The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 20, No. 4, pp.107-113, Aug. 31, 2020. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2020.20.4.107
JIIBC 2020-4-15

트리 기법을 사용하는 세미감독형 결함 예측 모델

Semi-supervised Model for Fault Prediction using Tree Methods

홍의석*

Euyseok Hong*

요 약 매우 많은 소프트웨어 결함 예측에 관한 연구들이 수행되어왔지만 대부분은 라벨 데이터를 훈련 데이터로 사용하는 감독형 모델들이었다. 언라벨 데이터만을 사용하는 비감독형 모델이나 언라벨 데이터와 매우 적은 라벨 데이터 정보를 함께 사용하는 세미감독형 모델에 관한 연구는 극소수에 불과하다. 본 논문은 Self-training 기법에 트리 알고리즘들을 사용하여 새로운 세미감독형 모델들을 제작하였다. 세미감독형 기법인 Self-training 모델에 트리 기법들을 사용하는 새로운 세미감독형 모델들을 제작하였다. 모델 평가 실험 결과 새롭게 제작한 트리 모델들이 기존 모델들보다 더 나은 성능을 보였으며, 특히 CollectiveWoods는 타 모델들에 비해 압도적으로 우월한 성능을 보였다. 또한 매우 적은 라벨 데이터 보유 상황에서도 매우 안정적인 성능을 보였다.

Abstract A number of studies have been conducted on predicting software faults, but most of them have been supervised models using labeled data as training data. Very few studies have been conducted on unsupervised models using only unlabeled data or semi-supervised models using enough unlabeled data and few labeled data. In this paper, we produced new semi-supervised models using tree algorithms in the self-training technique. As a result of the model performance evaluation experiment, the newly created tree models performed better than the existing models, and CollectiveWoods, in particular, outperformed other models. In addition, it showed very stable performance even in the case with very few labeled data.

Key Words : Fault prediction, Semi-supervised learning, CollectiveWoods

Ⅰ. 서 론

지난 수십년간 매우 많은 연구가 이루어진 소프트웨어 결함 예측 분야는 기계학습 알고리즘들을 사용하면서 많은 성과를 이루었다. 하지만 훈련 데이터가 없는 경우에 사용할 수 있는 비감독형 모델이나 결함 결과를 아는 라벨 데이터가 매우 적은 경우에 사용가능한 세미감독형 모델에 대한 연구 성과는 미미한 실정이다.

그림 1은 라벨 데이터와 라벨이 없는 데이터 집합을 함께 사용하여 학습을 하는 세미감독형 모델을 나타낸다. 이는 라벨 데이터 집합이 훈련 데이터 집합으로 사용하기에 너무 작아 감독형 학습을 하지 못할 때 적합한 모델

*정회원, 성신여자대학교 컴퓨터공학과
접수일자 2020년 5월 18일, 수정완료 2020년 7월 18일
게재확정일자 2020년 8월 7일

Received: 18 May, 2020 / Revised: 18 July, 2020 /
Accepted: 7 August, 2020
*Corresponding Author: hes@sungshin.ac.kr
Dept. of Computer Engineering, Sungshin Women's University, Korea

로 보통 작은 라벨 데이터 집합과 상대적으로 큰 언라벨 데이터 집합을 훈련 데이터 집합으로 사용한다. 훈련을 마친 모델은 입력 언라벨 데이터를 받아 결함경향성을 결정하며 그림은 결함과 비결함으로 라벨링하는 이진 결함 예측 모델을 나타내었다.

