# 소프트웨어 결함 데이터의 특징 부분 집합을 이용한 자기지도 학습

최정환[1], 류덕산[2]
[1]연세대학교, [2]전북대학교
jeongwhan.choi@yonsei.ac.kr, duksan.ryu@jbnu.ac.kr

# Self-Supervised Learning Using Feature Subsets of Software Defect Data

Jeongwhan Choi[1], Duksan Ryu[2]
[1]Yonsei University, [2]Jeonbuk National University

**요 약**

Software Defect Prediction (SDP)은 코드 리뷰와 테스팅을 위한 리소스를 효과적으로 할당하도록 도움을 준다. 자기지도 학습(self-supervised learning)은 데이터 속성을 더 잘 이해할 수 있는 멀티뷰 관점을 데이터 증강을 통해 제공할 수 있는 장점을 가진다. 본 연구의 목표는 멀티뷰 관점의 장점을 가진 Subsetting features of Tabular (SubTab) 기법이 SDP의 예측 성능 향상에 효과적인지 확인하는 것이다. Within-Project Defect Prediction(WPDP)에서 SDP 데이터의 subset들을 사용하여 멀티뷰 표현 학습으로 전환한다. 실험 결과 잠재 공간 상에서의 subset 들을 재구성하는 것이 기존 머신러닝 기반 베이스라인들 보다 더 좋은 성능을 보였다. 멀티뷰 표현 학습 태스크의 도움으로 소프트웨어 품질을 보장하는데 있어 클래스 불균형 문제를 해결하는 오버샘플링 기법의 사용 없이도 노력 비용을 감소할 수 있다. 따라서 자기지도 학습 기법이 효율적인 리소스 할당을 통해 소프트웨어 신뢰성을 향상시키는데 유용하다.

## 1. 서론

소프트웨어 품질을 보장하기 위해 소프트웨어 모듈 테스트에 많은 노력이 투자되었다. 하지만 제한된 리소스와 소프트웨어 시스템의 규모와 복잡성의 증가로 인해 점점 더 비용 효과적인 품질 제고 방안이 요구되고 있다. Software Defect Prediction (SDP)는 소프트웨어 데이터를 예측 모델에 적용하여 결함이 발생하기 쉬운 소프트웨어 모듈들을 효과적으로 식별하는 기법이다[1], [2]. 고성능의 결함 예측 기법은 품질 보증 관리자들이 코드 리뷰와 테스팅을 위한 리소스를 효과적으로 할당하는 것이 목표이다.

자기지도 학습(Self-supervised learning)은 많은 도메인에서 성공적으로 사용되어왔다. 이미지 처리 분야에서 데이터 증강 및 pretext task 생성을 통해 다운스트림 태스크에 도움이 되는 특징 표현을 포착하는데 있어서 큰 장점을 가져왔다[3]. 또한 데이터들의 관계를 통해 부족한 라벨을 생성할 수 있는 장점이 있다. 특히 데이터 증강 기법을 이용하여 데이터 속성을 잘 이해할 수 있는 멀티뷰 관점을 가질 수 있다[4], [5]. 하지만 테이블 데이터에 기반한 SDP 분야에는 자기지도 학습이 폭넓게 적용되지 못하였다[2]. 자기지도 학습에 사용되는 증강 방법들은 테이블 환경에 적합하지 않다. 따라서 자기지도 학습의 장점인 멀티뷰 관점이 SDP 에 효과적인지 확인이 필요하다.

본 연구에서는 subset 을 이용하는 자기지도 학습 기법인 Subsetting features of Tabular (SubTab) [6]을 Within-Project Defect Prediction (WPDP) 환경에서의 적용 가능성을 연구한다.

## 2. 관련 연구

최근 연구에서는 자기지도 학습을 적용하는 SDP 연구가 등장하고 있다[2], [7]-[9]. BUGLAB[9] 은 자기지도 학습을 이용하여 소스코드로부터 버그를 탐지하고 수정하는 연구를 제안한다. 또한 라벨이 없는 소스코드를 이해하기 위해 pre-trained 모델과 함께 자기지도 학습을 사용한다[10], [11]. 위의 연구들은 소프트웨어 척도 기반 SDP 연구가 아니라는 점에서 본 연구와 차이가 있다.

자기지도 학습은 텍스트와 이미지 데이터에 성공적으로 적용되었다. 주로 데이터 증강을 통해 우수한 분류 성능을 보였지만 테이블 데이터에서는 이러한 적용이 힘들기 때문에 연구가 활발하지 않다. 테이블 데이터를 사용하는 소프트웨어 척도 기반 SDP 연구에서 또한 자기지도 학습의 적용성을 보이는 연구 또한 활발하지 않다. 최근 SubTab[6]은 테이블 데이터 환경의 MNIST 데이터셋에서 State-of-the-art (SOTA) 성능을 보였다. 본 연구는 이 SubTab 을 기반하여 소프트웨어 척도 기반 SDP 모델을 제시한다.

