홍 의 석 교수지도
석사학위 청구논문

클러스터링 알고리즘을 사용한 비감독형 소프트웨어 품질 예측 모델

2016

성신여자대학교 대학원

컴퓨터학과

박 미 경

클러스터링 알고리즘을 사용한 비감독형 소프트웨어 품질 예측 모델

홍 의 석 교수 지도

이 논문을 석사학위논문으로 제출함

2015년 11월

성신여자대학교 대학원

컴퓨터학과

박 미 경

인 준 서

박미경의 석사학위 논문으로 인준함

2015년 11월

심사위원장 박 종 수 (인)

심 사 위 원 서 동 수 (인)

심 사 위 원 홍 의 석 (인)

성신여자대학교 대학원

논문 개요

소프트웨어 산업이 발전하면서 개발 프로세스 개선 및 평가 방법이 주목받고 있다. 고수준의 품질을 보장하기 위해서 품질에 영향을 미치는 핵심 부분들을 초기 프로세스에서 선정하고 이에 맞게 한정된 자원들을 분산 배치하여야 하므로 품질 예측 모델이 중요해지고 있다. 소프트웨어 품질 예측 모델을 통해 소프트웨어 프로세스에서 구현이 완료된 후에 나타날 문제를 미리 예측하여 고품질의 높은 신뢰성을 갖춘 시스템 구축 등을 가능하다.

소프트웨어의 특성을 나타내는 입력 모듈에 대해 결함 여부를 예측하는 결함 예측 모델 연구들은 대부분 훈련 데이터 집합이 필요한 감독형 모델에 관련된 것들이었다. 이와 달리, 과거 데이터 집합이 없거나 데이터 집합이 있더라도 현재 프로젝트와 성격이 다른 경우는 비감독형 모델이 필요하지만, 이들에 관한 연구는 모델 구축의 어려움 등으로 인해 극소수만이 존재한다. 모델 제작의 어려움 중 하나는 비감독형 모델을 구축하기 위해 사용되는 클러스터링 알고리즘의 클러스터 개수가 휴리스틱 하게 선택되어야 하는 것이며, 이 수는 예측 모델의 성능에 많은 영향을 미치게 된다.

본 논문에서는 1) 기존 비감독형 모델 연구들에서 사용되었던 K-means 알고리즘뿐만 아니라 사용하지 않은 대표적인 클러스터링 알고리즘인 EM, DBSCAN을 사용한 비감독형 모델과 2) 비감독형 모델 구축 프로세스에서 가장 많은 비용과 노력이 드는 분석 단계의 효율적인 수행을 위해 클러스터 수가 자동으로 결정되는 X-means, EM 알고리즘을 사용한 비감독형 모델들을 제작한다. 이를 실제 프로젝트 데이터 집합에 적용하고 모델의 성능 평가 및 기존의 연구들과의 비교를 통해 제작 모델들의 효용성을 보인다.

목 차

표 목 차

그 림 목 차

Ⅰ. 서 론

소프트웨어 산업이 발전할수록 소프트웨어 품질의 중요성이 강조되고 있다. 소프트웨어 프로세스에서 가장 중요한 것은 주어진 예산과 자원으로 고품질의 소프트웨어를 제작 및 유지보수 하는 것이다. 프로세스 후반 단계에서 발견한 결함은 초기 단계에서 발견한 결함에 비해 결함 수정에 있어 막대한 비용과 시간이 소요된다. 따라서 품질에 영향을 미치는 핵심 부분들을 초기 프로세스에서 선정하고 이에 맞게 한정된 자원들을 분산 배치한다면 고수준의 품질을 보장하는 소프트웨어를 제작할 수 있다. 이를 위해 소프트웨어 프로세스에서 구현이 완료된 후에 나타날 문제 부분을 미리 찾아내는 소프트웨어 품질 인자 예측이 주목받고 있다. 국내에서도 CMMi나 SPICE와 같은 소프트웨어 프로세스 평가 모델에 관한 관심이 증가하고 있으며, 특히 정량화에 의한 품질 예측 부분은 평가 모델들의 최상위 레벨에 속한 작업으로 예측 모델에 관한 연구의 필요성은 매우 높다.

품질 인자란 소프트웨어의 추상적인 품질 특성을 의미하며 유지 보수성, 신뢰성, 재사용성, 위험도 등이 있다[1]. 이는 소프트웨어의 구조적 특성을 나타내는 속성이나 규격을 측정한 기본 메트릭으로 정량화하여 추정할 수 있다. 품질 예측 연구 중 대다수의 품질 인자는 위험도에 관한 것들이었다. 설계 개체의 위험도란 개체가 구현되었을 때 갖는 결함경향성(fault-proneness)을 의미한다[2]. 위험도 예측 연구가 많은 이유는 생산된 소프트웨어에서 결함을 많이 일으킬 부분을 초기 단계에서 미리 찾아 적절한 자원 할당을 가능케 함으로써 프로젝트 전체 비용을 낮추고 소프트웨어 품질을 높이는 데 매우 유용하게 사용할 수 있기 때문이다. 본 연구에서도 위험도를 예측하는 결함 예측 모델로 한정한다.

소프트웨어 결함 예측 모델은 소프트웨어 코드의 라인 수나 복잡도 등을 계산하여 정량화한 메트릭과 소프트웨어 결함 간의 관계를 구축하여 소프트웨어의 결함을 예측하는 것이다. 즉, 메트릭 벡터로 수치화한 입력 모듈을 사용하여 모듈의 결함경향성을 결정한다. 이를 통해 소프트웨어 프로세스에서 구현이 완료된 후에 나타날 문제 부분을 미리 찾아냄으로써 적절한 자원 할당, 테스트 프로세스 개선, 최적의 리팩토링 후보 결정, 높은 신뢰성을 갖춘 시스템 구축 등이 가능하다[3].

메트릭 기반 결함 예측 모델에 관한 연구들은 소프트웨어 복잡도 연구가 한창이던 1980년대부터 계속되어 왔으며, 2000년대 중후반부터 관련 연구 결과 발표들이 급속히 많아졌다[3]. 이와 같은 현상의 주된 이유는 연구에 사용할 수 있는 공개 데이터 집합의 등장 때문이다. 대표적인 예로, 2000년대 초에 NASA IV&V 메트릭스 데이터 프로그램[1]의 데이터가 공개되고 2005년에 소프트웨어 예측 모델에 관련된 공개 데이터 집합들을 관리하는 PROMISE 레포지토리[2] 운영이 시작되었다.

최근까지 제안된 대부분의 예측 모델들은 훈련 데이터를 사용하여 학습하는 감독형 모델들이었다[3-5]. 훈련 데이터란 입력 모듈 데이터와 그에 대한 답(라벨)[3]이 함께 있는 데이터 집합을 의미하며, 감독형 모델은 이러한 입력 데이터에 대해 해당 답이 나오도록 학습한 모델을 의미한다. 이들은 학습에 판별분석, 회귀 분석 등과 같은 통계 방법이나 신경망 등과 같은 기계학습 알고리즘들을 사용한다. 이러한 훈련 모델들은 높은 예측 정확도를 보인다는 장점이 있지만, 과거 유사한 개발 환경에서 얻은 실제 프로젝트 데이터인 훈련 데이터 집합이 필요하다는 결정적인 문제점이 있다. 대부분의 개발 집단들은 이러한 훈련 데이터 집합을 보유하고 있지 않으며[6], 설

1) http://mdp.ivv.nasa.gov
2) http://promise.site.uottawa.ca/SERepository
3) 소프트웨어 결함 예측 모델에서는 결함 여부를 일컬음.

사 보유하고 있다 하더라도 모델을 적용하려는 현재 프로젝트의 참여 인력, 개발 환경이 과거 프로젝트와 유사하여야 한다.

