The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 19, No. 3, pp.127-133, Jun. 30, 2019. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2019.19.3.127
JIIBC 2019-3-17

세미감독형 학습 기법을 사용한 소프트웨어 결함 예측

Software Fault Prediction using Semi-supervised Learning Methods

홍의석*

Euyseok Hong*

요 약 소프트웨어 결함 예측 연구들의 대부분은 라벨 데이터를 훈련 데이터로 사용하는 감독형 모델에 관한 연구들이다. 감독형 모델은 높은 예측 성능을 지니지만 대부분 개발 집단들은 충분한 라벨 데이터를 보유하고 있지 않다. 언라벨 데이터만 훈련에 사용하는 비감독형 모델은 모델 구축이 어렵고 성능이 떨어진다. 훈련 데이터로 라벨 데이터와 언라벨 데이터를 모두 사용하는 세미 감독형 모델은 이들의 문제점을 해결한다. Self-training은 세미 감독형 기법들 중 여러 가정과 제약조건들이 가장 적은 기법이다. 본 논문은 Self-training 알고리즘들을 이용해 여러 모델들을 구현하였으며, Accuracy와 AUC를 이용하여 그들을 평가한 결과 YATSI 모델이 가장 좋은 성능을 보였다.

Abstract Most studies of software fault prediction have been about supervised learning models that use only labeled training data. Although supervised learning usually shows high prediction performance, most development groups do not have sufficient labeled data. Unsupervised learning models that use only unlabeled data for training are difficult to build and show poor performance. Semi-supervised learning models that use both labeled data and unlabeled data can solve these problems. Self-training technique requires the fewest assumptions and constraints among semi-supervised techniques. In this paper, we implemented several models using self-training algorithms and evaluated them using Accuracy and AUC. As a result, YATSI showed the best performance.

Key Words : Fault prediction, Semi-supervised learning, Self-training

Ⅰ. 서 론

소프트웨어 프로세스에서 주어진 비용과 자원을 이용하여 고품질의 소프트웨어를 기간 내에 개발하고 유지보수하는 것은 매우 중요하다. 이러한 목표를 달성하기 위해 분석이나 설계와 같은 초기 프로세스 단계에서 구현 단계 소프트웨어의 결함 정보들을 예측하는 결함 예측 모델은 성공적인 소프트웨어 프로젝트 수행을 위해 점점 더 중요해지고 있다. 대부분의 결함 예측 모델은 소프트웨어 구성 모듈들의 결함경향성 유무를 판단하는 이진 분류 모델이며, 결함 경향 부분들을 초기에 찾아냄으로써 적절한 자원 할당, 최적의 테스트 커버리지 결정, 성공적인 리팩토링 후보 결정 등을 가능케 한다[1].

그림 1은 결함을 예측하기 위해 데이터 집합을 입력으

*정회원, 성신여자대학교 정보시스템공학과
접수일자 2019년 4월 23일, 수정완료 2019년 5월 23일
게재확정일자 2019년 6월 7일

Received: 23 April, 2019 / Revised: 23 May, 2019 / Accepted: 7 June, 2019
*Corresponding Author: hes@sungshin.ac.kr
Dept. of Information Systems Engineering, Sungshin Women's University, Korea

로 받아 모델을 학습하는 훈련 단계와 결함 경향성을 결정하는 예측 단계를 보여준다. 이 때, L은 라벨 데이터이며 U는 언라벨 데이터(라벨 없는 데이터)이다. 결함 예측 모델에서 라벨은 모델의 출력값, 즉 결함경향여부를 의미하며 라벨 데이터는 입력값과 출력값이 있는 데이터를 의미하고 언라벨 데이터는 입력값만 있는 데이터를 의미한다. 결함 예측 모델은 데이터 집합의 종류에 따라 훈련 단계에 라벨 데이터만 사용하는 감독형 모델, 언라벨 데이터만 사용하는 비감독형 모델, 두 데이터를 함께 사용하는 세미 감독형 모델로 구분할 수 있다.

