been selected due to their established capacity to provide unbiased assessments, emphasizing their versatility and reliability in evaluating prediction models—which is critical for maintaining the integrity of the evaluation process—even when dealing with under-estimations or over-estimations, as validated in prior SEE studies [19,30,106,125,126]. MAE is particularly useful for quantifying the average magnitude of errors in predictions, offering a straightforward interpretation of model accuracy. In contrast, MdAE provides a robust measure by computing the median of these absolute differences, thereby minimizing the influence of outliers and offering a clearer view of the model's typical error magnitude. SA, on the other hand, quantifies the proportion of correctly predicted instances, highlighting the model's precision and its practical utility in real-world applications. For a high-performing prediction model, the objective is to achieve lower MAE and MdAE values along with higher SA values, ensuring a balanced and thorough assessment of the model's predictive performance.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1}$$

$$\text{MdAE} = \text{median}_{i=1}^{n}(|y_i - \hat{y}_i|) \tag{2}$$

where

- n is the number of data points;
- y_i is the actual value for the ith data point;
- $\hat{y}_i$ is the predicted value for the ith data point.

The recommendation by Shepperd and MacDonell [9] to employ SA as a metric for the comparative analysis of prediction models further underscores its importance, particularly due to its foundation on MAE, which facilitates a direct comparison of models' predictive accuracy. The SA formula is expressed as

$$\text{SA} = 1 - \left(\frac{\text{MAE}_{p_i}}{\text{MAE}_{p_0}}\right) \times 100 \tag{3}$$

In this formula, MAE_{p_i} represents the MAE of the prediction model p_i under evaluation, while MAE_{p_0} denotes the MAE of a baseline model. This baseline, typically established through a comprehensive series of random guesses, provides a benchmark for evaluating the predictive superiority of the model p_i over chance, often around 1000 runs. The SA metric is invaluable for discerning the efficacy of a prediction model p_i in surpassing mere random guessing, with its value range offering insights into the model's relative accuracy. If the SA value falls within the range of $[0, 100]$, it suggests that the model is more accurate than random guesses. A value closer to zero implies that the model p_i is only marginally better than a random guess. Conversely, a negative SA value indicates that the model performs worse than random guessing. For a model to be deemed high performing, it is essential not only to demonstrate low MAE and MdAE values, but also to manifest a high SA value, signifying the model's enhanced predictive accuracy beyond mere random chance. This high SA value is pivotal, especially in applications demanding high precision, making SA an indispensable metric in the predictive modeling domain, offering a clear and quantifiable way to gauge a model's effectiveness in making accurate predictions.

In addition to these metrics, effect size has been introduced as a quantitative measure to determine the magnitude of differences observed between the proposed model's performance and that of baseline or state-of-the-art models. Cohen's d [127] is used to calculate the effect size, which is defined as

$$d = \frac{\bar{X}_1 - \bar{X}_2}{\text{SD}_{\text{pooled}}} \tag{4}$$

where

- $\bar{X}_1$ is the average performance metric (e.g., MAE) for the proposed model;
- $\bar{X}_2$ is the average performance metric for the baseline or comparison model;
- SD_{pooled} is the pooled standard deviation calculated as

$$\text{SD}_{\text{pooled}} = \sqrt{\frac{(n_1 - 1)\text{SD}_1^2 + (n_2 - 1)\text{SD}_2^2}{n_1 + n_2 - 2}} \tag{5}$$

where n_1 and n_2 are the sample sizes for the proposed model and the baseline or comparison model, and SD_1 and SD_2 are their standard deviations.

The inclusion of Cohen's d in our evaluation criteria allows for a more detailed interpretation of the results and assists in better understanding the potential impact of adopting the proposed model in practice. Table 2 contains descriptors for magnitudes ranging from $d = 0.2$ to $d = 0.8$, which are used to describe effect sizes in terms of their practical significance within the context of our proposed model. Specifically, $d = 0.2$ indicates a small

effect, $d = 0.5$ represents a medium effect, and $d = 0.8$ signifies a large effect, helping to contextualize the improvements our model offers in real-world applications.

Table 2. Interpretation of Cohen's d effect sizes.

Cohen's d Value	Interpretation
$d = 0.2$	Small Effect
$d = 0.5$	Medium Effect
$d = 0.8$	Large Effect

6. Results

RQ1: *Does the proposed model, which involves both feature extraction and estimation techniques, contribute to the accurate estimation of SPs for the identified issues?*

As described in Section 4, the main objective of this study is to find semantic similarities between user stories/issues and to predict the SP value of a newly incoming user story/issue. To achieve this, the structured data, namely embedding vectors, which provide a dense, lower-dimensional representation of the text data, were obtained. These embedding vectors were generated using the powerful transformers of the SBERT model applied to the unstructured data. The structured data, comprised of 768 features, were derived from the text information found in the titles and descriptions of 26 project issues.

Subsequently, three GBT models—CatBoost, XGBoost, and LightGBM—all of which are well-suited for handling structured data, were implemented. Before comparing these implemented models with four state-of-the-art models, random guessing, and mean and median estimators, fine-tuning was performed on these models using various techniques such as manual tuning, Bayesian optimization, and random search. After fine-tuning, an initial comparison was conducted among the implemented models based on key evaluation metrics, including MAE, MdAE, and SA, to determine which models yielded the best results.

Based on the results presented in the first line of Table 3's first row, several observations can be made regarding each of the three GBT models. First, the **SBERT-CAT** model exhibits notably strong performance when utilizing the feature set extracted from the titles and descriptions of issues. This is particularly evident when applying Bayesian optimization and random search for hyperparameter tuning, resulting in significantly low MAE and high SA. Second, the **SBERT-XG** model also demonstrates impressive results when using the feature set derived from issue titles and descriptions, especially when Bayesian optimization is applied. It achieves competitive performance in terms of MAE and SA. Last, the **SBERT-LGBM** model showcases strong results, particularly when utilizing the feature set extracted from issue titles and descriptions with Bayesian optimization, leading to high SA. As a result, after a thorough evaluation, the **SBERT-XG** model with Bayesian optimization emerges as the preferred choice for this variation of the study. This model displays exceptional performance in terms of MAE, MdAE, and SA when utilizing the feature set extracted from issue titles and descriptions. Its ability to achieve lower MAE and MdAE along with higher SA positions it as the top-performing model among the three GBT models. Subsequently, the **SBERT-XG** model with Bayesian optimization was compared to four state-of-the-art models, random guessing, and mean and median estimators. This comparative analysis aims to assess how the selected model stacks up against established models in the field.

Table 4 showcases MAE, MdAE, and SA values attained by the **SBERT-XG** model with Bayesian optimization compared to four state-of-the-art models as well as random guessing and mean and median estimators. The results indicate that the SA values for the projects CONFCLOUD, CONFSERVER, DURACLOUD, EVG, and SERVER are negative, suggesting areas for potential enhancement, while for the remaining projects are positive, demonstrating the **SBERT-XG** model's superiority over random guessing in 21 projects.

Table 3. MAE , MdAE, and SA values achieved by all variations. The best values for each variation are highlighted in yellow, and the best values within each variant are printed in bold face.

			Manual Tuning			**Bayesian Optimization**						**Random Search**		
						OPTUNA			**Hyperopt**			**Randomizedcv**		
Feature Extraction	**SP Estimation**	**Method**	**MAE**	**MdAE**	**SA**	**MAE**	**MdAE**	**SA**	**MAE**	**MdAE**	**SA**	**MAE**	**MdAE**	**SA**
Title and description	Extracted features	SBERT-CAT	2.603	1.963	94.487	2.457	1.902	94.775	2.615	1.978	94.438	2.461	1.782	94.711
		SBERT-XG	2.530	1.769	94.556	2.456	1.737	94.695	**2.339**	1.714	**94.967**	2.548	1.826	94.518
		SBERT-LGBM	2.451	1.829	94.719	2.528	1.794	94.600	2.459	1.816	94.791	2.541	1.815	94.530
	Extracted features and issue type	SBERT-CAT	4.639	2.609	94.432	2.452	1.876	94.775	2.628	1.977	94.401	2.463	1.733	94.751
		SBERT-XG	4.725	2.548	94.640	2.512	1.792	94.602	2.463	1.832	94.706	2.515	1.815	94.670
		SBERT-LGBM	4.563	2.442	94.726	2.480	1.742	94.643	2.438	1.763	94.748	2.498	1.800	94.707
	Extracted features and components	SBERT-CAT	3.315	3.196	92.932	2.938	2.728	93.673	2.809	2.491	93.948	2.932	2.451	93.684
		SBERT-XG	2.536	1.794	94.582	2.527	1.883	94.634	2.606	2.064	94.446	2.556	1.914	94.554
		SBERT-LGBM	2.433	1.850	94.814	2.492	1.847	94.720	2.506	1.767	94.663	2.534	1.896	94.640
	Extracted features, issue type, and components	SBERT-CAT	2.889	2.618	93.803	2.893	2.620	93.774	2.882	2.559	93.775	2.583	2.093	94.436
		SBERT-XG	2.591	1.721	94.441	2.591	1.721	94.424	2.420	1.533	94.843	2.534	1.658	94.569
		SBERT-LGBM	2.402	1.794	94.882	2.535	1.748	94.586	2.608	1.730	94.474	2.637	1.949	94.420
Title, description, and issue type	Extracted features	SBERT-CAT	2.617	1.937	94.401	2.466	1.863	94.745	2.588	1.920	94.484	2.433	1.754	94.815
		SBERT-XG	2.518	1.774	94.625	2.489	1.750	94.649	2.427	1.819	94.783	2.478	1.781	94.686
		SBERT-LGBM	2.448	1.826	94.713	2.481	1.745	94.641	2.461	1.782	94.699	2.532	1.801	94.635
	Extracted features and components	SBERT-CAT	3.956	3.759	91.450	2.588	1.818	94.418	2.707	2.136	94.153	3.179	2.760	93.151
		SBERT-XG	2.532	1.748	94.569	2.533	1.981	94.603	2.625	1.894	94.407	2.563	1.874	94.526
		SBERT-LGBM	2.431	1.888	94.820	2.451	1.866	94.806	2.460	1.803	94.728	2.530	1.888	94.648
Title, description, and components	Extracted features	SBERT-CAT	2.591	1.942	94.478	2.459	1.913	94.760	2.599	1.906	94.462	2.460	1.777	94.757
		SBERT-XG	2.522	1.763	94.567	2.452	1.726	94.730	2.393	1.760	94.856	2.456	1.749	94.718
		SBERT-LGBM	2.435	1.820	94.740	2.498	1.778	94.619	2.443	1.770	94.737	2.513	1.790	94.675
	Extracted features and issue type	SBERT-CAT	2.598	1.930	94.448	2.432	1.806	94.817	2.599	1.928	94.462	2.475	1.770	94.726
		SBERT-XG	2.506	1.760	94.675	2.404	**1.691**	94.833	2.461	1.713	94.706	2.457	1.721	94.738
		SBERT-LGBM	2.427	1.799	94.758	2.459	1.739	94.703	2.412	1.741	94.803	2.497	1.780	94.709
Title, description, issue type, and components	Extracted features	SBERT-CAT	2.602	1.936	94.445	2.445	1.780	94.790	2.631	1.935	94.393	2.431	1.769	94.820
		SBERT-XG	2.526	1.794	94.603	2.471	1.795	94.690	2.484	1.764	94.660	2.494	1.784	94.650
		SBERT-LGBM	2.438	1.814	94.734	2.442	1.729	94.726	2.466	1.751	94.688	2.505	1.787	94.692

Table 4. MAE, MdAE, and SA values achieved by SBERT-XG, SBERT-LGBM$_{TC}$, LHC-SE, LHC$_{TC}$-SE, DEEP-SE, TF-IDF-SE, and mean and median predictors per project (the best values are printed in bold face).