[1]발표자, [2]교신 저자; 본 연구는 원자력안전위원회의 재원으로 한국원자력안전재단의 지원을 받아 수행한 원자력안전연구사업(No. 2105030)과 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원(NRF-2019R1G1A1005047)을 받아 수행된 연구사업의 결과임

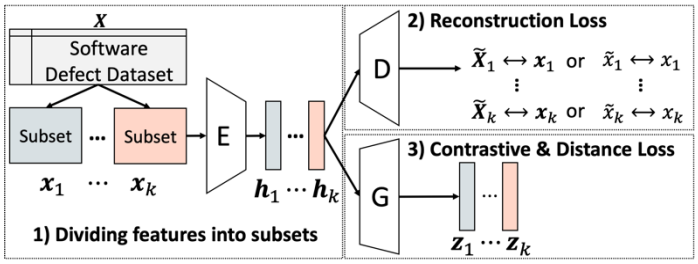


**그림 1**: SubTab을 이용한 SDP 프레임워크

## 3. 연구 방법

SubTab 을 이용한 SDP 프레임워크는 세 단계로 구성이 된다. 1) SDP 데이터셋 $\boldsymbol{X}$로부터 $k$개의 subset 으로 나눈다. 이웃하고 있는 subset 은 overlapping 영역을 가지며 해당 비율을 지정할 수 있다. 인코더 E 를 통해 $k$ 개 subset 들은 latent 벡터 $\boldsymbol{h}_k$로 표현이 된다. 2) 디코더 D 는 전체 테이블 $\tilde{\boldsymbol{X}}_k$ 혹은 subset $\tilde{\boldsymbol{x}}_k$ 로 재구성(reconstruct)하고 reconstruction loss 를 계산한다. 3) 프로젝션 네트워크 (Projection Network)인 G 는 프로젝션 $z_k$를 생성한다. 두 subset 의 같은 행의 샘플들인 $z_1$와 $z_2$ 는 positive 쌍으로, 나머지 행들은 negative 샘플로 고려할 수 있다. positive 와 negative 샘플을 사용하여 contrastive loss 를 계산할 수 있다. 또한 distance loss 를 사용하여 projection 된 subset 쌍 간의 distance 를 감소시키기 위해 mean squared error (MSE)를 사용한다. 테스트 시에는 테스트셋의 subset 을 인코더에 입력하고 모든 subset 에 대해 평균 값을 집계하여 joint 표현을 만든 후, 최종적으로 훈련된 logistic regression 모델을 통해 테스트 한다.

## 4. 실험 설정

### 4.1 연구 질문

WPDP에서 SubTab 의 효과를 확인하기 위해 통계적 검정을 이용하여 연구질문들을 설정한다.

***RQ1. SubTab 은 베이스라인 모델들보다 결함 예측 성능이 우수한가?***

- $H_0$: SubTab의 ROC-AUC 성능은 베이스라인과 유사하다.
- $H_A$: SubTab의 ROC-AUC 성능은 베이스라인보다 높다.

설정한 귀무가설과 대립가설은 위와 같으며. $H_0$ 을 기각할 수 있는지 확인하기 위해 ROC-AUC 성능을 기준으로 통계적 검정을 한다. 베이스라인 모델들은 logistic regression, random forest, XGBoost를 사용한다.

***RQ2. SubTab은 overlapping 비율과 subset 개수에 따른 성능 변화가 있는가?***

overlapping 비율과 subset 개수에 따른 성능 차이를 확인하기 위해 sensitivity 분석을 수행한다.

### 4.2 데이터셋과 평가척도

실험에서 사용하는 MORPH 데이터셋[12]은 표1과 같으며

**표 1**: 데이터셋 정보와 결함 비율을 나타낸다

| Dataset | ant | arc | camel | poi | redktor |
|---|---|---|---|---|---|
| Defect(%) | 16 | 11.54 | 3.83 | 59.49 | 15.34 |
| Total | 125 | 234 | 339 | 237 | 176 |
| **Dataset** | **skarbonka** | **tomcat** | **velocity** | **xalan** | **xerces** |
| Defect(%) | 20 | 8.97 | 75 | 16.13 | 15.21 |
| Total | 45 | 858 | 196 | 440 | 723 |

**표 2**: SubTab과 베이스라인에 대한 Cohen's D 효과크기 결과

| Effect size | ROC-AUC | FIR |
|---|---|---|
| Cohen's D | 1.14 | 0.82 |

불균형한 클래스를 포함하고 있다. 테스트셋 비율은 30%이며 10번의 실험을 진행한다. 불균형한 클래스 평가에 적합한 ROC-AUC를 평가척도로 사용한다[13]. File Inspection Reduction (FIR)을 사용하여 소프트웨어 인스펙션 감소 비율을 분석한다. 예측 모델을 사용하여 인스펙션 할 때, 동일한 Probability of Detection (PD)를 달성하기 위해 줄어든 인스펙션할 파일 수의 비율을 의미한다[14].