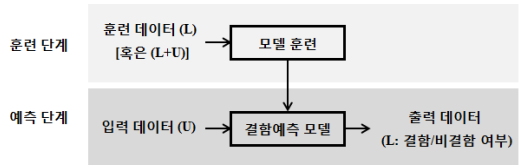

그림 1. 결함 예측 모델
Fig. 1. Fault Prediction Model

지난 수십년간 제안된 대부분의 예측 모델들은 라벨 데이터 집합을 사용하는 감독형 모델들이었다. 감독형 모델은 높은 예측 정확도를 보이지만 이는 훈련 데이터 수에 비례하고, 대부분 개발 집단은 대량의 라벨 데이터를 보유하고 있지 않다는 문제점이 있다. 따라서 이를 해결할 수 있는 비감독형 모델과 세미 감독형 모델들의 필요성이 대두되고 있지만 매우 극소수의 연구들만이 수행되었다. 주로 입력 데이터들을 몇 개의 클러스터로 나누어 특성을 정한 후 입력 모듈이 속하는 클러스터를 정하여 해당 모듈의 결함경향성을 결정하는 비감독형 모델은 전문가가 참여하는 클러스터 분석 단계의 어려움, 모델 구축의 어려움, 낮은 예측 성능 등의 문제점이 있다.

이와 달리 세미 감독형 모델은 라벨 데이터와 언라벨 데이터를 모두 훈련 데이터로 사용한다. 감독형 모델과 달리 대량의 라벨 데이터를 필요로 하지 않으며, 비감독형 모델에 비해 분석 및 모델 구축이 쉽다. 세미 감독형 방법 중 하나인 Self-training은 세미 감독형 기법의 대표적인 방법으로 구현이 비교적 간단하기 때문에 널리 사용된다. 또한 특정한 가정이 필요 없고 제약 조건들이 적다. 본 논문의 목적은 감독형과 비감독형 모델의 문제점들을 해결하기 위해 세미 감독형 기법인 Self-training을 사용하여 여러 가지 모델들을 구현해보고 그들의 예측 성능을 비교하여 세미 감독형 모델들의 효용성을 알아보는 것이다.

2장에서는 기존의 세미 감독형 모델을 포함한 결함 예측 연구들을 살펴보고, 3장에서는 구현 모델의 구조 및 사용 시나리오 대해 설명한다. 4장에서는 세미 감독형 모델들의 성능 실험 결과를 비교하고, 5장에서는 결론을 기술한다.

Ⅱ. 관련 연구

감독형 예측 모델들은 훈련 데이터 학습을 위해 로지스틱 회귀분석 같은 통계 기법들과 인공 신경망, 베이지안 분류기, 판단 트리, SVM, RF(Random Forest) 등과 같은 기계학습 기법들을 사용하였다[1]. 훈련 데이터가 없는, 즉 라벨 데이터가 없는 경우 사용 가능한 비감독형 모델들이 사용한 기법들은 K-means, X-means, EM, DBSCAN 등의 클러스터링 기법들이다[2]. 라벨 훈련 데이터 집합의 필요성 여부, 사람 전문가의 참여 여부, 예측 성능 차이에 의한 두 부류 모델들의 장단점은 명확히 존재하지만 아직 대부분 경우에 만족할만한 성능으로 사용할 수 있는 일반화된 모델은 존재하지 않는다. 세미 감독형 모델은 두 부류 모델들의 문제점들을 동시에 완화시키는 모델 형태이다.

