Journal of The Korea Society of Computer and Information
Vol. 21 No. 4, pp. 73-80, April 2016

www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2016.21.4.073

Severity-based Software Quality Prediction using Class Imbalanced Data

Euy-Seok Hong *, Mi-Kyeong Park **

Abstract

Most fault prediction models have class imbalance problems because training data usually contains much more non-fault class modules than fault class ones. This imbalanced distribution makes it difficult for the models to learn the minor class module data. Data imbalance is much higher when severity-based fault prediction is used. This is because high severity fault modules is a smaller subset of the fault modules. In this paper, we propose severity-based models to solve these problems using the three sampling methods, Resample, SpreadSubSample and SMOTE. Empirical results show that Resample method has typical over-fit problems, and SpreadSubSample method cannot enhance the prediction performance of the models. Unlike two methods, SMOTE method shows good performance in terms of AUC and FNR values. Especially J48 decision tree model using SMOTE outperforms other prediction models.

▸ Keyword : Data imbalance, Fault prediction, Severity, Sampling

I. Introduction

1980년대부터 계속되어온 소프트웨어 품질 예측에 관한 연구들은 대부분 메트릭 벡터 형태로 정량화 된 입력 개체의 결함경향성 유무를 판단하는 분류 모델 형태의 결함 예측 모델에 관한 것들이었다. 결함 데이터 수집의 어려움 때문에 2000년 이전의 연구들은 많지 않았으며, 비공개된 데이터의 사용으로 연구들 간의 비교나 평가가 어려웠다. 하지만 2000년대 초에 NASA IV&V MDP의 데이터 집합이 공개되고, 대표적인 소프트웨어 품질 관련 공개 데이터 집합인 PROMISE 레포지토리 운영이 시작되면서 연구들이 크게 늘어났다[1]. 현재는 많은 연구들을 정리하는 연구 결과들이 발표되고 있다[2].

결함 유무를 예측하는 모델은 구현 완료 후 또는 다음 릴리즈에서 결함이 발생할 문제 부분들을 미리 예측하므로 적절한 자원 할당 및 올바른 리팩토링 후보 결정 등을 통해 높은 신뢰성을 갖춘 시스템 구축을 가능케 한다[1]. 하지만 대부분의 모델들은 결함 유무만을 출력으로 하는 이진 분류 모델들로 결함이 가진 특성들을 전혀 고려하지 않는다는 문제점이 있다. 결함의 특성들 중 가장 중요한 결함 심각도는 소프트웨어 시스템과 사용자들에게 결함이 미치는 충격의 정도를 나타내는 척도이다[3]. 저심각 결함은 수행에 문제가 되지 않을 정도이지만, 고심각 결함은 시스템을 파괴시키거나 수행을 멈출 수도 있다. 따라서 심각도에 기반하여 고심각 개체와 저심각 개체를 분류하는 예측 모델은 자원 할당 등에 보다 자세한 정보를 제공함으로써 기존 예측 모델들보다 훨씬 유용하게 사용될 수 있다[4].

본 연구의 동기는 심각도에 기반한 결함 예측 모델을 제안한 선행 연구인 [5]의 연구 결과에 추가 실험을 통하여 분석한 평가 결과가 매우 좋지 않았기 때문이다. [5]의 제안 모델은 입력 모듈을 고심각(HSF: High Severity Fault-prone) 모듈, 저심각(LSF: Low Severity Fault-prone) 모듈, 비결함(NF: Not Fault-prone) 모듈로 분류하는 삼중 분류 모델이다. 데이터 집합으로는 NASA 데이터 집합들 중 모호성이 가장 적은[6] JM1, PC4를 사용하였으며 모델의 평가 척도로 전체 데이터에 대해 모델의 예측이 맞은 경우의 비율, 즉 Accuracy로 모델들의 예측 성능을 평가하였다.

Accuracy는 모델의 전체 예측 정확도를 반영하지만 심각도 기반 모델에서는 고심각 결함 모듈의 예측 정확도가 가장 중요하다.

• First Author: Euy-Seok Hong, Corresponding Author: Euy-Seok Hong
*Euy-Seok Hong (hes@sungshin.ac.kr), School of Information Technology, Sungshin Women's University
**Mi-Kyeong Park (parkmikyeong@gmail.com), Dept. of Computer Science, Sungshin Women's University
• Received: 2016. 04. 06, Revised: 2016. 04. 14, Accepted: 2016. 04. 27.
• This work was supported by the Sungshin University Research Grant of 2014.

따라서 [5]의 삼중 분류 모델을 Fig. 1과 같이 출력이 고심각(HSF)과 비고심각(Not HSF)으로 나뉘는 이진 분류 모델로 재해석하여 고심각 결함 모듈의 예측 정확도를 AUC(Area Under ROC)나 Type I/II 오류 등의 이진 분류 모델 성능 평가 척도들로 재평가하였다. 이와 같은 확장 분석 결과는 Accuracy 결과와는 다르게 해당 모델을 사용하지 못할 만큼 매우 좋지 않은 결과를 보였다. 이는 고심각 결함을 가진 모듈이 전체에 비해 극소수이기 때문에 생기는 데이터 불균형 문제라 생각된다. 데이터 불균형이란, 한 클래스에 속한 데이터의 수가 다른 클래스에 속한 데이터의 수보다 과도하게 많거나 현저하게 적은 현상을 의미한다. 불균형 데이터를 이용해 학습한다면, 분류의 정확성을 높이기 위해 다수 데이터를 가진 클래스에 편향된 학습을 하게 되고 따라서 소수 데이터를 가진 클래스의 예측률이 다수 데이터 클래스에 비해 매우 낮아진다.