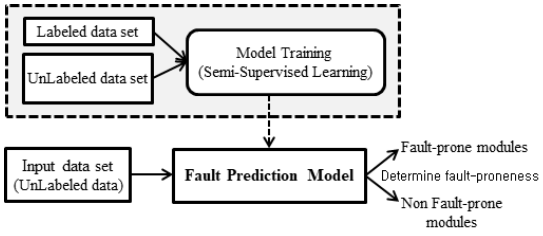

그림 1. 세미감독형 결함 예측 모델
Fig. 1. Semi-supervised fault prediction model

본 논문은 세미감독형 모델의 하나인 Self-training 모델 제작에 트리 기법들을 적용하여 새로운 모델들을 제작한다. 선행 연구인 [1]에서 여러 세미감독형 모델들을 제작하여 성능을 비교한 결과 YATSI 모델이 가장 뛰어난 성능을 보였다. 본 연구에서는 새롭게 제작한 트리 모델들을 YATSI 모델과 성능과 효용성 측면에서 비교 평가할 것이다.

II. 결함 예측 모델

수많은 감독형 결함 예측 모델들이 제안되었으며 가장 많이 사용된 대표적인 것들은 MLP, SVM, 베이지안 기법 등이었다[2]. 매우 극소수인 비감독형 모델들은 모두 클러스터링 기법들을 사용하였으며, 클러스터 수가 자동으로 결정되는가 여부 정도가 모델 간의 차이점이었다. 비감독형 모델은 전문가의 해석 및 참여가 필요하다.

세미감독형 모델들도 매우 극소수 발표 되었으며 Self-traing이나 Co-training 같은 세미감독형 알고리즘들을 사용하였다. 표 1은 세미감독형 연구들의 알고리즘 종류와 실험한 데이터 집합, 성능 측정 방법을 정리한 표이다. 7개의 연구들 중 3개의 모델이 Self-training 방법을 사용하였다. 대부분 연구들이 훈련 및 테스트 데이터 집합으로 NASA MDP 데이터 집합의 프로젝트들을 사용하였는데, 이 데이터들은 완전히 독립적으로 나눌 수 없으므로 Co-training을 적용하기 부적합하다. 또한 새로이 입력되는 데이터들의 분포를 알기는 어려우므로 Generative 알고리즘 역시 사용하기에 적합하지 않은 형태이다.

III. 모델 제작

1. Self-training 모델 구축

세미감독형 분류 기법에는 Generative 알고리즘, Self-training, Co-training, Transductive-SVM, 그래프기반 알고리즘이 있다[4]. 이들 대부분은 복잡한 제약조건들이 필요한데 결함 예측에 사용되는 데이터와 같이 결함 데이터가 비결함 데이터에 비해 상대적으로 매우 적은 데이터 집합에서는 이들 제약 조건들을 만족하기 어렵다. 따라서 단순한 반복 프로세스를 통해 라벨 데이터 정보를 데이터 집합에 점점 확산해 나가면서 완성 모델을 만들어 가는 Self-training 기법을 사용한다.

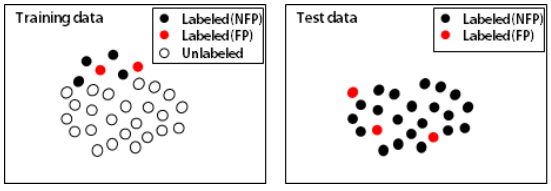

그림 2. 훈련 데이터와 테스트 데이터 구성
Fig. 2. Composition of training data and test data

본 논문에서 사용할 NASA 데이터 집합은 모든 모듈

표 1. 기존 세미 감독형 모델 연구들
Table 1. Previous studies about semi-supervised models

	분류	데이터 집합	성능 평가 척도
[3]	Self-training	PC1, PC3, PC4, KC1 (NASA)	AUC, PD
[4]	Self-training	JM1, KC2, CM1, PC1(NASA PROMISE)	AUC
[5]	Self-training	JM1, KC1, PC3, PC4, PC1 (NASA)	AUC, PD(recall or specificity)
[6]	Co-training	JM1, KC1, KC2, MW1, PC1, PC3, PC4, PC5 (NASA)	F1-measure, AUC
[7]	Co-training	Eclipse3.0, Eclipse2.0, JDT.Core, SWT, Lucene, Xalan	Precision, Recall, F-measure, Balance
[8]	Generative 알고리즘	JM1-8850 (JM1 정제, NASA 프로젝트))	Type I/II error, 예측실패율