매우 소수에 불과한 세미 감독형 모델 연구들은 모델 제안을 한 경우와 여기에 데이터 전처리 기법을 결합한 기법을 제안한 경우들이 있다. [3]은 EM 알고리즘을 이용하여 언라벨 데이터를 결측 정보로 간주하고 이를 추정된 값으로 대체하여 훈련 데이터로 사용하여 모델을 제작하였으며, 이 모델은 C4.5 감독형 모델보다 나은 성능을 보였다. [4]는 YATSI 알고리즘을 응용하여 인공 면역 기반의 YATSI 알고리즘을 제안하였다. [5]에서는 기존 Self-training 기법의 변형인 FTF(Fitting the Fits) 알고리즘을 제안하였다. 모델을 훈련하는데 언라벨 데이터 집합의 하위 집합만을 사용하는 기존 Self-training과 달리 FTF는 초기에 언라벨 데이터를 모두 라벨링 한 후 훈련에 사용하고 올바른 라벨을 갖도록 개선해나간다. RF를 기저 분류기로 한 FTF 세미 감독형 알고리즘은 RF 감독형 모델보다 더 나은 성능을 보였다.

몇몇 연구들은 새로운 학습 방법을 제안하는 동시에, 효율적으로 예측 성능을 높이기 위해 예측 데이터를 모델에 적용하기 전에 데이터 전처리 방법을 이용하였다[6,7,8]. [6]에서는 결함 모듈 수가 비결함 모듈 수보다 매우 적은 클래스 불균형 문제를 해결하는 세미 감독형 방법인 ROCUS(RandOm Committee with Under Sampling)

알고리즘을 제안하였다. ROCUS는 여러 학습기를 훈련시키고 학습기 간의 불일치를 통해 개선해나가는 disagreement 기반 세미 감독형 모델이다. [7]은 [5]를 변형한 FTcF(Fitting the Confident Fits) 알고리즘을 이용한 모델을 제안했다. 기존 Self-training과는 라벨 데이터 집합으로 추가된 데이터는 고정되므로 라벨 없는 데이터 수가 감소한다는 점에서 차이가 있다. [8]은 랜덤 샘플링을 거친 감독형 모델들과 앙상블 학습을 이용한 disagreement 기반의 세미 감독형 모델인 CoForest, 전문가가 샘플링을 한 세미 감독형 모델인 ACoForest의 성능을 비교하였으며, 실험 결과 ACoForest, CoForest, 감독형 모델 순으로 좋은 결과를 보였다.

Ⅲ. 모델 제작

1. 세미 감독형 학습

세미 감독형 기법은 입력 모듈에 대해 출력 값이 존재하는 라벨 데이터와, 출력 값이 없는 언라벨 데이터를 모두 훈련 데이터로 사용한다. 대개의 경우는 소수의 라벨 데이터와 상대적으로 다수의 언라벨 데이터가 사용된다. 세미 감독형 학습 기법은 크게 세미 감독형 분류와 세미 감독형 클러스터링으로 나뉘는데 본 논문은 분류가 목적이므로 세미 감독형 분류만을 다룬다.

세미 감독형 분류 기법은 크게 Generative 알고리즘, Self-training, Co-training, Transductive-SVM, 그래프기반 알고리즘의 5가지 영역으로 구분 한다[4]. 이들 기법들 사용하여 고성능 예측 모델을 구현하기 위해서는 각 기법마다 특정한 가정과 제약조건을 필요로 한다[5]. Co-training은 입력 데이터 집합이 완전하게 독립적인 2개 이상의 집합들로 나누어져야 함을 가정한다. 또한 그래프 기반의 알고리즘은 고밀도 지역에서 연결된 점들은 같은 클래스에 속하고, transductive-SVM으로 대표되는 밀도 기반 알고리즘들은 데이터가 자연스럽게 클러스터들로 분리되어야 한다고 가정한다. Generative 알고리즘 역시 데이터 분포를 알고 있음을 가정한다. 그러나 결함 예측을 하고자 하는 일반적인 데이터 집합은 이러한 가정들을 만족하기 어렵다. 소수의 라벨 데이터로 훈련한 모델로 새로운 라벨 데이터를 만들어 모델 완성까지 훈련을 반복하는 Self-training 방법은 특정한 가정이 필요 없으며 조건에 대한 제약이 적다. 이러한 이유로 본 논문에서는 세미 감독형 방법을 Self-training으로 국한한다.

