The Transactions of the Korea Information Processing Society
TKIPS Vol.13, No.5, pp.236~242

eISSN: 3022-7011
https://doi.org/10.3745/TKIPS.2024.13.5.236

Software Defect Prediction Based on SAINT

Sriman Mohapatra[†] · Eunjeong Ju[††] · Jeonghwa Lee[†††] · Duksan Ryu[††††]

ABSTRACT

Software Defect Prediction (SDP) enhances the efficiency of software development by proactively identifying modules likely to contain errors. A major challenge in SDP is improving prediction performance. Recent research has applied deep learning techniques to the field of SDP, with the SAINT model particularly gaining attention for its outstanding performance in analyzing structured data. This study compares the SAINT model with other leading models (XGBoost, Random Forest, CatBoost) and investigates the latest deep learning techniques applicable to SDP. SAINT consistently demonstrated superior performance, proving effective in improving defect prediction accuracy. These findings highlight the potential of the SAINT model to advance defect prediction methodologies in practical software development scenarios, and were achieved through a rigorous methodology including cross-validation, feature scaling, and comparative analysis.

Keywords : Transformer, SAINT, Software Defect Prediction

SAINT 기반의 소프트웨어 결함 예측

Sriman Mohapatra[†] · 주 은 정[††] · 이 정 화[†††] · 류 덕 산[††††]

요 약

소프트웨어 결함 예측(SDP)은 오류가 발생할 가능성이 있는 모듈을 사전에 식별하여 소프트웨어 개발의 효율을 높이고 있다. SDP에서의 주 과제는 예측 성능을 향상시키는것에 있다. 최근 연구에서는 딥러닝 기법이 소프트웨어 결함 예측(SDP) 분야에 적용되어 있으며, 특히 구조화된 데이터를 분석하는 데 뛰어난 성능을 보이고 있는 SAINT 모델이 주목받고 있다. 본 연구는 SAINT 모델을 다른 주요 모델(XGBoost, Random Forest, CatBoost)과 비교하여 SDP에 적용 가능한 최신 딥러닝 기법을 조사하였다. SAINT는 일관되게 우수한 성능을 보여주며 결함 예측 정확도 향상에 효과적임을 입증하였다. 이 연구 결과는 실용적인 소프트웨어 개발 상황에서 결함 예측 방법론을 발전시킬 수 있는 SAINT의 잠재력을 강조하며, 교차 검증, 특성 스케일링, 비교 분석 등을 포함한 철저한 방법론을 통해 수행되었다.

키워드 : 트랜스포머, SAINT, 소프트웨어 결함 예측

1. 서 론

Software Defect Prediction (SDP) plays a crucial role in the software development process by identifying modules that are likely to have errors and effectively allocating resources to enhance development efficiency. In this field, tabular datasets are primarily used, and various machine learning models are employed to predict software defects. The goal of SDP is to find highly effective defect prediction models to improve software quality. Recently, the Self-Attention and Intersample Attention Transformer (SAINT) model, equipped with self-attention and intersample attention mechanisms, has shown superior performance compared to traditional methods.

This research focuses on how the SAINT model effectively operates in software defect prediction. SAINT employs a hybrid attention model to learn the interactions among various components and utilizes this in integration with multi-task models and different modalities, such as images and text. These capabilities allow SAINT to be ver-

※ 본 연구는 원자력안전위원회의 재원으로 한국원자력안전재단의 지원을 받아 수행한 원자력안전연구사업(No. 2105030)과 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(NRF- 2022R1I1A3069233)과 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터 지원사업 (IITP-2024-2020-0-01795)의 연구결과로 수행되었음.
※ 이 논문은 2024년 KCSE 2024의 우수논문으로 "Software Defect Prediction Via SAINT"의 제목으로 발표된 논문을 확장한 것임.

