International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 3 (2026)

ijerst.editor@gmail.com

editor@ijerst.com

ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

AI-DRIVEN EFFORT AND TIME PREDICTION FOR SOFTWARE PROJECT TASKS

[1]Vadithya Suchithra, [2]Dr. M. Chandra Mohan
[1]MTech Student, [2]Professor
Department of CSE
JNTUH Kukatpally

ABSTRACT

Estimation of software project effort and software project time are very important in successful project planning, allocation of resources, budgeting and timely delivery. Traditional approaches to estimation are based on expert judgment and past experience, and may not be accurate due to uncertainty, evolving requirements and the lack of project information. This project aims to address these challenges by developing an AI-powered software effort and software completion time prediction framework based on advanced machine learning techniques.

The proposed system uses the CESAW software project dataset and uses comprehensive data preprocessing methods such as missing value treatment, categorical encoding, feature scaling, outlier removal, and feature engineering. In order to overcome the challenges of small dataset and make the model more robust, synthetic project data is generated based on Conditional Tabular Generative Adversarial Networks (CTGAN). Several regression models such as “Linear Regression”, “Decision Tree”, “Random Forest”, “Gradient Boosting”, “XGBoost” and “LightGBM” are trained and tested to find the most correct prediction model.

The standard regression metrics including Coefficient of Determination (R^2), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE) are used to assess the performance of the models. To increase transparency and trust in the predictions, SHAP (SHapley Additive exPlanations) is added to explain what role individual project features play in how much effort and development time they predict.

The results of experiments have provided evidence that the ensemble learning models have better prediction accuracy and reliability than the traditional estimation approach. Proposed framework based on AI is an effective decision support tool for software project manager where they can get accurate effort estimates, optimal resource planning, decreased project risk, and optimized project scheduling. The findings of this study underscore the potential of combining machine learning, synthetic data generation, and explainable AI in the software project management and estimation landscape.

I. INTRODUCTION

In software project management, software effort and software time estimation play a crucial role in the project planning, scheduling, budgeting and resource allocation. The conventional estimation techniques are largely based on expert judgment and manual computation, which can be subjective and lead to errors in the predictions, causing project delays and cost overruns.

With the recent progress in Artificial Intelligence (AI) and Machine Learning (ML), the creation of intelligent systems has become possible that can give you accurate and data-driven estimations. By examining the project data from past projects, machine learning algorithms can discover intricate relationships between project attributes, which can increase the precision of the predictions and facilitate better decision-making.

This project aims to create an AI based framework to predict the effort and completion time of software tasks, using the CESAW dataset. The data is pre-processed and engineered using different methods to enhance the data quality. Random Forest, XGBoost, LightGBM, Gradient Boosting Machine and Stacking Ensemble are implemented and evaluated with the help of MAE, RMSE and R^2 metrics.

Explainable Artificial Intelligence (XAI) is integrated to enhance transparency and interpretability, explaining model predictions. To further demonstrate the robustness of the models and overcome the scarcity of data, synthetic datasets are also generated based on the Synthetic Data Vault (SDV) framework.

The proposed framework will aim to offer an accurate, interpretable, and scalable solution for software effort and time estimation in order to help project managers to enhance planning, resource allocation and decision-making processes.

1.1 MOTIVATION

The key to the success of software development projects is to accurately estimate software development effort and software development time. Accurate estimates enable effective project planning, scheduling, budgeting and allocation of resources; while inaccurate estimates could cause delays, budget overruns, inefficient use of resources, and decreased software quality.

Most of the traditional estimation techniques are based on expert judgment and on past experiences, and are prone to the human nature, uncertainty and complexity of software projects. This means that there is a need for intelligent estimation techniques that are data driven and can make more accurate and reliable predictions.

With the latest developments in Artificial Intelligence (AI) and Machine Learning (ML), automated prediction systems have come into existence that can analyse historical project data and produce accurate predictions. These methods help project managers to make informed decisions and enhance the overall performance of a project.

1.2 OBJECTIVES

The main goal of this project is to use machine learning techniques to create an artificial intelligence-driven (AI) framework that can predict task effort and completion time for software tasks. The goal of the proposed framework is to increase estimation accuracy and to make the estimation process more transparent to the users, and to aid efficient software project planning and resource management.