### 4.3 전처리

베이스라인 모델 모두 SubTab의 실험 환경과 동일한 min-max 정규화를 사용한다. 소프트웨어 결함 데이터셋은 불균형한 클래스를 가지고 있기에 오버샘플링 중 가장 인기 있는 Synthetic Minority Over-sampling Technique (SMOTE)[15]을 포함한다. 오버샘플링된 샘플들을 mutual information 특징 선택 기법을 이용하여 가장 유용한 특징 5개를 선택하여 사용한다.

## 5. 실험 결과

### 5.1 *RQ1: SubTab 의 효과*

RQ1 에 대한 효과크기는 각 평가척도별로 표 2 와 같이 모두 0.8 이상의 값을 가지며 이를 통해 "효과크기가 크다"라고 해석할 수 있기에 대립가설 $H_A$ 를 수용할 수 있다. 표 3 에서 모든 데이터셋에 대한 SubTab 에 대한 실험 결과를 확인할 수 있고. SubTab 이 모든 프로젝트들에 대해서 베이스라인들보다 우수한 성능을 보인다. 특히 defect 비율이 극도로 낮은 camel 프로젝트에서 효과적인 성능 향상을 보인다. FIR 결과를 통해서도 인스펙션 노력 비용 감소 측면에서 효율적인 리소스 할당에 도움을 줄 수 있다.

### 5.2 *RQ2: Sensitivity 분석*

그림 2 를 통해 SubTab 은 subset 의 개수와 overlapping 비율에 따른 성능 변화가 있는 것을 확인할 수 있다. 불균형 문제를 가진 arc 데이터셋은 k 의 개수 7 로 많고 overlapping 비율이 0.25 로 적을 때 가장 최적의 성능을 보인다. 반면에 클래스가 균형을 갖춘 데이터셋은 overlapping 비율이 높고 k 가 5 일 때 가장 좋은 성능을 보인며 이를 통해 데이터별로 최적의 k 와 overlapping 비율이 있는 것을 확인하였다.

**표 3**: 모든 모델에 대한 ROC-AUC을 나타낸다. *는 SMOTE와 mutual information 특징 선택 기법을 함께 진행한 결과이다

| Models | ant | arc | camel | poi | redktor | skarbonka | tomcat | velocity | xalan | xerces |
|---|---|---|---|---|---|---|---|---|---|---|
| Logistic Regression | 0.7917 | 0.8988 | 0.6454 | 0.7187 | 0.6186 | 0.5833 | 0.7824 | 0.7166 | 0.7737 | 0.5782 |
| Logistic Regression* | 0.8281 | 0.9107 | 0.7551 | 0.5723 | 0.5442 | 0.6250 | 0.7695 | 0.7006 | 0.7910 | 0.5734 |
| Random Forest | 0.7708 | 0.8889 | 0.6913 | 0.7651 | 0.6814 | 0.4583 | 0.7910 | 0.7980 | 0.8010 | 0.7741 |
| Random Forest* | 0.7682 | 0.9226 | 0.6199 | 0.7734 | 0.7081 | 0.6250 | 0.7960 | 0.7834 | 0.7944 | 0.8062 |
| XGBoost | 0.7656 | 0.8175 | 0.5842 | 0.7211 | 0.5628 | 0.3333 | 0.7793 | 0.8619 | 0.5968 | 0.5214 |
| XGBoost* | 0.7839 | 0.7688 | 0.5612 | 0.7313 | 0.6721 | 0.5833 | 0.7633 | 0.8154 | 0.7698 | 0.7432 |
| **SubTab** | **0.8698** | **0.9643** | **0.8495** | **0.8348** | **0.7280** | **0.7500** | **0.8017** | **0.8881** | **0.8039** | **0.8390** |