† 비 회 원 : 전북대학교 소프트웨어공학과 석사과정
†† 정 회 원 : 전북대학교 소프트웨어공학과 학석사통합과정
††† 비 회 원 : 전북대학교 소프트웨어공학과 학석사통합과정
†††† 종신회원 : 전북대학교 소프트웨어공학과 부교수

Manuscript Received : March 28, 2024
Accepted : April 22, 2024

* Corresponding Author : Duksan Ryu(duksan.ryu@jbnu.ac.kr)

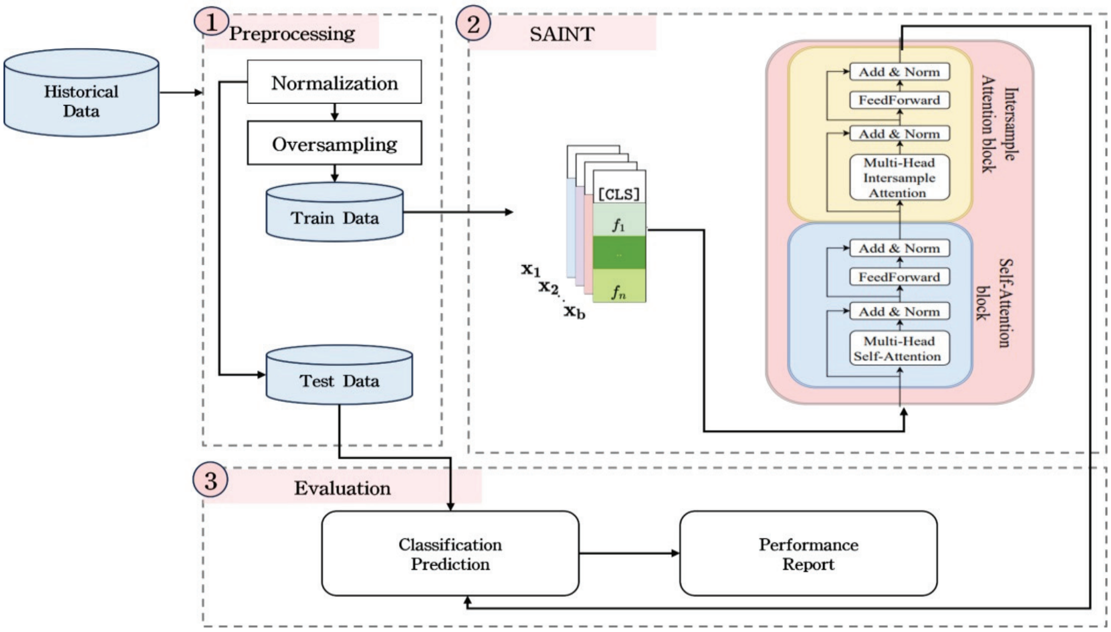

Fig. 1. Research Method

satile and perform well even in complex datasets. Particularly, SAINT introduces an innovative approach to processing tabular data, demonstrating excellent performance. However, it has not yet been applied in SDP, and this study aims to explore its application.

Through this research, we aim to empirically demonstrate how the SAINT-Transformer model is more effective than conventional SDP methods. The successful implementation of this model is expected to significantly improve the accuracy of defect prediction in software development, ultimately enhancing software quality and optimizing resource allocation.

2. Related Work

In recent years, Deep Learning (DL) has shown significant success across various industries, leading to its application in Software Defect Prediction (SDP). Pan et al. have utilized advanced Convolutional Neural Networks (CNNs) to enhance defect prediction capabilities. They specifically addressed certain code issues (PSC) using CNNs, advancing defect prediction. Concurrently, Lee et al. introduced a novel SDP model based on TabNet, taking a different approach from traditional models such as XGBoost (XGB) and Random Forest (RF). Evaluations comparing these models, including CNNs, highlighted the superior performance of TabNet. Inspired by these developments, studies have been conducted to further enhance SDP models using the FT-Transformer technique.

Somepalli et al. introduced SAINT, an innovative neural network architecture tailored for tabular data analysis. It integrates row attention mechanisms and contrastive pretraining techniques to improve the handling of tabular datasets. Additionally, in another related study, Kim et al. proposed an SDP technique based on the FT-Transformer, which surpassed other performance models such as XgBoost and Catboost. Similar to these research efforts, we investigate whether the recent DL method, namely SAINT, can be applied to SDP.

Chen et al. introduced XGBoost, an efficient and scalable implementation of gradient boosting decision trees. XGBoost has become one of the most popular machine learning libraries for structured/tabular data due to its superior performance, speed, and flexibility. It extends the classic gradient boosting framework with several novel features, including a highly optimized algorithm, regularization techniques, and parallelization, making it particularly effective for a range of data mining tasks including classification, regression, and ranking.

All the above studies effectively utilize deep learning techniques in structured data, demonstrating impressive performance. SAINT has also shown significant effects in analyzing structured data in other domains. Based on this, we apply SAINT to SDP to verify if it can outperform existing SDP models.