따라서 감독형 모델은 실세계에서 사용되기 어려우며, 훈련 데이터가 없는 경우에 사용할 수 있는 비감독형 모델이 필요하다. [3]은 결함 예측 모델 분야의 중요한 향후 연구 주제로 비감독형 모델에 관한 연구를 제시하였다.

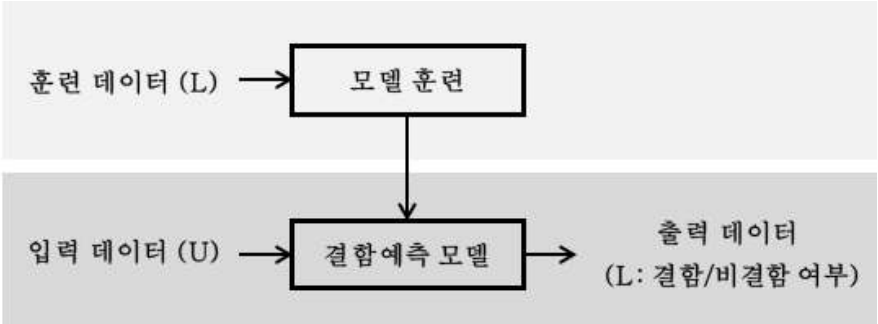

그림 1. 결함 예측 모델

[그림 1]은 입력 모듈들로 구성된 데이터 집합을 입력받아 각 모듈의 결함경향성을 결정하는 결함 예측 모델의 기능을 표현한 것이다. 이는 데이터를 입력받아 모델을 학습하는 훈련 단계와 결함경향성을 결정하는 예측 단계로 구성된다. 훈련 단계는 감독형 모델에만 적용되는 부분으로 모듈에 대한 결함 여부를 나타낸 라벨을 가진 훈련 데이터 집합(Labeled)이 있는 경우 이를 이용하여 훈련하는 것을 나타낸다. 비감독형 모델은 학습할 훈련 데이터가 없으므로 라벨이 없는 현재 입력 데이터(Unlabeled)만을 모델 구축에 사용하여야 한다. 따라서 입력 데이터의 분포만을 이용하여야 하며 이를 위해 클러스터링 알고리즘이 사용된다. 감독형 모델에 관한 연구들이 매우 많은 데 비해 입력 데이터 분석, 전문가에 의존, 모델 제작의 어려움 등의 이유로 인하여 비감독형 모델에 관한 연구들은 극소수에 불과하다.

비감독형 모델 제작이 어려운 이유 중 하나는 대부분의 클러스터링 알고리즘은 클러스터 수가 휴리스틱 하게 선택되어야 하며 그 수가 예측 모델의 성능에 영향을 미친다는 것에서 기인한다[7]. 이를 위해 데이터 집합에 관한 사전 지식에 기초하여 수를 결정하거나 반복적으로 여러 가지 경우의 수를 시도하여 가장 최적의 수를 결정하는 수고가 필요하다.

본 논문은 많지 않은 기존의 비감독형 모델 연구들에서 K-means 뿐만 아니라 사용하지 않았던 DBSCAN(Density-Based Spatial Clustering of Applications with Noise), EM(Expectation Maximization)과 같은 대표적인 클러스터링 방법을 사용한 모델들을 제작하는 실험Ⅰ과 클러스터 수 문제를 해결하기 위해 알고리즘 자체적으로 클러스터 수를 결정하는 EM, X-means(Extending K-means)를 사용하여 모델들을 제작하는 실험Ⅱ를 진행하고, 제작 모델들의 성능 평가 및 이전 연구 모델들과 성능을 비교하여 효용성을 알아본다.

본 논문은 총 5장으로 구성되어 있으며, 클러스터링 알고리즘을 사용하여 비감독형 소프트웨어 결함 예측 모델들을 제작하고 실제 프로젝트 데이터 집합을 적용하여 효용성을 보이는 것을 목표로 한다. 2장에서는 기존 소프트웨어 결함 예측 모델 연구들을 살펴보고, 3장에서는 모델 제작 방법 및 사용 시나리오, 클러스터링에 관해 설명한다. 4장에서는 모델 성능 실험을 위한 환경과 실험 과정, 결과에 대해 언급하며 5장에서는 결론 및 향후 연구에 관해 기술한다.

Ⅱ. 기존 연구

기존에 제안된 수많은 품질 예측 모델들의 대부분은 감독형 모델들이다. 감독형 모델 기법으로는 판별분석, 선형 회귀분석, 로지스틱 회귀 분석 등과 같은 통계 기법이나 분류 트리, 역전파 신경망, 랜덤 포레스트, 베이지안 분류기법, CBR(Case Based Reasoning), SVM(Support Vector Machine) 등과 같은 기계학습 기법들이다[3-5]. 이 연구들이 언급하지 않은 감독형 모델의 중요한 문제점은 대부분의 개발 집단이 훈련 데이터 집합을 보유하고 있지 않거나 제한된 형태로 보유하여 실제 개발 집단에서는 해당 모델들을 사용하기 어렵다는 것이다. 설사 훈련 데이터를 보유하고 있더라도 과거에 얻은 훈련 데이터가 현재 프로젝트의 데이터와 비슷한 분포를 가져야 모델을 사용할 수 있다는 큰 제약이 있다. 이는 예측 모델이 훈련 데이터에 대한 의존성 때문에 다른 시스템에서 사용되기 어렵다는 모델의 범용성 실험 결과에서도 알 수 있다[8].

매우 많은 감독형 모델들이 제안되었지만 어느 모델이 가장 우수한 모델이라는 결론은 내려지지 않았다. 즉, 일반화된 모델은 존재하지 않는다. 왜냐하면, 각 훈련 알고리즘들의 성능은 훈련 데이터 집합의 특성에 크게 의존하기 때문이다. 그렇기 때문에 수많은 모델 중 어느 모델을 선택하는가도 성능에 큰 영향을 미치며, 모델 선정은 매우 어려운 문제이다. 이를 해결하기 위해 여러 모델에 대한 선택부터 결함 예측, 평가를 지원하는 결함 예측 프레임워크에 대한 연구들도 등장하였다[9, 10].

또 다른 최근 연구 방향으로 제한된 훈련 데이터 집합이 존재할 때 이를 이용하는 예측 모델에 관한 연구들이 있다. 즉 완전한 모델을 구축하기에는 충분하지 않은 적은 수의 라벨 데이터와 이에 비해 다수의 라벨이 없는 데

이터가 함께 존재하는 경우이다. 이런 경우에는 이 두 가지 형태의 데이터들을 사용하기 위해 세미 감독형 학습 기법이 사용된다. [11]에서는 초기 설정값에 큰 영향을 받는 K-means 클러스터링 기법의 문제를 해결하기 위해 라벨 있는 데이터를 활용하여 가장 적절한 초기 클러스터 위치를 지정해주는 기법을 사용한 모델을 제작하였다. 또한 이들은 불완전한 데이터를 처리하기에 적합한 EM 알고리즘 기반의 세미 감독형 모델을 제안하였다[12]. [13]에서는 두 단계를 거쳐 모듈 구축을 하는 YATSI(Yet Another Two Stage Idea) 알고리즘을 이용하여 나이브 베이지안을 제외하고는 감독형 모델보다 높은 성능을 보임을 밝혔다. [표 1]은 세미 감독형 모델 연구와 그 기법을 나타낸다.

