Synthetic Open-source Agile Software Estimation Performance

Nevena Rankovic [1], Dragica Rankovic [2] and Mirjana Ivanovic [3]

1Tilburg University, School of Humanities and Digital Sciences, Department of Cognitive Science and Artificial Intelligence, Warandelaan 2, 5037 AB Tilburg, The Netherlands
2University of Business Academy in Novi Sad, Department of mathematics, informatics and statistics, Dusana Popovica 21, 18 000 Nis, Serbia
3University of Novi Sad, Faculty of Sciences, Department of mathematics and informatics, Trg Dositeja Obradovica 4, 21 000, Novi Sad, Serbia

Abstract

In this paper, we investigate whether Software Development Effort Estimations (SDEEs) predictions can be improved using commonly used machine learning algorithms such as Linear Regression, Decision Tree Regression, Random Forest Regression, XGBoost Regression, CatBoost Regression, and LightGBM Regression. To prevent the data leakage and enhance the TAWOS agile open-source software project dataset using Tabular Variational Autoencoder (TVAE) and Truncation Normal Data distribution we also apply additional scaling. Hyperparameter optimization with Optuna was conducted on 21 model-data combinations based on 5-fold cross-validated adjusted R^2, mean squared prediction error (MSPE), and Pearson's correlation coefficient. The Random Forest Regressor trained on TVAE-augmented data achieved the best results, with an adjusted R^2 of 0.59, a Pearson's correlation of 0.81, and an MSPE of 140011, indicating strong predictive accuracy. The CatBoost Regressor on regular data ranked second, with an adjusted R^2 of 0.39, a Pearson's correlation of 0.74, and an MSPE of 200011. The Decision Tree Regressor, despite a high training correlation, performed the worst, with an adjusted R^2 of 0.35, a Pearson's correlation of 0.76, and an MSPE of 234500, indicating weaker performance. Ultimately, we aimed to reduce the gap between expected and actual software development efforts, thereby minimizing associated risks. The results of this study can significantly enhance software development project planning and management.

Keywords

software estimation, regression models, synthetic data generation, hyperparameter optimization.

1. Introduction

Designing and developing software requires both high-quality data and high accuracy to ensure the overall success of the project. Consequently, Software Development Effort Estimation (SDEE) is vital in project management, determining project feasibility, and impacting the distribution of funds [1]. Errors and inaccuracies in SDEE often lead to misjudgments of investment, potentially causing underfunding in successful projects or overspending in unsuccessful endeavors [2]. Therefore, many technical leaders, software engineers, and software development teams benefit from well-fitting SDEE, which improves overall project outcomes by providing more accurate estimations [3]. Our research addresses this challenge by combining state-of-the-art regression models with data augmentation techniques, distinguishing our approach from existing methodologies. Previous works, such as [4] on machine learning for effort estimation and [5] on Random Forests with different parametric models data, have laid important groundwork. However, our focus on data augmentation through TVAE and hyperparameter tuning provides a unique and potentially more reliable approach for generating precise results. The newest approach in estimating software project development is closely related to agile methodologies such as SCRUM, Kanban, Extreme Programming (XP), Crystal methods and similar. Previous research has shown initial evidence that Story Points (SPs) estimated by human experts may not accurately reflect the effort needed to realize agile software projects, although it is still a widely accepted measurement [6]. In the context of Agile software development, practitioners have introduced and used Story Points (SP) as an Agile-specific software size measurement unit. Unlike Function Point Analysis (FPA) and Use Case Point Analysis (UCP), SP does not follow a formal method of measurement. Instead, developers use them as a relative measure to

SQAMIA 2024: Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, September 9-11, 2024, Novi Sad, Serbia
EMAIL: N.Rankovic@uvt.nl (N.R); dragica.d.rankovic@gmail.com (D.R); mira@dmi.uns.ac.rs (M.I)
ORCID: 0000-0002-9910-5886 (N.R); 0000-0002-4464-0726 (D.R); 0000-0003-1946-0384 (M.I)

maintain the relative difference of stories in size by assigning a point value to each user story. One common approach to determining the story point value of a user story is to select one of the smallest stories in the backlog and assign it one story point. More complex and larger user stories then receive more points considering their size. Thus, any user story assigned two SP is considered twice as large as a user story assigned one SP. SP estimations need to be consistent throughout the project to ensure reliability [6]. However, SP's estimated value is often inaccurate, making it harder for the model to estimate the required effort to complete a project. Hence, there have been many attempts to increase its accuracy, recently in the form of machine and deep learning models. Many models have already been experimented with, and this study seeks to elaborate on this subject, evaluating the performance of additional five machine learning models (MLMs) and comparing them to a baseline model.