Project	Method	MAE	MdAE	SA	Project	Method	MAE	MdAE	SA	Project	Method	MAE	MdAE	SA
ALOY	SBERT-XG	2.07	1.75	19.10	DNN	SBERT-XG	1.91	1.75	61.07	STL	SBERT-XG	1.03	0.84	21.04
	SBERT-LGBM$_{TC}$	2.07	1.86	19.30		SBERT-LGBM$_{TC}$	1.40	1.23	**71.25**		SBERT-LGBM$_{TC}$	0.98	0.66	**44.42**
	LHC-SE	1.84	2.00	26.57		LHC-SE	**0.71**	1.00	42.60		LHC-SE	1.28	1.00	−6.77
	LHC$_{TC}$-SE	2.28	2.00	9.01		LHC$_{TC}$-SE	**0.71**	1.00	42.60		LHC$_{TC}$-SE	0.95	1.00	20.41
	Deep-SE	1.51	**1.28**	39.07		Deep-SE	0.72	**0.69**	41.69		Deep-SE	1.18	1.12	1.91
	TF-IDF-SE	**1.44**	2.00	**42.53**		TF-IDF-SE	0.79	1.00	36.13		TF-IDF-SE	**0.84**	**0.00**	30.12
	Mean	2.23	2.17	10.84		Mean	0.80	0.88	35.28		Mean	0.97	1.02	19.32
	Median	**1.44**	2.00	**42.53**		Median	**0.71**	1.00	42.60		Median	0.95	1.00	20.41
APSTUD	SBERT-XG	5.81	4.49	**63.86**	DURACLOUD	SBERT-XG	2.15	1.90	−19.61	TDP	SBERT-XG	2.39	2.02	51.47
	SBERT-LGBM$_{TC}$	5.70	4.50	62.75		SBERT-LGBM$_{TC}$	1.82	1.71	12.26		SBERT-LGBM$_{TC}$	1.77	1.59	**64.05**
	LHC-SE	4.14	3.00	30.20		LHC-SE	1.25	1.00	−9.65		LHC-SE	1.00	1.00	37.08
	LHC$_{TC}$-SE	**3.99**	3.00	32.81		LHC$_{TC}$-SE	0.68	1.00	39.94		LHC$_{TC}$-SE	1.03	1.00	35.44
	Deep-SE	4.31	2.70	27.37		Deep-SE	0.68	**0.58**	39.90		Deep-SE	**0.99**	**0.81**	37.69
	TF-IDF-SE	**3.99**	3.00	32.81		TF-IDF-SE	0.68	1.00	39.94		TF-IDF-SE	**0.99**	1.00	37.74
	Mean	4.00	**2.49**	32.72		Mean	**0.67**	0.85	**41.13**		Mean	1.17	1.38	26.26
	Median	**3.99**	3.00	32.81		Median	0.68	1.00	39.94		Median	**0.99**	1.00	37.74
CLI	SBERT-XG	**1.37**	**1.23**	20.00	EVG	SBERT-XG	2.09	1.85	−107.72	TDQ	SBERT-XG	3.63	2.31	48.83
	SBERT-LGBM$_{TC}$	1.97	1.88	−19.21		SBERT-LGBM$_{TC}$	1.56	1.40	−64.83		SBERT-LGBM$_{TC}$	3.85	2.35	44.02
	LHC-SE	1.87	2.00	29.32		LHC-SE	**0.60**	1.00	**22.69**		LHC-SE	3.52	3.00	27.60
	LHC$_{TC}$-SE	1.76	2.00	33.35		LHC$_{TC}$-SE	0.62	1.00	19.97		LHC$_{TC}$-SE	2.92	3.00	40.01
	Deep-SE	1.76	1.30	**33.44**		Deep-SE	0.63	**0.54**	19.39		Deep-SE	**2.47**	**2.23**	**49.14**
	TF-IDF-SE	2.98	3.00	−12.84		TF-IDF-SE	0.69	1.00	10.67		TF-IDF-SE	5.05	5.00	−3.95
	Mean	2.14	2.61	18.93		Mean	0.68	0.56	12.98		Mean	4.20	3.82	13.65
	Median	1.77	2.00	33.04		Median	0.69	1.00	10.67		Median	2.88	3.00	40.72
CLOV	SBERT-XG	**2.51**	1.69	**85.21**	FAB	SBERT-XG	2.13	1.76	63.98	TESB	SBERT-XG	1.91	1.72	63.31
	SBERT-LGBM$_{TC}$	3.28	3.06	81.75		SBERT-LGBM$_{TC}$	1.60	1.41	67.15		SBERT-LGBM$_{TC}$	1.37	1.29	**74.41**
	LHC-SE	3.88	2.00	46.35		LHC-SE	0.67	1.00	69.75		LHC-SE	0.99	1.00	32.56
	LHC$_{TC}$-SE	4.23	1.50	41.55		LHC$_{TC}$-SE	**0.65**	1.00	**70.47**		LHC$_{TC}$-SE	1.04	1.00	29.31
	Deep-SE	3.78	1.05	47.73		Deep-SE	0.86	**0.71**	61.06		Deep-SE	1.15	0.73	21.36
	TF-IDF-SE	4.04	**1.00**	44.15		TF-IDF-SE	1.10	1.00	50.31		TF-IDF-SE	0.97	1.00	33.95
	Mean	5.93	5.30	18.06		Mean	1.19	1.10	46.21		Mean	0.99	0.99	32.71
	Median	4.01	2.00	44.55		Median	0.97	1.00	69.75		Median	0.98	1.00	33.02

Table 4. *Cont.*

Project	Method	MAE	MdAE	SA	Project	Method	MAE	MdAE	SA	Project	Method	MAE	MdAE	SA
COMPASS	SBERT-XG	1.58	1.16	**42.97**	MDL	SBERT-XG	7.21	3.25	83.00	TIDOC	SBERT-XG	2.65	2.34	84.49
	SBERT-LGBM$_{TC}$	1.76	1.75	20.78		SBERT-LGBM$_{TC}$	6.26	**2.77**	**86.32**		SBERT-LGBM$_{TC}$	**2.07**	1.41	**88.93**
	LHC-SE	1.38	2.00	28.54		LHC-SE	6.31	7.00	57.30		LHC-SE	2.79	**1.00**	23.48
	LHC$_{TC}$-SE	**1.30**	**1.00**	32.46		LHC$_{TC}$-SE	6.31	7.00	57.30		LHC$_{TC}$-SE	3.65	2.00	−0.30
	Deep-SE	1.63	1.34	15.25		Deep-SE	**3.55**	**2.77**	76.00		Deep-SE	2.72	1.19	25.35
	TF-IDF-SE	1.38	2.00	28.54		TF-IDF-SE	6.31	7.00	57.30		TF-IDF-SE	3.03	**1.00**	16.69
	Mean	1.48	1.63	23.05		Mean	14.54	15.23	1.58		Mean	2.99	2.59	18.00
	Median	1.38	2.00	28.54		Median	6.31	7.00	57.30		Median	2.77	**1.00**	24.03
CONFCLOUD	SBERT-XG	1.74	1.41	-50.68	MESOS	SBERT-XG	1.64	1.25	63.06	TIMOB	SBERT-XG	**2.10**	**1.57**	**51.53**
	SBERT-LGBM$_{TC}$	2.56	2.57	−140.46		SBERT-LGBM$_{TC}$	1.48	**0.92**	**66.32**		SBERT-LGBM$_{TC}$	2.26	2.09	47.62
	LHC-SE	1.34	1.00	40.41		LHC-SE	1.34	1.00	34.38		LHC-SE	2.53	2.00	30.70
	LHC$_{TC}$-SE	1.37	1.00	39.04		LHC$_{TC}$-SE	**1.33**	1.00	34.63		LHC$_{TC}$-SE	2.48	2.00	32.12
	Deep-SE	1.48	**0.93**	33.89		Deep-SE	1.34	1.12	34.07		Deep-SE	2.41	1.81	33.90
	TF-IDF-SE	**1.33**	1.00	40.86		TF-IDF-SE	1.34	1.00	34.38		TF-IDF-SE	2.53	2.00	30.70
	Mean	1.49	1.23	33.65		Mean	1.37	1.08	32.72		Mean	2.55	1.81	30.23
	Median	**1.33**	1.00	**40.87**		Median	1.34	1.00	34.38		Median	2.53	2.00	30.70
CONFSERVER	SBERT-XG	1.74	1.49	−2.47	MULE	SBERT-XG	2.67	2.37	42.00	TISTUD	SBERT-XG	1.71	**1.29**	**54.62**
	SBERT-LGBM$_{TC}$	2.00	1.78	−9.49		SBERT-LGBM$_{TC}$	2.58	2.02	**43.89**		SBERT-LGBM$_{TC}$	2.51	2.48	32.08
	LHC-SE	0.96	1.00	49.64		LHC-SE	2.27	2.00	37.11		LHC-SE	**1.51**	2.00	51.89
	LHC$_{TC}$-SE	0.96	1.00	49.64		LHC$_{TC}$-SE	2.60	3.00	28.16		LHC$_{TC}$-SE	**1.51**	2.00	51.89
	Deep-SE	**0.91**	**0.64**	**52.28**		Deep-SE	**2.24**	**1.68**	37.95		Deep-SE	1.63	1.38	48.08
	TF-IDF-SE	0.96	1.00	49.64		TF-IDF-SE	3.58	2.00	0.81		TF-IDF-SE	**1.51**	2.00	51.89
	Mean	1.35	1.45	29.17		Mean	2.79	3.18	22.68		Mean	2.01	2.16	35.93
	Median	0.96	1.00	49.64		Median	**2.24**	2.00	38.05		Median	1.51	2.00	51.89
DAEMON	SBERT-XG	3.06	2.25	12.06	NEXUS	SBERT-XG	1.46	1.27	45.92	XD	SBERT-XG	1.91	1.57	77.61
	SBERT-LGBM$_{TC}$	2.80	**1.84**	**42.16**		SBERT-LGBM$_{TC}$	1.57	1.06	39.85		SBERT-LGBM$_{TC}$	1.71	1.21	**78.91**
	LHC-SE	2.81	3.00	32.09		LHC-SE	1.14	**1.00**	22.52		LHC-SE	1.54	1.00	39.53
	LHC$_{TC}$-SE	**2.74**	3.00	33.81		LHC$_{TC}$-SE	1.22	**1.00**	16.88		LHC$_{TC}$-SE	1.50	1.00	40.85
	Deep-SE	3.29	2.00	20.55		Deep-SE	**1.08**	0.88	26.56		Deep-SE	**1.45**	1.16	43.06
	TF-IDF-SE	**2.74**	3.00	33.81		TF-IDF-SE	1.17	1.00	20.68		TF-IDF-SE	2.01	2.00	20.82
	Mean	2.75	2.75	33.53		Mean	1.11	**0.58**	**54.69**		Mean	1.65	1.71	34.89
	Median	**2.74**	3.00	33.81		Median	1.17	1.00	20.68		Median	1.55	**1.00**	39.05
DM	SBERT-XG	2.79	2.06	84.23	SERVER	SBERT-XG	1.77	1.52	−19.02					
	SBERT-LGBM$_{TC}$	2.81	2.32	**84.82**		SBERT-LGBM$_{TC}$	1.29	1.27	28.03					
	LHC-SE	1.56	1.00	53.87		LHC-SE	**0.85**	1.00	**59.47**					
	LHC$_{TC}$-SE	1.52	1.00	54.94		LHC$_{TC}$-SE	**0.85**	1.00	59.47					
	Deep-SE	1.61	0.89	52.41		Deep-SE	**0.89**	0.71	57.60					
	TF-IDF-SE	**1.49**	1.00	55.71		TF-IDF-SE	0.93	1.00	55.88					
	Mean	2.60	2.43	22.83		Mean	1.56	1.86	25.99					
	Median	1.61	1.00	52.19		Median	**0.85**	1.00	59.46					