표 1. 기존 세미 감독형 연구와 기법

연구	기법
[12], [14]	EM 기반의 세미 감독형 알고리즘
[13]	Immune 기반의 YATSI 알고리즘
[15]	FTF(Fitting the Fits)
[16]	ROCUS(RandOm Committee with Under-Sampling)
[17]	FTcT(Fitting the Confident Fits)
[18]	CoForest, ACoForest

세미 감독형 모델은 라벨이 존재하는 모듈을 통해 데이터 분포만을 이용하는 비감독형 모델보다 좀 더 정확하게 결함 여부를 알 수는 있지만 여전히 결함 여부를 아는 데이터가 존재해야 하며 이는 높은 비용을 필요로 한다. 또한, 세미 감독형 방법들은 높은 예측 성능을 내기 위해 특정한 가정이 필요하며 그에 따른 제약이 존재한다[15]. 예를 들어, 입력 데이터 집합이

완전하게 독립적인 2개 이상의 집합으로 나누어져야 하고 고밀도의 지역에서 연결된 점들은 같은 클래스에 속한다는 것을 가정해야 한다. 혹은 데이터 분포를 알고 있다 가정해야 한다는 한계가 있다.

감독형·세미 감독형을 사용할 수 없는 특수한 경우에 비감독형 모델이 필요하다. 비감독형 모델은 입력 데이터의 분석 과정 때문에 모델 구축이 어렵고, 예측 성능이 감독형 모델보다 떨어지기 때문에 높은 필요성에도 불구하고 극소수의 연구들만이 수행되었다. [19] 및 [20]에서는 각 클러스터의 대표 속성과 통계 정보를 토대로 결함경향성을 결정하여 Neural-Gas와 K-means 클러스터링 알고리즘의 성능을 비교하였다. 그 결과, 대체적으로 Neural-Gas보다 K-means가 낮은 오류율을 보였다. [21]은 비감독형 학습 신경망인 SOM(Self Organizing Map)을 이용한 모델을 제시하고, 모델의 예측 성능에 많이 사용되는 감독형 모델인 오류 역전파 신경망 모델과 비교하였으나 실제 데이터를 사용하여 검증하지 못했다는 문제점이 있다. [19-21]은 비감독형 모델의 시초가 되는 연구인 점에서 의미가 있지만 가장 좋은 클러스터링 결과를 얻기 위해 전문가에게 의존해야 하고 이를 통하여 클러스터 수를 인위적으로 설정해야 한다는 문제점이 있다.

이러한 문제를 해결하기 위해, 클러스터 성능에 큰 영향을 미치는 클러스터 수 설정을 최적화하기 위해 [22], [23]에서는 클러스터 수가 자동으로 결정되는 클러스터링 알고리즘을 사용하였다. [22]에서는 X-means 알고리즘을 사용한 모델이 클러스터 수를 설정해야 하는 Pure Metrics Thresholds methods, Fuzzy C-means, K-means를 사용한 모델들보다 나은 성능을 보임을 밝혔다. [23]에서는 K-means를 변형한 QDK(Quad Tree-based K-means)알고리즘을 이용하여 K-means의 문제 중 하나였던 불안정한 초기 값 할당 문제를 안정화시켰다. QDK는 DD(SAS2004), GM(Global K-means) 알고리즘보다 초기화 성능이 좋았다. 그러나 이 두

연구는 PROMISE 중 기존 연구에서 많이 사용되었던 NASA 데이터 집합을 사용하지 않고, 입력 모듈의 수가 상대적으로 적은 AR3, AR4, AR5 데이터 집합을 이용했다. 또한 클러스터 수를 직접 선정해야하는 알고리즘들만을 비교 대상으로 삼아, 본 연구에서는 클러스터 수를 자동으로 결정하는 알고리즘들을 서로 비교한다.

대표적인 클러스터링 방법들을 사용한 실험Ⅰ은 [19]를 비교 연구로, 자동으로 클러스터 수를 결정하는 클러스터링 방법을 사용한 실험Ⅱ는 [22], [23]을 비교 연구로 하여 구축 모델의 효용성을 알아본다.

Ⅲ. 모델 제작

1. 모델 제작 및 사용 시나리오

[그림 2]는 정제되지 않은 초기 원본 데이터 집합을 입력으로 하여 데이터 전처리 → 클러스터링 → 분석 과정을 거쳐 비감독형 모델을 구축하고 이를 이용하여 결함경향성을 예측하는 전체 프로세스를 나타낸 것이다.

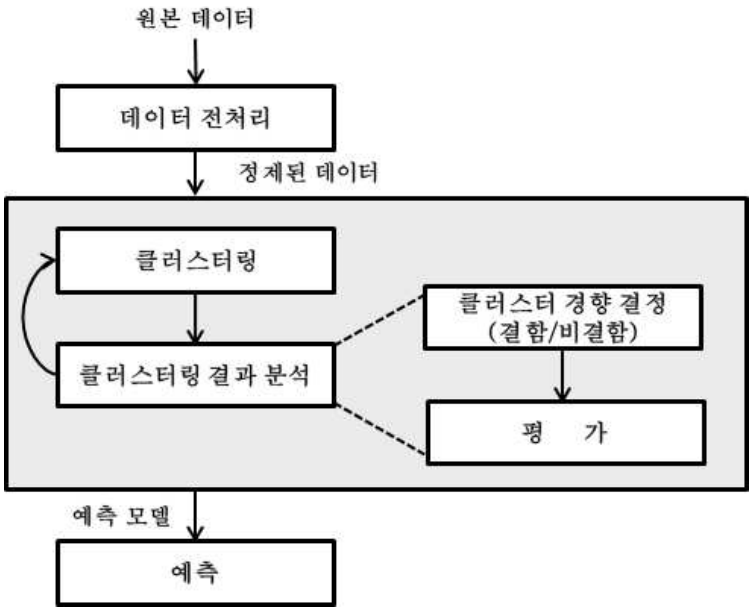

그림 2. 모델 구축 프로세스

클러스터링 알고리즘을 이용한 비감독형 모델 제작 시 사용하는 데이터 집합은 각 모듈에 대한 결함 유무를 나타내는 라벨이 없는 상태이다. 정제되지 않은 데이터를 정규화하고 차원을 축소하는 전처리 과정을 거친 후 클러스터링 단계에서 입력 데이터 집합에 대한 클러스터링이 이루어지며, 분

석 단계에서는 전문가가 클러스터링 결과 클러스터를 분석한다. 분석 결과로 결함경향 클러스터와 비결함경향 클러스터가 결정되며, 각 클러스터의 성질과 다른 성질을 갖는 모듈의 비율을 계산함으로써 모델 평가가 이루어진다. 보통 클러스터 수를 변화하여 좋은 결과가 나올 때까지 클러스터링-분석 단계(회색 영역)를 반복하여 진행한다.

1) 데이터 전처리 단계

개발 집단에서 예측하고자 하는 품질, 즉 결함경향성의 대상은 설계 명세나 초기 코드이고 예측을 위해서는 이들을 소프트웨어 메트릭 벡터 형태로 정량화해야 한다. 정량화된 초기 원본 데이터는 바로 모델의 입력으로 사용되지 않고 데이터 정제 작업을 하는 데이터 전처리 과정을 거치게 된다. 데이터 정제 작업이란, 데이터를 사용하기 전에 초기 데이터에 존재하는 문제점을 해결하여 문법적으로나 의미상으로 완전한 데이터 집합을 만드는 과정이다. 이는 모델의 예측 성능에 매우 큰 영향을 끼치므로 올바른 정제 작업은 매우 중요하다[24]. 단계는 모든 케이스가 같은 상수 속성 삭제, 같은 의미를 갖는 중복 속성 삭제, 데이터 값이 존재하지 않는 결측값 처리, 속성 간의 비정상적인 관계나 비정상적인 속성값을 갖지 않도록 하는 데이터 무결성 처리, 중복 또는 모순 케이스 처리 작업으로 구성된다. 전처리 과정에는 정제 작업 외에 모든 데이터 값을 [0, 1]로 바꾸어 상대적 가중치를 조정하는 정규화와 메트릭 벡터의 차원이 큰 경우 효율적인 수행을 위한 차원 축소 과정도 포함된다.