이 라벨 데이터이므로 일부를 언라벨 데이터로 가정하고 모델을 제작하였다. 이들은 추후에 테스트 데이터로 사용된다. 그림 2는 30개의 모듈을 갖는 데이터 집합에서의 훈련 데이터와 테스트 데이터를 나타낸다. 훈련 데이터의 라벨 데이터는 전체의 20%인 6개이고, 나머지는 라벨 없는 데이터로 24개가 존재한다. 테스트 데이터는 훈련 데이터에서 라벨 없는 데이터로 취급한 24개의 데이터를 실제 출력 값을 붙인 라벨 데이터로 사용한다.

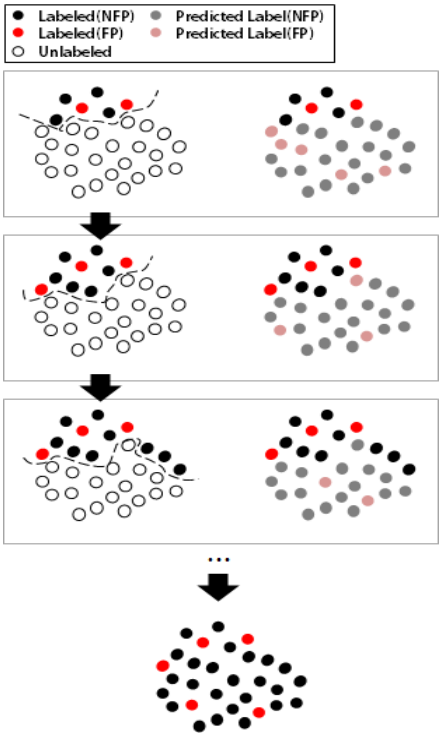

그림 3. Self-training 과정
Fig. 3. Self-training process

Self-training 기법은 라벨 데이터로 모델을 훈련시켜 새로운 라벨 데이터를 만들어 내는 기법이다. 초기 훈련 데이터 집합은 출력값을 아는 라벨 데이터와 언라벨 데이터로 구성된다. 라벨 데이터의 수가 많으면 더 나은 결과를 얻겠지만 대부분의 경우는 라벨 데이터가 언라벨 데이터에 비해 매우 적다. Self-training 기법은 학습 알고리즘을 선정하고 이를 이용하여 훈련 데이터 집합의 라벨 데이터를 학습하여 초기 모델을 만든다. 그 후 이 모델로 언라벨 데이터를 모두 라벨링 한다. 언라벨 데이터의 라벨링 결과를 보고 가장 예측이 잘된 데이터들을 선정하여 다음 단계의 훈련 데이터 집합에 첨가한다. 이 반복 프로세스를 모든 훈련 데이터가 모두 라벨 데이터가 되어서 더 이상 변화가 없을 때까지 계속한다.

그림 3은 Self-training 모델의 훈련 과정을 그림으로 나타낸 것이다. 사각형은 한 번의 훈련 단계를 나타내며 사각형 왼쪽의 라벨 데이터(색이 있는 점으로 나타냄)로 훈련을 하고, 언라벨 데이터(색이 없는 점으로 나타냄)를 라벨링하여 오른쪽처럼 나타나게 된다(옅은 색은 예측한 라벨링 결과를 나타냄).