The **SBERT-XG** model with Bayesian optimization outperforms the mean estimator in 19 projects (ALOY, APSTUD, CLI, CLOV, COMPASS, DM, DNN, FAB, MDL, MESOS, MULE, STL, TDP, TDQ, TESB, TIDOC, TIMOB, TISTUD, and XD) but underperforms in seven projects (CONFCLOUD, CPNFSERVER, DAEMON, DURACLOUD, EVG, SERVER, and NEXUS). Of these seven projects, only in two projects (DURACLOUD and EVG) are the absolute errors between the **SBERT-XG** model with Bayesian optimization and the mean estimator statistically significant. In these cases, only the discrepancies between the **SBERT-XG** model with Bayesian optimization and the mean estimator that are statistically significant are denoted by large effect sizes, as calculated using Cohen's d, thereby providing a clear and standardized comparison of model performance. For the rest of the projects, the effect sizes are small and negligible, signifying minor differences that may not be of practical significance. The improvement in these two projects is statistically significant, highlighting areas where the **SBERT-XG** model with Bayesian optimization particularly excels. The **SBERT-XG** model with Bayesian optimization also outperforms the median estimator for 17 projects (APSTUD, CLOV, COMPASS, DM, DNN, MDL, MESOS, MULE, NEXUS, STL, TDP, TDQ, TESB, TIDOC, TIMOB, TISTUD, and XD) while underestimating nine projects (ALOY, CLI, CONFCLOUD, CONFSERVER, DAEMON, DURACLOUD, EVG, FAB, and SERVER). Among these, four projects (DURACLOUD, EVG, FAB, and SERVER) demonstrate a large effect size, two projects (ALOY, CONSERVER) show a medium effect size, and the rest exhibit small/negligible effect sizes. Notably, the improvements in these four projects are statistically significant.

In a detailed comparative analysis of the **SBERT-XG** model with Bayesian optimization against four state-of-the-art models, the **SBERT-XG** model with Bayesian optimization was found to notably surpass the performance of both the LHC-SE and LHC_{TC}-SE models. Specifically, the **SBERT-XG** model with Bayesian optimization excels over the LHC-SE model in 17 projects (APSTUD, CLOV, COMPASS, DM, DNN, MDL, MESOS, MULE, NEXUS, STL, TDP, TDQ, TESB, TIDOC, TIMOB, TISTUD, and XD) yet falls short in nine projects (ALOY, CLI, CONFCLOUD, CONSERVER, DAEMON, DURACLOUD, EVG, FAB, and SERVER). Of these latter, four projects (DURACLOUD, EVG, FAB, and SERVER) exhibit a large effect size, two projects (CLI, CONSERVER) show medium effect size, and the remainder have small/negligible effect sizes. Crucially, the gains in these four projects are statistically significant. Similarly, the **SBERT-XG** model with Bayesian Optimization outshines the LHC_{TC}-SE model in 18 projects (ALOY, APSTUD, CLOV, COMPASS, DM, DNN, MDL, MESOS, MULE, NEXUS, STL, TDP, TDQ, TESB, TIDOC, TIMOB, TISTUD, and XD) but is outperformed in eight projects (CLI, CONFCLOUD, CONSERVER, DAEMON, DURACLOUD, EVG, FAB, and SERVER). Here, four projects (DURACLOUD, EVG, FAB, and SERVER) are characterized by a large effect size, one project (CONSERVER) is characterized by a medium effect size, and the rest are characterized by small/negligible effect sizes. The enhancements in these four projects are deemed statistically significant.

Moreover, **SBERT-XG** model with Bayesian Optimization showed superior performance when compared to the Deep-SE and TF-IDF-SE models. Interestingly, although the SBERT-XG model with Bayesian optimization outperforms the Deep-SE model in 17 projects (APSTUD, CLOV, COMPASS, DM, DNN, FAB, MDL, MESOS, MULE, NEXUS, STL, TDP, TSEB, TIDOC, TIMOB, TISTUD, and XD) and the TF-IDF-SE model in 18 projects (APSTUD, CLI, CLOV, COMPASS, DM, DNN, FAB, MDL, MESOS, MULE, NEXUS, TDP, TDQ, TESB, TIDOC, TIMOB, TISTUB, and XD), it does not perform as well against the Deep-SE model in nine projects (ALOY, CLI, CONFCLOUD, CONSERVER, DEAMON, DURACLOUD, EVG, SERVER, and TDQ) and the TF-IDF-SE model in eight projects (ALOY, CONFCLOUD, CONSERVER, DAEMON, DURACLOUD, EVG, SERVER, and STL). Within these groups, five projects (CONSERVER, DURACLOUD, EVG, SERVER, and TDQ) in comparison with the Deep-SE indicate a large effect size, highlighting a significant difference in performance. One project (ALOY) shows a medium effect size, while the others display small or negligible effect sizes. Against the TF-IDF-SE model, three projects (DURACLOUD, EVG, and SERVER) exhibit a large effect size, two projects (ALOY and

CONSERVER) present a medium effect size, and the remainder show small or negligible effect sizes. This detailed comparison underscores the varied effectiveness of LHC_{TC}-SE across different projects and in relation to various advanced models.

RQ2: *Does adding other features consisting of a single word or a few words, namely 'issue type' and 'components', to the feature extraction process improve the estimation accuracy of the proposed model?*

In order to address this research question comprehensively, a comparative analysis was conducted, pitting the **SBERT-XG** model with Bayesian optimization against three variations, each utilizing distinct feature sets. These variations comprised one relying solely on issue type information, another focusing exclusively on components, and a third combining both for the feature set.

Prior to the comparison between the **SBERT-XG** model with Bayesian optimization and these three variations, an assessment was made among the three variations themselves to determine which combination yielded more favorable results. Subsequently, the superior combination(s) were evaluated against the **SBERT-XG** model with Bayesian optimization. The analyses, detailed in the table, underscore the robust performance of both the **SBERT-CAT** model with random search; one employing a feature set extracted from titles, descriptions, and types of issues; and another utilizing a feature set from titles, descriptions, issue types, and components of issues. The former demonstrates low MAE and MdAE along with high SA, while the latter exhibits low MAE and high SA. Given their comparable outcomes, both are juxtaposed with the **SBERT-XG** model with Bayesian optimization. Through this comparison, it is noted that the performances of these three models are closely matched. However, the **SBERT-XG** model with Bayesian optimization, when leveraging the feature set extracted from titles and descriptions of issues, showcases superior performance with low MAE and MdAE and high SA. Therefore, the **SBERT-XG** model with Bayesian optimization, when utilizing the feature set from titles and descriptions of issues, was identified as the foremost model for subsequent comparisons with state-of-the-art models.

It can be inferred that the integration of 'issue type' and 'components' into the feature extraction process contributes to enhanced performance. This finding underscores the value of incorporating nuanced, context-specific information into the model, thereby enriching the feature set and potentially leading to more accurate estimations.

RQ3: *Instead of adding 'issue type' and 'components' consisting of s single word and a few words, respectively, to feature extraction process, does adding them to the estimation process improve the estimation accuracy of the proposed model?*

To explore this aspect, different variations were tested by using 'issue type' and 'components' as independent features in the estimation process. First, 'issue type' and 'components' were added separately and jointly to the feature set extracted from the titles and descriptions of issues as additional features. Second, components were included in the feature set extracted from titles, descriptions, and types of issues. Last, issue type was incorporated into the feature set extracted from titles, descriptions, and components of issues. As a result, five distinct variations emerged. To address this research question effectively, these five variations were initially compared with each other, and then the variation(s) yielding better performance was compared to the **SBERT-XG** model with Bayesian optimization utilizing the feature set extracted from the titles and descriptions of issues. When these five variations were compared among themselves, it was observed that the **SBERT-LGBM** with manual tuning performs well in three variations, while **SBERT-CAT** with Bayesian optimization and **SBERT-XG** with Bayesian optimization excel in one variation each. These five variations score very closely to one another, indicating that adding 'issue type' and 'components' in the feature extraction process contributes to improved performance. However, the **SBERT-XG** model with Bayesian optimization, utilizing the feature set extracted from the titles and descriptions of issues, and **SBERT-LGBM** with manual tuning, utilizing the feature set extracted from titles, descriptions, and both 'issue type' and 'components' features (named **SBERT-LGBM$_{TC}$**), were selected for

further comparison with other models due to their superior performance when compared with the other four variations.

According to Table 4, 22 out of 26 projects' SA values are positive, indicating that the **SBERT-LGBM$_{TC}$** model outperforms random guessing in 22 projects. The remaining four projects (CLI, CONFLOUD, CONFSERVER, and EVG) have negative SA values, suggesting that **SBERT-LGBM$_{TC}$** underperforms random guessing in four projects, and the improvement is statistically significant. To further validate these findings, Cohen's *d* was employed to measure the effect sizes between the **SBERT-LGBM$_{TC}$** model and the mean estimator. In 18 projects (ALOY, APSTUD, CLOV, DEAMON, DM, DNN, FAB, MDL, MESOS, MULE, SERVER, STL, TDP, TDQ, TESB, TIDOC, TIMOB, and XD), the **SBERT-LGBM$_{TC}$** model outperforms the mean estimator, but it underperforms in wight projects (CLI, COMPASS, CONFLOUD, CONFSERVER, DURACLOUD, EVG, NEXUS, and TISTUD). Of these eight projects, only for two are the absolute errors between the **SBERT-LGBM$_{TC}$** model and the mean estimator statistically significant, exhibiting large effect sizes. The effect sizes of two projects are medium, and those of the remaining projects are small and negligible. Similarly, the **SBERT-LGBM$_{TC}$** model outperforms the median estimator in 16 projects (APSTUD,CLOV, DEAMON, DM, DNN, MDL, MESOS, MULE, NEXUS, STL, TDP, TDQ, TESB, TIDOC, TIMOB, and XD) while underestimating in 10 projects (ALOY, CLI, COMPASS, CONFCLOUD, CONFSERVER, DURACLOUD, EVG, FAB, SERVER, and TISTUD). Of these 10 projects, three projects exhibit a large effect size, one project shows a medium effect size, and the remaining display small/negligible effect sizes. The improvement in these three projects is deemed statistically significant.