2) 클러스터링 단계

클러스터링 단계에서는 정제된 전체 입력 데이터 인스턴스들을 유사한 인

스턴스들끼리 클러스터로 묶어주는 작업이 수행된다. 모델 구축 프로세스의 예를 보여주는 [그림 3]에서 점은 하나의 데이터 인스턴스이며 하나의 모듈을 의미한다. 흰 점은 비결함경향 모듈을, 검은 점은 결함경향 모듈을 나타내며 [그림 3]에서는 클러스터링 후 모듈들이 5개의 클러스터로 클러스터링 된다.

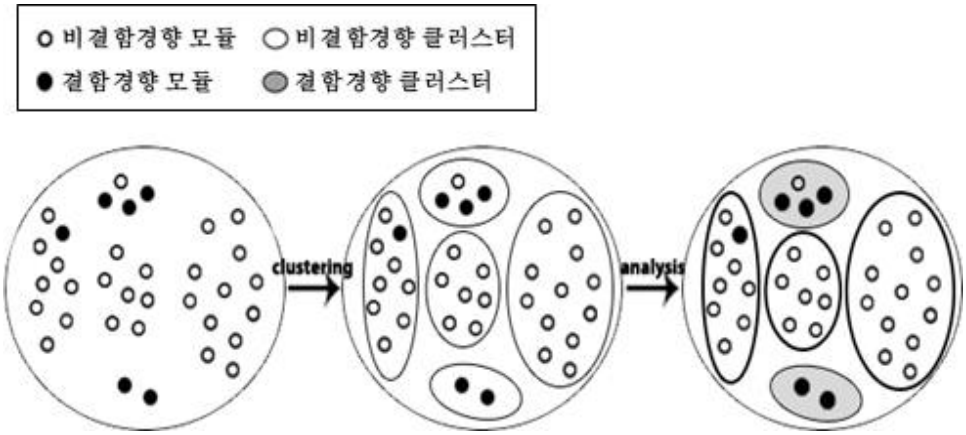

그림 3. 모델 제작 예

3) 분석 단계

분석 단계에서는 클러스터링 단계에서 결정된 클러스터들에 대해 전문가의 분석이 이루어진다. [그림 2]에서 분석 단계를 두 단계로 나눈 것처럼 분석 과정에서 결함경향 클러스터와 비결함경향 클러스터가 결정되며, 각 클러스터의 성질과 다른 성질을 갖는 모듈의 비율을 계산함으로써[4] 제작 모델의 평가가 이루어진다. 전문가의 판단은 결정론적인 알고리즘에 의한 것이 아니므로 주관적인 결과를 낳을 수 있지만 입력 데이터에 대한 라벨을 알 수 없으므로 클러스터의 해석을 자동화하는 것은 불가능하다. 따라서 기

4) 모듈의 성질을 사람이 판단하므로 입력 데이터 집합이 큰 경우 정확한 계산은 어렵지만, 개략적인 결과로 평가할 수 있었음.

존 비감독형 모델 연구들은 모두 전문가의 분석으로 이 단계를 수행했으며 이 점이 비감독형 모델의 가장 중요한 한계점이다. 이로 생기는 문제점을 줄이기 위해 전문가는 반드시 입력 데이터를 구성하는 소프트웨어 메트릭 전문가여야 하며, 두 사람 이상이 작업하여 서로 교차 검증을 하여야 한다. [그림 3]의 분석 단계에서는 두 개의 클러스터를 결함경향 클러스터로 결정하였으며 실제 이 클러스터들에 결함경향 모듈이 많이 있으므로 올바르게 분석하였음을 알 수 있다.

클러스터링 알고리즘은 대부분 클러스터 수가 자동으로 결정되지 않는다. 클러스터 수가 너무 작으면 패턴 분류가 되지 않아 결함경향 모듈들과 비결함경향 모듈들이 같은 클러스터에 속할 가능성이 있고, 클러스터 수가 너무 크면 사람의 분석 노력이 많이 든다. 이럴 경우 [그림 2]에 그려진 사이클과 같이 여러 번의 클러스터링-분석 단계를 거쳐[5] 가장 좋은 평가를 얻은 결과가 최종 완성된 모델이 된다.

그러나 이러한 과정을 반복하는 것은 비용과 시간이 많이 들기 때문에 실험Ⅱ에서는 가장 좋은 결과를 만드는 클러스터 수를 선택하기 위해 사람의 판단 없이 알고리즘 자체에서 클러스터 수를 결정하는 알고리즘을 사용하여 모델을 제작한다. 따라서 실험Ⅱ에서는 클러스터링 단계에서 분석 단계까지 반복하는 불필요한 영역을 수행할 필요 없이 적합한 개수를 결정할 수 있다. 전문가는 반복 없이 오직 한 번만 결함경향성 분석을 하게 된다.

4) 예측 단계

모델 제작 프로세스는 현재 입력 데이터 집합을 가지고 이루어지므로 완성 모델이 결정되면 입력 모듈의 예측도 끝나게 된다. 결함경향 클러스터에 속하는 모듈들은 결함경향 모듈로 비결함경향 클러스터에 속하는 모듈들은

5) 보통은 클러스터 수를 변화하면서 반복함.

비결함경향 모듈로 예측된 것이다. [그림 3]의 예를 보면 두 개의 결함경향 클러스터 중 위에 있는 클러스터에는 세 개의 결함경향 모듈과 한 개의 비결함경향 모듈이 존재하는데 비결함경향 모듈은 결함경향 모듈로 예측되므로 이는 모델의 예측 오류가 된다.

가끔 완성된 모델이 새로운 입력 모듈의 결함 예측에 사용되어야 하는 경우가 생긴다. 매우 큰 입력 데이터 집합의 부분 집합을 가지고 모델을 제작한 경우나 소프트웨어 진화 프로세스에서 이전 릴리즈를 사용하여 완성된 모델에 다음 릴리즈의 입력 데이터를 사용해야 하는 경우 등이 이에 해당한다. 이 같은 경우 완성된 모델에 새로운 입력 모듈이 들어오면 가장 가까운 클러스터의 성질에 따라 입력 모듈의 결함경향성이 결정된다. [그림 4]에서 각 입력 모듈은 가장 가까운 클러스터의 결함경향성으로 결정된다. 즉, 세 개의 입력 모듈 중 위, 아래 두 개의 모듈은 결함경향 모듈, 다른 하나는 비결함경향 모듈로 예측된다.

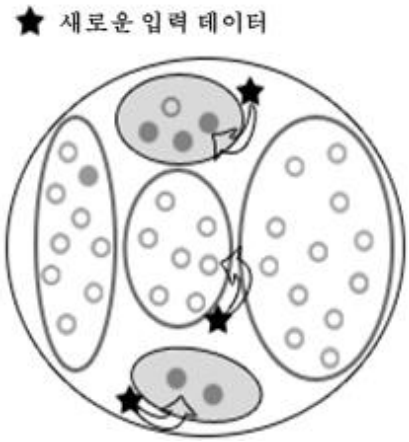

그림 4. 새로운 입력 모듈 예측

2. 클러스터링 알고리즘

본 논문의 목적은 일반적으로 많이 사용되는 대표적인 클러스터링 알고리즘 중 기존 결함 예측 모델 연구에서 사용하지 않았던 알고리즘 및 클러스터 수를 자동으로 결정하는 알고리즘들을 사용한 모델을 제작해 평가하는 것이다. 클러스터링 알고리즘은 입력 데이터의 분포만을 이용하는 비결함 예측 모델 연구 핵심 내용이다.

클러스터는 유사한 특성을 가진 인스턴스들의 집단이다. 즉, 두 개의 인스턴스가 같은 클러스터가 아니라면, 이 인스턴스들은 상이한 특성을 갖게 된다. 클러스터링은 이러한 성질을 갖는 클러스터들을 만들어 가는 과정을 말한다. [그림 5]는 클러스터링 알고리즘들을 분류한 것이다.