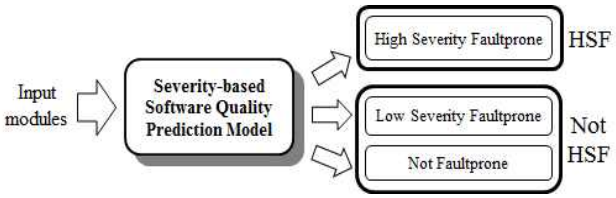

Fig. 1. Translation of [5] to Binary Classification Model

Table 1은 [5]가 사용한 데이터 집합에서 가장 중요한 출력 클래스인 고심각 모듈은 저심각이나 비결함 모듈에 비해 극소수에 불과하여 데이터 불균형 문제를 야기할 수 있다는 것을 보여준다. 본 논문의 목적은 심각도 기반 결함 예측 모델과 같은 다중 출력 모델에서 가장 중요한 소수 클래스 예측 성능을 높이기 위해 클래스 불균형 데이터를 처리하는 기법들을 적용해보고 일반 모델들과의 성능 평가 비교를 통해 이 기법들의 유용성을 평가하는 것이다.

Table 1. Imbalanced data sets used in [5] (SEVERITY1)

Data set	# modules	NF	LSF	HSF
PC4	1458	1280 (87.79%)	120 (8.23%)	58 (3.97%)
JM1	10878	8776 (80.67%)	1759 (16.17%)	343 (3.15%)

2장에서는 기존의 결함 예측 연구들과 불균형 데이터를 사용한 연구들을 살펴보고, 3장에서는 선행 연구인 [5]의 확장 분석 결과의 문제점을 자세히 언급하고 사용하고자 하는 샘플링 모델들을 설명한다. 4장에서는 모델 성능 실험 결과를 언급하고, 5장에서는 결론을 기술한다.

II. Related works

1. Software fault prediction

현재까지 제안된 수많은 결함 예측 모델들은 학습 데이터의 유무에 따라 크게 세가지로 분류할 수 있다. 세가지 형태는 훈련 데이터를 사용하여 학습하는 감독형 모델, 훈련 데이터가 없어서 클러스터링 알고리즘 등을 사용하는 비감독형 모델, 학습하기에 충분치 않은 제한된 훈련 데이터만 사용하는 세미감독형 모델이다. 기존 연구들의 거의 대부분은 감독형 모델에 대한 것들이었고, 다른 두 모델 형태에 관한 연구들은 극소수에 불과하다. 왜냐하면 입력 데이터 분석과정에서 사람이 개입해야하는 어려움이 있고 예측 성능이 감독형 모델에 비해 떨어지기 때문이다. 하지만 훈련 데이터 집합을 보유하지 않은 많은 개발 집단이 사용할 수 있다는 큰 장점 때문에 최근에 새로운 연구 결과들이 계속 등장하고 있다.

훈련 데이터란 입력 개체를 정량화한 메트릭 값과 그에 대한 결함 정보 출력값이 함께 있는 데이터를 의미하며, 과거 유사 프로젝트나 현재 프로젝트의 이전 릴리즈 개발 과정의 산물로 얻어진다. 감독형 모델 연구들이 훈련 데이터를 학습하는데 사용한 분류 알고리즘들은 주로 기계학습이나 통계 기법들이었다 [2]. 기계학습 기법들 중 가장 많이 사용된 것들은 판단 트리, 베이지안 분류기, SVM, 신경망이었고 통계 기법으로는 로지스틱 회귀분석이 주로 사용되었다. 몇몇 연구들에 불과하지만 비감독형 모델이 모델 구축에 주로 사용한 기법은 K-means, X-means, EM, DBSCAN 등의 클러스터링 알고리즘들이다 [7]. 세미감독형 모델은 여러 알고리즘들을 혼용하였다.

2. Prediction based on defect severity

결함 심각도 값에 대한 합의된 표준안은 없다. 많은 개발 집단에서 내부적으로 사용하는 심각도 값은 심각한 단계를 나타내는 서수 스케일(ordinal scale) 형태의 심각도 레벨 형태이다. 예를 들면, NASA 데이터 집합은 5단계의 심각도 레벨값을 포함하고 있으며 심각도 1이 가장 심각한 결함을, 심각도 5가 가장 가벼운 결함을 나타낸다.

심각도에 기반한 연구들은 모두 감독형 모델들이었다. [5]는 NASA의 JM1과 PC4를, [8]과 [9]는 NASA의 KC1 프로젝트를 훈련 데이터로 사용하였다. [5]는 학습 기법으로 신경망, 베이지안 모델, 판단 트리를 [8]은 로지스틱 회귀분석과 베이지안 모델, 랜덤 포리스트, NNge를 [9]는 로지스틱 회귀분석과 판단 트리, 신경망을 사용하였다. [8]은 심각도 1을 고심각 결함, 2~5를 저심각 결함으로 분류하였고 [9]는 심각도 1을 고심각, 2를 중심각, 3~5를 저심각 결함으로 분류하였다. [5]는 심각도 1을 고심각 결함, 2~5를 저심각 결함으로 분류한 데이터를 SEVERITY1, 심각도 1~2를 고심각 결함, 3~5를 저심각 결함으로 분류한 데이터를 SEVERITY2라 정의하여 두 가지 출력에 대한 실험을 수행하였다. 모듈의 심각도 결정은 모듈이 가진 결함의 심각도에 따라 결정하였다.

[8], [9]는 성능 평가에서 이진 분류 모델 형태를 사용하였다. 두 연구 모두 저심각 예측이 고심각 예측보다 좋은 성능을 보였고, [9]는 중심각 예측의 성능이 가장 좋았다. [5]는 출력

이 고심각, 저심각, 비결함으로 나뉘는 삼중 분류 모델을 사용하였으며 Accuracy를 사용한 평가 결과는 역전파 신경망 모델이 가장 좋았다.