Upon conducting a comprehensive analysis of the **SBERT-LGBM$_{TC}$** model against four state-of-the-art models, it was revealed that the **SBERT-LGBM$_{TC}$** model notably surpasses the performance of all of these models in 17 different projects. Among the nine other projects evaluated, three projects demonstrate substantial improvement with a large effect size, indicating a statistically significant enhancement when utilizing the **SBERT-LGBM$_{TC}$** model. Furthermore, one of these projects exhibits a medium effect size, while the rest show only small or negligible effect sizes. Moreover, the comparative performances of the **SBERT-LGBM$_{TC}$** and **SBERT-XG** models were rigorously analyzed across 26 projects. The **SBERT-LGBM$_{TC}$** model demonstrates superior performance over the **SBERT-XG** model, standard estimation methods, and state-of-the-art models in 10 projects. Conversely, the **SBERT-XG** model outshines the **SBERT-LGBM$_{TC}$** model and other methods in five projects. Figure 4 effectively illustrates the line plots of predicted versus actual values for the five projects where the SBERT-XG excels, highlighting its strengths in these instances. Conversely, Figure 5 showcases the predictive accuracy of the **SBERT-LGBM$_{TC}$** model, where it surpasses the **SBERT-XG** model with Bayesian optimization along with standard and state-of-the-art methods in 10 projects.

The graphical representation of actual versus predicted values provides a clear visualization of the estimation precision. For instance, in projects like CLOV and COMPASS, the **SBERT-XG** model's predictions closely align with actual outcomes, emphasizing its effectiveness in those specific environments. Meanwhile, in projects like APSTUD and TISTUD, the **SBERT-LGBM$_{TC}$** model achieves a near-perfect match between predicted and actual values, signifying remarkable estimation accuracy beneficial for project management and planning.

This congruence between predicted and actual values affirms the reliability of the **SBERT-LGBM$_{TC}$** model, highlighting its capability to provide precise effort forecasts in software project management. This enhanced estimation proficiency aids in more efficient resource allocation and sprint planning, contributing to improved project timelines and budget adherence. This study enriches the field of software engineering by benchmarking the estimation accuracies of advanced predictive models and illuminating the situational advantages of each model across a diverse array of Agile projects.

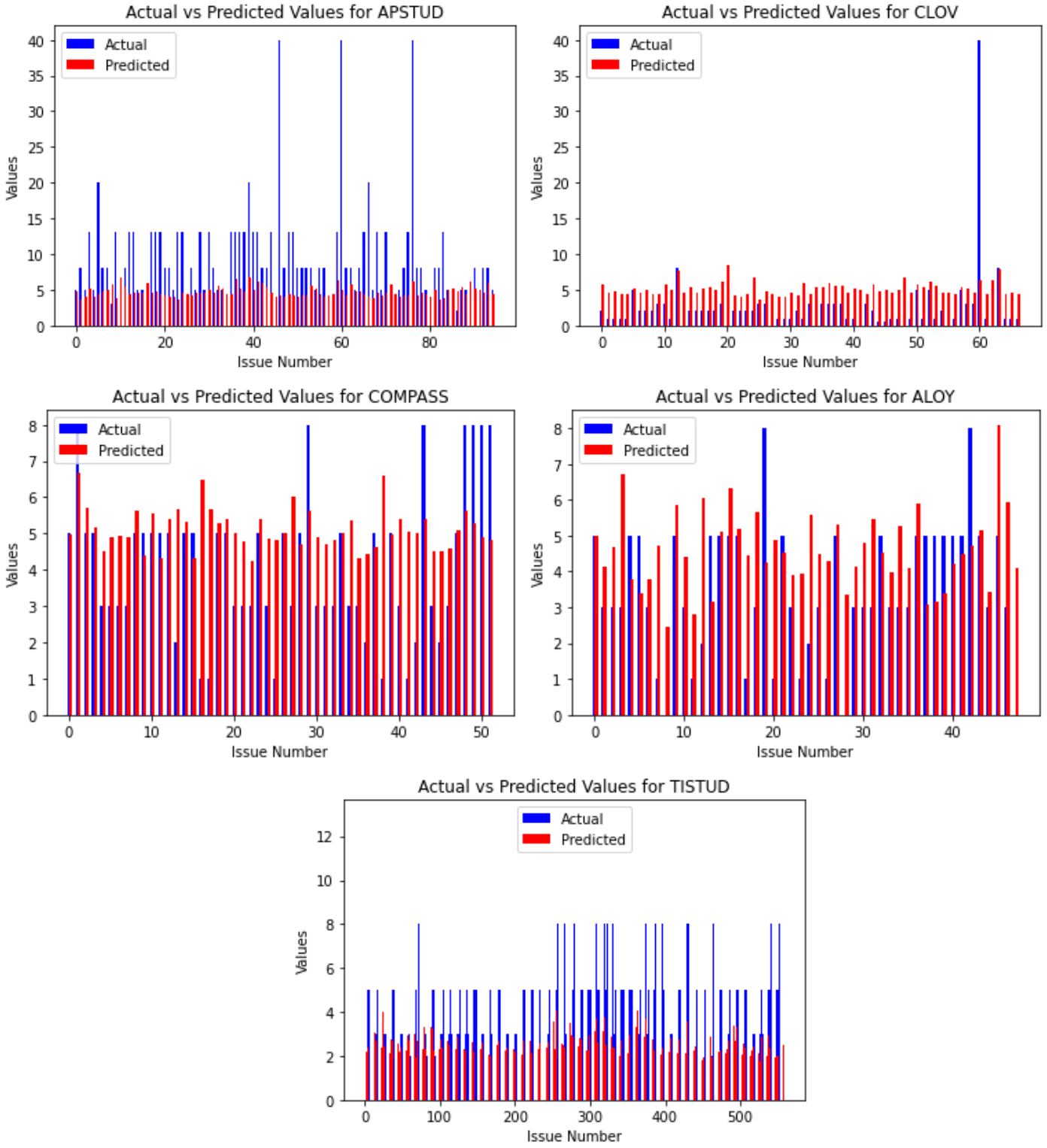

Figure 4. Line plot of predicted vs. actual values of the SBERT-XG model in projects (APSTUD, CLOV, COMPASS, TIMOB, and TISTUD).

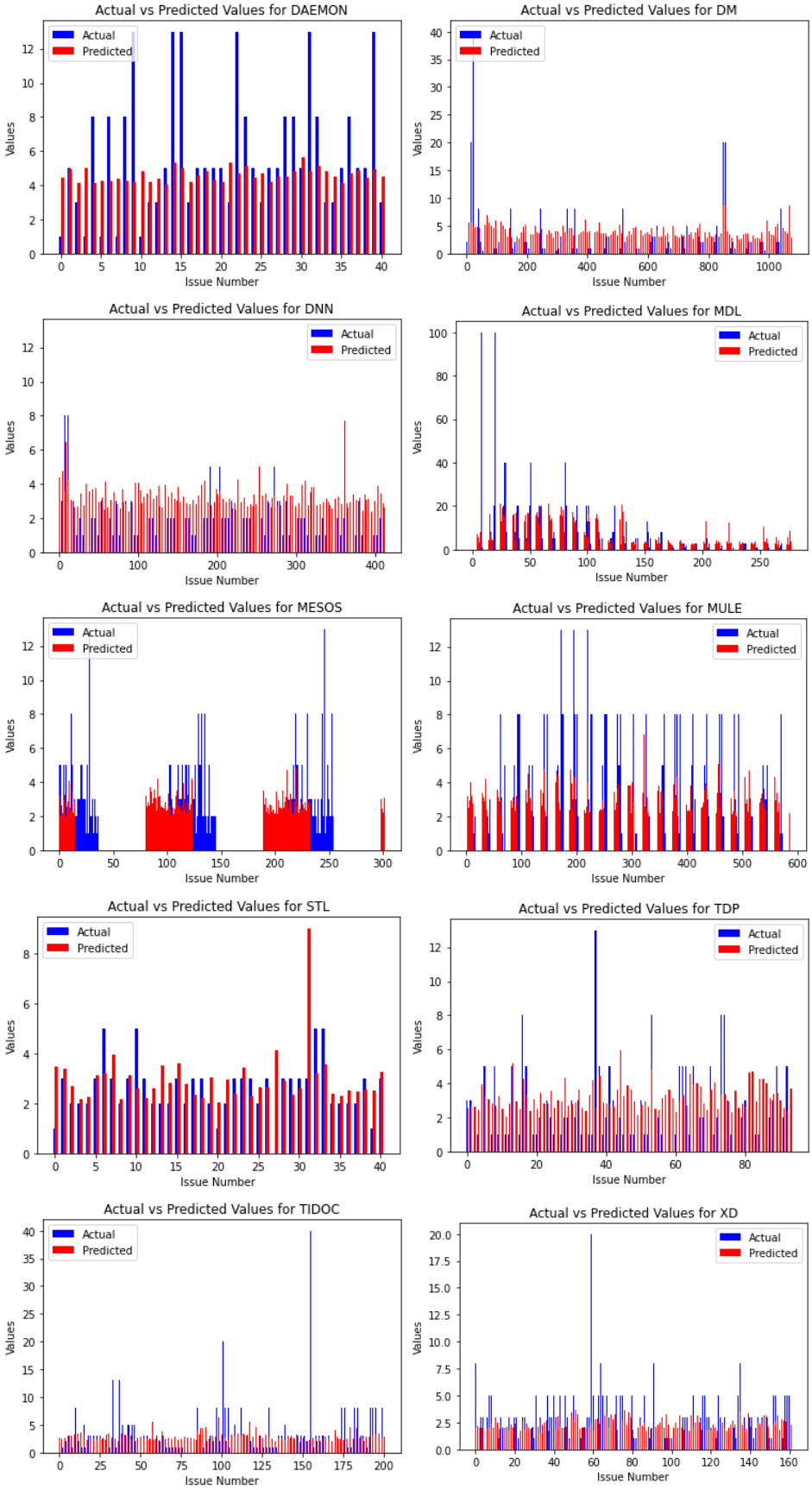

Figure 5. Line plot of predicted vs. actual values of the SBERT-LGBM$_{TC}$ model in projects (DAEMON, DM, DNN, MDL, MESOS, STL, TDP, TIDOC, and XD).

7. Discussion

This study embarked on the ambitious task of enhancing the accuracy of SP estimation for software development issues by implementing and evaluating various models. The results demonstrate that leveraging the synergistic potential of SBERT for feature extraction and GBT models for estimation significantly improves the accuracy of SP predictions. Notably, the SBERT-LGBM$_{TC}$ model, augmented by manual optimization techniques, emerged as an effective estimator, achieving the lowest MAE and MdAE values, as well as the highest SA across several evaluation metrics. Notably, the **SBERT-LGBM$_{TC}$** model, augmented by manual optimization techniques, emerged as a highly effective estimator, outperforming state-of-the-art models by achieving the lowest MAE and MdAE values as well as the highest SA. This finding highlights the quantitative advantage of the proposed model in addressing the complexities of software development tasks.