The project has specific objectives as follows:

1. To study and analyse the CESAW software project data sets and determine the factors that affect the effort and duration of software tasks.
2. To clean data for model building by data preprocessing techniques like missing value treatment, outlier removal, feature extraction, feature engineering, categorical encoding, and feature scaling to enhance data quality and model performance.
3. To build predictive models by applying different machine learning algorithms such as Random Forest (RF), XGBoost, LightGBM (LGBM), Gradient Boosting Machine (GBM), and Stacking Ensemble models for the prediction of Software effort and time.
4. Assess and compare the performance of various Machine Learning models on common evaluation parameters like Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), and the coefficient of determination (R^2 Score).
5. To apply Explainable Artificial Intelligence (XAI) approaches to explain the model prediction and to find top features influencing the prediction results with SHAP (SHapley Additive Explanations).

1.3 SCOPE

In this project, the scope is limited to creating an Artificial Intelligence (AI) based model for software task effort and completion time

prediction based on the historical data of software projects from the datasets of CESAW (Task and Time) datasets. In order to enhance the quality of data and the prediction accuracy, various data preprocessing techniques were employed in the framework that includes handling missing values, removing outliers, feature engineering, encoding, and feature scaling.

A number of machine learning models like Random Forest (RF), XGBoost, LightGBM (LGBM), Gradient Boosting Machine (GBM) and Stacking Ensemble are implemented and tested using regression metrics like RMSE, MAE, and R^2 Score to determine the best prediction model.

The project also incorporates Explainable Artificial Intelligence (XAI) with SHAP (SHapley Additive Explanations), which helps to improve the transparency and interpretability of model predictions. In addition, synthetic data is created based on the Synthetic Data Vault (SDV) to evaluate model robustness, enhance the generalization ability and aid privacy-preserving data analysis.

II. LITERATURE REVIEW

Estimating software effort and time is one of the most important activities in software project management, which can have a significant effect on project planning, scheduling, budgeting and resource allocation. Organizations can finish projects in a timely and cost-effective manner without compromising the quality of their software with accurate estimation. But software projects are plagued by uncertainties like shifting requirements, differing team skill sets and changing technologies, which makes it difficult to accurately estimate such projects. Traditional estimation methods are mainly based on experience and the expertise of the individual assessing, potentially leading to subjectivity and inconsistencies.

As Artificial Intelligence (AI) and Machine Learning (ML) technologies continue to evolve, researchers have been spending more effort on the development of estimation techniques based on data. Using machine learning algorithms, historical project information can be entered to determine complex relationships between the project attributes and to predict software effort and completion time with accuracy. Ensemble learning models, explainable AI techniques, and synthetic data generation have been shown to enhance the accuracy of estimation, enhance robustness, and enhance model reliability, respectively, in recent studies.

In this chapter, we extensively review the related works on software effort and time estimation, machine learning-based prediction models, Explainable Artificial Intelligence (XAI), and the challenges in current research and the motivation behind the proposed work.

2.1 Software Effort and Time Estimation

The term software effort estimation means predicting the effort necessary to complete a software development task and the term time estimation means estimating the time needed to complete a software development task. These estimates are critical for planning, cost control, risk analysis and allocating resources. The oversight of estimation can cause the project to be delayed, run over budget, and produce software that is of lower quality.

Researchers have devised several methods to enhance accuracy of their estimates. Early approaches involved a lot of human knowledge and manual calculations, and recent studies have used machine learning models that learn from project data in the past. As the number of software engineering datasets has grown, data-driven software engineering estimation techniques have gained more traction.

2.1.1 Importance of Software Effort Estimation

One of the most crucial elements of software project management is software effort estimation. It can aid the project managers to identify the amount of developers needed, to calculate the cost of the project, distribute the resources and set a realistic deadline. Project planning becomes more accurate and schedule overruns and budget violations are less likely with accurate effort estimates.

Effort estimation is also very important in the study of productivity and performance evaluation. Estimation results are employed in organizations to gauge the feasibility of a project, manage risks and optimise the use of resources. Effort estimation is a critical process in software projects, especially when they are becoming more complex, in order to ensure a successful project outcome.

2.1.2 Time Estimation in Software Projects

Time estimation is a process of forecasting the time necessary to perform the software development tasks. It is used for scheduling, planning milestones and monitoring progress. By coordinating development activities, ensuring customer expectations are met, and ensuring timely delivery of projects, accurate time estimates help organizations coordinate their development activities.