3. Prediction using class imbalanced data

소프트웨어 시스템을 구성하는 개체들 중 결함 개체가 비결함 개체에 비해 매우 소수이므로 결함이 없는 다수의 개체 데이터가 예측 모델의 성능을 좌우하게 된다. 따라서 비결함 개체에 대한 예측 성능보다 중요시해야 할 결함 개체에 대한 예측 성능이 좋지 못하다. 이러한 클래스 불균형 문제를 해결하기 위해 몇몇 연구들에서 사용된 기법은 샘플링 기법이다.

[10]은 결함 예측 모델에 샘플링을 적용한 최초의 연구로 선형판별분석, 로지스틱 회귀분석, 신경망, 판단 트리 알고리즘을 샘플링 기법들(ROS, SMOTE, RUS, ONESS)을 사용하여 전처리한 MIS 소프트웨어 데이터 집합에 적용하여 결함 예측 성능을 높였다. 4가지 샘플링 기법들은 모두 선형판별분석, 로지스틱 회귀분석에서 성능을 향상시켰으나, 신경망, 판단 트리에서는 항상 향상되지는 않았다. 이는 샘플링 기법이 항상 높은 성능에 도움이 되는 건 아니라는 것을 의미한다. [11]은 감독형 모델에서 훈련 데이터 수집의 어려움과 함께 클래스 불균형 문제를 지적하였다. 이를 해결하기 위해 새로운 세미감독형 방법인 ROCUS(RandOm Committee with Under-Sampling) 알고리즘을 제안하였다. ROCUS는 여러 학습기를 훈련시키고 학습기 간의 불일치를 통해 개선해나가는 disagreement 기반 세미감독형 모델이다. 결과적으로 샘플링을 거친 모델이 클래스 불균형 문제를 무시한 다른 세미감독형 알고리즘(ROCA)의 성능보다 좋고, 라벨 없는 데이터를 이용한 클래스 불균형 방법의 성능보다 뛰어남을 밝혔다. [12]는 샘플링으로 데이터를 전처리한 후 여러 가지 다른 형태의 알고리즘들을 적용하여 결과를 비교하였다. 랜덤 샘플링을 거친 감독형 모델(로지스틱 회귀분석, 베이지안 모델, 판단 트리)과 세미감독형 모델(CoForest), 전문가가 데이터 특성을 고려하여 샘플링 한 세미감독형 모델(ACoForest)을 비교 실험하였다. 실험 결과 ACoForest, CoForest, 감독형 모델 순의 성능 평가 결과를 보였으나 ACoForest는 전문가가 직접 데이터를 샘플링 해야 한다는 단점이 있다. [13]은 클래스 불균형 해결을 위해 Resampling, 임계치 이동(Threshold moving), 앙상블 기법들을 실험하였고 그 결과 AdaBoost.NC가 가장 좋은 성능을 보임을 보였다.

III. The Proposed Model

1. Problems of the previous model

[5]는 모델의 평가 실험을 위해 예측 모델 연구들에서 많이 사용된 데이터 마이닝 도구인 WEKA[14]의 Experimenter를 이용하였으며 모델의 입력 메트릭은 전체와 CFS(Correlation based Feature Selection) 기법으로 차원 축소한 두 경우, 출력은 SEVERITY1, SEVERITY2의 두 경우를 모두 실험하였다. 사용한 알고리즘들은 예측 모델 연구에 가장 많이 사용되어온 신경망(MLP: MultiLayer Perceptron), 베이지안 모델의 Naïve Bayes(NB), 판단 트리 모델인 J48이다. 세 개의 출력 클래스인 HSF, LSF, NF 중 HSF 예측 성능 평가를 수행하기 위해 추가한 척도는 AUC와 FNR(False Negative Rate)이다. AUC는 최근 예측 모델 평가에 많이 사용되고 있는 척도로 True Positive(TP)와 True Negative(TN)를 동시에 나타내는 ROC curve 면적을 측정한 것이다. 이 때, x축은 FPR(False Positive Rate), y축은 TPR(True Positive Rate)이 된다. AUC 수치에 따라 성능이 아주 안 좋거나(AUC<0.5), 덜 정확한(0.5<AUC<=0.7), 정확한(0.7<AUC<=0.9), 매우 정확한(0.9<AUC<1), 완벽한(AUC=1) 예측으로 분류할 수 있다[15]. FNR은 HSF 모듈을 LSF나 NF 모듈로 잘못 예측한 비율로 Type I/II 오류에서 Type II 오류를 의미한다. Not HSF인 모듈을 HSF로 잘못 예측하는 것보다 HSF 모듈을 Not HSF로 잘못 예측하는 것이 개발 프로세스에서 훨씬 많은 비용을 요구하므로 Type II 오류가 Type I 오류보다 중요한 척도이다.

Table 2는 JM1을 사용한 추가 실험 결과를 나타낸다. 추가 실험에서도 Accuracy(ACC) 결과는 [5]의 결과와 거의 일치하였다. 하지만 FNR은 상당수 모델들이 0.9를 넘을 정도로 좋지 않은 결과를 보였다. 이와 같은 결과는 극소수에 속하는 고심각 예측보다 저심각 예측 성능이 좋았던 기존 연구들인 [8], [9]의 결과들과 유사하다. FNR 결과는 해당 모델을 고심각 결함 예측 모델로 사용하기 어렵다는 것을 의미하며 원인은 클래스 불균형 문제 때문이다. 기존 학습 모델들은 세가지 클래스에 대해 가중치를 두고 학습하지 않으므로, 다수의 NF 모듈에 대해 많은 학습을 하고 극소수의 HSF 모듈에 대해 적은 학습을 한다. 따라서 HSF 모듈에 대한 예측 성능은 상대적으로 낮아질 수밖에 없다.