2. 모델 시나리오

Self-training 모델 구축은 데이터 클리닝 및 정규화나 속성 선정 등의 데이터 전처리 과정을 거친 훈련 데이터를 이용해 반복 훈련을 통해 이루어진다. 이 때 어떤 훈련 알고리즘을 적용하느냐에 따라 모델의 성능이 좌우된다. 감독형 학습 방법과 비슷한 과정을 거치지만 훈련 데이터의 수를 점차 늘려가면서 여러 번의 훈련 과정을 반복한다는 점에서 차이가 있다.

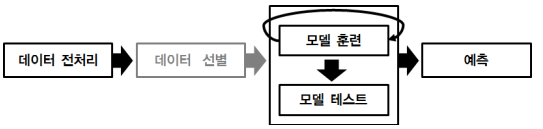

그림 2. Self-training 모델 구축 프로세스
Fig. 2. Construction process of the Self-training model

결함 예측 모델 연구에서 가장 많이 사용되어온 공개 데이터 집합은 NASA MDP 데이터 집합이다. 그림 2에서 데이터 선별은 NASA 데이터 집합과 같이 모든 데이터가 라벨 데이터인 경우 모델 구축을 위해 훈련 데이터 집합 선별 및 그 중 라벨 데이터의 선별 과정을 의미한다. 이 과정은 라벨 데이터가 소수 존재하는 실제 경우에는 필요 없으므로 엷게 표시하였다. 라벨 데이터의 비율은 예측 성능에 큰 영향을 미치므로 훈련 데이터에서 라벨 데이터와 언라벨 데이터의 비율을 다르게 하여 모델의 성능을 비교한다. 초기 훈련 데이터 집합의 라벨 데이터로 훈련한 모델은 훈련 데이터의 언라벨 데이터를 예측하여 라벨링한다. 새롭게 라벨이 정해진 데이터 중 가장 예측이 잘된 데이터들을 선정하여 다음 단계의 훈련에 라벨 데이터로 추가한다. 이 과정을 반복 하면서 여러 번의 훈련 과정을 거쳐 훈련 데이터의 라벨 데이터의 수를 늘려나가며, 이 반복 과정은 훈련 데이터 집합이 모두 라벨 데이터로 되어 변화하지 않을 때까지 계속된다. 완성된 모델은 테스트 데이터로 테스트를 거쳐 예측에 사용된다.

3. 사용 알고리즘

MARSDEN 프로젝트[9]에서는 여러 가지 세미 감독형 알고리즘을 제공하며 이들은 예측 모델 연구들에 많이 사용된 WEKA[10]에서 제공하는 'Collective' API를 사용하여 코드로 구현 가능하다. WEKA에서는 Collective 분류기를 functions, lazy, meta, tree의 4개 카테고리로 분류하였으며 각 카테고리는 몇 개의 세미 감독형 알고리즘들을 가지고 있다. 표 1은 이들 정보들을 나타내며 짙게 나타낸 알고리즘들은 세미 감독형 기법들이다.

표 1. WEKA의 세미 감독형 알고리즘
Table 1. Semi-supervised learning algorithms in WEKA

분류 그룹	알고리즘
functions	LLGC
lazy	CollectiveIBk
meta	AdvanceCollective
	Chopper
	CollectiveEM
	CollectiveNeighbor
	CollectiveWrapper
	FilteredCollectiveClassifier
	SimpleCollective
	TwoStageCollective
	YATSI
	Weighting
trees	CollectiveForest
	CollectiveTree
	CollectiveWoods

표 1에서 선택된 세미 감독형 알고리즘들은 반복 훈련 시 사용할 새로운 라벨 데이터 선정 방법 및 추후 제거 여부 등에 차이가 있지만 모두 선정하였다. 다만 meta 분류기들은 기본 기저 분류기가 J48이고, tree 모델들은 각각 RandomForest, CollectiveForest를 사용하였으므로 구현 및 실험에서 tree 모델들을 제외하였다.