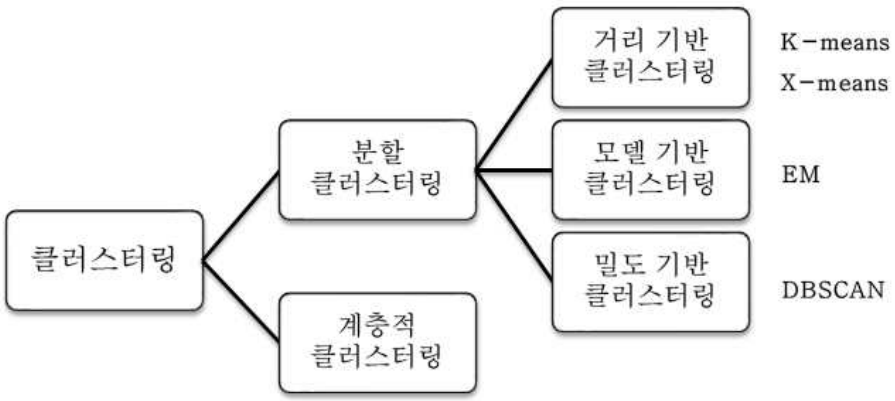

그림 5. 클러스터링 알고리즘 분류

일반적으로 클러스터링 기법은 계층 클러스터링과 분할 클러스터링 두 가지로 구분한다. 분할 클러스터링은 다시 세부적으로 거리 기반, 밀도 기반, 모델 기반의 알고리즘 등으로 구분할 수 있다[25]. 분류의 오른쪽에 쓰여 있는 알고리즘은 각 분류의 대표적인 클러스터링 알고리즘을 나타낸다. 거

리 기반에는 K-means와 X-means, 모델 기반에는 EM, 밀도 기반 클러스터링에는 DBSCAN이 대표적이다[26].

1) 계층 클러스터링

계층 클러스터링은 크게 응집분석 클러스터링과 분할분석 클러스터링으로 나눌 수 있다. 응집분석 클러스터링은 인스턴스를 각각 하나의 클러스터로 간주하고 인스턴스의 특성이 가까운 클러스터끼리 차례로 합해나가는 방법이고, 분할분석 클러스터링은 모든 인스턴스를 하나의 클러스터로 보고 차례로 인스턴스의 특성이 먼 것끼리 나누는 방법이다[27].

2) 분할 클러스터링

계층 클러스터링과 구분하여 비계층 클러스터링이라고도 하며, 각 클러스터가 적어도 하나의 인스턴스를 포함하도록 전체 인스턴스를 지정한 클러스터 수만큼의 클러스터로 나누는 것이다. 대표적인 분할 클러스터링으로는 K-means, DBSCAN, EM 알고리즘이 있다.

■ K-means 클러스터링

K-means는 주어진 인스턴스를 지정된 수만큼의 클러스터로 묶는 알고리즘으로, 그 수만큼의 초기 중심점을 지정한 뒤 중심점에서 가장 근접한 인스턴스 항목을 할당하면서 시작한다. 할당된 모든 클러스터의 평균 위치로 중심점을 이동 후 재할당 하게 되는데, 클러스터와 인스턴스 항목에 변화가 없거나 최대 반복 횟수에 도달할 때까지 반복한다.

K-means는 가장 간단하면서 효과적인 성능을 보이기 때문에 많이 사용된다. 그러나 전역 최적해에 수렴하지 않고 지역 최적해에 수렴하므로 초기

중심점에 민감하며 클러스터 수를 미리 설정해야 하는 한계점이 있다.

■ DBSCAN 클러스터링

K-means가 거리 기반을 사용하는 것에 반해 DBSCAN(Density-Based Spatial Clustering of Applications with Noise)은 밀도 개념을 사용하여 클러스터링을 수행한다. 클러스터의 밀도 결정하기 위해 2개의 매개변수, 즉 점의 이웃 범위를 나타내는 반경과 최소 이웃의 수가 있어야 한다. 인스턴스들의 일정 반경 내에 최소 이웃의 수를 만족하여 어느 정도의 밀도를 형성하게 되면 인스턴스들을 연결하는 방식으로 클러스터링을 수행한다.

DBSCAN은 밀도 기반 정의를 이용하기 때문에 잡음에 강하고 불규칙한 모양과 크기의 클러스터링이 가능하다. 그러나 고차원의 데이터에서는 반경 산출이 어렵다. 다차원의 데이터뿐만 아니라 고밀도 데이터에서는 군집화가 어렵다.

■ EM 클러스터링

EM(Expectation Maximization)은 보이지 않는 변수에 의존하는 확률 모델에서 모수들의 최대우도 추정치를 찾고자 하는 알고리즘이다. 즉, EM은 라벨 없는 인스턴스를 결측 정보로 간주하고 이를 추정된 값으로 대체함으로써 라벨이 있는 훈련 데이터처럼 사용하게 된다. EM 알고리즘은 임의로 설정된 클러스터 수만큼 클러스터를 초기화하는 것에서부터 시작한다. 이때, 각 클러스터는 평균과 분산을 가지고 있다. 기대(E) 단계에서는 각 인스턴스가 클러스터에 포함될 확률을 계산한다. 최대화(M) 단계에서는 클러스터의 우도를 최대화하기 위해 분포의 모수를 구하며 기대-최대화 단계를 우도의 변화가 적을 때까지 반복한다.

EM 역시 K-means와 마찬가지로 초기 값에 상당히 민감하다. 혼합 성분의 개수 역시 미리 지정해야 하지만 교차 검증을 통해 알고리즘 자체적으로 결정할 수도 있다.

■ X-means 클러스터링

X-means는 K-means를 확장한 버전으로 하나의 클러스터에서 시작하여 반복적으로 클러스터를 분할해 나가는 하향식 알고리즘이다. 이 때, 분할을 위한 기준으로 BIC(Bayesian Information Criterion) 점수를 사용한다. X-means 알고리즘은 초기 K를 정해주면 K-means 클러스터링을 수행한 후, 각 클러스터를 두 개로 나누는 작업을 반복한다. 이때 분할 전후의 BIC 점수를 비교하여 분할 후의 BIC 점수가 개선되지 않으면 분할을 멈춘다. 더 나눌 클러스터가 없으면 최종 클러스터를 반환한다.

X-means는 알고리즘이 스스로 클러스터의 개수를 최적화한다는 장점을 가지며 K-means 알고리즘보다 효율적이며 좋은 성능을 보인다[3]. 그러나 BIC 점수를 계산해야 하므로 K-means보다 속도가 느리다.

Ⅳ. 모델 성능 실험

1. 실험 환경

1) 데이터 집합 및 속성 선정

실험에는 결함 관련 연구에 가장 많이 사용된 공개 데이터 집합 중 [19]의 모델과 비교하기 위해 NASA 데이터를, [22], [23]의 모델과 비교하기 위해 AR* 데이터를 사용하였다[5]. 실험Ⅰ에서는 PROMISE 초기 버전을 사용하였으며, 데이터 집합을 구성하는 여러 프로젝트 중에서 결측값이 적고, 모듈 수가 많으며 결함에 관련된 속성들의 모호성이 적은 JM1을 선택하였다[28]. JM1 프로젝트는 총 10,885개의 모듈로 구성된 C언어로 구현된 프로젝트이다. 그중 결함 모듈은 2,106개, 비결함 모듈은 8,779개로 결함 모듈이 전체에서 대략 20%정도를 차지한다. 속성은 Cyclomatic Complexcity로 대표되는 McCabe 기반 메트릭들, Halstead Volume(v), Halstead Length(l) 등과 같은 Halstead 기반 메트릭들과 코드 사이즈 메트릭들을 포함한 21개의 속성으로 구성된다.