2. 사용 알고리즘

기계학습에서 Collective 분류란 네트워크에 존재하는 알려지지 않은(unknown) 노드들을 알려진(known) 노드들 정보와 네트워크 구조로 분류하는 기법을 의미한다. 네트워크 구조 정보를 사용한다는 것은 각 노드들의 연관성 정보를 이용한다는 의미이다. 연관성은 분류하고자하는 노드의 라벨과 속성들 간이나 해당 노드의 라벨과 다른 노드들(라벨 노드들 또는 언라벨 노드들)의 속성들 간의 연관성을 의미한다[9]. 알려지지 않은 노드를 언라벨 데이터, 알려진 노드를 라벨 데이터라 하면 라벨 데이터를 기반으로 하여 언라벨 데이터를 분류한다는 점은 세미 감독형 기법임을 의미한다. 따라서 연관성 사용 유무에서 전통적인 분류 기법들과 차이가 있지만 세미 감독형 모델 구현은 WEKA의 Collective API를 사용하여 제작하였다[10].

표 2는 WEKA의 Collective 분류 기법들 중 Self-training 기법을 사용한 알고리즘만을 나타낸 것이다. 이들 중 meta 그룹에 속하는 모델들은 [1]에서 기존 세미감독형 연구 결과들과 성능 비교를 하였으며 그 결과 YATSI가 가장 좋은 성능을 보였다. 이 알고리즘들은 초기 훈련 후에 훈련 데이터에 추가 및 제거할 데이터 선정 방법이나 가중치를 주는 방법 등이 각기 다르지만 모두 Self-training 방법을 따른다. 또한 Self-training으로 모델을 구축하더라도 필터를 사용하여 예측 성능을 높이거나 다른 분류기와 비교하기 위해 구현된 알고리즘은 선정하지 않았다.

표 2. Self-training을 사용하는 세미 감독형 모델들
Table 2. Semi-supervised models using Self-training

분류 그룹	알고리즘
meta	Chopper CollectiveEM SimpleCollective YATSI Weighting
trees	CollectiveForest CollectiveWoods

트리 그룹의 Collective 모델에는 CollectiveTree 모델도 있지만 Self-training 조건 때문에 제외하였다. 트리 그룹 모델들은 공통적으로 각각의 분류기 가지고 테스트 집합을 같은 수의 객체들을 갖는 폴드(folds)로 나눈다. 첫 번째 반복에서 본래의 훈련 데이터로 학습을 하고 테스트 집합의 모든 객체들의 라벨이 할당된다. 그 중 가장 잘 라벨링 된 폴드가 훈련 데이터에 추가되고 다시 모델은 새로운 데이터가 추가된 훈련 데이터로 학습해 나간다. CollectiveForest의 경우 여러 의사결정 트리들을 만들고 가장 나은 모델을 선택하는 앙상블 모델인 Random Forest를, CollectiveWoods의 경우 CollectiveForest 자체를 기저 분류기로 사용한다. 트리 그룹의 모델들은 세미 감독형 기법은 같고 기저 분류기만 다른 것이라고 할 수 있다. 결국 CollectiveWoods는 CollectiveForest의 업그레이드 버전이라고 할 수 있다.

Ⅳ. 실험 및 결과

1. 데이터 집합 및 평가 척도

여러 프로젝트 데이터들로 구성된 NASA MDP 공개 데이터 집합들 중 내부 데이터간의 불일치성이 가장 적어 모호성 부분에서 가장 좋은 평가를 받는 데이터 집합은 JM1, PC4이다[11]. 하지만 NASA 데이터 집합을 사용한 수많은 결함 예측 연구들이 가장 많이 사용한 프로젝트는 PC1과 CM1이다[2]. 이는 해당 연구들이 프로젝트 데이터에 대한 모호성 분석을 안 한 부정적인 이유도 있지만 프로젝트 크기의 적정성(344 모듈, 759 모듈), 결함 비율의 적정성(12%, 8%), 데이터 정제 작업의 규모(데이터의 문제로 얼마나 많은 정제 작업이 드는지 여부), 프로젝트의 특성(상용화된 프로그래밍 언어 사용, 상용화된 분야의 응용 프로그램 여부) 등을 살펴보고 결정한 긍정적인 이유도 있다. 고려사항들 중 결함 모듈이 극소수라 결함 비율이 5% 이내로 매우 작아지면 데이터 불균형 문제가 심각해져서 샘플링 기법 등을 사용해야하는 부담이 발생한다. 따라서 본 논문은 PC1과 CM1의 PROMISE 레포지토리 버전 중 [12]의 DS' 버전을 사용하였다.