Table 2. Extended results of the previous study

Data	Measure	Attribute Selection	J48	NB	MLP
SEVERITY1	ACC	All	0.78	0.79	0.81
		CFS	0.80	0.79	0.81
	AUC	All	0.53	0.76	0.76
		CFS	0.60	0.75	0.76
	FNR	All	0.90	0.86	0.99
		CFS	0.92	0.86	0.99
SEVERITY2	ACC	All	0.78	0.79	0.81
		CFS	0.79	0.78	0.81
	AUC	All	0.55	0.68	0.68
		CFS	0.56	0.68	0.69
	FNR	All	0.87	0.87	0.97
		CFS	0.92	0.86	0.98

2. Sampling Model

일반적으로 샘플링이란 모집단에서 표본을 추출하는 것을 뜻한다. 본 연구에서는 출력 클래스 간 데이터 수의 균형을 맞추기 위해 데이터에 적절한 가중치를 결정하는 작업을 뜻한다. 샘플링 방법은 크게 Over-sampling과 Under-sampling으로 구분할 수 있다. Over-sampling은 소수의 클래스 데이터를 랜덤하게 데이터 집합에 더하여 소수 클래스에 더 큰 가중치를 부여하는 방법이며, Under-sampling은 다수 클래스 데이터에서 데이터를 랜덤하게 삭제하여 더 작은 가중치를 부여하는 방법이다. 본 연구에서 사용한 WEKA는 대표적인 샘플링 방법들을 제공하고 있으며 본 연구에서 사용할 기법들은 다음과 같다.

Table 3. Representative sampling methods of WEKA

Method	Explanation
Resample	데이터의 랜덤 종속 표본 추출
SpreadSubsample	클래스 빈도 사이에 주어진 분포로 랜덤 표본 추출
SMOTE	소수 클래스 데이터들의 합성을 통한 표본 추출

2.1 Resample

Resample은 한 번 표본이 되었던 데이터를 배제하지 않고 다시 표본이 될 수 있도록 허용하는 Over-sampling 방법이며, 소수 클래스 데이터를 다시 추가한다. 클래스 분포를 유지하도록 샘플링 할 수도 있으며(클래스 비율은 같게 하되, 사이즈만 변경), 모든 클래스 분포가 동일해지도록 샘플링 할 수도 있다. 본 연구에서는 클래스 불균형 문제를 해결하기 위해 샘플링 방법을 사용하므로, 클래스 분포를 유지하는 것이 아닌 모든 클래스 분포가 동일해지도록 샘플링 하였다. Fig. 2는 3개의 소수 클래스 데이터를 다시 추가하여 두 클래스의 분포를 같게 한 예이다. 추가된 데이터는 원본 데이터와 같은 위치에 그려져야 하나 이해를 돕기 위해 다른 위치에 점선으로 표시하였다.

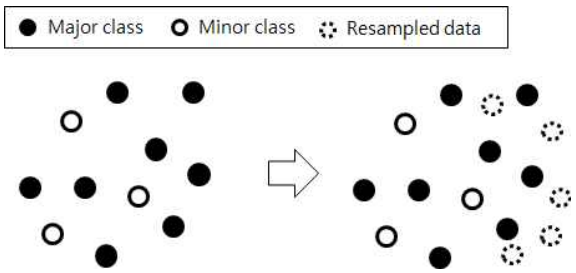

Fig. 2. Example of Resample

2.2 SpreadSubSample

SpreadSubSample는 데이터 집합의 임의의 표본을 생성하는 기법으로, 데이터의 수가 가장 적은 소수 클래스와 가장 빈도가 높은 다수 클래스 사이의 빈도수 차이 값을 조절할 수 있다(최대 "Spread"를 지정할 수 있다). 예를 들어, 2를 최대 Spread로 한다면 이 클래스 간의 비율을 1대 2로 하여 희소 클래스 데이터가 1일 때, 가장 수가 많은 클래스 데이터를 2로 임의의 표본을 생성한다. 이때, 최대 비율은 다수 클래스 데이터의 수를 넘지 않는다. 즉, 다수 클래스 데이터를 모두 사용하거나 비율에 맞게 랜덤하게 제거하는 Under-sampling 기법이다.

실험에서는 최대 Spread를 1, 2, 5로 설정하여 1:1, 1:2, 1:5일 때를 실험하였다. Fig. 3은 2개의 다수 클래스 데이터를 랜덤하게 삭제하여 소수 클래스 데이터와 다수 클래스 데이터 비율이 1대 2가 되도록 한 예이다.

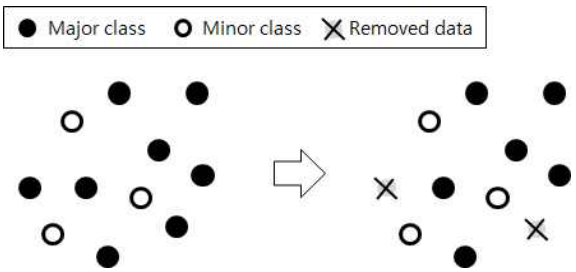

Fig. 3. Example of SpreadSubSample

그러나 이러한 샘플링 방법들은 몇가지 문제점을 가지고 있다. Over-sampling 방법은 소수 클래스 데이터를 복제하여 훈련 데이터에 과적합(Over-fit)하는 문제를 일으킬 수 있으며 Under-sampling 방법은 다수의 클래스 데이터의 유용하고 중요한 정보가 삭제될 수 있다. 즉, Resample과 같이 대체를 허용하는 Over-sampling 기법은 똑같은 인스턴스들을 추가하고, 모델은 그 인스턴스들을 반복 학습하게 되므로 결국 과적합하게 된다. 또한 SpreadSubsample과 같은 Under-sampling 기법은 다수 클래스 데이터를 랜덤하게 삭제하므로 유용하고 중요한 정보를 버릴 수도 있다. 이런 문제점들을 해결하기 위해 몇몇 샘플링 방법들이 제안되었지만 구현하기 쉽고 간단하여 가장 많이 이용되는 기법은 SMOTE 샘플링 기법이다[16].