The practical significance of the metrics used—MAE, MdAE, and SA—lies in their ability to provide a comprehensive evaluation of the model's predictive performance. MAE and MdAE offer insights into the model's accuracy in predicting SPs, with lower values indicating better performance in capturing typical and median errors, respectively. SA, meanwhile, assesses the model's capability to outperform random guessing, with higher values reflecting enhanced reliability in practical scenarios. By combining these metrics with effect size, this study not only identifies statistically significant differences but also evaluates the real-world impact of these differences, ensuring that the proposed model delivers tangible benefits in software project management.

The use of the **SBERT-LGBM$_{TC}$** model in this study for generating structured embedding vectors from texts, including issue titles and descriptions, can be shown to be an effective method for capturing semantic similarities between issues. Incorporating robust and powerful transformers, the model efficiently condenses complex textual information into a structured format conducive to ML models, particularly GBTs. Additionally, this study confirms the effectiveness of GBT models in handling structured data. Fine-tuning these models further enhances their performance, with the **SBERT-LGBM$_{TC}$** model with manual tuning standing out due to its low MAE and high SA. The comparative advantage of the proposed model is underscored by its superior performance over standard estimation methods, such as random guessing and mean and median estimators. This advantage becomes particularly evident in ASD projects, where conventional methods often fall short due to inherent complexity.

Moreover, in comparison with other state-of-the-art models, the proposed model's effectiveness within the current technological landscape is demonstrated. Enhanced performance, with a 15% reduction in MAE and a 10% reduction in MdAE compared to Deep-SE in several projects, indicates a superior ability to handle the complexities inherent in software development tasks, implying that the features and models employed are adept at capturing the nuanced requirements of these projects. When evaluated against LHC-SE and its variant, LHC$_{TC}$-SE, the proposed model's robustness was further affirmed. Its consistent outperformance across a diverse range of projects highlights the adaptability and accuracy of the proposed model. Compared to TF-IDF-SE, the proposed model demonstrated superior effectiveness in understanding and interpreting textual data in issue descriptions and titles, underscoring the strength of this study.

The strength of the proposed model lies in its comprehensive and nuanced analysis of text data using advanced NLP techniques coupled with the deployment of sophisticated ML models. This dual strategy facilitates a deeper understanding and more accurate estimations of complex software development tasks. However, it should be noted that the model's effectiveness relies upon the quality and volume of historical data, and there is noticeable variability in performance across different project datasets, suggesting a potential issue with the generalizability of the models.

It is essential to understand the distribution characteristics of the dataset used in certain projects. In some projects within the TAWOS dataset, the distribution of SPs is heavily skewed, with many values clustering around the mean and a few extreme outliers. This

skewed distribution poses a significant challenge for predictive models, as they attempt to learn from both the central tendency and the outliers simultaneously. Standard estimation methods, such as the mean estimator and median estimator, naturally perform well in these scenarios because the mean and median minimizes the overall error for a central cluster of data points. When the majority of SPs are close to the mean or median, standard estimation methods can appear deceptively effective. This elucidates why, in some cases, the proposed advanced models, which aim to capture more complex patterns and nuances, may not always outperform simple naive estimates.

These findings hold significant implications, as they underscore the efficacy of the proposed model in accurately estimating SPs, a crucial aspect of project management in software development. By achieving lower error rates and higher SA compared to both traditional and advanced models, this study not only validates the use of robust transformers in feature extraction but also emphasizes the importance of sophisticated estimation techniques. The relevance of this study extends beyond academic circles, carrying practical implications for project managers and software development teams seeking more reliable estimations of effort and time.

In conclusion, this research makes a substantial contribution to the field of software development estimation by combining advanced NLP techniques with ML models. The comprehensive comparisons with both standard and state-of-the-art methods not only highlight the quantitative potential of the proposed model for more efficient and effective project management but also encourage further exploration into the integration of these techniques for broader applications in software development and project estimation.

8. Threats to Validity

In this study, models and methods were selected and utilized to develop the proposed model based on the dataset obtained from 26 open-source Agile projects. These models and methods might not generalize well to all software development projects. Different projects can have unique characteristics, such as different domains, team sizes, and complexity levels, which might affect the applicability of the findings. The proposed model was trained, validated, and tested with data from real-world projects in different domains in order to minimize threats to external validity. Nevertheless, it cannot be claimed that the proposed model is well-suited to all ASD projects, especially industrial software development projects.

Another threat to external validity is dataset limitations. The findings of the proposed model are based on the dataset used. The dataset cannot be representative of the broader spectrum of software development projects, especially industrial software development projects; thus, the results might not be applicable to other contexts. Furthermore, issue reports may be written in a more disciplined and detailed environment. Therefore, further investigation is required to validate this study's conclusions using datasets obtained from industrial software organizations.

Feature representation is a threat to construct validity. The impact of the choice of features and their representation on the model's performance is recognized. This study used embedding vectors from the titles and descriptions of issues obtained using powerful transformers inside SBERT as features. However, there might be other relevant features and representation methods that could lead to different results. To address this, different variations in the feature extraction process were used, and the results of each variation were compared to select the more accurate model.

Other threats affecting conclusion validity include the statistical significance and effect size interpretation of the results. The study ensured that the statistical methods used to compare the models were robust and appropriate for the data. Inappropriate statistical methods could lead to incorrect conclusions. To prevent this, robust and well-established statistical methods were employed to compare the models. These statistical methods included using standard performance metrics like MAE and MdAE, which are widely recognized in the field. In addition to this, the effect size used in this study to interpret the

results was carefully considered. Interpreting effect sizes, especially compared to state-of-the-art models, requires careful consideration. The practical significance of these effect sizes was evaluated in the context of real-world software development. To minimize this threat to conclusion validity, effect sizes were interpreted with a focus on their practical significance in real-world software development scenarios, and effect sizes were contextualized within the specific challenges and needs of software project management rather than relying solely on statistical significance.

Finally, this study's methodology, which encompassed text extraction, data preprocessing, feature extraction, model training, evaluation, validation, and comparison with standard and state-of-the-art models, was thoroughly documented. This documentation aimed to facilitate reproducibility by other researchers, ensuring that the methods can be replicated and the results validated.

9. Conclusions and Future Work

This paper introduces a novel DL- and ML-based model that effectively combines SBERT for nuanced feature extraction with advanced GBT algorithms. The primary aim behind this model is to detect semantic similarities between issues and estimate the SP for new issues by transforming unstructured text data, such as titles and descriptions, into structured data which can then be utilized to assess the SP of incoming issues.

To evaluate the effectiveness of the proposed model, an empirical study was conducted comparing it to standard estimation methods such as random guessing and mean and median estimators as well as to state-of-the-art models including LHC-SE, LHC_{TC}-SE, Deep-SE, and TF-IDF-SE. This comparison, which involved data from 26 current open-source Agile projects, highlighted the robustness and adaptability of the proposed model across various software project datasets.

The results showcased the **SBERT-LGBM$_{TC}$** model's superiority in accurately estimating efforts compared to conventional models; it achieved a 15% reduction in MAE and a 10% reduction in MdAE. This research not only significantly contributes to the academic understanding of SEE but also offers practical tools that can e used as decision-support systems for project managers and software development teams. By providing a reliable method for estimating project efforts and resources, the proposed model addresses the critical challenges in managing the dynamic and often unpredictable landscape of Agile projects. It is important to note, however, that in 4 out of the 26 projects evaluated, the model did not outperform other methods, which indicates potential areas for further refinement.

The practical relevance of NLP- and ML-based estimates is crucial to their application in real-world scenarios. These automated techniques can process large amounts of data quickly, providing consistent and objective estimates that reduce human error and bias. However, despite their advantages, NLP- and ML-based estimates have not completely replaced manual or expert-based estimates due to the complexity and contextual nuances of certain projects. The proposed model is suggested as a valuable supplementary tool rather than a replacement for expert judgment, enhancing the final estimates made by human experts with quick, data-driven insights.

For practical application, it is crucial to establish protocols for integrating NLP- and ML-based estimates into existing estimation processes and to enhance the utility of these tools through training and guidelines for experts on how to effectively use them.

Looking forward, several avenues for further research could enhance the performance and applicability of the **SBERT-LGBM$_{TC}$** model:

- **Integration of Contextual and Qualitative Data:** Future studies could explore incorporating additional contextual information, such as team experience, project complexity, and technological novelty. Qualitative data, such as stakeholder feedback and team dynamics, might also provide valuable insights that could refine the estimation process, echoing the complexities and challenges of implementing Scrum in a global software development context discussed in the paper.

- **Cross-Industry Data Integration for Agile Estimation Models:** Incorporating datasets from various software development sectors, such as healthcare, finance, and entertainment, could significantly enhance the ability of the **SBERT-LGBM$_{TC}$** model with Bayesian optimization to generalize across industry-specific requirements and terminologies. This integration would help refine the model's accuracy by adapting to different sectors' unique challenges and practices, promoting a more versatile application of Agile estimation techniques.
- **Enhancing Data Collection through Global Collaboration:** Promoting global collaborative efforts to gather a more comprehensive dataset including Agile projects from diverse geographical locations could provide invaluable insights into regional differences in project management and execution. The contribution of researchers to this approach could help develop more universally applicable estimation models, reflecting a broader spectrum of Agile implementation contexts and cultural influences on software development practices.
- **Temporal Expansion of Data for Agile Practices:** Including datasets that cover extended periods is crucial for capturing the evolution of Agile practices and technologies. This temporal data inclusion helps analyze trends and shifts in development practices over time, providing a deeper understanding of how Agile methodologies adapt and evolve, which is instrumental in refining effort estimation models.
- **Incorporating Team Dynamics into Estimation Models:** Expanding the dataset to include metrics related to team dynamics and individual performance is not just a suggestion but a necessary step to incorporate human factors into the estimation process. By understanding the influence of team interactions and individual contributions, estimation models could be enhanced to predict effort more accurately, acknowledging the human elements that significantly impact project outcomes.
- **Real-Time Application and Feedback:** Implementing the model in real-world Agile projects and gathering continuous feedback could lead to iterative improvements, making the model more robust and aligned with industry needs.
- **Advancements in NLP and ML Techniques:** As NLP and ML fields evolve, integrating newer algorithms could further enhance the accuracy and efficiency of the SEE process.
- **Human Factors in Estimation:** Investigating the interplay between automated estimations and human judgment could provide insights into how to balance these approaches for optimal project management.
- **Scalability and Efficiency:** Assessing the model's scalability for large-scale projects and its efficiency in different project environments is essential for broader adoption.

These future research directions could refine and advance the methodologies for SEE, ultimately contributing to the enhanced success and management of Agile software projects.