3. Methodology

In this investigation, the application of the SelfAttention and Intersample Attention Transformer (SAINT) to Software Defect Prediction (SDP) involves a meticulous process. To address overfitting concerns, cross-validation is employed, effectively partitioning the dataset into training and testing subsets. The model undergoes training on the designated training data and is subsequently evaluated using separate test data. To augment the model's performance, feature scaling is incorporated, utilizing the MIN-MAX normalization technique. This normalization ensures that the features maintain a uniform range, typically between 0 and 1. By implementing this normalization specifically on the training data, the varying scales across features are standardized. This not only facilitates effective learning but also diminishes the impact of disparate scales on the model. In summary, our study meticulously applies SAINT to SDP, integrating rigorous cross-validation and MIN-MAX normalization for feature scaling. This comprehensive approach enhances the model's robustness and predictive effectiveness, particularly in the realm of software defect prediction.

This comprehensive approach enhances the model's robustness and predictive effectiveness, particularly in the realm of software defect prediction. In this research, the proposed method, SAINT, draws inspiration from the transformer encoder introduced by Vaswani et al., originally designed for natural language processing.

The transformer architecture takes a sequence of feature embeddings as input and outputs contextual representations of the same dimension. A visual representation of SAINT is illustrated in [Figure 1]. SAINT is structured as a stack of L identical stages, each comprising a self-attention transformer block and an intersample attention transformer block. Each stage's self-attention transformer block closely resembles the encoder from a prior study. This block encompasses a multi-head self-attention layer (MSA) with h heads, followed by two fully connected feed-forward (FF) layers featuring a GELU non-linearity.

A skip connection and layer normalization (LN) are included in each layer to enhance model stability. On the other hand, the intersample attention transformer block mirrors the self-attention transformer block, with the distinction that the self-attention layer is substituted with an intersample attention layer (MISA). Further details on the intersample attention layer are expounded upon in the subsequent subsection. The SAINT pipeline, with a single stage (L = 1) and a batch of b inputs, is characterized by a set of equations. Multi-head self-attention is denoted by MSA, multi-head intersample attention by MISA, feedforward layers by FF, and layer normalization by LN. This research introduces SAINT as a transformative paradigm, leveraging a stack of transformer-based stages that incorporate both self-attention and intersample attention mechanisms. The adoption of this architecture aims to overcome the limitations of conventional models, offering a more nuanced and effective approach for the specific challenges posed by the dataset under consideration.

$$Z_i^{(1)} = L_1 V\left(\left(MSA\left(E-\left(x_i\right)\right)\right) + E\left(x_i\right)\right. \tag{1}$$

$$z_i{}^{(2)} = FF_1 - \left(z_i{}^{(1)}\right) + z_i{}^{(1)} \tag{2}$$

$$z_i^{(3)} = LN\left(MISA\left(\left\{Z_i^{(2)}\right\}_{J=1}^{b}\right)\right) + z_i^{(2)} \tag{3}$$

$$r_i = LN\left(FF_2\left(z_i^{(3)}\right)\right) + Z_i^{(3)} \tag{4}$$

Algorithm 1 encapsulates the SAINT, expressed in code for processing features denoted as x within the SAINT module. Its transforms features into structured tokens and consistently tokenize them. Uniformly process the tokenized features (lines 1-3). It uses the tokenized features to transform data through the Transformer module (lines 4-6). It redefines the [CLS] token and top-level tokens to facilitate subsequent predictions.

Algorithm 1. SAINT

```
Algorithm1. SAINT
Input : Train Data X
Output : Data Y Predicted for fault
1: /*Preprocessing*/
2: X is Oversampled
3: MIN-MAX ← X
4: X is Normalization
5: Feature Tokenizer ← X
6: X is Tokenized
7: T = X
8: T1 = T[CLS]
9: /*[CLS]is top-level token containing the
      contents of all data*/
10: Transformer ← T1
11: T2 = Newly defined T1 in Transformer Model
```

Redefine the [CLS] token within the algorithm. It uses the [CLS] token to make predictions using the transformed data (lines 7-9).

The SAINT formulation in this paper succinctly describes the key processes of tokenization, data transformation, and predictive steps. The formulation constituting the SAINT used in this paper is as follows:

$$T_J = b_J + F_J(x_J) \in RdF_J : X_J \to R_d \quad (5)$$

$$T_0 = stack[[CLS]T] \;\; T_i = F_i(T_{i-1}) \quad (6)$$

4. Experimental Setups

4.1 Research Questions

RQ1: How does SAINT's defect prediction performance compare to other techniques?