2.3 SMOTE

SMOTE(Synthetic Minority Over-sampling TEchnique)는 소수 클래스의 데이터를 합성하여 새로운 데이터를 만들어 내는 Over-sampling 기법이다. SMOTE는 먼저 소수 클래스의 데이터 표본들을 취한 뒤 이 표본들의 k 최근접 이웃을 찾는다. 그리고 현재 표본들과 이들 k개 이웃들 간의 차이를 구하고, 이 차이에 0~1 사이의 임의의 값을 곱하여 원래 표본에 더한다. 이렇게 만든 새로운 표본들을 훈련 데이터에 추가하며, 이는 결과적으로 기존의 표본들을 주변의 이웃을 고려해 약간씩 이동시킨 표본들을 추가하는 방식으로 동작하는 것을 의미한다. 개체 1이 (10, 10)의 값을 가지고 개체 2가 (10, 12)의 쌍을 가진다고 할 때, SMOTE 샘플링을 한다면 두 개체의 평균값인 (10, 11)인 데이터가 추가될 수 있다. 따라서 유사하지만 똑같지는 않는 인스턴스를 생성하므로 하나의 인스턴스에 대한 과적합이 일어나지 않을 것이고 보다 일반적으로 분류할 수 있게 된다. 즉, 데이터 중복 문제를 해결이 가능하다. Fig. 4는 SMOTE 기법으로 소수 클래스 데이터를 2배 늘려 균형을 이루도록 불균형 데이터 집합을 샘플링한 결과를 나

타낸 것이다. 새로운 데이터는 기존 데이터에 과적합 하지 않도록 소수 클래스의 데이터 거리의 평균값을 구해 생성한다.

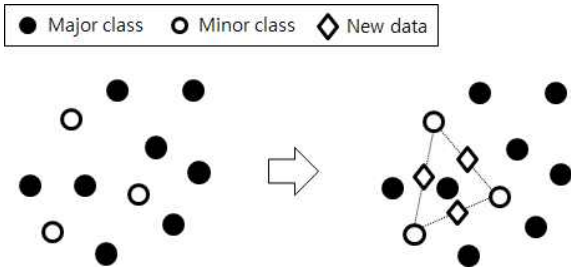

Fig. 4. Example of SMOTE

IV. Experiments

1. Data Set and Performance Measures

선행 연구인 [5]의 실험 결과와 비교 평가가 필요하므로 본 연구 역시 공개 데이터 집합들 중 결함 관련 연구에 가장 많이 사용된 NASA 데이터 집합을 사용하였다. 실험에는 [5]가 사용한 JM1, PC4 프로젝트들 중 JM1만을 사용하였다. 그 이유는 JM1이 PC4보다 10배 정도 큰 코드 사이즈를 가지므로 PC4보다 실험 데이터로 적당하며, PC4의 모듈들은 심각도 값으로 1~3까지의 값만을 가지고 있어 1~5까지의 값을 모두 가지고 있는 JM1에 비해 원본 심각도 데이터에 값에 한계가 있을 수 있다는 점이다. 또한 Table 1에서 보듯이 JM1의 HSF 클래스 모듈 비율도 PC4와 같이 충분히 작다.

JM1 프로젝트는 C언어로 구현된 실시간 시스템이며 315 KLOC와 10878개의 모듈로 구성된 대형 시스템이다. HSF 모듈은 HSF 결함이 하나 이상 있는 모듈이고, LSF 모듈은 HSF 결함이 없고 LSF 결함이 하나 이상 있는 모듈이며, NF 모듈은 결함이 없는 모듈이다. SEVERITY1은 심각도 1인 결함만을 HSF 결함으로, SEVERITY2는 심각도 1 또는 2인 결함을 HSF 결함으로 본 데이터 집합이다. SEVERITY1에서 전체 모듈들 중 HSF 모듈은 343개, LSF 모듈은 1759개, NF 모듈은 8776개이다.

JM1의 입력 메트릭들은 Halstead 계열, McCabe 계열, LOC 계열의 전통적인 복잡도 관련 메트릭들이다. 입력 메트릭이 21차원이므로 작지 않고, 입력들 중에는 다른 메트릭들을 조합한 파생 메트릭들도 존재하므로 효율적인 알고리즘 수행을 위해 입력 데이터의 차원 축소가 필요하다. 따라서 결함 예측 분야에서 가장 일반적으로 많이 사용된 차원 축소 기법으로, 전처리 과정에서 정규화한 입력 데이터의 모든 속성의 조합들과 결함 예측 출력치의 상관도를 평가하여 가장 최적의 조합을 찾아내는 CFS 속성 선정 기법을 적용하였다[2]. Table 4는 JM1의 전체 입력 데이터와 CFS에 의해 선정된 데이터를 나타낸다. 모델 평가 실험에는 이들 두가지 입력 형태를 모두 사용하였다.

모델의 평가 척도는 Accuracy 외에 3.1절에서 언급한 AUC와 FNR을 사용하였다. Table 5는 HSF 예측 관점에서 본 Confusion matrix를 나타낸 것이다. 즉, HSF 예측이 모델 성능에 가장 중요한 부분이므로 HSF 모듈 예측 성공 여부에 따른 이진 분류 형태로 수식을 정의한 것이다. FNR은 HSF 모듈 예측에 실패하는 오류율을 의미한다.