The software development time is influenced by various factors such as complexity of the tasks, experience of the developers, team collaboration, the size of the software, and the methodology used for its development. These factors can be input into machine learning models and based on past project data, time predictions can be made accurately.

2.1.3 Prediction of Task Level efforts and durations.

Most traditional software estimation studies are concentrated on the project level estimation in which the effort and duration of the entire software project are predicted. Project level predictions may not, however, be detailed enough to support the day-to-day project management.

Task-level prediction estimates the time and effort needed to perform each individual software development task, e.g., requirement analysis, coding, testing, debugging, documentation, and maintenance. These detailed estimates allow project managers to make better use of project resources, track the progress of tasks, and identify possible delays.

Recently, it was shown that machine learning algorithms are effective in forecasting effort and duration at task-level by learning patterns based on past tasks. This approach gives more detailed and specific information, than project level estimation methods.

2.2 Machine Learning Models for Software Prediction

Because of its ability to handle large amounts of data and uncover complex relationships between project variables, machine learning has become a popular technology used for software effort and time estimation. Machine learning models learn directly from historical project data and learn and adapt as they go.

Several machine learning approaches have been used for software estimation problems: each approach has different strengths and weaknesses for prediction accuracy, interpretability, and computational efficiency.

2.2.1 Random Forest for Effort Estimation

Random Forest is an ensemble learning algorithm that generates multiple decision trees for better prediction. A decision tree is trained on a random portion of the data and the overall prediction is computed as the average of the predictions of the individual trees.

Random Forest is a machine learning algorithm that is the most popular for software effort estimation due to its ability to deal with high dimensional data, non-linear relationships, and avoid overfitting. It is an algorithm that is well suited for noisy and missing data, particularly in real-world software engineering data.

Random Forest models have been reported to perform well in software effort estimation task in several studies. It is also useful for obtaining feature importance, thereby enhancing model interpretability.

2.2.2 XGBoost and LightGBM Models

XGBoost (Extreme Gradient Boosting) is a highly effective machine learning algorithm that utilizes gradient boosting techniques. It is sequential in the creation of predicting models, each of which tries to eliminate the errors made by the preceding models. XGBoost is highly predictive, scalable and efficient.

LightGBM (Light Gradient Boosting Machine) is another powerful boosting algorithm that aims to enhance the training time and memory usage. LightGBM grows trees in a leaf-wise

manner, which results in high accuracy while using less computation.

In software effort estimation research, XGBoost and LightGBM have been found to be superior. Given their capability of managing advanced feature interactions and huge data sets, they are very useful for predictive modelling use.

The concept of ensemble learning and stacking methods is explained. Ensemble learning and stacking methods are explained.

Ensemble learning is the technique that involves using more than one machine learning model to make predictions to get better accuracy and robustness. Ensemble methods combine the benefits of various algorithms, and frequently yield superior performance.

One of the most popular ensemble methods is stacking, where multiple base models make predictions which are then aggregated by a meta-model. This way prediction error and the performance of generalization is reduced.

The stacking approach has been implemented with success in software effort estimation, such as stacking Random Forest, XGBoost, and LightGBM algorithms, and predict the effort with a high level of accuracy.

We need to explain AI and get people in the loop.2.3 Explainable Artificial Intelligence (XAI)

While good predictive accuracy can be obtained by using machine learning models, many sophisticated algorithms are considered black boxes. This lack of transparency can diminish trust in the user, or make it hard to see why predictions are made.

Explainable Artificial Intelligence (XAI) tackles this by offering techniques that clarify the functioning and results of the models.

2.3.2 Model Interpretability via Exploration of Data and Models

Interpretability is crucial for systems based on Artificial Intelligence to generate trust. In software project management, project managers need explanations of the prediction results so that they can make appropriate decisions on resource allocation, scheduling and risk management.

Explainable models enable stakeholders to gain insights into how different attributes of a project are related to predicted effort and duration values. It helps in effective planning and boosts the trust in decision-making based on AI.

2.3.2 SHAP-Based Feature Explanation

SHAP (SHapley Additive Explanations) is one of the most popular explainability methods in machine learning. It is predicated on the game theory notion of cooperation and assigns contribution scores to the individual features.

SHAP provides both global and local explanations. Global explanations are used to find the most important features for the entire set of data, and local explanations explain how particular features affect individual predictions. Visualizations created by SHAP can assist the user to understand the behaviour of the model and to identify important factors impacting software effort and time estimation.