This paper aims to answer to what extent the models CatBoost, XGBoost, RandomForest, LightGBM, and Decision Trees can, in comparison to the Linear Regression baseline model, accurately estimate the story points required for SDEE within the TAWOS dataset. To further elaborate on specific aspects of model specialization, we will also seek to compare the machine learning models' abilities to estimate story points using different evaluation methods such as Pearson's correlation, Mean Squared Prediction Error (MSPE), and adjusted R^2. We will examine how hyperparameter tuning affects these metrics and if data augmentation techniques such as min-max scaling, transformation to a truncated normal distribution, and the use of a Tabular Variational Autoencoder (TVAE) can optimize machine learning models and enhance their accuracy. Therefore, the research contributions (RCs), along with their underlying motivations, are as follows:

Main RC: To what extent can additional regression models, in comparison to the baseline model, accurately estimate the story points required for SDEE within the TAWOS dataset?

Sub-RC1: How do regression models compare in their ability to estimate the story points required for software development within the TAWOS dataset, considering evaluation metrics such as Pearson's correlation, MSPE, and adjusted R^2?

Sub-RC2: How can multiple data augmentation techniques, including min- max scaling, transformation to a truncated normal distribution, and the use of a TVAE, be employed to optimize regression models for enhanced prediction accuracy of actual effort in SDEE?

Sub-RC3: How does hyperparameter tuning influence the performance of regression models in the context of SDEE within the TAWOS dataset?

The rest of the paper is organized as follows: Section 1 provides an overview of the current state-of-the-art literature in the field of software project estimation, focusing on machine learning and deep learning methods, and the poor use of any data augmentation techniques. Section 2 presents the methodology pipeline. Section 3 discusses the research findings. Section 4 delves into a detailed discussion of the obtained results. Concluding remarks, along with limitations and future directions, are provided in Section 5.

2. Related work

Historically, numerous software projects either failed or were left unfinished due to inadequate processes. Commonly employed methods included similarity-based estimation, the analysis and synthesis method, expert knowledge-based estimation, and various parametric techniques [7] Researchers and practitioners are increasingly aware of previous machine learning effort estimation techniques and are evaluating which methods yield more accurate results based on evaluation measures, datasets, and other attributes [8]. In [8] the authors investigated the performance of machine learning ensemble and solo techniques on various datasets. Analysis of 35 studies shows machine learning as the top choice for ensemble effort estimation due to promising error metrics. Additionally, machine learning-based software fault prediction (SFP) methods outperform traditional statistical approaches [9]. Empirical evidence suggests these techniques effectively identify fault proneness [10]. Techniques like Naïve Bayes, Random Forests, Logistic Regression, and decision trees are predominant for predictive estimations [11]. An automated text mining framework to investigate trends in 1015 papers on software development effort and cost estimation (SDECE) was proposed by study [12]. They found that artificial neural networks, fuzzy logic, regression, analogy-based approaches, and the COCOMO method are the most utilized for SDECE, with NASA and ISBSG datasets being the most employed. In [13] the authors assessed project duration estimation using Support Vector Regression and Multiple Linear Regression, finding Support Vector Regression significantly more precise. Predictive models using regression analysis, such as Decision Tree Regression, Extreme Gradient Boosting Regression, Bayesian Ridge Regression, and Support Vector Regression, were evaluated, with Bayesian Ridge Regression producing the best results.

Recent advancements in machine learning, also increased the popularity of using deep learning in software estimation field [14], [15], [16]. Surveyed defect prediction using deep learning, highlighting techniques for automatic extraction of code information and trends in effort and cost estimation was presented in [14]. Program analysis methods often have high false positive or negative rates [17]. Despite DL's promising results in automated vulnerability identification with up to 95% accuracy, they often perform below expectations. The authors in [18] discuss current DL-based vulnerability prediction challenges and future research directions. Due to traditional features' limitations in capturing semantic information, recent studies incorporate semantic features in defect prediction models [19].

To summarize, reducing model complexity while maintaining accuracy is a desirable state of each software industry use case. State-of-the-art models can offer a more efficient and effective approach to modeling complex systems. Data augmentation techniques help us continue performing analysis and understanding outcomes by providing more accurate and straightforward models.

3. Methodology & Experimental Setup

In this section, we will describe the steps for conducting the experimental part of our research. An illustration of the methodology pipeline is given in Figure 1. The TAWOS dataset comprises 31960 issues from 26 projects from repositories such as Atlassian, Apache, Appcelerator, Hyperledger, MongoDB, Sonatype, Moodle, Talendforge, and similar sources, where different programming languages such as Java, Python, C#, Go and others were used for different projects. It offers detailed information on versioning, issue tracking, developer assignments, and resolution times. This data is invaluable for research in software testing, maintenance, and task optimization. It includes version details like name, description, release date, and status (archived or released) and tracks issues through affected and fix versions. Developer assignment data helps in recommending the best developer for new issues and optimizing task assignments based on work-load. The dataset also provides issue status transitions, enabling the analysis of bug fix times and triage, thereby supporting advanced research using machine learning models.