Ⅳ. 실험 및 결과

1. 데이터 집합 및 평가 척도

NASA 데이터 집합들은 13개의 프로젝트들로 구성되어있으며 각 프로젝트는 여러 개의 데이터 파일들로 구성된다. 본 연구에서는 실험 결과를 기존 관련 연구와 비교하기 위해 NASA 프로젝트들 중 결함 예측 연구들에 가장 많이 사용된 PC1과 CM1을 사용하였다[1]. 사용한 데이터 버전은 MDP 원본 데이터와 이를 정제한 PROMISE 레포지토리 데이터 중 후자를 사용하였다. 초기 PROMISE 데이터 집합은 [11]에서 제안한 방법으로 데이터 정제 작업이 이루어졌으며 결과물은 DS'와 DS" 두가지 형태이다. 전자는 상수 및 중복 속성 삭제, 결측값 처리, 데이터 무결성 처리를 마친 정제 데이터고 후자는 DS'에 중복 데이터(입출력이 모든 같은 케이스) 및 모순 데이터(입력은 같은데 출력이 다른 케이스)를 삭제한 데이터 집합이다. 실제 중복과 모순 데이터는 존재 가능하고 예측 모델은 이들 데이터도 처리 가능해야하므로 본 연구는 DS' 버전을 사용한다. 표 2는 두 데이터집합의 정보를 나타낸 것으로 결함 모듈 수는 10%로 내외로 적당하다.

표 2. 실험 데이터 집합 정보
Table 2. Properties of experimental data sets

데이터 집합	모듈 수	결함 모듈수
CM1	344	42 (12%)
PC1	759	61 (8%)

CM1, PC1의 입력 메트릭, 즉 속성들은 Halstead 계열, McCabe 계열, LOC 계열의 복잡도 관련 메트릭들이다. CFS(Correlation based Feature Selection)를 이용한 속성 선정 작업을 통해 이들을 CM1은 8개, PC4는 10개로 축소하였으며, 모델 평가 실험은 전체 속성을 사용한 경우와 차원 축소를 한 경우 두가지 형태로 시행하였다.

예측 모델의 성능 평가를 위한 평가 척도로 Accuracy와 AUC(Area Under ROC curve)를 사용한다. Accuracy는 전체 데이터에 대해 모델의 예측이 맞은 경우의 비율로 모델의 전체 예측 정확도를 나타내지만 결함 관련 데이터 집합은 결함 모듈 수가 항상 비결함 모듈 수보다 매우 작은 불균형 데이터 집합이므로 이를 보완하기 위해 최근 예측 모델 평가에 매우 많이 사용되는 척도인 AUC를 함께 사용한다. ROC 곡선은 실제 결함이 있는 모듈에 대해 결함이 있다고 예측하는 TPR(True Positive Rate)과 실제 결함이 없는 모듈에 대해 결함이 있다고 잘못 예측하는 FPR(False Positive Rate) 값들을 표현하며, 이 곡선을 정량화 한 AUC는 1에 가까울수록 TPR은 높고 FPR은 낮은 좋은 예측 성능을 의미한다.

2. 평가 실험

CM1, PC1은 이미 라벨을 알고 있으므로 이 중 일부를 언라벨 데이터로 취급하여 훈련 데이터에 사용하며, 이를 다시 답을 아는 라벨 데이터 형태로 테스트 데이터로 사용한다. 훈련 데이터의 언라벨 데이터의 비율은 95%, 90%, 80% 로 실험한다. 각 경우에 소수인 라벨 데이터의 비율은 5%, 10%, 20%가 된다.