Table 4. Input metrics for prediction model

Attribute Selection		Input metrics
All	CFS	LOC_BLANK, LOC_CODE_AND_COMMENT, LOC_COMMENTS, CYCLOMATIC_COMPLEXITY, DESIGN_COMPLEXITY, ESSENTIAL_COMPLEXITY, HALSTEAD_CONTENT, LOC_TOTAL
		BRANCH_COUNT,LOC_EXECUTABLE, HALSTEAD_DIFFICULTY, HALSTEAD_EFFORT HALSTEAD_ERROR_EST, HALSTEAD_LENGTH HALSTEAD_LEVEL,HALSTEAD_PROG_TIME HALSTEAD_VOLUME,NUM_OPERANDS, NUM_OPERATORS NUM_UNIQUE_OPERANDS, NUM_UNIQUE_OPERATORS

Table 5. Confusion matrix based on HSF

		Predicted Class		
		HSF	LSF	NF
Actual Class	HSF	True Positive(TP)	False Negative(FN)	
	LSF	False Positive(FP)	True Negative(TN)	
	NF			

$$ACC = \frac{TP + TN}{TP + FP + FN + TN} \qquad FNR = \frac{FN}{TP + FN}$$

2. Size of Sampling results

샘플링은 모델 학습 전에 데이터에 진행되는 작업이므로 각 샘플링 방법에 의해 데이터 집합의 크기가 어떻게 결정되었는지를 살펴보아야 한다. Table 6은 본 연구의 실험에서 사용된 샘플링 방법들의 결과로 얻어진 데이터 집합의 크기이다.

Table 6. Sampling Size

Data	Sampling	Exp.id	#HSF	#LSF	#NF
SEVERITY1	Non-sampling		343	1759	8776
	Resample	①	1813	1813	1813
		②	3626	3626	3626
	SpreadSubSample	③	343	343	343
		④	343	686	686
		⑤	343	1715	1715
	SMOTE	⑥	686	1759	8776
		⑦	1029	1759	8776
SEVERITY2	Non-sampling		506	1596	8776
	Resample	①	1813	1813	1813
		②	3626	3626	3626
	SpreadSubSample	③	506	506	506
		④	506	1012	1012
		⑤	506	2530	2530
	SMOTE	⑥	1012	1596	8776
		⑦	1518	1596	8776

실험은 크게 SEVERITY1과 SEVERITY2로 나누어 진행되었으며 각 실험을 지칭하기 쉽게 하기 위하여 id를 붙였다. #HSF/#LSF/#NF는 샘플링에 의해 조정된 HSF/LSF/NF 모듈의 수를 나타낸다. Resample은 샘플링을 통해 HSF, LSF, NF 클래스의 모듈 수를 동일하게 일정한 수로 만들어 클래스 간 균형을 맞춘다. 10878개의 50%인 5439개를 1813개씩 세 개의 클래스로 균등하게 나눈 ①과 10878개를 3626개씩 세 개의 클래스로 균등하게 나눈 ②의 두 가지 경우로 실험을 하였다. Resample은 전체 모듈을 균등하게 나누기 때문에 SEVERITY1과 SEVERITY2 두 경우 모두 모듈의 수가 동일하다. SpreadSubSmaple에서는 기존의 HSF 모듈 수를 1의 비율로 고정하고 이와 LSF 모듈, NF 모듈의 비율이 각각 1:1인 ③, 1:2인 ④, 1:5인 ⑤의 3가지 경우를 실험하였다. 즉, SEVERITY1을 사용한 실험의 데이터 크기는 343/343/343, 343/686/686, 343/1715/1715이고, SEVERITY2를 사용한 실험은 506/506/506, 506/1012/1012, 506/2530/2530이 된다. SMOTE 기법에서는 LSF와 NF 모듈을 그대로 둔 채 소수 클래스인 HSF 모듈의 수치를 조절한다. 이 때, HSF 모듈을 그대로 복사하는 것이 아니라 두 모듈의 평균치로 새로운 모듈 데이터를 만들어 낸다. 기존 HSF 모듈의 수를 각각 2배, 3배로 늘린 실험 ⑥과 ⑦을 수행하였다. SEVERITY1을 사용한 실험의 데이터 크기는 686/1759/8776, 1029/1759/8776이고, SEVERITY2를 사용한 실험의 경우는 1012/1596/8776, 1518/1596/8776이 된다.

3. Experimental Results

평가 실험에 사용한 학습 모델과 속성 선정 방법, 출력 해석에 따른 두 개의 데이터 집합인 SEVERITY1, SEVERITY2를 사용한 것들은 모두 Table 2를 결과로 얻어낸 선행 연구에 대한 확장 실험 경우와 일치한다. 각 모델의 구조와 초기 설정값은 WEKA의 기본값을 이용하였다. 실험의 정확도를 높이기 위해 일반적인 10 폴드 교차 검증(10-fold cross validation)을 10회 반복하였다. 최종적으로 각 모델의 평가 결과는 10번 반복된 교차검증 결과들의 평균값이 된다.

SEVERITY1과 SEVERITY2를 사용한 경우의 전체 실험 결과를 Table 7과 Table 8에 나타내었다. 입력 메트릭을 전체로 한 경우와 CFS를 사용하여 속성 선정을 한 경우의 결과값들은 비슷한 값을 보이며 유의미한 차이가 없다.

선행 연구 결과인 Table 2와 각 샘플링 기법을 적용한 결과들을 비교해 보면, Resample의 경우에는 J48이 매우 좋은 결과를 보인다. ACC는 비슷한 결과를 보이지만 AUC는 거의 1에 가까워졌고, 0.9를 넘었던 FNR 오류율이 0에 가까워진 엄청난 성능 향상을 보인다. 하지만 NB와 MLP 모델은 Resample 기법을 사용하여도 성능 향상을 보이지 않았다. FNR은 약간 나은 결과를 보였으나, AUC는 약간 안좋은 결과를, ACC는 매우 안좋은 결과를 보였다. ACC 결과는 Resample 데이터 집합이 동일 데이터 값들을 많이 지니고 있으므로 NB, MLP 같은 일반화된 학습 성공에 안 좋은 결과를 끼쳤다는 이유로 분석된다. 판단 트리는 훈련 데이터가 커서 큰 트리를 생성하는 경우 다른 모델들에 비해 과적합 문제에 쉽게 빠진다고 알려져 있다 [17]. 최근 약 20년간의 결함 예측 분야의 가장 중요한 연구들을 검토한 [2]의 분석 결과를 보면 J48을 사용한 성공적인 8개 연구 모델들의 AUC 값이 최소 0.78, 최대 0.93, 평균 0.83으로 모두 본 실험 결과보다 매우 낮다. 또한 JM1을 이용한 모델들 중 가장 좋은 성능을 보인 랜덤 포리스트는 평균 0.747의 AUC 값 성능을 보였다. 따라서 J48 모델의 과도한 성능 폭등은 판단 트리에 대해 중복 데이터들이 훈련 과정에서 과적합된 결과라 보인다. 결국 Resample 기법은 NB와 MLP 모델에서 샘플링을 사용하지 않은 모델보다 좋지 않은 결과를 보이므로 선행 연구 문제를 해결하는 안정된 기법이라 볼 수 없다.