NASA 데이터 집합에서 각 모듈의 정량화에 사용된 소프트웨어 메트릭들은 Halstead의 Software Science, McCabe의 Cyclomatic Complexity, LOC 같은 SIZE 형태의 복잡도 메트릭들이다. 차원축소 작업은 예측 모델 연구들에서 가장 많이 사용하는 CfsSubsetEval을 사용하였으며, 표 3은 두 프로젝트 데이터의 차원축소로 선정된 속성들을 나타낸 것이다.

표 3. CFS의 차원축소 결과
Table 3. Dimensionality reduction results with CFS

데이터 집합	선정 속성들
CM1	LOC_COMMENTS/CYCLOMATIC_DENSITY/LOC_EXECUTABLE/HALSTEAD_CONTENT/NUM_UNIQUE_OPERANDS/NUM_UNIQUE_OPERATORS/PERCENT_COMMENTS/LOC_TOTAL
PC1	LOC_BLANK/LOC_CODE_AND_COMMENT/LOC_COMMENTS/CYCLOMATIC_DENSITY/LOC_EXECUTABLE/PARAMETER_COUNT/HALSTEAD_CONTENT/NODE_COUNT/NORMALIZED_CYLOMATIC_COMPLEXITY/NUM_UNIQUE_OPERANDS

올바르게 예측한 객체의 수에 대한 비율인 정확도(Accuracy, ACC)는 모델의 전체적인 성능 평가에는 적당하지만 결함 모듈 수가 비결함 모듈 수보다 매우 적은 결함예측모델의 평가에는 부족한 정보를 제공한다. 왜냐하면 결함 모듈을 정확히 예측하는 것이 비결함 모듈을 예측하는 것보다 더 중요하기 때문이다. 따라서 이를 보완하는 AUC(Area Under ROC curve)를 함께 사용한다.

AUC는 최근 예측 모델 평가에 많이 사용되고 있는 척도로 True Positive(TP)와 True Negative(TN)를 동시에 나타내는 ROC curve 면적을 측정한 것이다. ROC 곡선을 하나의 값으로 정량화 한 AUC는 0과 1사이의 값이며 값이 1에 가까울수록 TPR은 높고 FPR은 낮은 좋은 분류기라 할 수 있다. 적용 분야에 따라 차이가 있지만 일반적인 분류기는 AUC 값에 따라 다음과 같이 성능을 평가할 수 있다: AUC≤0.5: 아주 안좋은; 0.5〈AUC≤0.7: 덜 정확한, 0.7〈AUC≤0.9: 정확한; 0.9〈AUC〈1.0: 매우 정확한; AUC=1[13]. [1]의 실험 결과에서 YATSI 모델을 제외한 모든 모델들의 AUC 결과는 0.6 미만이었으며, 모든 경우에서 라벨 데이터를 20%로 하고 CFS로 차원축소를 한 한 개 경우에만 YATSI의 AUC가 0.7 이상의 성능을 보였다.

2. 성능 평가 실험

평가는 [1]의 평가 실험에서 가장 좋은 성능을 보인 YATSI 모델과 트리 그룹 모델들과의 성능 평가 비교를 통하여 실시하였다. 비교를 위하여 실험 환경은 [1]과 같게 하였다. 훈련 데이터의 라벨 데이터 비율은 5%, 10%, 20%로 랜덤하게 선정하여 각각 10번씩 실험하여 평균값을 결과값으로 하였으며, 실험은 차원축소 유무에 따라 두가지 경우 각각 실험하였다. 실험 결과는 표 4, 표 5에

나타냈으며 두 결과표는 차원축소를 안한 경우, 즉 전체 속성을 사용한 경우와 차원축소를 한 경우의 결과를 나타낸다. 음영 부분은 해당 데이터 집합 사용 시 가장 좋은 결과 부분을 나타낸 것이다. 예를 들면, 표 4에서 라벨 데이터를 5%로 한 경우의 정확도는 CM1의 경우 CollectiveForest가 81.90으로, PC1의 경우는 CollectiveWoods가 90.87로 세 모델들 중 가장 좋은 결과를 보인다는 것이다.