H_{10}:The defect prediction performance of SAINT is similar to that of other models.

H_{1A} The defect prediction performance of SAINT is superior to other models.

RQ2 : Does the hyperparameters of SAINT have an impact on defect prediction performance?

H_{20} The hyperparameters of SAINT do not affect defect prediction performance.

H_{2A} The hyperparameters of SAINT affect defect prediction performance.

performance compared to alternative techniques, quantifying its superiority in defect prediction. Through rigorous experimentation and comparative analysis, this study provides insights into SAINT's effectiveness as a defect prediction model, contributing valuable information to the field of software reliability.

4.2 Dataset

To evaluate the performance of the model, opensource data sets (AEEEM, ReLink,PROMISE and AUDI). Detailed information of each data is shown in Table 1

Table 1. Dataset

Dataset	Project	Instances	Buggy(%)	No of metrics	Granularity
AEEEM	EQ	324	129(39.81%)	61	Class
	JDT	997	206(20.66%)	61	Class
	LC	691	64(9.26%)	61	Class
Relink	Apache	194	98(50.52%)	26	Class
	Zxing	399	110(29.5%)	26	Class
	Safe	56	22(39.22%)	26	Class
Promise	Ant	769	187(24.32%)	20	Class
	Xerces	454	43(9.47%)	20	Class
AUDI	Project A	1909	191(4.45%)	13	Class
	Project K	2516	374(15%)	13	Class
	Project L	2896	76(2.62%)	13	Class

Table 2. Confusion Matrix

		Predicted class	
		Defective	Clean
Actual class	Defective	TP(True Positive)	FN(False Negative)
	Clean	FP(False Positive)	TN(True Negative)

4.3 Data Pre-processing

This study employs SAINT, XGBoost (XGB), Random Forest (RF), and CatBoost for software defect prediction. To tackle imbalanced data, MIN-MAX scaling and 1:1 ratio learning is applied to all models. Moreover, Synthetic Minority Over-sampling Technique (SMOTE) is used specifically for XGB and RF to enhance minority class representation, boosting model robustness. Performance evaluation involves rigorous 10-fold cross-validation repeated three times for each technique, totaling 30 evaluations. Evaluation metrics such as PD,

PF, balance, and FIR are computed and averaged across folds, ensuring a comprehensive and statistically sound assessment of predictive capabilities. The study aims to compare the performance of SAINT, XGBoost (XGB), Random Forest (RF), and CatBoost, different gradient boosting algorithms, in software defect prediction. This comparative analysis provides insights into the effectiveness of various machine learning techniques in addressing the challenges of software defect prediction tasks.

4.4 Performance Matrix

In this study, evaluation metrics are derived from the confusion matrix. PD (Probability of Detection) measures the ratio of correctly identified defective instances to the total number of actual defective instances, while PF (Probability of False Detection) quantifies the ratio of non-defective instances misclassified as defective to the total number of non-defective instances. To address class imbalance, the study employs the Balance Metric,Balance=

$=1-\sqrt{\frac{(0-PF)^2+(1-PD)^2}{2}}$)Moreover, FIR (FIR(File Inspection Reduction $=\frac{PD-FI}{PD}$)was measured to analyze the effectiveness of reducing code inspection efforts. $=\frac{PD-FI}{PD}$. where FI is the ratio of the number of files to be examined to the entire file.

5. Experimental Setups

5.1 RQ1: How does SAINT's defect prediction performance compare to other techniques?

In our study, SAINT consistently outperformed XGBoost, Random Forest, and CatBoost (Table 3), demonstrating higher average Precision (PD) of 0.8375, lower False Positive Rate (PF) of 0.2067, and superior average balance of 0.83528. This indicates that SAINT achieves higher accuracy in identifying positive instances while minimizing false positive predictions, resulting in a better balance between precision and recall.

Furthermore, SAINT outperformed XGBoost (FIR: 0.4599), Random Forest (FIR: 0.2676), and CatBoost (FIR: 0.2468) in terms of False Instance Rate (FIR). The lower FIR values for SAINT suggest that it more effectively minimizes false predictions, leading to a higher overall accuracy in software defect prediction. These findings underscore SAINT's consistent effectiveness in software defect prediction, showcasing its superiority over other gradient boosting algorithms such as XGBoost, Random Forest, and CatBoost. The detailed analysis of performance metrics emphasizes SAINT's robustness and reliability in addressing the challenges of software defect prediction task.
In addition, it showed higher performance in ope source project data as shown in Table 4.