2.3.3 Explainability in Software Project Management

SHAP can be incorporated into software estimation systems to help the project manager understand the reasons behind a specific effort/duration estimate. This helps to clarify, aid decision making and support risk analysis.

Explainable AI is also a helpful tool for researchers to validate machine learning models and understand the potential biases in the predictions that are made.2.4 Synthetic Data Generation and Validation.

III. PROPOSED SYSTEM

The estimation of software effort and time plays an important role in software project management because it directly affects the project planning, scheduling, budgeting and resource allocation. When estimating a software project accurately, companies can be more likely to complete the project on time and within budget while causing software delays, cost overruns, poor resource use and lower software quality if the estimate is incorrect. In traditional estimation methods, the estimation process is often subject to experts' judgment and prior experiences that can lead to

subjectivity and inconsistencies. The recent developments in Artificial Intelligence (AI) and Machine Learning (ML) have opened new avenues for enhancing the accuracy of estimations using data-driven methods. Machine learning algorithms can process data from previous software projects, discover intricate patterns, and make accurate forecasts for future software development endeavours. These smart prediction models assist the project managers in resource allocation, scheduling and project monitoring decisions. This chapter introduces the proposed AImatic software task effort and completion time prediction system. The framework proposed here involves the use of CESAW data along with the data preprocessing, feature engineering, machine learning model development, explainable artificial intelligence, and synthetic data generation techniques. Several of the machine learning models, such as Random Forest, XGBoost, LightGBM, Gradient Boosting Machine, and Stacking Ensemble, are used to make accurate predictions at the task level. Furthermore, for the models, it is integrated with Explainable Artificial Intelligence (XAI) to enhance the model transparency and offer insights into the factors that influence prediction outcomes. In addition to real datasets, synthetic datasets created with the Synthetic Data Vault (SDV) framework are also used to assess the model's robustness and generalization. The proposed system is designed to offer an accurate, interpretable and scalable solution for software effort and time estimation with the support of effective project planning and management.

3.1 SYSTEM ARCHITECTURE

3.1.1 Architecture of the Effort and Time prediction of Software project task

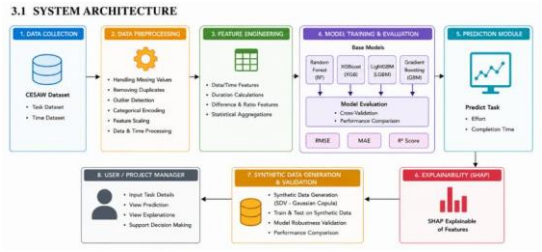

Fig. 3.1.1 Architecture of Effort and Time prediction

Figure 3.1.1 outlines the workflow, which has been described in detail through the following sequential steps:

The sequential steps of the proposed system:

Step 1: Data Collection Data collection of software project data from CESAW dataset is the initial step in the process. The dataset contains Task and Time datasets, with data related to the software development activities, software development phases, team members, and records of how tasks are executed.

Step 2: Data Preprocessing The data collected is used for data pre-processing to enhance data quality. It involves the following steps: missing value treatment, data cleansing of duplicate entries, outlier detection and treatment, encoding categorical attributes, numerical feature scaling, and date and time processing.

Step 3: Feature Engineering The pre-processed data is enriched with meaningful information using feature engineering techniques. These encompass features such as date/time attributes, calculation of task durations, creation of difference and ratio attributes, and statistical aggregation of features to improve prediction accuracy.

Step 4: Model Training and Evaluation

Various machine learning models, such as Random Forest (RF), XGBoost (XGB), LightGBM (LGBM) and Gradient Boosting Machine (GBM), are trained using the engineered features. The models are validated using the cross-validation method and the following performance measures: RMSE, MAE, R2 Score.

Step 5: Prediction Module The model that best fits the data is used to estimate software task effort and completion time. These forecasts help project managers in good planning and scheduling. The sixth step is to explainability with SHAP.SHAP is used for explainability in Step 6. SHAP (SHapley Additive Explanations) is used to explain model predictions, by showing the contribution of each feature to the outcome of the prediction, making the system more transparent and trusted to trust.

Step 7: Synthetic Data Generation & Validation
The SDV framework is used to generate synthetic datasets, based on the Gaussian Copula model. The trained models are tested on synthetic data to evaluate their robustness, generalisation and performance uniformity.
Step 8: User/Project Manager Decision Support
Last but not least, the prediction results and explanations using SHAP values are communicated to the project managers. This allows for informed decision making in project planning, scheduling, resource allocation and risk management.