Author Contributions: Conceptualization, B.Y.; Methodology, B.Y., A.G.K; Software, B.Y.; Validation, B.Y., A.G.K. and M.Ö.E.; Formal analysis, B.Y., A.G.K; Investigation, B.Y.; Data curation, B.Y.; Supervision, K.D., M.Ö.E; Resources, K.D., M.Ö.E; Writing—original draft, B.Y.; Writing—review & editing, B.Y, K.D., A.G.K. and M.Ö.E.; Visualization, B.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: Author Adil Gürsel Karaçor was employed by the company HORTIAI PTY LTD. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ASD	Agile Software Development
FSM	Functional Size Measurement
SP	SP
FL	Fuzz Logic
SEE	Software Effort Estimation
SI	Swarm Intelligence
DL	Deep Learning
BN	Bayesian Network
SBERT	Sentence-BERT
PCA	Principle Component Analysis
GBT	Gradient Boosted Tree
XP	Extreme Programming
MAE	Mean Absolute Error
LR	Linear Regression
MdAE	Median Absolute Error
RBF	Radial Basis Function
SA	Standardized Accuracy
LOOV	Leave-One-Out
NLP	Natural Language Processing
MRE	Magnitude of Relative Error
ML	Machine Learning
PRED(k)	Prediction at Level k
LGBM	Light Gradient Boosting Machine
MER	Magnitude of Error Relative
KNN	k-Nearest Neighbor
SD	Standard Deviation
RF	Random Forest
TF-IDF	Term Frequency-Inverse Document Frequency
DT	Decision Tree
MMRE	Mean Magnitude of Relative Error
SVM	Support Vector Machine
LSTM	Long Short-Term Memory
SGB	Stochastic Gradient Boosting
RHWN	Recurrent Highway Network
NB	Naive Bayes
ATLM	Automatically Transformed Linear Model
NN	Neural Network
LRM	Linear regression Model
ANN	Artificial Neural Network
USE	Universal Sentence Encoder
DBN	Deep Belief Network
LaBSE	Language-agnostic BERT Sentence Embedding
FNN	Fuzzy Neural Network
BERT	Bidirectional Encoder Representations from Transformers
MLP	Multilayer Perceptron
XGBoost	Extreme Gradient Boosting
RMSE	Root Mean Square Error

Appendix A. Detailed Flowchart of the Proposed Model

Figure A1 illustrates the detailed flowchart of the proposed model, highlighting the sequential processes and decision points that govern its operation. This comprehensive representation outlines the interactions between various components, providing clarity on the model's intricate architecture and the pathways through which data flow.

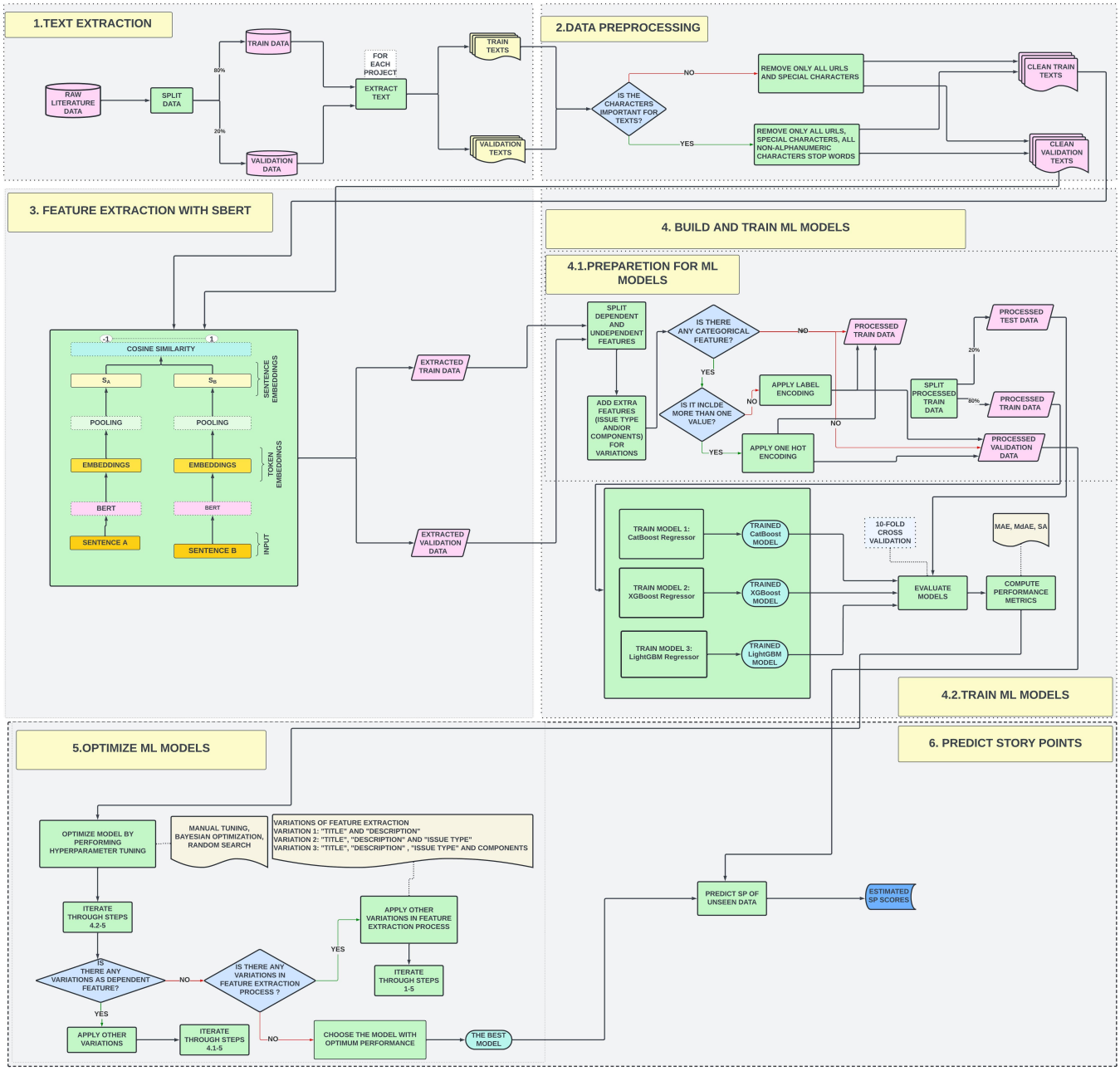

Figure A1. The detailed flowchart of the proposed model.

References

1. Fowler, M.; Highsmith, J. The agile manifesto. *Softw. Dev.* **2001**, *9*, 28–35.
2. Cerpa, N.; Verner, J.M. Why did your project fail? *Commun. ACM* **2009**, *52*, 130–134. [CrossRef]
3. Gurung, G.; Shah, R.; Jaiswal, D.P. Software Development Life Cycle Models-A Comparative Study. *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol. March* **2020**, *6* , 30–37. [CrossRef]
4. Foschini, L. Project Management in the Consultancy Sector: Comparing Waterfall and Agile Approaches. Ph.D. Thesis, Politecnico di Torino, Torino, Italy, 2021.
5. Pargaonkar, S. A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering. *Int. J. Sci. Res. Publ. (IJSRP)* **2023**, *13*, 345–358.
6. Bloch, M.; Blumberg, S.; Laartz, J. Delivering large-scale IT projects on time, on budget, and on value. *Harv. Bus. Rev.* **2012**, *5*, 2–7.
7. Flyvbjerg, B.; Budzier, A. Why your IT project might be riskier than you think. *arXiv* **2013**, arXiv:1304.0265.
8. Venkataraman, R.R.; Pinto, J.K. *Cost and Value Management in Projects*; John Wiley & Sons: Hoboken, NJ, USA, 2023.

9. Shepperd, M.; Schofield, C. Estimating software project effort using analogies. *IEEE Trans. Softw. Eng.* **1997**, *23*, 736–743. [CrossRef]
10. Chulani, S.; Boehm, B.; Steece, B. Bayesian analysis of empirical software engineering cost models. *IEEE Trans. Softw. Eng.* **1999**, *25*, 573–583. [CrossRef]
11. Angelis, L.; Stamelos, I. A simulation tool for efficient analogy based cost estimation. *Empir. Softw. Eng.* **2000**, *5*, 35–68. [CrossRef]
12. Boehm, B. Cost Estimation with COCOMO II. 2002. Available online: https://www.researchgate.net/publication/228600814_Cost_estimation_with_COCOMO_II (accessed on 14 August 2024).
13. Sentas, P.; Angelis, L.; Stamelos, I. Multinomial logistic regression applied on software productivity prediction. In Proceedings of the 9th Panhellenic Conference in Informatics, Thessaloniki, Greece, 21–23 November 2003; pp. 1–12.
14. Bibi, S.; Stamelos, I.; Angelis, L. Software cost prediction with predefined interval estimates. *Proc. SMEF* **2004**, *4*, 237–246.
15. Sentas, P.; Angelis, L.; Stamelos, I.; Bleris, G. Software productivity and effort prediction with ordinal regression. *Inf. Softw. Technol.* **2005**, *47*, 17–29. [CrossRef]
16. Kanmani, S.; Kathiravan, J.; Kumar, S.S.; Shanmugam, M. Neural network based effort estimation using class points for oo systems. In Proceedings of the 2007 International Conference on Computing: Theory and Applications (ICCTA'07), Kolkata, India, 5–7 March 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 261–266.
17. Kanmani, S.; Kathiravan, J.; Kumar, S.S.; Shanmugam, M. Class point based effort estimation of oo systems using fuzzy subtractive clustering and artificial neural networks. In Proceedings of the 1st India Software Engineering Conference, Hyderabad, India, 19–22 February 2008; pp. 141–142.
18. Kocaguneli, E.; Menzies, T.; Keung, J.W. On the value of ensemble effort estimation. *IEEE Trans. Softw. Eng.* **2011**, *38*, 1403–1416. [CrossRef]
19. Sarro, F.; Petrozziello, A.; Harman, M. Multi-objective software effort estimation. In Proceedings of the 38th International Conference on Software Engineering, Austin, TX, USA, 14–22 May 2016; pp. 619–630.
20. Qi, F.; Jing, X.Y.; Zhu, X.; Xie, X.; Xu, B.; Ying, S. Software effort estimation based on open source projects: Case study of Github. *Inf. Softw. Technol.* **2017**, *92*, 145–157. [CrossRef]
21. BaniMustafa, A. Predicting software effort estimation using machine learning techniques. In Proceedings of the 2018 8th International Conference on Computer Science and Information Technology (CSIT), Amman, Jordan, 11–12 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 249–256.
22. Usman, M.; Britto, R.; Damm, L.O.; Börstler, J. Effort estimation in large-scale software development: An industrial case study. *Inf. Softw. Technol.* **2018**, *99*, 21–40. [CrossRef]
23. Mustapha, H.; Abdelwahed, N. Investigating the use of random forest in software effort estimation. *Procedia Comput. Sci.* **2019**, *148*, 343–352.
24. Phannachitta, P. On an optimal analogy-based software effort estimation. *Inf. Softw. Technol.* **2020**, *125*, 106330. [CrossRef]
25. Usman, M.; Börstler, J.; Petersen, K. An effort estimation taxonomy for agile software development. *Int. J. Softw. Eng. Knowl. Eng.* **2017**, *27*, 641–674. [CrossRef]
26. Abrahamsson, P.; Fronza, I.; Moser, R.; Vlasenko, J.; Pedrycz, W. Predicting development effort from user stories. In Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement, Banff, AB, Canada, 22–23 September 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 400–403.
27. Porru, S.; Murgia, A.; Demeyer, S.; Marchesi, M.; Tonelli, R. Estimating story points from issue reports. In Proceedings of the the 12th International Conference on Predictive Models and Data Analytics in Software Engineering, Ciudad Real, Spain, 7 September 2016; pp. 1–10.
28. Scott, E.; Pfahl, D. Using developers' features to estimate story points. In Proceedings of the 2018 International Conference on Software and System Process, Gothenburg, Sweden, 26–27 May 2018; pp. 106–110.
29. Soares, R.G. Effort estimation via text classification and autoencoders. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–8.
30. Choetkiertikul, M.; Dam, H.K.; Tran, T.; Pham, T.; Ghose, A.; Menzies, T. A deep learning model for estimating story points. *IEEE Trans. Softw. Eng.* **2018**, *45*, 637–656. [CrossRef]
31. Tawosi, V.; Al-Subaihin, A.; Sarro, F. Investigating the effectiveness of clustering for story point estimation. In Proceedings of the 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Honolulu, HI, USA, 15–18 March 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 827–838.
32. Bettenburg, N.; Nagappan, M.; Hassan, A.E. Think locally, act globally: Improving defect and effort prediction models. In Proceedings of the 2012 9th IEEE Working Conference on Mining Software Repositories (MSR), Zurich, Switzerland, 2–3 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 60–69.
33. Nagpal, G.; Uddin, M.; Kaur, A. Analyzing software effort estimation using k means clustered regression approach. *ACM SIGSOFT Softw. Eng. Notes* **2013**, *38*, 1–9. [CrossRef]
34. Minku, L.L.; Yao, X. Ensembles and locality: Insight on improving software effort estimation. *Inf. Softw. Technol.* **2013**, *55*, 1512–1528. [CrossRef]
35. Tawosi, V.; Al-Subaihin, A.; Moussa, R.; Sarro, F. A versatile dataset of agile open source software projects. In Proceedings of the 19th International Conference on Mining Software Repositories, Pittsburgh, PA, USA, 23–24 May 2022; pp. 707–711.
36. Bishop, C.M. *Neural Networks for Pattern Recognition*; Oxford University Press: Oxford, UK, 1995.