SpreadSubSample을 사용한 경우는 세 모델이 비슷한 성능을 보였다. 하지만 선행 연구의 결과와 비교해 보면 세 모델 모두 ACC는 매우 안 좋아졌으며, AUC는 J48만 약간 좋아지고 나머지는

Table 7. Results when SEVERITY1 data is used

			J48			NB			MLP		
Attribute Selection	Sampling	Exp.id	ACC	AUC	FNR	ACC	AUC	FNR	ACC	AUC	FNR
All	Resample	①	0.78	0.96	0.03	0.41	0.72	0.86	0.49	0.74	0.49
		②	0.86	0.98	0.01	0.41	0.72	0.86	0.50	0.75	0.49
	SpreadSubSample	③	0.44	0.61	0.53	0.42	0.71	0.81	0.45	0.71	0.57
		④	0.49	0.60	0.69	0.48	0.71	0.81	0.49	0.71	0.82
		⑤	0.56	0.59	0.84	0.51	0.71	0.86	0.56	0.71	0.91
	SMOTE	⑥	0.76	0.65	0.51	0.77	0.76	0.86	0.78	0.77	0.91
		⑦	0.76	0.72	0.55	0.75	0.76	0.84	0.77	0.77	0.87
CFS	Resample	①	0.77	0.96	0.03	0.42	0.70	0.83	0.48	0.73	0.52
		②	0.84	0.98	0.01	0.42	0.71	0.84	0.49	0.73	0.52
	SpreadSubSample	③	0.48	0.63	0.52	0.46	0.71	0.79	0.48	0.72	0.61
		④	0.48	0.61	0.75	0.47	0.69	0.81	0.49	0.69	0.78
		⑤	0.58	0.62	0.86	0.51	0.71	0.86	0.56	0.69	0.92
	SMOTE	⑥	0.77	0.73	0.58	0.75	0.74	0.85	0.78	0.76	0.96
		⑦	0.78	0.64	0.57	0.77	0.74	0.86	0.76	0.76	0.87

Table 8. Results when SEVERITY2 data is used

			J48			NB			MLP		
Attribute Selection	Sampling	Exp.id	ACC	AUC	FNR	ACC	AUC	FNR	ACC	AUC	FNR
All	Resample	①	0.77	0.94	0.07	0.42	0.65	0.84	0.46	0.65	0.74
		②	0.86	0.98	0.01	0.41	0.63	0.85	0.46	0.65	0.71
	SpreadSubSample	③	0.46	0.62	0.54	0.41	0.63	0.84	0.44	0.62	0.69
		④	0.48	0.61	0.70	0.45	0.63	0.86	0.49	0.61	0.88
		⑤	0.78	0.55	0.87	0.79	0.68	0.87	0.81	0.68	0.97
	SMOTE	⑥	0.76	0.66	0.65	0.76	0.67	0.86	0.77	0.69	0.82
		⑦	0.76	0.76	0.47	0.73	0.67	0.86	0.75	0.69	0.87
CFS	Resample	①	0.76	0.94	0.07	0.42	0.65	0.83	0.46	0.65	0.69
		②	0.85	0.98	0.01	0.41	0.64	0.84	0.47	0.64	0.69
	SpreadSubSample	③	0.49	0.65	0.59	0.42	0.64	0.81	0.43	0.63	0.68
		④	0.51	0.61	0.77	0.47	0.64	0.85	0.49	0.62	0.88
		⑤	0.59	0.59	0.83	0.59	0.65	0.86	0.60	0.64	0.92
	SMOTE	⑥	0.77	0.67	0.68	0.76	0.68	0.85	0.77	0.69	0.80
		⑦	0.76	0.74	0.54	0.73	0.68	0.84	0.75	0.69	0.84

안 좋아졌으며, FNR은 약간 좋아졌다. 하지만 FNR도 대부분이 0.6을 훨씬 넘는 0.8이나 0.9의 결과를 보임으로써 HSF 모듈을 예측 하는 모델로는 사용하기 어렵다. 따라서 SpreadSubSample 기법 역시 모델의 성능 향상에 안정적인 도움을 주지 못했다.

SMOTE 기법을 사용한 경우는 세 개의 모델들이 다른 성능을 보였다. 세 모델 모두 ACC의 경우는 0.7~0.8로 유사한 성능을 보였고, AUC는 J48보다 나머지 두 모델들이 매우 근소하게 좋은 성능을 보였으나 유의미한 차이는 아니므로 유사한 결과를 보였다 할 수 있다. 하지만 HSF 모듈을 예측하는 중요한 오류율인 FNR은 J48의 결과가 나머지 두 모델보다 현격하게 좋았다. J48은 SEVERITY2의 실험 ⑥에서 0.6대의 성능을 보였고 나머지는 모두 0.5대의 결과를 보였다. 좋은 FNR 결과를 보였던 SpreadSubSample과 SMOTE를 비교하면 실험 ③의 경우 FNR이 SMOTE 결과보다 근소하게 좋은 경우가 있지만 전체적으로는 근소하게 안 좋았고, AUC 역시 안 좋았으며, ACC는 매우 낮았다. SMOTE 기법을 사용한 결과를 선행 연구와 비교해 보면 J48은 ACC는 근소하게 떨어졌으나 유사하고, AUC는 꽤 좋아졌고, FNR은 현격하게 좋아졌다. NB 역시 ACC는 근소하게 떨어졌으나 AUC와 FNR은 거의 같았다. MLP는 ACC는 근소하게 떨어졌고 FNR은 거의 같았지만 AUC가 꽤 좋아졌다. 따라서 SMOTE 기법이 모든 예측 모델들의 성능을 유의미하게 향상시키지는 못했지만 J48 모델의 성능은 매우 높였다.