라벨 데이터와 언라벨 데이터의 비율의 영향은 예상대로 라벨 데이터가 더 많아 훈련에 사용할 수 있는 정보가 많아지면 성능이 높아졌다. 5%와 10%의 차이보다 20%로 높아지면서 더 많은 차이를 보임을 알 수 있다. 음영 부분을 보면 차원축소에 관계없이 PC1에서는 모든 경우에 CollectiveWoods가 가장 좋은 성능을 보이며, CM1에서도 거의 모든 경우에 가장 좋은 성능을 보인다. 두 평가 척도들 중에서는 AUC가 다른 모델들보다 월등하게 높으며, 결과도 0.7보다 크며, 0.8을 넘는 경우도 있어 정확한 성능에 가까워졌음을 볼 수 있다. 특히 라벨 데이터 비율 상승에 따른 성능 향상이 다른 모델들보다 적음에도 불구하고 압도적인 성능을 보인다는 것은 상대적으로 5%인 경우에 더욱 압도적인 차이를 보인다는 의미이다. 이는 라벨 데이터 정보가 매우 적은 상황, 즉 거의 비감독형 모델을 사용해야할 상황에서도 매우 적은 라벨 데이터만 가지고도 CollectiveWoods는 훌륭한 성능을 낼 수 있다는 가능성을 보여준다.

차원축소 작업은 YATSI 모델에는 어느 정도 긍정적인 영향을 주지만 트리그룹에는 전혀 긍정적인 영향을 미치지 않는다. 오히려 두 모델들은 전체 속성을 사용한 경우 좀 더 나은 결과를 보이며 특히 CollectiveWoods는 더 나은 결과를 보였다. 이는 기계학습 분야에서 어떤 데이터 형태에서도 좋은 성능을 잃지 않는 모델이 필요하다는 관점에서 볼 때 CollectiveWoods는 이에 매우 만족스럽게 부합한다고 볼 수 있다. 실험 결과를 요약해 보면 CollectiveWoods는 Collective 분류를 사용한 [1]의 5개 모델들과 트리 그룹 2개 모델들 중에서 모든 면에서 가장 좋은 성능 결과를 보였으며, 차원축소에 영향을 받지 않고, 매우 적은 라벨 데이터를 사용한 경우에도 좋은 성능을 보이는 안정적인 모델이다.

3. 기저분류기 변화에 따른 실험

평가 실험에서 J48을 사용한 YATSI와 달리 트리그룹은 다른 기저 분류기를 사용하였으므로, 정확하게 모델들의 성능을 비교하기에는 문제점이 있다. 그러므로 기저 분류기가 달라 생기는 성능의 차이를 고려해보기 위해 YATSI의 기저 분류기를 Random Forest(RF)로 변경하

표 4. 전체 속성을 사용한 경우 실험 결과
Table 4. Experimental results when all attributes are used

알고리즘	프로젝트	5%		10%		20%	
		ACC	AUC	ACC	AUC	ACC	AUC
YATSI	CM1	81.44	0.56	80.94	0.62	85.78	0.62
	PC1	89.89	0.61	90.75	0.60	91.25	0.67
CollectiveForest	CM1	81.90	0.60	84.87	0.68	85.45	0.69
	PC1	89.18	0.57	90.19	0.74	90.89	0.79
CollectiveWoods	CM1	81.47	0.61	85.77	0.69	86.51	0.71
	PC1	90.87	0.76	91.10	0.75	91.38	0.79