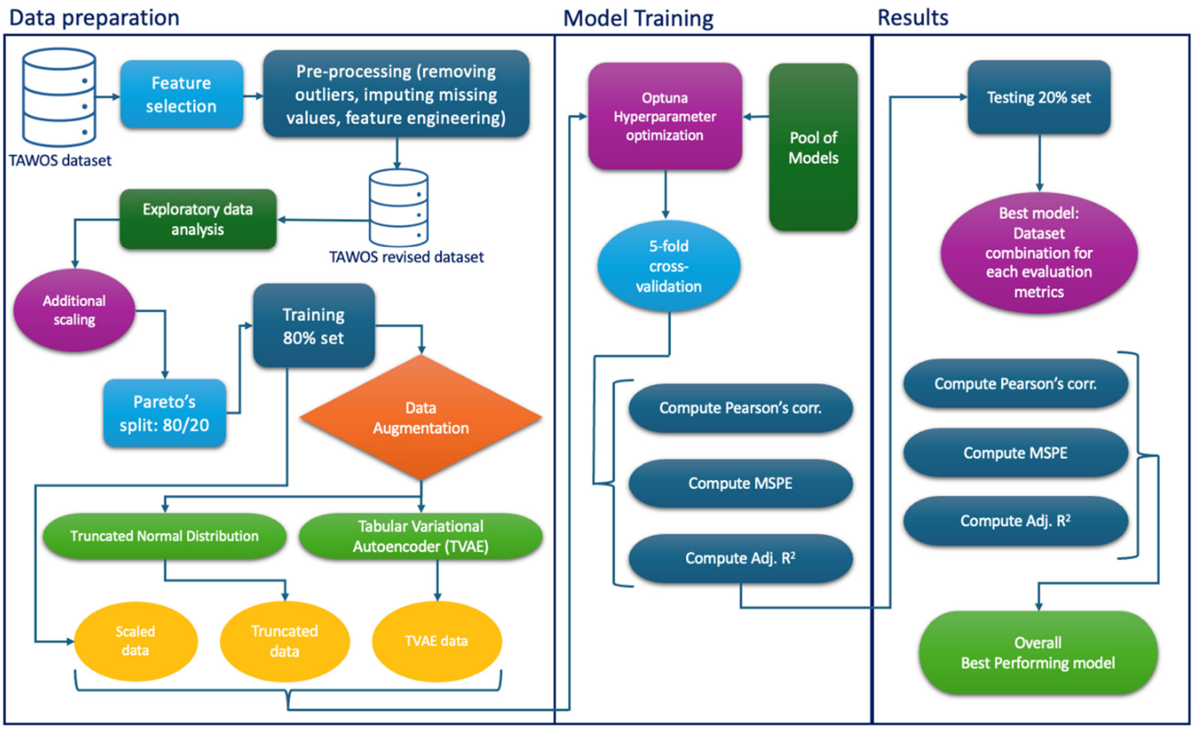

Figure 1. Methodology pipeline.

While #Components and #Developers are useful for detecting the anomalies in specific components produced for each user story from different developer, #Change Log is likely to be more directly correlated with the effort involved in the project (as reflected in story points) because it captures the actual work being done in terms of changes and updates to the project. This creates a fuller picture when combined with #Issues and #Bugs. Finally, the distribution of the target variable Story Points can be seen in Figure 2, along with descriptive statistics of the input features in Table 1. The smooth line shows the probability density, indicating how story points are distributed in the dataset. This helps to see the trend and pattern of the target variable distribution. "Target Value" on the x-axis represents the values of story points. "Frequency" on the y-axis shows how many times a particular target value appears in the dataset. The quantification step represents the width of the interval for grouping story points. Story points are grouped into bins (intervals) to better display the data distribution. This visualization helps to see how the values are distributed in specific ranges. The graph shows that there were about 40 projects in the first bin interval, but this does not mean that there were projects with exactly 0 story points. The interval from 0 to 500 can include projects with story points between 0 and 500. The minimum mark on the x-axis is 0 for histogram visualization, which often starts at 0 for clearer representation and interpretation of the distribution. Therefore, the x-axis mark does not mean that there were projects with exactly 0 story points.

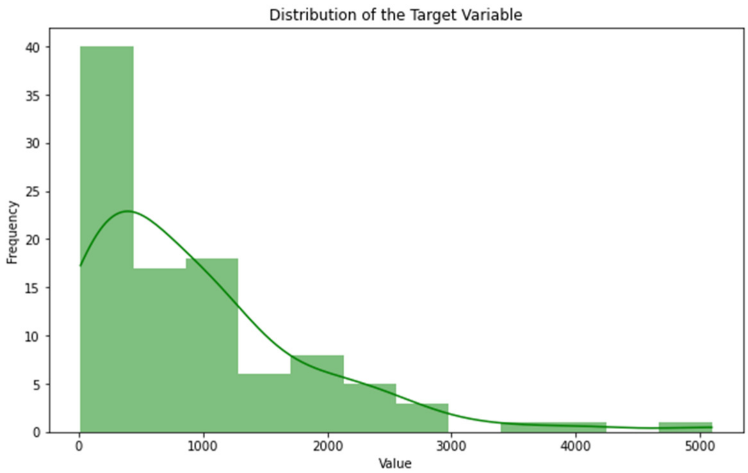

Figure 2. Story Points distribution in TAWOS dataset.