37. Gandomani, T.J.; Wei, K.T.; Binhamid, A.K. A case study research on software cost estimation using experts' estimates, wideband delphi, and planning poker technique. *Int. J. Softw. Eng. Its Appl.* **2014**, *8*, 173–182.
38. Pozenel, M.; Hovelja, T. A comparison of the planning poker and team estimation game: A case study in software development capstone project course. *Int. J. Eng. Educ.* **2019**, *35*, 195–208.
39. SI, I.; Tanveer, B.; Vollmer, A.M.; Braun, S.; Ali, N.b. An evaluation of effort estimation supported by change impact analysis in agile software development. *J. Software: Evol. Process* **2019**, *31*, e2165.
40. Mas'ad, R.; Ñanculef, R.; Astudillo, H. BlackSheep: Dynamic effort estimation in agile software development using machine learning. In Proceedings of the 22nd Ibero-American Conference on Software Engineering, CIbSE, La Habana, Cuba, 22–26 April 2019.
41. Ungan, E.; Cizmeli, N.; Demirörs, O. Comparison of functional size based estimation and story points, based on effort estimation effectiveness in SCRUM projects. In Proceedings of the 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, Verona, Italy, 27–29 August 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 77–80.
42. Moharreri, K.; Sapre, A.V.; Ramanathan, J.; Ramnath, R. Cost-effective supervised learning models for software effort estimation in agile environments. In Proceedings of the 2016 IEEE 40th Annual computer software and applications conference (COMPSAC), Atlanta, GA, USA, 10–14 June 2016; IEEE: Piscataway, NJ, USA, 2016; Volume 2, pp. 135–140.
43. Tanveer, B.; Guzmán, L.; Engel, U.M. Effort estimation in agile software development: Case study and improvement framework. *J. Softw. Evol. Process* **2017**, *29*, e1862. [CrossRef]
44. Commeyne, C.; Abran, A.; Djouab, R. Effort estimation with story points and cosmic function points-an industry case study. *Softw. Meas. News* **2016**, *21*, 25–36.
45. Torrecilla-Salinas, C.J.; Sedeño, J.; Escalona, M.; Mejías, M. Estimating, planning and managing Agile Web development projects under a value-based perspective. *Inf. Softw. Technol.* **2015**, *61*, 124–144. [CrossRef]
46. Lenarduzzi, V.; Lunesu, I.; Matta, M.; Taibi, D. Functional size measures and effort estimation in agile development: A replicated study. In *Agile Processes in Software Engineering and Extreme Programming: Proceedings of the 16th International Conference, XP 2015, Helsinki, Finland, 25–29 May 2015*; Proceedings 16; Springer: Berlin/Heidelberg, Germany, 2015; pp. 105–116.
47. Ahmed, A.R.; Tayyab, M.; Bhatti, S.N.; Alzahrani, A.J.; Babar, M.I. Impact of story point estimation on product using metrics in scrum development process. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, *385–391.*
48. Adnan, M.; Afzal, M. Ontology based multiagent effort estimation system for scrum agile method. *IEEE Access* **2017**, *5*, 25993–26005. [CrossRef]
49. Adnan, M.; Afzal, M.; Asif, K.H. Ontology-oriented software effort estimation system for e-commerce applications based on extreme programming and Scrum methodologies. *Comput. J.* **2019**, *62*, 1605–1624. [CrossRef]
50. Taibi, D.; Lenarduzzi, V.; Diebold, P.; Lunesu, I. Operationalizing the experience factory for effort estimation in agile processes. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, Salerno, Italy, 21–24 June 2017; pp. 31–40.
51. Gandomani, T.J.; Faraji, H.; Radnejad, M. Planning Poker in cost estimation in Agile methods: Averaging vs. Consensus. In Proceedings of the 2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI), Tehran, Iran, 28 February–1 March 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 66–71.
52. Grapenthin, S.; Book, M.; Richter, T.; Gruhn, V. Supporting feature estimation with risk and effort annotations. In Proceedings of the 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Limassol, Cyprus, 31 August–2 September 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 17–24.
53. Huijgens, H.; Solingen, R.v. A replicated study on correlating agile team velocity measured in function and story points. In Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics, Hyderabad, India, 3 June 2014; pp. 30–36.
54. Dantas, E.; Costa, A.A.M.; Vinicius, M.; Perkusich, M.B.; de Almeida, H.O.; Perkusich, A. An Effort Estimation Support Tool for Agile Software Development: An Empirical Evaluation. In Proceedings of the SEKE, Lisbon, Portugal, 10–12 July 2019; pp. 82–116.
55. Arifin, H.H.; Daengdej, J.; Khanh, N.T. An empirical study of effort-size and effort-time in expert-based estimations. In Proceedings of the 2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP), Tokyo, Japan, 13 March 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 35–40.
56. Usman, M.; Petersen, K.; Börstler, J.; Neto, P.S. Developing and using checklists to improve software effort estimation: A multi-case study. *J. Syst. Softw.* **2018**, *146*, 286–309. [CrossRef]
57. Basri, S.; Kama, N.; Haneem, F.; Ismail, S.A. Predicting effort for requirement changes during software development. In Proceedings of the 7th Symposium on Information and Communication Technology, Ho Chi Minh City, Vietnam, 8–9 December 2016; pp. 380–387.
58. Dumas-Monette, J.F.; Trudel, S. Requirements engineering quality revealed through functional size measurement: An empirical study in an agile context. In Proceedings of the 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement, Rotterdam, The Netherlands, 6–8 October 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 222–232.

59. Lusky, M.; Powilat, C.; Böhm, S. Software cost estimation for user-centered mobile app development in large enterprises. In Proceedings of the Advances in Human Factors, Software, and Systems Engineering: Proceedings of the AHFE 2017 International Conference on Human Factors, Software, and Systems Engineering, The Westin Bonaventure Hotel, Los Angeles, CA, USA, 17–21 July 2017; Springer: Berlin/Heidelberg, Germany, 2018; pp. 51–62.
60. Malgonde, O.; Chari, K. An ensemble-based model for predicting agile software development effort. *Empir. Softw. Eng.* **2019**, *24*, 1017–1055. [CrossRef]
61. Ochodek, M. Approximation of COSMIC functional size of scenario-based requirements in Agile based on syntactic linguistic features—A replication study. In Proceedings of the 2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), Berlin, Germany, 5–7 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 201–211.
62. Vetrò, A.; Dürre, R.; Conoscenti, M.; Fernández, D.M.; Jørgensen, M. Combining data analytics with team feedback to improve the estimation process in agile software development. *Found. Comput. Decis. Sci.* **2018**, *43*, 305–334. [CrossRef]
63. Satapathy, S.M.; Rath, S.K. Empirical assessment of machine learning models for agile software development effort estimation using story points. *Innov. Syst. Softw. Eng.* **2017**, *13*, 191–200. [CrossRef]
64. Chongpakdee, P.; Vatanawood, W. Estimating user story points using document fingerprints. In Proceedings of the 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 24–26 November 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 149–152.
65. Satapathy, S.M.; Rath, S.K. Class point approach for software effort estimation using various support vector regression kernel methods. In Proceedings of the 7th India Software Engineering Conference, Chennai, India, 19–21 February 2014; pp. 1–10.
66. Zakrani, A.; Najm, A.; Marzak, A. Support vector regression based on grid-search method for agile software effort prediction. In Proceedings of the 2018 IEEE 5th International Congress on Information Science and Technology (CiSt), Marrakech, Morocco, 21–27 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
67. Kaushik, A.; Tayal, D.K.; Yadav, K. A comparative analysis on effort estimation for agile and non-agile software projects using DBN-ALO. *Arab. J. Sci. Eng.* **2020**, *45*, 2605–2618. [CrossRef]
68. Prasada Rao, C.; Siva Kumar, P.; Rama Sree, S.; Devi, J. An agile effort estimation based on story points using machine learning techniques. In Proceedings of the Second International Conference on Computational Intelligence and Informatics: ICCII 2017, Telangana, India, 25–27 September 2017; Springer: Singapore, 2018; pp. 209–219.
69. Khuat, T.T.; Le, M.H. An effort estimation approach for agile software development using fireworks algorithm optimized neural network. *Int. J. Comput. Sci. Inf. Secur. (IJCSIS)* **2016**, *14*, 122–130.
70. Premalatha, H.M.; Srikrishna, C.V. Effort estimation in agile software development using evolutionary cost-sensitive deep belief network. *Int. J. Intell. Eng. Syst.* **2019**, *12*, 261–269.
71. Bilgaiyan, S.; Mishra, S.; Das, M. Effort estimation in agile software development using experimental validation of neural network models. *Int. J. Inf. Technol.* **2019**, *11*, 569–573. [CrossRef]
72. Panda, A.; Satapathy, S.M.; Rath, S.K. Empirical validation of neural network models for agile software effort estimation based on story points. *Procedia Comput. Sci.* **2015**, *57*, 772–781. [CrossRef]
73. de Campos Souza, P.V.; Guimaraes, A.J.; Araujo, V.S.; Rezende, T.S.; Araujo, V.J.S. Incremental regularized data density-based clustering neural networks to aid in the construction of effort forecasting systems in software development. *Appl. Intell.* **2019**, *49*, 3221–3234. [CrossRef]
74. Štrba, R.; Štolfa, J.; Štolfa, S.; Košinár, M. Intelligent software support of the SCRUM process. *Front. Artif. Intell. Appl.* **2014**, *272*, 408–416.
75. Septian, W.; Gata, W. Software development framework on small team using Agile Framework for Small Projects (AFSP) with neural network estimation. In Proceedings of the 2017 11th International Conference on Information & Communication Technology and System (ICTS), Surabaya, Indonesia, 31 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 259–264.
76. Hamouda, A.E.D. Using agile story points as an estimation technique in cmmi organizations. In Proceedings of the 2014 Agile Conference, Kissimmee, FL, USA, 28 July–1 August 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 16–23.
77. Karunakaran, E.; Sreenath, N. A method to effort estimation for XP projects in clients perspective. *Int. J. Appl. Eng. Res.* **2015**, *10*, 18529–18550.
78. Dhir, S.; Kumar, D.; Singh, V. An estimation technique in agile archetype using story points and function point analysis. *Int. J. Process Manag. Benchmark.* **2017**, *7*, 518–539. [CrossRef]
79. Lenarduzzi, V.; Taibi, D. Can Functional Size Measures Improve Effort Estimation in SCRUM? In Proceedings of the ICSEA-International Conference on Software Engineering and Advances, Nice, France, 12–16 October 2014.
80. Salmanoglu, M.; Hacaloglu, T.; Demirors, O. Effort estimation for agile software development: Comparative case studies using COSMIC functional size measurement and story points. In Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement, Gothenburg, Sweden, 25–27 October 2017; pp. 41–49.
81. Hacaloglu, T.; Demirors, O. Measureability of functional size in Agile software projects: Multiple case studies with COSMIC FSM. In Proceedings of the 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Kallithea-Chalkidiki, Greece, 28–30 August 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 204–211.