Table 5 compares the effect sizes between SAINT and the XGBoost, Random Forest (RF), and CatBoost techniques. SAINT exhibited a medium-sized difference in the PD metric compared to XGBoost. For RF, SAINT demonstrated a large-sized difference in the PD metric. For CatBoost, SAINT showed a small-sized difference in the PD metric.

This result demonstrates that SAINT's defect prediction performance is superior to traditional machine learning techniques, thereby proving advancements in defect prediction methodologies.

Table 3. Comparison of a Performance

	MODEL			
Metric	SAINT	XGB	RF	CAT BOOST
PD	0.8375	0.7999	0.7000	0.7750
PF	0.2087	0.4297	0.3702	0.3207
BALANCE	0.8352	0.6634	0.6618	0.6492
FIR	0.4806	0.4599	0.2676	0.2468

Table 4. Comparison of Performances Datasets

Dataset	Project	PD	PF	BALANCE	FIR
AEEEM	EQ	0.8375	0.2087	0.8352	0.4806
	JDT	0.8973	0.3062	0.8225	0.4814
	LC	0.7907	0.1896	0.8013	0.4238
Relink	Apache	0.7083	0.2869	0.8963	0.4814
	Zxing	0.6953	0.2747	0.8638	0.4008
	Safe	0.7072	0.3001	0.8153	0.4653
PROMISE	Ant	0.7008	0.1547	0.7894	0.4023
	Xerces	0.7845	0.2152	0.7584	0.3907
AUDI	ProjectA	0.7550	0.1976	0.8001	0.3990
	ProjectK	0.8013	0.1978	0.7983	0.3874
	ProjectL	0.7963	0.2376	0.8132	0.4354

Table 5. Comparison of Effect Size

SAINT	Model			
	PD	PF	BALANCE	FIR
XgBoost	0.6613(M)	-0.7434(L)	2.0652(L)	0.3217(S)
RF	2.0841(L)	-1.2115(L)	2.2714(L)	1.8023(L)
CatBoost	0.3781(S)	-0.9461(L)	2.0987(L)	2.2521(L)

5.2 RQ2: Does the hyperparameters of SAINT have an impact on defect prediction performance?

Table 6 illustrates the range and default value of the parameter “n_blocks” for the SAINT model. The parameter “n_blocks” in the SAINT model represents the number of Transformer blocks utilized. To assess performance variations based on the difference in the number of Transformer blocks used for tokenized data, the range and default values of the parameter “n_blocks” are specified accordingly

Table 7 presents the optimal parameter values for the SAINT model based on the best performance in terms of the PD metric across different projects.

By finding the optimal hyperparameter values, we were able to identify performance variations across projects

Table 6. Range and Default Values of Parameters

Parameter	Range	Default
n_blocks	{1,2,3}	3

Table 7. Parameters Performances Across Different Projects

Dataset	Project	Best Performing Parameter
		n_blocks
AEEEM	EQ	2
	JDT	3
	LC	2
Relink	Apache	3
	Zxing	1
Promise	Safe	2
	Ant	3
	Xerces	2
Audi	ProjectA	2
	ProjectK	3
	ProjectL	2

and hyperparameters in SAINT, and recognized the need to adjust parameters to fit specific values for the data. Through this process, we can derive the optimal values that allow SAINT to effectively perform in SDP. This approach has led to improved defect prediction performance in SAINT.

6. Threats to Validity

This study's limitations include the potential compositional threat stemming from examining only four performance metrics (PD, PF, BALANCE, and FIR). By focusing solely on these metrics, there is a risk of overlooking other important aspects of model performance, which could impact the overall interpretation and generalizability of the findings. Additionally, the study acknowledges threats to validity arising from the limited diversity of the dataset used and the narrow scope of comparison with only XGBoost and Random Forest models. Future research efforts should aim to address these limitations by exploring datasets with greater diversity in terms of software projects, domains, and defect types. Moreover, incorporating a broader range of baseline methods beyond just XGBoost and Random Forest would provide a more comprehensive understanding of SAINT's performance relative to other state-of-the-art approaches. By addressing thes e limitations, future studies can enhance the robustness and applicability of the findings, ultimately contributing to a more nuanced understanding of SAINT's effectiveness and its potential impact on software defect prediction tasks.