The Table 1. presents the descriptive statistics for the chosen TAWOS dataset [6], following thorough data cleaning, which involved the removal of missing values and the detection and removal of outliers at both upper and lower bounds. The average values for the metrics are: 11567 issues, 5399 bugs, 227815 change log entries, and 1584 story points per project. The standard deviations indicate considerable variability, particularly in the #Change Log (358604.554) and Story Points (3226.417), suggesting diverse project activities. Minimum and maximum values further illustrate the range of data, with issues ranging from 313 to 66.741 and story points from 209 to 20.664. Outliers were identified, where for example an entry with 1608.633 change log entries was identified and removed as an upper outlier. Additional techniques, such as data normalization and feature scaling, can be applied at this stage to ensure all features contribute equally to the analysis.

Table 1 TAWOS dataset descriptive statistics.

Term	#Issues	#Bugs	#Change Log	Story Points
Mean	11567.3409	5399.8636	227815.2500	1584.6364
Std. Deviation	14593.71297	8146.67080	358604.55400	3226.41663
Min.	313.00	123.00	4062.00	209.00
Max.	66741.00	41355.00	1608633.00	20664.00

Table 2 gives an overview of the terms and definitions for each of the chosen features and target variable. The correlation heatmap highlights several key relationships in the dataset, as shown in Figure 3. Notably, there is a strong positive correlation (0.60) between the scaling factor #Issues and the #Bugs, indicating that these variables tend to increase together. Additionally, both #Bugs and #Change Log show moderate positive correlations with actual effort of Story Points with coefficients of 0.39 and 0.47. This suggests that both #Bugs and #Change Log are factors influencing the actual effort required, which in the case of Story Points is usually expressed on W1 to W10 weighting scale where W1 reflects the weight of 1 and W10 the weight of 10 points. The histogram of the target variable, reveals a highly skewed distribution.

Table 2. Description of the features and target variable.

Valid projects without missing values	Definition
#Issues	Work items or tasks to be completed, including user stories and bugs.
#Bugs	Defects or problems in the software that need fixing.
#Change Log	Record of all notable changes made to the project.
Story Points	Measure of effort required to complete a user story or task, aiding in estimation and planning.

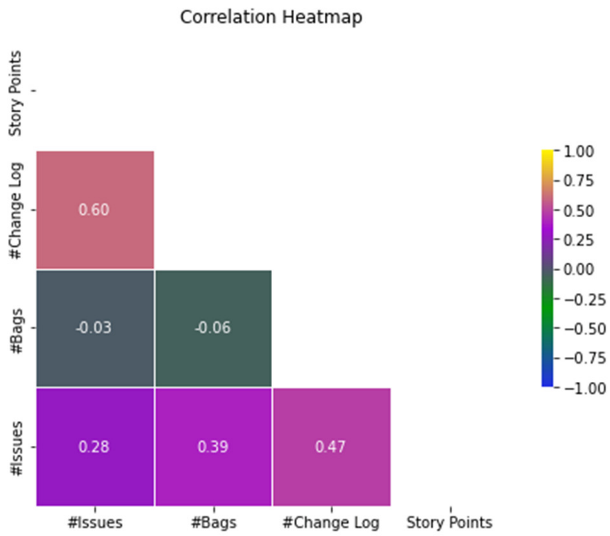

Figure 3. Correlation between the input features of the revised TAWOS dataset and Story Points.

Most projects require relatively low effort. However, there are a few projects that demand significantly higher effort, resulting in a long tail to the right. The data was split using an 80/20 Pareto Split, out of which 80% of the data was used for training and the remaining 20% of the data for the final testing of the models on unseen data. Additionally, it was experimented with 60/20/20 ratio, but better results were observed with 80/20 after numerous trial-error attempts. Moreover, the models were evaluated using 10-fold cross-validation, where the dataset is split into 10 parts, each serving as a test set while the rest are used for training, repeated 10 times. This mitigates the risk of relying on a single split and provides a reliable performance estimate. Additionally, statistical significance tests confirmed that performance differences were not due to random chance, ensuring robust evaluation. The whole data set was scaled using the Min-Max algorithm in order to have a uniform scale for all features and create more homogenous nature of the dataset. Data augmentation techniques such as Tabular Variational AutoEncoder (TVAE) and Truncated Normal Distribution were implemented. Truncated Normal Distribution is a statistical technique used to enlarge the dataset by creating new data points within a specific range derived from the normal distribution, again tailoring the statistical characteristics of the data from the original dataset. TVAE is a type of generative model tailored specifically for creating synthetic data. By these adjustments three different datasets were created, namely, scaled, TVAE and Truncated, to see which performed the best. In this research, we used six different machine learning models and trained them on each dataset variant. These models were XGBoost Regression, Random Forest Regression, CatBoost Regression, Decision Trees Regression, and LightGBM Regression. Linear Regression was also included as a baseline model to compare with the other models, totaling six models. To further increase the accuracy of the model's predictions we conducted hyperparameter optimization using Optuna [20]. Evaluation metrics such as Mean Squared Prediction Error (MSPE), adjusted R^2, and Pearson's correlation coefficient were implemented with 5-fold cross-validation, resulting in three model/dataset combinations with the most optimal performance.