실험Ⅱ에서는 터키 White-Goods 제조사 출처의 AR3, AR4, AR5[6] PROMISE 데이터 집합을 사용한다. 각 데이터 집합은 JM1의 메트릭에서 *ESSENTIAL_COMPLEXITY, HALSTEAD_CONTENT*를 제외하고 10개가 더 추가된 29개 메트릭을 가진다. AR3는 55개의 비결함 모듈과 8개의 결함 모듈로 63개의 모듈로 구성되어 있으며 AR4는 87개의 비결함 모듈과 20개의 결함 모듈로 107개의 모듈을, AR5는 28개의 비결함 모듈과 8개의

6) PROMISE DATA. http://promisedata.org

결함 모듈로 총 36개의 모듈로 구성되어있다.

표 2. 모델 입력 메트릭

프로젝트명		메트릭
AR* (29)	JM1 (21)	**LOC_TOTAL**, ***LOC_CODE_AND_COMMENT***, **LOC_COMMENTS**, **CYCLOMATIC_COMPLEXITY**, **ESSENTIAL_COMPLEXITY**, **DESIGN_COMPLEXITY**, **LOC_BLANK**, BRANCH_COUNT, *NUM_UNIQUE_OPERANDS*, *HALSTEAD_LEVEL*, NUM_OPERANDS, NUM_UNIQUE_OPERATORS, NUM_OPERANDS, LOC_EXECUTABLE, HALSTEAD_CONTENT, HALSTEAD_LENGTH, HALSTEAD_VOLUME, HALSTEAD_DIFFICULTY, HALSTEAD_EFFORT, HALSTEAD_PROG_TIME, HALSTEAD_ERROR_EST.
	HALSTEAD_VOCABULARY, DECISION_COUNT, FORMAL_PARAMETERS, CONDITION_COUNT, *MULTIPLE_CONDITION_COUNT*, CALL_PAIRS, *CYCLOMATIC_DENSITY*, *DECISION_DENSITY*, *DESIGN_DENSITY*, NORMALIZED_CYCLOMATIC_COMPLEXITY.	

전처리 과정에서 속성 선정을 위해 정규화한 데이터 집합에 가능한 모든 속성의 조합을 고려하여 최적의 조합을 찾아내는 CFS(Correlation-based Feature Selection) 속성 선정 기법을 적용하였다. 이는 클러스터링 알고리즘을 효율적으로 수행하기 위해 입력 데이터의 차원을 축소하기 위한 과정이다[29]. 차원 축소 없이 전체 속성을 사용하는 NotReduction 버전과 CFS 기법을 통해 차원을 축소한 Reduction 버전 두 가지 경우를 실험에 사용하였다. 전체 속성들로 구성된 메트릭 벡터는 [표 2]와 같고 차원 축소 결과 선정된 JM1 속성은 굵은 글씨(8개)로, AR* 속성은 이탤릭체(6개)로 나타내었다.

2) 사용 클러스터링 알고리즘

본 논문에서는 클러스터링을 위해 WEKA[7] 데이터 마이닝 도구를 사용하였다. [그림 5]와 같이 클러스터링 분류에 속하는 대표적인 알고리즘을 분류 당 하나씩 선정하였다. 즉, 분할 클러스터링 중 거리기반 알고리즘에서는 K-means, 밀도 기반 알고리즘에서는 DBSCAN, 모델 기반 알고리즘에서는 EM 알고리즘을 선정하였다. 계층 클러스터링은 여러 고사양 PC 플랫폼에서 실험하였지만 적절한 학습 결과를 얻지 못하여 실험에서 제외하였다. 2장에서 기술한 비감독형 모델 중 실제 프로젝트 데이터 집합을 사용한 [19], [22], [23]에서는 모두 K-means 또는 이를 변형한 알고리즘을 사용한 모델이므로 본 논문에서는 K-means와 비교해 DBSCAN과 EM을 사용한 모델이 어느 정도의 예측 성능을 보이는지를 실험한다. 또한, 클러스터 수를 입력하지 않는 클러스터링 알고리즘으로 X-means, EM만을 실험에 사용하였는데, 클러스터 개수를 미리 지정하지 않는 클러스터링 알고리즘들(CLOPE, CobWeb, DBSCAN, OPTICS 등)은 클러스터 수를 직접 지정하지 않아도 그 수에 많은 영향을 미치는 다른 파라미터를 선택해야하기 때문이다. 이 값들을 선택하는 것은 클러스터 수를 휴리스틱 하게 선택하는 것과 다를 바 없으며 연구의 목적과 맞지 않다. 따라서 X-means와 EM 알고리즘을 사용하여 [22], [23]의 모델과 비교한다.

EM은 생성하고자 하는 클러스터 수를 지정할 수도 있고, 교차 검증을 이용해 알고리즘 자체적으로 결정할 수도 있으므로 실험Ⅰ, Ⅱ에서 모두 사용한다. 편의를 위해 직접 클러스터 수를 선정하는 EM은 H-EM(Heuristic EM)으로, 알고리즘 자체적으로 수를 결정하는 EM은 A-EM(Automatic EM)으로 표기한다.

---

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

K-means와 H-EM 알고리즘의 실행을 위해서는 클러스터 수(K)를, DBSCAN은 사용자 정의 거리(E, Epsilon)와 범위 내의 최소한 이웃해야 하는 점의 수(MP, Minimum Points)를 설정해야 한다. 주어진 값들을 어떻게 설정하느냐에 따라 클러스터링 결과는 완전히 달라진다. K-means, H-EM의 K를 2부터 최대 20개로 설정하였으며, 이들과의 비교를 위해 DBSCAN의 클러스터 수도 2개 이상 20개 이하로 클러스터링 되도록 E와 MP를 설정하였다. 그 결과 E가 0.01, 0.03, 0.06인 경우와 E 각각에 대해 MP가 10, 50, 100인 경우를 실험하였다. 실험에 사용하는 4가지 클러스터링 알고리즘 모두 클러스터 수 이외의 구조 및 초기 설정값은 적절하게 설정되어 있는 WEKA의 기본값[30]을 이용하였다.

3) 분석 단계의 자동화

평가 실험에서 실험의 용이성을 위하여 분석 단계 중 클러스터링 결과를 결함경향 클러스터와 비결함경향 클러스터로 결정하는 부분을 자동화하였다. 자동화가 가능한 이유는 비감독형 모델이 사용하는 일반적인 입력 데이터 집합과는 달리 JM1 및 AR* 프로젝트는 결함여부 라벨이 존재하는 데이터 집합이기 때문이다.

[표 3]은 결함경향 클러스터를 자동으로 결정하는 알고리즘을 기술한 것이다. 알고리즘에서 사용한 변수와 함수들의 정의는 다음과 같다- RC(Remaining Clusters)는 WHILE 루프를 거치면서 처리되지 않은 현재 클러스터 집합을, FPC(Fault-Prone Clusters)는 결정된 결함경향 클러스터 집합을 의미한다. maxFR(Maximum Fault-Prone Module Rate)은 알고리즘 사용자가 정하는 값으로 전체 모듈에 대한 결함경향 모듈의 비율의 상한값을 의미하며, 본 논문의 실험에서는 20%로 하였다. 따라서 알고리즘에 의해 선정되는 결함경향 모듈은 전체 모듈의 20% 내에서 결정된다. R_{CS}는

전체 모듈에 대한 클러스터 집합 CS(Cluster Set)에 속한 모듈의 비율이며, $r_{fpm}(c)$와 $r_{nfpm}(c)$는 클러스터 c에 속하는 모듈 중 결함경향 모듈과 비결함경향 모듈의 비율을 반환한다. 마지막으로 maxNfp(CS)는 클러스터 집합 CS에서 r_{fpm}이 가장 큰 클러스터를 반환하며, CS가 ∅인 경우는 null을 반환한다.