V. Conclusions

훈련 데이터 집합을 사용하는 결함 예측 모델은 비결함 데이터에 비해 결함 데이터가 소수이므로 학습 모델로 하여금 비결함 데이터를 더 잘 학습하게 하는 데이터 불균형 문제가 발생한다. 이는 결함을 고심각과 저심각으로 분류하는 심각도 기반 예측 모델에서는 더욱 심각한 문제가 된다. 왜냐하면 고심각 결함 데이터는 결함 데이터에 비해 더 극소수이기 때문이다. 이와 같은 문제점은 심각도 기반 모델을 제안한, 본 본문의 선행 연구의 결과를 확장 분석한 실험에서 확연하게 나타났다.

본 논문에서는 데이터 불균형 문제를 해결하고자 대표적인 샘플링 방법들인 Resample, SpreadSubSample, SMOTE 기법들을 사용하여 심각도 기반 결함 예측 모델들을 제작하고 성능을 평가하였다. 실험 결과, 이론에 기초한 예상대로 Resample 기법은 과적합된 결과를 보였고, SpreadSubSample을 사용한 모델들은 성능 결과가 좋지 않았다. 이론적으로 가장 안정적인 결과를 보이리라 예상한 SMOTE 기법이 전체적으로 좋은 결과를 보였으며, 그들 중 판단 트리 알고리즘인 J48 모델이 가장 좋은 결과를 보였다. J48-SMOTE 모델은 선행 연구와 비슷한 ACC를 보였으나, 더 나은 0.7 정도의 AUC와 훨씬 향상된 FNR 결과를 보임으로써 선행 연구의 문제점들을 상당 부분 해결하였다. 향후 연구는 SMOTE 기법을 수정하여 베이지안 모델과 신경망 모델의 성능도 유의미하게 향상시키는 것이다.

REFERENCES

[1] C. Catal, “Software fault prediction: A literature review and current trends,” Expert Systems with Applications, Vol.38, No.4, pp.4626-4636, April 2011.

[2] R. Malhotra, “A systematic review of machine learning techniques for software fault prediction,” Applied Soft. Computing Vol.27, pp.504-518, Feb. 2015.

[3] D. E. Harter, C. F. Kemerer and S. A. Slaughter, “Does Software Process Improvement Reduce the Severity of

Defects? A Longitudinal Field Study," IEEE Trans. Software Eng., Vol.38, No.4, pp. 810-827, July 2012.
[4] Y. Zhou and H. Leung, "Empirical analysis of object-oriented design metrics for predicting high and low severity faults," IEEE Trans. Software Eng., Vol.32, No.10, pp.771-789, Oct. 2006.
[5] E. S. Hong, "Software Quality Prediction based on Defect Severity," Journal of the Korea Society of Computer and Information, Vol.20, No.5, pp. 73-81, May 2015.
[6] E. S. Hong, "Ambiguity Analysis of Defectiveness in NASA MDP data sets," Journal of the Korea Society of IT Services, Vol.12, No.2, pp.361-371, June 2013.
[7] E. S. Hong and M. K. Park, "Unsupervised learning model for fault prediction using representative clustering algorithms," KIPS Trans. Software and Data Engineering, Vol.3, No.2, pp.57-64, Feb. 2014.
[8] Y. Zhou and H. Leung, "Empirical analysis of object-oriented design metrics for predicting high and low severity faults," IEEE Trans. Software Eng., Vol.32, No.10, pp.771-789, Oct. 2006.
[9] Y. Singh, A. Kaur and R. Malhotra, "Empirical validation of object-oriented metrics for predicting fault proneness models," Software Quality Journal, Vol.18, pp.3-35, March 2010.
[10] Y. Kamei, A. Moden, S. Matsumoto, T. Kakimoto and K. Matsumoto, "The Effects of Over and Under Sampling on Fault-prone Module Detection," proc. ESEM, pp.196-204, 2007.
[11] Y. Jiang, M. Li and Z. Zhou, "Software defect detection with ROCUS," Journal of Computer Science and Technology, Vol.26, No.2, pp.328-342, March 2011.
[12] 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.
[13] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," IEEE Trans. Reliability, Vol.62, No.2, pp.434-443, June 2013.
[14] WEKA (Waikato Environment for Knowledge Analysis) *http://www.cs.waikato.ac.nz/~ml/weka/*
[15] T. Fawcett, "An introduction to ROC analysis," Pattern recognition letters, Vol.27, No.8, pp.861- 874, June 2006.
[16] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of Artificial Intelligence Research, Vol.16, No.1, pp.321-357, Jan. 2002.
[17] L. Rokach and O. Maimon, "Top-Down Induction of Decision Trees Classifiers - A Survey," IEEE Trans. Systems, Man, and Cybernetics, Part C, Vol.35, No.4, pp. 476-487, Nov. 2005.

Authors

Euy-Seok Hong received the B.S., M.S. and Ph.D. degrees in computer science from Seoul National University, Korea, in 1992, 1994 and 1999, respectively. He joined the faculty of the Sungshin Women's University, Korea, in 2002. His research interests include software quality and prediction models in software engineering.

Mi-Kyeong Park received the B.S. degree in information technology and the M.S. degree in computer science from Sungshin Women's University, Korea, in 2014 and 2016, respectively. Her research interests include software engineering and data mining.