4. Results

In this section, we will delve into the details of the results obtained by each model and compare the overall results. We recorded the training time for each model and calculated the average. The average training times are presented in Table 3. Three models were identified as the best performers, each excelling in a different metric. The performance of these models on the test set is presented in Table 4 and Figure 7. The hyperparameters for the final models are detailed in Table 5. As indicated in the tables and figure, these models are labeled 1 through 3 and are further described below. The residuals for all three models in Figure 5, while showing some increase, suggest that each model captures the underlying trend reasonably well, with Model 3 being the most accurate. Model 3's smaller residuals indicate it predicts Story Points with greater precision, making it the best performing model among the three. Despite some areas for improvement, all models demonstrate their ability to follow the data's trend and provide useful predictions.

Table 3. Average running times per each model.

Model	#Average running time(s)
Linear Regression (Baseline)	0.021
Decision Tree Regressor	0.026
Random Forest Regressor	3.524
XGBoost Regressor	0.925
CatBoost Regressor	1.599
LightGBM Regressor	0.417

Table 4. Best-performing models and final results on test data measured by chosen evaluation metrics.

Model No.	Evaluation metrics	Model	Synthetic and scaled	Adj. R^2	MSPE	Pearsons' Corr.
1	Average Adj. R^2 (0.55)	Random Forest Regressor	TVAE	0.59	140011	0.81
2	Average MSPE (328288)	CatBoost Regressor	Scaled	0.39	200011	0.74
3	Average Pearson's Corr. (0.77)	Decision Tree Regressor	Scaled	0.350	234500	0.66

4.1. Model 1 (Random Forest Regressor, TVAE)

Model 1 was trained on TVAE-augmented data and achieved the best performance among all models. It was evaluated based on the 5-fold cross-validated average adjusted R^2 (0.55) on its training data. On the final 20% test set, it maintained strong performance, achieving an adjusted R^2 of 0.59, a Pearson's correlation coefficient of 0.81, and an MSPE of 140001. These results indicate a good fit, strong alignment with observed data, and enhanced prediction accuracy.

4.2. Model 2 (CatBoost Regressor, unaugmented)

Model 2 was trained on unaugmented data and achieved the second-highest performance across the three test metrics on the final 20% split. Based on the 5-fold cross-validated average MSPE (328288) on its training data, it obtained an adjusted R^2 of 0.39 and a Pearson's correlation coefficient of 0.74 on the final test data. These metrics indicate a weak goodness-of-fit and moderate alignment with the observed data. Additionally, its lower average MSPE of 200011 suggests moderate prediction accuracy.

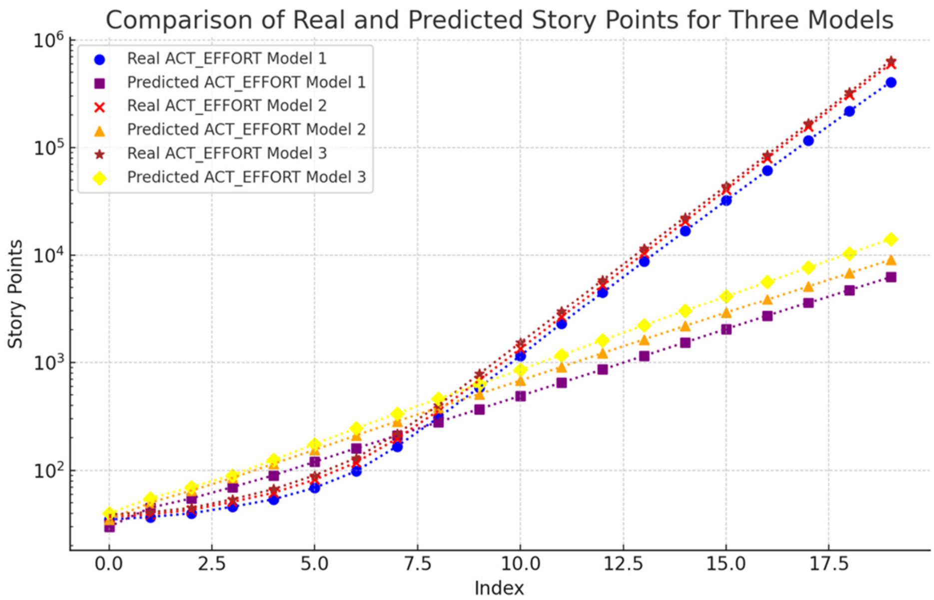

Figure 4. Model performance comparisons including logarithmic fit on actual Vs. predicted values.