IV. RESULTS AND DISCUSSION

In the following chapter, the results of experiments conducted with the proposed AI-Driven Effort and Time Prediction framework for software project tasks are presented and discussed. The framework consists of five major steps: Data preprocessing, Feature engineering, Machine learning model development, Synthetic data generation, Performance evaluation and Explainability analysis.
The experimental study was carried out by the original CESAW datasets and the synthetic datasets generated from CESAW. Several machine learning models, such as Random Forest (RF), XGBoost (XGB), LightGBM (LGBM), Gradient Boosting Machine (GBM), and Stacking Ensemble models, were trained and evaluated to predict software task effort and completion time. These models were evaluated using typical regression measures including the Root Mean Square Error (RMSE), the Mean Absolute Error (MAE), and R2 Score.
The experimental findings clearly illustrate the usefulness of the proposed framework for accurately predicting the effort and time consumption in the completion of software tasks. LightGBM obtained the best prediction performance on the Task Dataset with an R^2 score of 0.9981 while Gradient Boosting Machine (GBM) outperformed the other models on the Time Dataset with an R^2 score of 0.9998. A meaningful model validation and robustness analysis was possible due to the successful preservation of the statistical properties of the synthetic datasets generated.
Additionally, the results showed that the models developed using boosting methods were superior in terms of prediction accuracy compared to the conventional machine learning methods. The use of SHAP-based explainability analysis helped gain insights into feature importance and model decision-making processes, fostering transparency and interpretability of the proposed framework.
Overall, the results obtained confirm the reliability, robustness, and applicability of the proposed AI-based framework for software effort and time estimation that could contribute to the success of software project planning, scheduling, and resource allocation in software development environments.

RESULTS

Dataset Preprocessing Results

Initial Dataset

The original CESAW Task Dataset contained **61,817 records** and **12 features** used for task effort prediction.

Fig 5.1.1 Dataset Preprocessing Results

After Outlier Removal

After removing outliers and performing initial feature engineering, the dataset was reduced to **59,695 records**, while the number of features increased to **19**.

Fig 5.1.1 After Outlier Removal

Training and Testing Datasets

Additional feature extraction and preprocessing increased the feature count to **34**. The dataset was then split into **80% training data (47,756 records)** and **20% testing data (11,939 records)** for model development and evaluation.

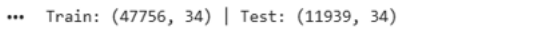

Fig 5.1.1 Training and Testing Datasets

5.1.2 Task Effort Prediction Results

The performance of the tuned machine learning models on the Task Dataset is summarized below.

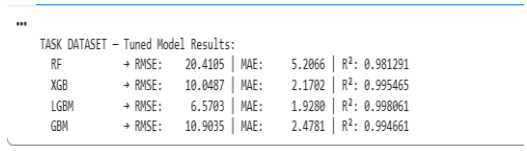

Fig. 5.1.2 Task Effort result

The results show that **LGBM achieved the best performance**, with the lowest RMSE (6.5703) and MAE (1.9280), and the highest R^2 score (0.9981). This indicates that LGBM provided the most accurate task effort predictions among the evaluated models.

5.2 DISCUSSION

The outcomes of the experiments show that the proposed AI-Driven Effort and Time Prediction framework is very efficient in predicting software development effort and time. Pre-processing techniques such as date feature extraction, feature engineering, and outlier removal were crucial for ensuring the data's quality and improving the model's performance. More features were created to better represent the complex relationships in the data set.

In the evaluated models, boosting-based models (LightGBM, XGBoost, Gradient Boosting Machine) outperformed Random Forest. LightGBM showed the best performance in "task effort" prediction and Gradient Boosting Machine was the best in "time" prediction. The models showed good R^2 values on the original data sets, suggesting that they captured the underlying patterns and relationships present in the project data.

The synthetic data created with the Gaussian Copula Synthesizer maintained the structure and statistical attributes of the original data. The predictive accuracy on synthetic data was slightly below what was obtained from the original data sets, however, important relationships were still observed, and predictions made with reasonable accuracy. This illustrates the value of synthetic data for model validation, experimentation, and when real project data is not available.

SHAP-based explainability analysis was also used to improve transparency of the proposed framework, highlighting the most important features that influence effort and time predictions. This offers important insights to project managers and software teams, so they understand what affects the results of their project estimates.