표 3. 결함경향성 클러스터를 결정하는 알고리즘

```
RC ← all clusters
FPC ← ∅
maxc ← maxNfp(RC)

WHILE RC !=∅ AND r_fpm(maxc) ≥ r_nfpm(maxc) DO
   IF R_{FPC∪{maxc}} ≤ maxFR THEN
      FPC ← FPC U {maxc}
   ENDIF
   RC ← RC − {maxc}
   maxc ← maxNfp(RC)
ENDWHILE
```

[표 3]의 알고리즘은 결함경향 클러스터 집합인 FPC를 구하는 것이 목적이다. 현재 RC에서 r_{fpm}이 가장 큰 클러스터를 maxc로 하고, maxc가 FPC에 들어갈 수 있는지를 검사하여 조건을 만족하면 FPC에 포함하는 것이 WHILE 루프의 한 단계이다. FPC에 들어갈 수 있는 조건은 maxc 내의 결함경향 모듈 비율이 비결함경향 모듈 비율보다 높고, maxc가 포함된 FPC의 전체 모듈에 대한 비율이 maxFR보다 낮아야 한다는 것이다. 루프는 더 이상 고려 대상인 클러스터가 없거나, 선정된 maxc의 비결함경향 모듈 비

율이 결함경향 모듈 비율보다 커지면 종료된다. [표 4]는 알고리즘이 [그림 3]의 클러스터들에 적용된 결과를 나타낸다. #은 클러스터의 모듈 수를, #nfpm과 #fpm은 클러스터의 비결함경향 모듈 수와 결함경향 모듈 수를 의미한다. 처음 maxc는 r_{fpm}이 가장 큰 클러스터 3이 되며, R_3은 6%로 maxFR 값인 20%보다 작으므로 클러스터 3은 FPC에 들어간다. 루프의 다음 단계의 maxc는 클러스터 1이 되며 R_1은 11%로, 기존 FPC에 들어가도 R_{FPC}는 17%로 maxFR보다 작으므로 클러스터 1도 FPC에 들어간다. 그다음 maxc는 클러스터 0이 되는데 $r_{fpm}(0)<r_{nfpm}(0)$이므로 클러스터 0은 FPC에 들어가지 못하고 WHILE 루프는 끝난다. 결과적으로 결함경향 클러스터 결과는 클러스터 3과 1로 [그림 3]의 그림 결과와 일치한다.

표 4. 결함경향성 클러스터를 결정하는 예

Cluster(K=5)			nfpm		fpm	
c	#	R_C	#nfpm	$r_{nfpm}(c)$	#fpm	$r_{fpm}(c)$
0	9	26%	8	89%	1	11%
1	4	11%	1	25%	3	75%
2	7	20%	7	100%	0	0%
3	2	6%	0	0%	2	100%
4	12	34%	12	100%	0	0%

4) 평가 척도 및 방법

실험은 일반적인 10겹 교차 검증을 사용하며 실험의 정확도를 높이기 위해 이를 10회 반복하였다. 즉, 훈련 데이터 집합과 검증 데이터 집합을 바

꾸면서 10회씩 10회 반복하여 한 모델 당 100회 실행하였다.

예측 모델의 성능 평가 수행을 위해 많은 평가 척도가 사용되고 있으나 본 논문에서는 [19], [22], [23]에서 사용한 TER(Total Error Rate), FPR(False Positive Rate), FNR(False Negative Rate)을 사용한다. TER은 전체 모듈 중에서 잘못 예측한 모듈의 비율이다. FPR은 비결함경향 모듈을 결함경향 모듈로 예측하는 오류율을, FNR은 결함경향 모듈을 비결함경향 모듈로 예측하는 오류율을 뜻한다. 결함경향이 있는 모듈을 비결함 모듈로 예측하는 것은 소프트웨어 품질에 심각한 영향을 주기 때문에 FNR이 FPR보다 신중히 살펴야 할 오류 척도이다. 아래 [표 5]는 결함 예측을 통해 네 가지의 결과를 나타낸 오차 행렬(Confusion Matrix)이고 수식 (1), (2), (3)은 각각 TER, FPR, FNR을 나타낸다.

표 5. Confusion Matrix

	Predicted Class		
Actual Class		Fault-Prone	Non Fault-Prone
	Fault-Prone	True Positive(TP)	False Negative(FN)
	Non Fault-Prone	False Positive(FP)	True Negative(TN)

$$TER = \frac{FP + FN}{TP + FP + FN + TN} \quad (1)$$

$$FPR = \frac{FP}{FP + TN} \quad (2)$$

$$FNR = \frac{FN}{TP + FN} \quad (3)$$

2. 실험 과정

표 6. JM1-Reduction에 대한 EM 결과 및 결함경향성

클러스터		실제 클러스터		클러스터 분석				경향
		개체 개수		결함		비결함		
4개	0	6915	64%	825	12%	6090	88%	NFP
	1	709	7%	333	47%	376	53%	FP
	2	3096	28%	852	28%	2244	72%	NFP
	3	165	2%	96	58%	69	42%	FP
5개	0	6596	61%	766	12%	5830	88%	NFP
	1	3244	30%	832	26%	2412	74%	NFP
	2	369	3%	167	45%	202	55%	FP
	3	587	5%	276	47%	311	53%	FP
	4	89	1%	65	73%	24	27%	FP
6개	0	3188	29%	766	24%	2422	76%	NFP
	1	393	4%	176	45%	217	55%	FP
	2	58	1%	43	74%	15	26%	FP
	3	136	1%	78	57%	58	43%	FP
	4	707	6%	307	43%	400	57%	FP
	5	6403	59%	736	11%	5667	89%	NFP
7개	0	5990	55%	662	11%	5328	89%	NFP
	1	521	5%	211	40%	310	60%	FP
	2	102	1%	55	54%	47	46%	FP
	3	266	2%	133	50%	133	50%	FP
	4	40	0%	29	73%	11	28%	FP
	5	3124	29%	694	22%	2430	78%	NFP
	6	842	8%	321	38%	521	62%	FP

[표 6]은 JM1-Reduction 데이터를 입력으로 하는 EM 알고리즘 수행 결과와 클러스터 예측을 위한 클러스터 경향을 나타낸 표이다. 클러스터 수

를 2개부터 20개까지 다양하게 바꾸면서 클러스터를 분석한다. [표 6]은 4개에서 7개까지로 일부분을 나타내었다. 클러스터의 수가 4개일 때는 3번 클러스터에 속한 결함 모듈 비율이 58%로 가장 높으므로 3번 클러스터를 FPC로 라벨링한다. 이때, 결함 클러스터의 비율은 전체의 2%가 된다. 1번 클러스터가 전체에서 7%를 차지하고, 47%로 2번째로 높은 결함 모듈 비율을 가지므로 FPC로 라벨링 한다. 그다음으로 높은 2번 클러스터는 전체 모듈의 28%이므로 FP에 포함할 수 없다. 따라서 FP 경향을 보이는 클러스터는 1번과 3번 클러스터가 된다.

표 7. JM1-Reduction에 대한 EM 수행 분석 결과

K	Confusion Matrix				평가 척도		
	TP	FP	FN	TN	TER(%)	FNR(%)	FPR(%)
4	430	1677	444	8334	19.5	79.6	8.0
5	809	1597	236	8243	16.8	75.8	9.6
6	605	1501	689	8090	20.1	71.3	11.9
7	750	1356	1021	7758	21.8	64.4	16.3

[표 7]은 [표 6]에 대한 Confusion Matrix와 분석 결과이다. 5개의 군집으로 클러스터링될 때 정확도가 가장 높으며 클러스터 수 K가 7개일 때 FNR이 가장 낮았다.