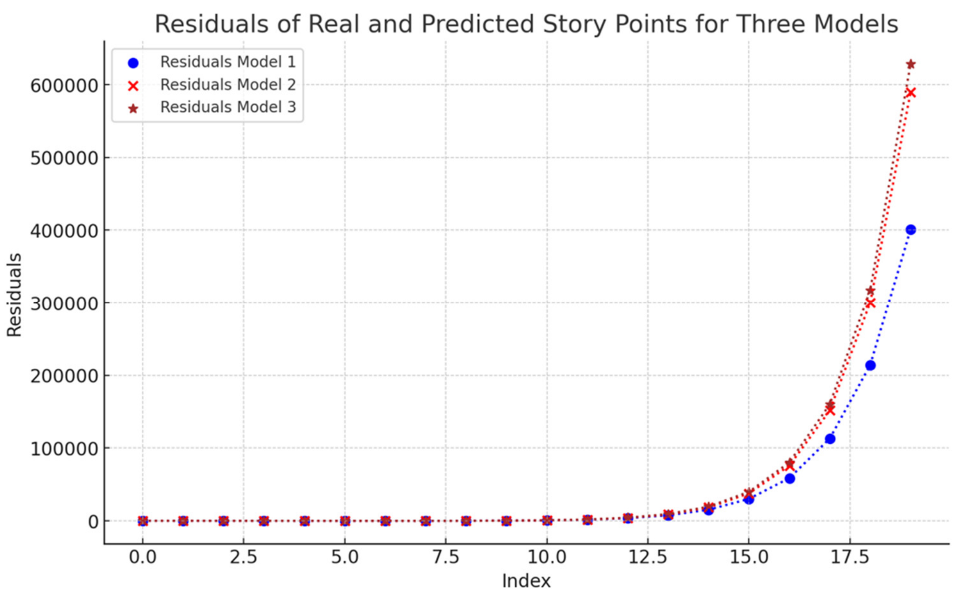

Figure 5. Model performance comparisons including logarithmic fit on actual Vs. predicted values.

4.3. Model 3 (Decision Tree Regressor, unaugmented)

Model 3 was trained on the original data and exhibited the weakest performance among the three models on the final 20% test split, despite achieving a high 5-fold cross-validated average Pearson's correlation coefficient during training (0.77). On the test set, the model resulted in a poor adjusted R^2 of 0.35 and a Pearson's correlation coefficient of 0.66, indicating weak goodness-of-fit and poor alignment with the observed data. Furthermore, its average MSPE of 234500 suggests moderate prediction accuracy. Consistent with our hypothesis, the baseline linear regression model is consistently outperformed by all other models across all evaluation metrics (adjusted R^2, MSPE, and Pearson's correlation coefficient). This is evidenced by the top-performing models being the Random Forest Regressor (Model 1), CatBoost Regressor (Model 2), and Decision Tree Regressor (Model 3). These models co-incidentally follow a clear order of efficacy as shown in Figure 6 and Figure 7. Our findings also suggest that data augmentation can be beneficial for the TAWOS dataset. Model 1, trained on the TVAE-augmented data, achieved the best performance on the final test set for all three metrics, indicating the potential effectiveness of TVAE in improving performance on the TAWOS dataset. Examining the predicted data, as well as the residuals, revealed a trend where the models exhibit greater accuracy in predicting software projects with lower actual effort values. Conversely, projects with higher actual effort proved more challenging to predict. This phenomenon can be also seen in Figure 5.

Table 5. The hyperparameters combination settings for three best-performing models.

Model	Hyperparameters
1	'bootstrap': True, 'ccp_alpha': 0.0, 'criterion': 'squared_error', 'max_depth': 21, 'max_features': 'sqrt', 'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 5, 'min_weight_fraction_leaf': 0.0, 'monotonic_cst': None, 'n_estimators': 435, 'n_jobs': None, 'oob_score': False, 'random_state': None, 'verbose': 0, 'warm_start': False
2	'learning_rate': 0.091, 'depth': 3, 'l2_leaf_reg': 0.734, 'loss_function': 'RMSE', 'border_count': 446, 'verbose': False, 'random_strength': 0.976, 'bagging_temperature': 0.479, 'num_trees': 896
3	'ccp_alpha': 0.0, 'criterion': 'poisson', 'max_depth': 21, 'max_features': 0.920, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 2, 'min_samples_split': 4, 'min_weight_fraction_leaf': 0.0, 'monotonic_cst': None, 'random_state': None, 'splitter': 'best'

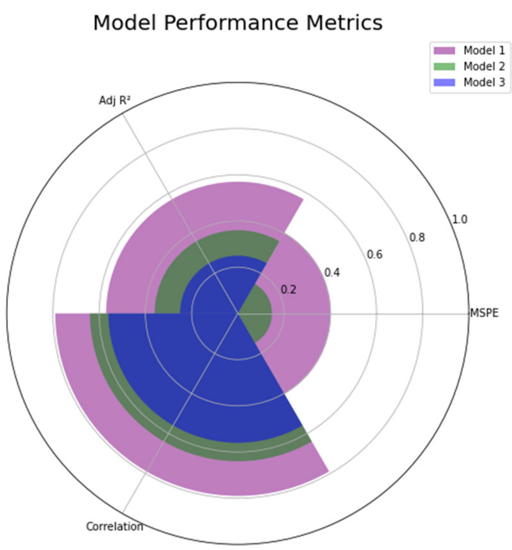

Figure 6. Correlation between the input features of the revised TAWOS dataset and Story Points.

Figure 7. Model 1 Vs. Model 2 Vs. Model 3 model preformance.