실험에서는 라벨 데이터, 언라벨 데이터를 비율대로 랜덤하게 10번씩 반복하여 선택하였고 평균 Accuracy와 AUC를 측정하였다. 하나의 분류기를 이용하여 라벨 데이터의 비율을 5%, 10%, 20%로 10번씩, 정규화 적용 및 미적용 데이터 집합에 전체 속성을 사용하는 경우와 CFS로 축소된 속성을 사용하는 경우까지 총 120번의 실험을 진행하였다.

표 3. [4]와 본 연구의 YATSI(J48) 실험 결과
Table 3. YATSI(J48) results of [4] and this study

데이터 집합	모델 YATSI	5%	10%	20%
CM1	[4]	0.56	0.60	0.60
	본 연구	0.56	0.62	0.62
PC1	[4]	0.59	0.60	0.66
	본 연구	0.61	0.60	0.67

표 3은 AUC 결과를 [4]의 실험 결과와 비교한 표이다. [4]에서는 기저 분류기를 다르게 하여 YATSI 알고리즘의 성능을 AUC로 측정하였다. 본 실험에서는 J48을 기저 분류기로 하였기 때문에 [4]의 YATSI(J48)의 결과와만 비교하였다. [4]보다 더 많은 정제 과정을 거친 데이터 집합을 사용하였기 때문에 약간의 차이를 보이지만 유사한 결과를 보임을 알 수 있다.

전체 실험 결과는 표 4와 표 5에 나타내었다. 표 4는 전체 속성을 가지고 실험한 결과이며, 표 5는 속성 선정을 적용한 실험의 결과이다. 정규화를 한 데이터 집합과 정규화를 하지 않은 집합은 같은 결과를 보였기 때문에 하나의 결과만을 나타내었다. 많은 경우에 CFS를 이용해 속성 간의 연관 노이즈를 제거한 표 5가 조금 더 나은 결과를 보였다. 명암을 넣은 부분은 해당 데이터 집합 사용 시 가장 좋은 성능을 보인 알고리즘 결과 부분을 나타낸다. 예를 들면 전체 속성을 사용하고 훈련 데이터의 라벨 데이터를 5% 사용하였을 경우 가장 좋은 AUC 결과는 YATSI와 Weighting 알고리즘이 보인 0.61이다.

속성 선정 기법 사용 유무와 상관없이 대부분 모델에서 라벨 데이터의 사용 비율이 커질수록 예측 정확도는 높아지나 탁월하게 좋아지지는 않는다. 속성 선정을 적용한 CM1에서 SimpleCollective 알고리즘은 라벨 데이

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

알고리즘	프로젝트	5%		10%		20%	
		Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
Chopper	CM1	84.31	0.53	82.81	0.55	85.25	0.55
	PC1	89.22	0.57	90.22	0.58	89.80	0.57
CollectiveEM	CM1	81.56	0.52	81.10	0.58	82.84	0.56
	PC1	88.97	0.60	90.29	0.57	89.67	0.58
SimpleCollective	CM1	69.72	0.49	71.26	0.50	71.05	0.50
	PC1	67.18	0.47	73.13	0.48	70.97	0.52
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
Weighting	CM1	82.02	0.52	81.26	0.58	82.95	0.58
	PC1	89.89	0.61	90.34	0.56	89.93	0.57

표 5. 속성 선정을 사용한 경우 실험 결과
Table 5. Experimental results when the attribute selections are used

알고리즘	프로젝트	5%		10%		20%	
		Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
Chopper	CM1	82.63	0.55	84.74	0.54	85.20	0.51
	PC1	89.49	0.58	90.16	0.56	90.82	0.57
CollectiveEM	CM1	82.08	0.56	82.06	0.57	85.31	0.53
	PC1	89.39	0.59	89.19	0.57	90.58	0.61
SimpleCollective	CM1	82.91	0.58	85.58	0.53	84.62	0.49
	PC1	90.26	0.54	90.47	0.49	89.09	0.50
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
Weighting	CM1	82.63	0.56	81.68	0.57	85.64	0.54
	PC1	89.67	0.57	89.37	0.56	91.00	0.55