7. Conclusion

In conclusion, our study introduces the Self-Attention and Intersample Attention Transformer (SAINT) model for Software Defect Prediction (SDP) and provides a comprehensive evaluation of its performance. Through meticulous experimentation and comparative analysis, SAINT emerges as a robust and promising approach for enhancing defect prediction accuracy. Our findings reveal that SAINT consistently surpasses traditional methods such as XGBoost, Random Forest and CatBoost across various evaluation metrics. This consistent superiority underscores SAINT's efficacy in capturing complex patterns and dependencies within software defect data, thereby improving predictive capabilities significantly. Furthermore, SAINT offers valuable insights into the Sapplication of deep learning techniques in SDP, highlighting its potential to advance the field. Its ability to handle high-dimensional data and adapt to diverse software development scenarios underscores its versatility and practical utility. Overall, our research contributes to the growing body of knowledge in SDP by demonstrating the effectiveness of advanced machine learning approaches like SAINT. By paving the way for further exploration and refinement, we envision SAINT playing a pivotal role in real-world software development contexts, facilitating the creation of more reliable and resilient software systems.

References

[1] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol.35, No.8, pp.6679-6687, 2021.

[2] C. Pan, M. Lu, B. Xu, and H. Gao. "An improved CNN model for within-project software defect prediction," *Applied Sciences*, 2019.

[3] J. W. Lee, J. W. Choi, D. S. Ryu, and S. T. Kim, "TabNet based Software Defect Prediction," in *Proceedings of the Korean Information Science Society Annual Conference*, pp.1255-1257, 2021.

[4] G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, and T. Goldstein, "Saint: Improved neural networks for tabular data via row attention and contrastive pre-training," *arXiv preprint arXiv:2106.01342,* 2021.

[5] S. J. Kim, E. J. Ju, J. W. Choi, and D. S. Ryu. "Software defect prediction based on Ft-Transformer," In *Proceedings of the Korea Information Science Society Conference,* 2022.

[6] J. Chen, E. Jiménez-Ruiz, I. Horrocks, and C. Sutton. "Learning s mantic annotations for tabular data," *arXiv preprint arXiv:1906.00781*, 2019.

[7] A. V. Dorogush, V. Ershov, and A. Gulin. "CatBoost: gradient boosting with categorical features support," *arXiv preprint arXiv:1810.11363*, 2018.

[8] J. Ho, N. Kalchbrenner, D. Weissenborn, and T. Salimans. "Axial attention in multidimensional transformers," *arXiv preprint arXiv:1912.12180*, 2019.

[9] Y. LeCunet, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE,* 86.11: pp.2278-2324, 1998.

[10] T. Chen, C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.785-794, 2016.

[11] K. Clark, M. T. Luong, Q. V. Le, and C. D. Manning. "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[12] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the NIPS 2003 feature selection challenge," in *Advances in Neural Information Processing Systems, 17,* 2004.

[13] L. Katzir, G. Elidan, and R. EI-Yaniv, "Net-dnf: Effective deep modeling of tabular data," *International Conference on Learning Representations, 2020.*

[14] A. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv: 1807.03748*, 2018.

Sriman Mohapatra

https://orcid.org/0009-0006-2174-801X

e-mail : srimanmohapatra0@jbnu.ac.kr

Master's degree in software engineering at Chonbuk National University from 2023. His research interests include SE4AI, AI4SE, AI/LLM based software analysis, software defect prediction, software reliability, software metrics, software quality assurance.

Eunjeong Ju

https://orcid.org/0009-0006-4795-4620

e-mail : jeju3146@jbun.ac.kr

Bachelor's degree in software engineering at Chonbuk National University from 2021. Her research interests include SE4AI, AI4SE, AI/LLM based software analysis, software defect prediction, Includes software reliability.

Jeonghwa Lee

https://orcid.org/0009-0006-7140-8363

e-mail : dlwjdghk133@jbnu.ac.kr

Bachelor's degree in software engineering at Chonbuk National University from 2021. Her research interests include SFront-end Design & Verification Methodology, LLM4SE, and software defect prediction.

Duksan Ryu

https://orcid.org/0000-0002-9556-0873

e-mail : duksan.ryu@jbnu.ac.kr

Master's degree in Software Engineering, dual degree from KAIST and Carnegie Mellon University in 2012. PhD in Computer Science from KAIST in 2016. From September 2018 to now, he is an associate professor of software engineering at Chonbuk National University. His research interests include SE4AI, AI4SE, AI/LLM based software analysis, software defect prediction, software reliability, software metrics, software quality assurance.