표 5. 차원 축소를 사용한 경우 실험 결과
Table 5. Experimental results when the dimensionality reductions are used

알고리즘	프로젝트	5%		10%		20%	
		ACC	AUC	ACC	AUC	ACC	AUC
YATSI	CM1	82.39	0.60	81.29	0.64	85.96	0.61
	PC1	90.01	0.62	90.47	0.64	92.08	0.71
CollectiveForest	CM1	80.86	0.64	81.55	0.66	84.47	0.65
	PC1	88.82	0.75	89.55	0.73	90.30	0.79
CollectiveWoods	CM1	81.65	0.67	83.68	0.71	85.31	0.71
	PC1	90.54	0.79	90.81	0.75	90.92	0.81

표 6. 기저 분류기를 Random Forest로 바꾼 YATSI 모델 결과
Table 6. Experimental results when the Random Forest is used as a base classifier for YATSI

차원축소	알고리즘	프로젝트	5%		10%		20%	
			ACC	AUC	ACC	AUC	ACC	AUC
전체 속성	YATSI(RF)	CM1	80.37	0.58	82.55	0.61	86.55	0.61
		PC1	89.72	0.61	90.69	0.62	91.35	0.68
	YATSI(J48)	CM1	81.44	0.56	80.94	0.62	85.78	0.62
		PC1	89.89	0.61	90.75	0.60	91.25	0.67
속성 선정	YATSI(RF)	CM1	82.63	0.64	81.48	0.64	85.45	0.64
		PC1	90.43	0.64	91.13	0.65	91.53	0.73
	YATSI(J48)	CM1	82.39	0.60	81.29	0.64	85.96	0.61
		PC1	90.01	0.62	90.47	0.64	92.08	0.71

여 트리 그룹과 성능을 비교 평가하였다.

표 6은 YATSI의 기저 분류기를 RF와 J48로 설정한 결과 값을 비교한 것이다. RF를 사용한 결과, J48을 사용했을 때보다 조금 나은 결과를 보이지만 오히려 J48을 사용했을 때 더 좋은 보이는 경우도 있다. 같은 기저 분류기인 Random Forest를 사용한 CollectiveForest와 비교했을 때, CollectiveForest가 여전히 더 나은 성능을 보여며, 이는 기저분류기에 변화를 주어 비교하여도 여전히 CollectiveWoods가 가장 좋은 모델임을 의미한다.

V. 결 론

소프트웨어 결함 예측 분야는 수십 년간 굉장히 많은 연구들이 수행된 분야로 인공지능의 기계학습 알고리즘들이 많이 적용되면서 높은 성능을 가진 모델들이 많이 제안되었다. 하지만 구현이나 운용에 큰 어려움이 있는 비감독형 모델과 세미감독형 모델에 대한 연구 성과들은 극소수에 불과하다. 본 연구는 선행연구 결과로 제작된 모델들과 트리 기법을 사용하여 새롭게 제작한 모델들의 성능을 비교 평가하였다. 모델들은 Self-training 기법들을 사용한 세미감독형 모델들이며, 선행 연구에서 가장 좋은 결과를 보인 YATSI 모델과 새로운 트리 모델들인 CollectiveForest 및 CollectiveWoods의 성능을 비교하였다. 그 결과 트리 모델들 모두 선행 모델들보다 나은 성능을 보였으며, 특히 CollectiveWoods는 탁월한 성능을 보였다. CollectiveWoods는 차원축소를 한 결과 성능 향상이 없었으므로 차원축소가 필요치 않은 모델이며, 라벨 데이터가 매우 적은 5%인 경우에도 성능이 많이 떨어지지 않는 특성을 보였다. 이는 라벨 데이터를 구하기 매우 어려운 많은 개발 집단들이 매우 소수의 라벨 데이터만 보유한 상황에서도 예측 성능이 매우 떨어지는 비감독형 모델보다 훨씬 나은 성능을 보이고, 차원축소에도 영향을 받지 않는 안정적인 CollectiveWoods 모델을 사용할 수 있다는 것을 의미한다.