5. Discussion

The results highlight that advanced machine learning models outperform the baseline linear regression model across the performance metrics. Specifically, the Random Forest Regression (Model 1), CatBoost Regressor (Model 2), and Decision Tree Regressor (Model 3) emerged as the best-performing models, each having its unique strengths and weaknesses, provoking further discussion. An important aspect of our study was analyzing the correlation between input features and the target variable, Story Point. The correlation heatmap indicated strong positive correlations between several key variables (scaling factor #Issues and #Bugs at 0.60, #Bugs and #Change Log with Story Points at 0.39 and 0.47). The Random Forest Regression model (Model 1), based on a 5-fold cross-validated average, achieved the highest adjusted R^2 score of 0.55 when trained on the TVAE-augmented dataset. This result underscores the model's ability to handle complex data environments by effectively capturing underlying data patterns. The data augmentation through TVAE appears to have provided a more detailed dataset that allowed the Random Forest model to make more accurate predictions. However, this also raises questions about the computational complexity and the potential for overfitting, based on the high-dimensional nature of the augmented data itself. The final evaluation on the 20% data split showed an adjusted R^2 of 0.59, a Pearson's correlation coefficient of 0.81, and an MSPE of 140011, indicating strong performance The CatBoost Regression model (Model 2) performed well in minimizing the Mean Squared Prediction Error (MSPE) with a value of 328288 on the scaled dataset. This highlights its potential for practical applications in software project management and suggests that it can effectively handle imbalanced data distributions and various feature scales, making it a versatile model. However, its relatively lower adjusted R^2 score may indicate that it may not fully capture the variability in the data, pointing to potential areas for improvement in feature selection and model tuning. The final evaluation revealed an adjusted R^2 of 0.39, a Pearson's correlation coefficient of 0.74, and an MSPE of 200011, reflecting moderate predictive accuracy but highlighting areas for improvement. Decision Tree Regressor (Model 3) demonstrated a strong linear relationship between predicted and actual values with a correlation coefficient of 0.66. However, its lower adjusted R^2 score of 0.35 and MSPE of 234500 suggest potential overfitting scenarios, where the model fits the training data but fails to generalize to new data. The aforementioned final evaluation on the 20% data split indicates that it might benefit from more sophisticated ensemble methods to potentially improve the algorithm's generalization capabilities. Despite lower adjusted R^2 performance, the Decision Tree's high correlation coefficient proves its potential in mapping relationships within the dataset. Data augmentation, particularly using TVAE, was found to enhance the performance of the Random Forest model. The synthetic data generated by TVAE improved the original dataset, suggesting that data preprocessing and augmentation could be beneficial for improving model accuracy. The comprehensive search for hyperparameter tuning using Optuna, involving 300 trials for each model-data pair and each metric, ensured that our models were fine-tuned to their optimal configurations. This process showcased that it can influence model performance, as shown by the improved metrics across the models. For instance, the Random Forest model's hyperparameters were tuned to balance depth and feature selection, which enhanced its performance on the augmented dataset. The results suggest that their effectiveness varied based on the differences in handling feature interactions, which indicates the potential need for a more refined feature engineering for hyperparameter tuning.

6. Conclusion

In Software Development Effort Estimation (SDEE), our approach significantly enhances predictive accuracy and addresses data variability challenges. We leverage advanced machine learning models like Random Forest Regressor, CatBoost Regressor, and Decision Tree Regressor, combined with data augmentation techniques such as Tabular Variational AutoEncoder (TVAE) and hyperparameter optimization using Optuna, to achieve notable improvements. Our methodology demonstrated superior reliability in effort predictions on the open-source agile TAWOS dataset, outperforming the Linear Regression baseline. The utilization of these sophisticated models ensures a robust handling of complex data patterns, leading to more accurate and reliable effort estimations. Furthermore, the integration of TVAE helps in overcoming data sparsity issues, providing a richer and more comprehensive dataset for training. Overall, our enhanced approach sets a new benchmark in SDEE, promoting efficiency and precision in project management and planning. The results of this study can significantly enhance software development project planning and management. Improved effort predictions allow for more accurate resource and deadline estimations, reducing delays and budget overruns. Using a Random Forest regressor on TVAE-augmented data provides high predictive accuracy, increases estimate reliability, and reduces the risk of resource assessment errors. These models help organizations better manage risks, optimize costs, and improve project efficiency.

6.1. Limitations

Despite these promising results, several limitations warrant attention. The computational complexity associated with TVAE augmentation and hyperparameter optimization is considerable, potentially restricting the scalability of our approach for larger datasets or real-time applications. Additionally, the skewed distribution of effort in the dataset, with most projects requiring relatively low effort and a few demanding significantly higher effort, poses challenges in achieving consistent predictive accuracy across all project types.