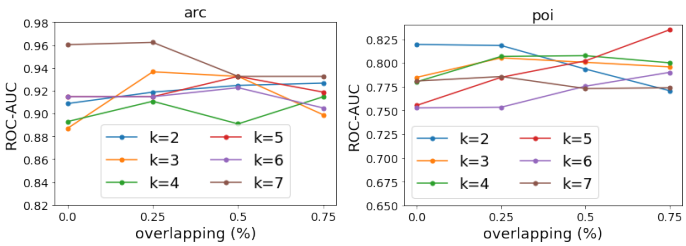


**그림 2**: (좌) arc 데이터셋과 (우) poi 데이터셋에 대한 subset 개수 k 와 overlapping 비율에 대한 sensitivity 분석

## 6. 위협 요소

불균형한 클래스는 구성 유효에 대한 위협이기에 불균형한 클래스에 적합한 ROC-AUC를 평가척도로 사용한다. 세가지 베이스라인을 사용한 것은 내부 유효성에 대한 위협이기에 향후 다른 모델 실험을 추가할 계획이다.

## 7. 결론

자기지도 학습은 멀티뷰 관점을 통해 데이터 속성을 잘 이해할 수 있는 장점이 있다. 본 연구는 이러한 장점이 SDP 에 적용 가능한지 확인하였다. SDP 데이터셋을 여러 subset 으로 나누어 이용하여 결함 예측을 멀티뷰 표현 학습 태스크로 변경할 수 있다. SubTab 을 이용하여 여러 subset 으로부터 데이터를 다시 재구성하는 것이 특징 선택과 결함 데이터의 만연한 문제인 클래스 불균형을 해결하는 SMOTE 를 이용한 베이스라인 모델 보다 우수한 성능을 보였다. 추후 SubTab 에 기반한 하이퍼파라미터 최적화 프레임워크를 연구할 계획이다.

## 참고 문헌


[1] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, and A. Bener, “Defect prediction from static code features: Current results, limitations, new approaches,” *Automated Software Engineering*, vol. 17, no. 4, pp. 375–407, 2010.

[2] C. Watson, N. Cooper, D. N. Palacio, K. Moran, and D. Poshyvanyk, “A Systematic Literature Review on the Use of Deep Learning in Software Engineering Research,” *arXiv Preprint*, Sep. 2020.

[3] L. Jing and Y. Tian, “Self-Supervised Visual Feature Learning With Deep Neural Networks: A Survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 11, pp. 4037–4058, 2020.

[4] L. Ericsson, H. Gouk, C. C. Loy, and T. M. Hospedales, “Self-Supervised Representation Learning: Introduction, Advances and Challenges,” *arXiv Preprint*, 2021, pp. 1–19.

[5] Y.-H. H. Tsai, Y. Wu, R. Salakhutdinov, and L.-P. Morency, “Self-supervised Learning from a Multi-view Perspective,” in *ICLR*, 2021, pp. 1–18.

[6] T. Ucar, E. Hajiramezanali, and L. Edwards, “SubTab: Subsetting Features of Tabular Data for Self-Supervised Representation Learning,” in *NeurIPS,* 2021.

[7] E. N. Akimova *et al.*, “A survey on software defect prediction using deep learning,” *Mathematics*, vol. 9, no. 11, pp. 1–14, 2021.

[8] S. Omri and C. Sinz, “Deep Learning for Software Defect Prediction: A Survey,” *Proc. - 2020 IEEE/ACM 42nd Int. Conf. Softw. Eng. Work. ICSEW 2020*, pp. 209–214, 2020.

[9] M. Allamanis, H. Jackson-Flux, and M. Brockschmidt, “Self-Supervised Bug Detection and Repair,” in *NeurIPS*, 2021.

[10] D. Guo et al., “GraphCodeBERT: Pre-training Code Representations with Data Flow,” in *ICLR*, 2021, pp. 57–74.

[11] R.-M. Karampatsis and C. Sutton, “SCELMo: Source Code Embeddings from Language Models,” *arXiv Preprint*, 2020, pp. 1–12.

[12] M. Jureczko and L. Madeyski, “Towards identifying software project clusters with regard to defect prediction,” in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering - PROMISE ’10*, 2010, p. 1.

[13] Z. Li, X.-Y. Jing, and X. Zhu, “Progress on approaches to software defect prediction,” *IET Softw.*, vol. 12, no. 3, pp. 161–175, 2018.

[14] Y. Shin, A. Meneely, L. Williams, and J. A. Osborne, “Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities,” *IEEE Trans. Softw. Eng.*, vol. 37, no. 6, pp. 772–787, 2011.

[15] X. L. Yang, D. Lo, X. Xia, Q. Huang, and J. L. Sun, “High-Impact Bug Report Identification with Imbalanced Learning Strategies,” *J. Comput. Sci. Technol.*, vol. 32, no. 1, pp. 181–198, 2017.