References

[1] E. Hong, “Software Fault Prediction using Semi-supervised Learning Methods,” Journal of the Institute of Internet, Broadcasting and Communication, Vol.19, No.3, pp.127-133, June 2019. DOI: https://doi.org/10.7236/JIIBC.2019.19.3.127

[2] R. Malhotra, “A systematic review of machine learning techniques for software fault prediction,” Applied Soft. Computing Vol.27, pp.504-518, 2015. DOI: https://doi.org/10.1016/j.asoc.2014.11.023

[3] H. Lu, B. Cukic, and M. Culp, “A Semi-Supervised Approach to Software Defect Prediction,” Proc. of COMPSAC, Sept. 2014. DOI: https://doi.org/10.1109/COMPSAC.2014.65

[4] C. Catal and D. Banu. “Unlabelled extra data do not always mean extra performance for semi- supervised fault prediction," Expert Systems, Vol.26 No.5, pp.458-47, Nov. 2009. DOI:https://doi.org/10.1111/j.1468-0394.2009.00509.x

[5] H. Lu, B. Cukic, and M. Culp, “An iterative semi-supervised approach to software fault prediction," Proc. of PROMISE ’11, 2011. DOI: https://doi.org/10.1145/2020390.2020405

[6] Y. Jiang, M. Li, and Z.H. Zhou, “Software defect detection with ROCUS,” Journal of Computer Science and Technology, Vol.26 No.2, pp.328-342. March 2011. DOI: https://doi.org/10.1007/s11390-011-9439-0

[7] M. Li, H. Zhang, R. Wu, and Z. H. Zhou, “Sample-based software defect prediction with active and semi-supervised learning,” Automated Software Engineering, Vol.19, No.2, pp.201-230, June 2012.

DOI: https://doi.org/10.1007/s10515-011-0092-1

[8] N. Seliya and T.M. Khoshgoftaar, "Software quality estimation with limited fault data: a semi- supervised learning perspective," Software Quality Journal Vol.15 No.3, pp.327-344, Sept. 2007.
DOI: https://doi.org/10.1007/s11219-007-9013-8

[9] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," AI magazine, Vol. 29, No.3, pp.93-106, 2008.
DOI: https://doi.org/10.1609/aimag.v29i3.2157

[10] E. Frank, M. A. Hall, and I. H. Witten, The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

[11] E. Hong, "Ambiguity Analysis of Defectiveness in NASA MDP data sets," Journal of Information Technology Services, Vol.12, No.2, pp.361-371, 2013.
DOI: https://doi.org/10.9716/KITS.2013.12.2.361

[12] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data Quality : Some Comments on the NASA Software Defect Data Sets," IEEE Trans. Software Engineering, Vol.39, No.9, pp.1208-1215. Sept. 2013.
DOI: https://doi.org/10.1109/TSE.2013.11

[13] T. Fawcett, "An introduction to ROC analysis," Pattern recognition letters, Vol.27, No.8, pp.861- 874, June 2006.
DOI: https://doi.org/10.1016/j.patrec.2005.10.010

[14] Eun-Mi Kim, "Adaptive Network Model for the Recognition of Software Quality Attributes," Journal of KIIT, Vol.15, No.11, pp.103-109, 2017.
DOI: https://10.14801/jkiit.2017.15.11.103

저 자 소 개

홍 의 석(정회원)

- 1992년 서울대학교 계산통계학과 전산과학전공 학사
- 1994년 서울대학교 계산통계학과 전산과학전공 석사
- 1999년 서울대학교 계산통계학과 전산과학전공 박사
- 현재 성신여자대학교 컴퓨터공학과 교수
- 관심 분야: 소프트웨어 품질 예측 모델, 소프트웨어 메트릭, MSR 등

※ 이 논문은 2018년도 성신여자대학교 학술연구조성비 지원에 의하여 연구되었음.