6.2. Future directions

Future research should focus on further enhancing SDEE models. This includes expanding the dataset through advanced feature engineering techniques to improve predictive power. Incorporating datasets from diverse sources and adopting different approaches such as Functional Point Analysis and Use Case Point Analysis can increase the models' applicability across various project types. Moreover, adapting the models for real-time effort estimation is crucial to ensure accurate predictions in dynamic project environments. Addressing these limitations will enable future work to build on our findings and develop more effective and scalable SDEE solutions. For instance, exploring the use of different representations of graph neural networks or subsets of recurrent neural networks such as Fuzzy Cognitive Maps to perform *WHAT-IF* simulations for various scenarios could significantly enhance the robustness and applicability of SDEE models [21]. Ultimately, this would contribute to more successful software project management and resource allocation.

Acknowledgement

This work is partially supported by CERCIRAS "Connecting Education and Research Communities for an Innovative Resource Aware Society" COST Action CA19135 funded by COST Association.

References

[1] Feizpour, E., Tahayori, H., & Sami, A. CoBRA without experts: New paradigm for software development effort estimation using COCOMO metrics. Journal of Software: Evolution and Process, 35(12), e2569, 2023.

[2] Sunil Kumar Gouda, Prithvi Raj, et al. A methodology for software cost estimation using machine learning techniques: International conference on recent trends in artificial intelligence, iot, smart cities & applications (icaisc-2020), 2020.

[3] Jianglin Huang, Yan-Fu Li, and Min Xie. An empirical analysis of data preprocessing for machine learning-based software cost estimation. Information and software Technology, 67:108–127, 2015.

[4] Yasir Mahmood, Nazri Kama, Azri Azmi, Ahmad Salman Khan, and Mazlan Ali. Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. Software: Practice and experience, 52(1):39–65, 2022.

[5] Ilham Cahya Suherman, Riyanarto Sarno, et al. Implementation of random forest regression for cocomo ii effort estimation. In 2020 international seminar on application for technology of information and communication (iSemantic), pages 476–481.IEEE, 2020.

[6] Vali Tawosi, Afnan Al-Subaihin, Rebecca Moussa, and Federica Sarro. A versatile dataset of agile open source software projects. In Proceedings of the 19th International Conference on Mining Software Repositories, pages 707–711, 2022.

[7] Rankovic, N., Rankovic, D., Ivanovic, M., & Lazic, L. A new approach to software effort estimation using different artificial neural network architectures and Taguchi orthogonal arrays. *IEEE access*, *9*, 26926-26936, 2021.

[8] Mahmood, Y., Kama, N., Azmi, A., Khan, A. S., & Ali, M. Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. Software: Practice and experience, 52(1), 39-65, 2022.

[9] Althar, R. R., & Samanta, D. The realist approach for evaluation of computational intelligence in software engineering. Innovations in Systems and Software Engineering, 17(1), 17-27, 2021.

[10] Pandey, S. K., Mishra, R. B., & Tripathi, A. K. Machine learning based methods for software fault prediction: A survey. Expert Systems with Applications, 172, 114595, 2021.

[11] Garg, Y. Comparative analysis of machine learning techniques in effort estimation. In 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON) (Vol. 1, pp. 401-405). IEEE, May 2022.

[12] Jadhav, A., Kaur, M., & Akter, F. Evolution of software development effort and cost estimation techniques: five decades study using automated text mining approach. Mathematical Problems in Engineering, 2022(1), 5782587, 2022.

[13] Van Hai, V., Javed, M., Abbas, Z., Ari, M., & Bílá, M. On the Software Projects' Duration Estimation Using Support Vector Regression. In Proceedings of the Computational Methods in Systems and Software (pp. 288-298). Cham: Springer International Publishing, 2022.

[14] Nevendra, M., & Singh, P. (2021). Software defect prediction using deep learning. Acta Polytechnica Hungarica, 18(10), 173-189.

[15] Alghanim, F., Azzeh, M., El-Hassan, A., & Qattous, H. (2022). Software defect density prediction using deep learning. *IEEE Access*, *10*, 114629-114641.

[16] Yang, Y., Xia, X., Lo, D., & Grundy, J. A survey on deep learning for software engineering. *ACM Computing Surveys (CSUR)*, *54*(10s), 1-73, 2022.

[17] Chakraborty, S., Krishna, R., Ding, Y., & Ray, B. Deep learning based vulnerability detection: Are we there yet?. IEEE Transactions on Software Engineering, 48(9), 3280-3296, 2021.
[18] Tadapaneni, P., Nadella, N. C., Divyanjali, M., & Sangeetha, Y. Software defect prediction based on machine learning and deep learning. In 2022 International Conference on Inventive Computation Technologies (ICICT) (pp. 116-122). IEEE, July, 2022.
[19] Wang, H., Zhuang, W., & Zhang, X. Software defect prediction based on gated hierarchical LSTMs. IEEE Transactions on Reliability, 70(2), 711-727, 2021.
[20] Rimal, Y., Sharma, N., & Alsadoon, A. The accuracy of machine learning models relies on hyperparameter tuning: student result classification using random forest, randomized search, grid search, bayesian, genetic, and optuna algorithms. *Multimedia Tools and Applications*, 1-16, 2024.
[21] Maden, A., & Yücenur, G. N. Evaluation of sustainable metaverse characteristics using scenario-based fuzzy cognitive map. Computers in Human Behavior, 152, 108090, 2024.