In general, the experimental results confirm the efficiency of the suggested framework for obtaining accurate, interpretable and scalable software project effort and time's prediction. Together, these sophisticated machine learning models, synthetic data creation, and explainable AI approaches create a comprehensive solution to assist project planning and decision-making processes.

V. CONCLUSION AND FUTURE SCOPE

Conclusion

This project introduced an AI-Driven Effort and Time Prediction Framework for software project estimation based on the CESAW dataset and synthetic data generation techniques. To enhance the accuracy and interpretability of predictions, the proposed framework integrated data preprocessing, feature engineering, training of machine learning models, generation of synthetic data, and the incorporation of explainable AI techniques.

For task effort and time prediction, multiple machine learning models such as Random Forest (RF), XGBoost (XGB), LightGBM (LGBM), and Gradient Boosting Machine (GBM) were analyzed. The results of the experiments confirmed that boost-based models outperformed others, and LightGBM outperformed other boost-based models for the task effort prediction, while GBM outperformed the other boost-based models for time prediction. The synthetic datasets generated were able to capture the attributes of the original CESAW datasets and further validate and test the model.

Additionally, explainability analysis using the SHAP method gave the information on what factors affect the prediction results of the proposed framework, increasing the

transparency and reliability of the proposed framework. The study overall shows that machine learning and explainable AI can be used effectively to assist software project effort estimation and time estimation, which can significantly help organizations to improve their software project planning, scheduling, and decision-making processes.

Future Scope

- The proposed framework can be expanded to further improve prediction accuracy, scalability, and real-world applicability in several ways.
- Use of deep learning models, including CNNs, LSTMs, and Transformer models, to extract intricate project patterns.
- Increased access to bigger and broader software project data sets from various organizations and development contexts.
- Owners can use real-time prediction systems which will continuously update the estimates throughout the project.
- Investigation of other methods of synthetic data creation such as GANs and diffusion-based models to enhance synthetic data quality.
- Inclusion of project risk factors, team dynamics, developer experience, and productivity as extra predictive features.
- Implementation of the framework as an online or cloud-based project management and software development decision support tool.
- Improvement of explainability mechanisms to increase explainability with high-level XAI techniques to generate detailed information about prediction results.
- Obviously, the extensions of the framework to accommodate tasks related to cost estimation, defect prediction, project scheduling, and resource optimization.
- Further improvements to the proposed framework can further reinforce the framework's applicability, and also help improve the intelligence and data-driven nature of software project management practices.

REFERENCES

[1] A. O. Sousa, J. P. Leal, P. S. Lima, and F. M. Coutinho, “Applying Machine Learning to Estimate the Effort and Duration of Individual Tasks in Software Projects,” IEEE Access, vol. 11, pp. 1–15, 2023.

[2] M. Rahman, P. P. Roy, M. Ali, T. Gonçalves, and H. Sarwar, “Software Effort Estimation Using Machine Learning Technique,” International Journal of Advanced Computer Science and Applications, vol. 14, no. 4, pp. 91–100, 2023.

[3] M. Shepperd and S. MacDonell, “Evaluating Prediction Systems in Software Project Estimation,” Information and Software Technology, vol. 54, no. 8, pp. 820–827, 2012.

[4] E. Mendes, I. Watson, C. Triggs, N. Mosley, and S. Counsell, “A Comparative Study of Cost Estimation Models for Web Hypermedia Applications,” Empirical Software Engineering, vol. 8, no. 2, pp. 163–196, 2003.

[5] T. Menzies, Z. Chen, J. Hihn, and K. Lum, “Selecting Best Practices for Effort Estimation,” IEEE Transactions on Software Engineering, vol. 32, no. 11, pp. 883–895, 2006.

[6] L. Breiman, “Random Forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[7] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, 2016.

[8] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Y. Liu, “LightGBM: A Highly Efficient Gradient Boosting Decision Tree,” Advances in Neural Information Processing Systems (NeurIPS), vol. 30, pp. 3146–3154, 2017.

[9] S. M. Lundberg and S. I. Lee, “A Unified Approach to Interpreting Model Predictions,” Advances in Neural Information Processing

Systems (NeurIPS), vol. 30, pp. 4765–4774, 2017.

[10] N. Patki, R. Wedge, and K. Veeramachaneni, "The Synthetic Data Vault," IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 399–410, 2016.