터 비율을 10%로 한 경우 Accuracy와 AUC가 85.58, 0.53이지만 20%인 경우 84.62, 0.49을 보여 라벨 데이터사용 비율이 더 낮은 경우에서 더 좋은 예측 결과를 내는 예외적인 결과를 보이기도 한다. 이는 라벨 데이터의 비율이 낮더라도 의미 있는 결과를 보일 수 있으며, 대량의 학습 데이터가 존재하지 않는 경우에 사용할 수 있는 가능성을 보이는 경우이다.

모델간의 성능 비교는 Accuracy로 평가한 경우는 YATSI가 가장 좋은 결과를 보이지만 몇몇 경우에는 다른 모델들의 성능이 좋다. 특히 속성 선정을 한 경우는 라벨 데이터가 5%, 10%인 경우에 두 데이터 집합 모두에서 SimpleCollective 알고리즘이 가장 좋은 Accuracy 결과를 보였다. 하지만 전체 속성을 사용한 경우에는 SimpleCollective는 매우 안 좋은 성능을 보였으므로 안정적인 모델로 보기엔 어려움이 있다. YATSI는 다른 모델이 가장 좋은 결과를 보인 경우에도 좋은 결과를 보였으며, 전체적으로 가장 좋은 결과도 많이 보인 가장 좋은 모델이다. 결함 데이터가 불균형 데이터라는 관점에서 더 중요한 평가 척도인 AUC로 평가한 경우는 모든 경우에서 YATSI가 가장 좋은 결과를 보였다. 따라서 실험 결과로는 YATSI가 가장 성능이 좋고 안정적인 세미 감독형 모델이다.

V. 결 론

기존의 매우 많은 연구들이 결함 예측 분야에서 수행되었지만 세미 감독형 모델에 관한 연구들은 극소수에 불과하다. 결함 예측 모델의 중요성이 커지면서, 예측 성능은 높지만 대량의 훈련 데이터가 필요한 감독형 모델과 분석에 있어 많은 비용과 노력이 들고 예측 정확도가 낮은 비감독형 모델 외에 이 두 모델들의 장점을 취한 세미 감독형 모델의 필요성이 빠르게 증가하고 있다. 본 논문은 세미 감독형 기법들 중 복잡한 가정의 제약들이 없어 쉽게 사용될 수 있는 Self-training 기법을 사용한 여러 예측 모델들을 제작하고 그들의 성능을 Accuracy와 AUC를 사용하여 평가하였다. 그 결과 YATSI 모델이 가장 좋은 성능을 보였다. 향후 연구는 세미 감독형 기법을 확장하여 트리 그룹 기법들을 사용한 모델들과 J48 외의 다른 여러 기저분류기들을 사용한 모델들의 성능 평가 실험을 하는 것이다.

References

[1] 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

[2] E. Hong, "Severity-based Fault Prediction using Unsupervised Learning," Journal of the Institute of Internet, Broadcasting and Communication, Vol.18, No.3, pp.151-157, June 2018.
DOI: https://doi.org/10.7236/JIIBC.2018.18.3.151.

[3] 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.

[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] 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

[8] 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

[9] MARSDEN project http://www.cs.waikato.ac.nz/fracpete/marsden/

[10] WEKA (Waikato Environment for Knowledge Analysis) http://www.cs.waikato.ac.nz/~ml/weka/

[11] 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

[12] Eun-Mi Kim, "Adaptive Network Model for the Recognition of Software Quality Attributes," Journal of KIIT, Vol.15, No.11, pp.103-109, 2017.

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 정보시스템공학과 교수

- 관심 분야 : 소프트웨어 품질 예측 모델, 소프트웨어 메트릭, MSR 등

※ 이 논문은 2017년도 성신여자대학교 학술연구조성비 지원에 의하여 연구되었음.