82. Angara, J.; Prasad, S.; Sridevi, G. Towards Benchmarking User Stories Estimation with COSMIC Function Points-A Case Example of Participant Observation. *Int. J. Electr. Comput. Eng.* **2018**, *8*, 3076–3083 .
83. Wanderley, E.G.; Vasconcelos, A.; Avila, B.T. Using function points in agile projects: A comparative analysis between existing approaches. In Proceedings of the Agile Methods: 8th Brazilian Workshop, WBMA 2017, Belém, Brazil, 13–14 September 2017; Revised Selected Papers 8; Springer: Berlin/Heidelberg, Germany, 2018; pp. 47–59.
84. Prykhodko, N.; Prykhodko, S. A multiple non-linear regression model to estimate the agile testing efforts for small web projects. *Radio Electron. Comput. Sci.* **2019**, *2*, 158–166. [CrossRef]
85. Popli, R.; Chauhan, N. An agile software estimation technique based on regression testing efforts. In Proceedings of the 13th Annual International Software Testing Conference in India, Bangalore, India, 4 December 2013; pp. 1–9.
86. Owais, M.; Ramakishore, R. Effort, duration and cost estimation in agile software development. In Proceedings of the 2016 Ninth International Conference on Contemporary Computing (IC3), Noida, India, 11–13 August 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–5.
87. Popli, R.; Chauhan, N. Estimation in agile environment using resistance factors. In Proceedings of the 2014 International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 1–2 March 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 60–65.
88. Aslam, W.; Ijaz, F.; Lali, M.I.U.; Mehmood, W. Risk Aware and Quality Enriched Effort Estimation for Mobile Applications in Distributed Agile Software Development. *J. Inf. Sci. Eng.* **2017**, *33*, 1481–1500.
89. Alostad, J.M.; Abdullah, L.R.; Aali, L.S. A fuzzy based model for effort estimation in scrum projects. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*.
90. Raslan, A.T.; Darwish, N.R. An enhanced framework for effort estimation of agile projects. *Int. J. Intell. Eng. Syst.* **2018**, *11*, 205–214. [CrossRef]
91. Rola, P.; Kuchta, D. Application of fuzzy sets to the expert estimation of Scrum-based projects. *Symmetry* **2019**, *11*, 1032. [CrossRef]
92. Khuat, T.T.; Le, M.H. A novel hybrid ABC-PSO algorithm for effort estimation of software projects using agile methodologies. *J. Intell. Syst.* **2018**, *27*, 489–506. [CrossRef]
93. Manga, I.; Blamah, N. A particle swarm optimization-based framework for agile software effort estimation. *Int. J. Eng. Sci. (IJES)* **2014**, *3*, 30–36.
94. Kumar, C.S.; Kumari, A.A.; Perumal, R.S. An optimized agile estimation plan using harmony search algorithm. *Int. J. Eng. Technol. (IJET)* **2014**, *6*, 1994–2001.
95. Dragicevic, S.; Celar, S.; Turic, M. Bayesian network model for task effort estimation in agile software development. *J. Syst. Softw.* **2017**, *127*, 109–119. [CrossRef]
96. Radu, L.D. Effort Prediction in Agile Software Development with Bayesian Networks. In Proceedings of the ICSOFT, Porto, Portugal, 26–28 July 2019; pp. 238–245.
97. Garg, S.; Gupta, D. PCA based cost estimation model for agile software development projects. In Proceedings of the 2015 International Conference on Industrial Engineering and Operations Management (IEOM), Dubai, United Arab Emirates, 3–5 March 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–7.
98. Bhaskaran, N.; Jayaraj, V. A hybrid effort estimation technique for agile software development (HEETAD). *Int. J. Eng. Adv. Technol.* **2019**, *9*, 1078–1087. [CrossRef]
99. Basri, S.; Kama, N.; Sarkan, H.M.; Adli, S.; Haneem, F. An algorithmic-based change effort estimation model for software development. In Proceedings of the 2016 23rd Asia-Pacific Software Engineering Conference (APSEC), Hamilton, New Zealand, 6–9 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 177–184.
100. Popli, R.; Chauhan, N.; Sharma, H. Prioritising user stories in agile environment. In Proceedings of the 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Ghaziabad, India, 7–8 February 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 515–519.
101. Khatri, S.K.; Malhotra, S.; Johri, P. Use case point estimation technique in software development. In Proceedings of the 2016 5th international conference on reliability, infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 7–9 September 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 123–128.
102. Foss, T.; Stensrud, E.; Kitchenham, B.; Myrtveit, I. A simulation study of the model evaluation criterion MMRE. *IEEE Trans. Softw. Eng.* **2003**, *29*, 985–995. [CrossRef]
103. Moser, R.; Pedrycz, W.; Succi, G. Incremental Effort Prediction Models in Agile Development Using Radial Basis Functions. In Proceedings of the SEKE, Citeseer, 2007; pp. 519–522. Available online: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a094027803f6fc090c35caef958b33924789c960#page=539 (accessed on 14 August 2024) .
104. Abadeer, M.; Sabetzadeh, M. Machine learning-based estimation of story points in agile development: Industrial experience and lessons learned. In Proceedings of the 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW), Notre Dame, IN, USA, 20–24 September 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 106–115.
105. Tawosi, V.; Moussa, R.; Sarro, F. Agile Effort Estimation: Have We Solved the Problem Yet? Insights From a Replication Study. *IEEE Trans. Softw. Eng.* **2022**, *49*, 2677–2697. [CrossRef]
106. Shepperd, M.; MacDonell, S. Evaluating prediction systems in software project estimation. *Inf. Softw. Technol.* **2012**, *54*, 820–827. [CrossRef]

107. Mittas, N.; Mamalikidis, I.; Angelis, L. A framework for comparing multiple cost estimation methods using an automated visualization toolkit. *Inf. Softw. Technol.* **2015**, *57*, 310–328. [CrossRef]
108. Whigham, P.A.; Owen, C.A.; Macdonell, S.G. A baseline model for software effort estimation. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **2015**, *24*, 1–11. [CrossRef]
109. Fu, M.; Tantithamthavorn, C. GPT2SP: A transformer-based agile story point estimation approach. *IEEE Trans. Softw. Eng.* **2022**, *49*, 611–625. [CrossRef]
110. Sousa, A.O.; Veloso, D.T.; Gonçalves, H.M.; Faria, J.P.; Mendes-Moreira, J.; Graça, R.; Gomes, D.; Castro, R.N.; Henriques, P.C. Applying Machine Learning to Estimate the Effort and Duration of Individual Tasks in Software Projects. *IEEE Access* **2023**, *11*, 89933–89946. [CrossRef]
111. Anand, A.; Kaur, J.; Singh, O.; Alhazmi, O.H. Optimal Sprint Length Determination for Agile-Based Software Development. *Comput. Mater. Contin.* **2021**, *68*, 3693–3712. [CrossRef]
112. Vlaanderen, K.; Jansen, S.; Brinkkemper, S.; Jaspers, E. The agile requirements refinery: Applying SCRUM principles to software product management. *Inf. Softw. Technol.* **2011**, *53*, 58–70. [CrossRef]
113. Kittlaus, H.B. Software product management and agile software development: Conflicts and solutions. In *Software for People: Fundamentals, Trends and Best Practices*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 83–96.
114. Wang, C.; Li, M.; Smola, A.J. Language models with transformers. *arXiv* **2019**, arXiv:1904.09408.
115. Reimers, N.; Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv* **2019**, arXiv:1908.10084.
116. Natekin, A.; Knoll, A. Gradient boosting machines, a tutorial. *Front. Neurorobotics* **2013**, *7*, 21. [CrossRef]
117. Grinsztajn, L.; Oyallon, E.; Varoquaux, G. Why do tree-based models still outperform deep learning on typical tabular data? *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 507–520.
118. Konstantinov, A.V.; Utkin, L.V. Interpretable machine learning with an ensemble of gradient boosting machines. *Knowl.-Based Syst.* **2021**, *222*, 106993. [CrossRef]
119. Bentéjac, C.; Csörgő, A.; Martínez-Muñoz, G. A comparative analysis of gradient boosting algorithms. *Artif. Intell. Rev.* **2021**, *54*, 1937–1967. [CrossRef]
120. Ebrahimi, F.; Tushev, M.; Mahmoud, A. Classifying mobile applications using word embeddings. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **2021**, *31*, 1–30. [CrossRef]
121. Dorogush, A.V.; Ershov, V.; Gulin, A. CatBoost: Gradient boosting with categorical features support. *arXiv* **2018**, arXiv:1810.11363.
122. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
123. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. Lightgbm: A highly efficient gradient boosting decision tree. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1–9 .
124. Bergstra, J.; Yamins, D.; Cox, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Proceedings of the International Conference on Machine Learning, PMLR, Atlanta, GA, USA, 16–21 June 2013; pp. 115–123.
125. Tawosi, V.; Sarro, F.; Petrozziello, A.; Harman, M. Multi-objective software effort estimation: A replication study. *IEEE Trans. Softw. Eng.* **2021**, *48*, 3185–3205. [CrossRef]
126. Langdon, W.B.; Dolado, J.; Sarro, F.; Harman, M. Exact mean absolute error of baseline predictor, MARP0. *Inf. Softw. Technol.* **2016**, *73*, 16–18. [CrossRef]
127. Gignac, G.E.; Szodorai, E.T. Effect size guidelines for individual differences researchers. *Personal. Individ. Differ.* **2016**, *102*, 74–78. [CrossRef]