이와 같이 클러스터 결과에 대해 결함경향성 여부를 판단하고 직접 라벨과 비교해보면서 클러스터링을 평가하는 작업을 반복해나간다. 실험 I 에서는 K-means, EM, DBSCAN 3개 알고리즘에 대해 전체 속성을 사용하는 NotReduction 버전과 CFS로 속성 차원을 축소한 Reduction 버전을 입력으로 하여 총 6가지 경우를 실험한다. 클러스터 수를 2개에서 20개로 변경해가며 실험하므로 경우마다 19번의 클러스터링 및 클러스터링 결과 분석

([그림 2]의 회색 영역)을 반복해야 한다. 그러나 실험Ⅱ에서는 자동으로 클러스터 수가 결정되기 때문에 간단하게 한 번만 수행하면 된다.

3. 실험 결과

실험Ⅰ, Ⅱ 모두 CFS를 사용하여 차원 축소를 한 Reduction의 경우가 전체 속성을 사용한 NotReduction을 사용한 모델의 성능이 더 좋으므로 실험 결과는 CFS를 사용하여 차원 축소한 경우의 결과만을 나타내었다. TER은 모델의 전체적인 예측 성능을 나타내는 값이지만 FNR이 상대적으로 FPR보다 중요하므로 TER이 낮아도 FNR이 높으면 좋은 성능을 보인다고 할 수는 없다. 즉, 모델 평가는 TER의 관점과 FNR의 관점 모두에서 이루어져야 한다.

1) 실험Ⅰ 결과

[표 8]과 [표 9]는 K-means와 H-EM의 실행 결과 중 의미 있는 성능을 보인 것들을 나타낸 것이다. K-means의 TER과 FPR은 클러스터 수 K가 4일 때 가장 낮았고, FNR은 K가 11일 때 가장 낮았다. 이를 NASA의 JM1을 사용한 [19]의 실험 결과[8]와 비교해 보았을 때 K가 4인 경우는 TER과 FPR이 더 좋은 대신 FNR이 좋지 않았고, K가 11인 경우는 FPR이 조금 좋지 않았지만, 나머지는 비슷한 결과를 보였다. 이 같은 차이는 [19]에서 평가 실험에 사용한 JM1은 NASA 데이터 원본을 정제한 것으로 본 실험에서 사용한 PROMISE의 JM1과 다른 데이터 정제 작업을 거쳤기

8) [19]의 결과는 그래프로 나타나 있어 정확한 수치를 판단하기 어렵지만 대략 (TER, FPR, FNR)은 (23, 13, 62)로 판독됨.

때문이라 판단된다.

K-means와 마찬가지로 [표 9]의 H-EM 역시 K가 4인 경우 가장 좋은 TER과 FPR값을 보였지만 FNR은 극도로 나쁜 결과를 보였다. FNR은 K가 13인 경우에 가장 좋았지만 K가 6인 경우와 큰 차이는 없었다. 전체적으로 H-EM이 K-means에 비해 TER 면에서는 좋은 결과를 보이고, FNR 면에서는 좋은 결과를 보이지 못했다. 하지만 그 차이는 매우 미미하므로 본 실험 결과로는 K-means와 H-EM 중 어느 것이 유의미하게 좋은 성능을 보인다고 할 수는 없다.

표 8. K-means 알고리즘을 사용한 모델 결과

K	TER	FPR	FNR
4	19.6	4	84.66
11	25.71	17.33	60.64
13	23.2	12.6	67.38
15	23.32	13.46	64.39

표 9. H-EM 알고리즘을 사용한 모델 결과

K	TER	FPR	FNR
4	19.29	0	99.57
6	20.34	17.09	61.68
11	20.5	6.74	77.84
13	25.83	17.8	59.31

DBSCAN은 K-means, H-EM과 달리 클러스터 수를 설정하지 않고 클러스터 반경을 나타내는 E와 하나의 클러스터를 이루는 데 필요한 최소한

의 점의 개수를 나타내는 MP의 값에 따라 클러스터 수가 결정된다. 이 역시 클러스터 수를 결정해야 하는 것과 같이 전문가의 분석이 필요하므로 자동으로 클러스터링이 되는 클러스터 알고리즘이라 할 수 없다. 따라서 DBSCAN은 휴리스틱 알고리즘으로 분류하여 실험Ⅰ에서 K-means와 H-EM과 함께 비교한다. [표 10]은 E가 0.01, 0.03, 0.06일 때 MP가 10, 50, 100인 9가지 경우의 결과이다. 9번의 클러스터링 실험 모두에서 클러스터링이 되지 않은 모듈들이 존재하였는데, 그러한 모듈들은 잡음으로 처리하여 하나의 클러스터로 할당하여 실험 결과를 내었다. TER이 가장 낮은 결과는 21.4이지만 전체적인 TER의 값은 K-means, H-EM보다 높다. FNR의 가장 낮은 결과는 63.2로 역시 K-means, EM의 가장 낮은 결과보다 높지만, 전체적으로 볼 때는 안 좋다고 보기는 어렵다.

표 10. DBSCAN 알고리즘을 사용한 모델 결과

MP \ E		0.01	0.03	0.06
10	TER	27.8	21.4	26.2
	FPR	13.4	9.31	14.9
	FNR	88.1	71.6	73.5
50	TER	33.4	23.3	26.3
	FPR	22.1	13.1	16
	FNR	85.0	65.8	69.5
100	TER	69.1	25.6	22.1
	FPR	5.22	16.6	7.7
	FNR	84.4	63.2	82.3

다만 전체적으로 균일한 결과를 보이지 않는데, 이는 연속적인 수치인 밀

도를 이산화했기 때문이다. 또한 DBSCAN이 약간 낮은 성능을 보이는 것은 클러스터링이 안 된 부분들이 존재하기 때문이라 판단된다.

2) 실험Ⅱ 결과

표 11. X-means와 A-EM 모델의 결과

Model	Datasets	K	TER	FPR	FNR
X-means	AR3	3	9.52	7.27	25
	AR4	4	13.08	2.3	60
	AR5	2	13.89	14.9	12.5
A-EM	AR3	5	12.7	10.91	25
	AR4	7	13.08	2.3	60
	AR5	5	16.67	14.29	25

[표 11]은 X-means와 A-EM 모델의 결과를 나타낸다. 클러스터 수 K는 [표 9], [표 10]과 달리 전문가가 결정한 것이 아니라 클러스터링 알고리즘 자체에서 자동으로 결정된 수이다. [표 11]에서 보듯이 X-means의 경우에 총 오류율이 10 안팎으로 높은 정확도를 보인다. FNR은 데이터마다 차이가 존재하지만, 실험 I 에서보다 월등히 좋은 결과를 내었다.

[표 12]는 [22]에서 이용한 X-means[22], [23]에서 이용한 QDK를 모델의 성능 실험 결과를 요약한 것이다. X-means[22], QDK는 AR4를 제외한 AR3, AR5 데이터 집합에서 거의 비슷한 예측 성능을 보인다. AR4 데이터 집합에서 QDK는 TER 측면에서 좋은 성능을 보였고 X-means는 FNR 측면에서 좋은 성능을 보였다.

표 12. 기존 연구 X-means[22]와 QDK 모델의 성능

	X-means[22]			QDK		
데이터 집합	TER	FPR	FNR	TER	FPR	FNR
AR3	33.33	34.55	25	33.33	34.54	25
AR4	37.38	44.83	5	12.14	4.59	45
AR5	13.89	14.29	12.5	13.88	14.28	12.5

[그림 6], [그림 7]은 X-means, A-EM, X-means[22], QDK의 모델 결과를 함께 비교한 결과이다. 본 연구의 X-means와 X-means[22]는 같은 클러스터링 알고리즘을 사용하지만, 입력 데이터 정제 과정과 분석 단계의 자동화 부분에서 차이가 난다. X-means[22]은 분석 자동화 단계를 거치지 않으며, 29개의 메트릭 중 LoC, CC, UOp, UOpndn, TOp, TOpnd의 6가지 메트릭만 사용한다.

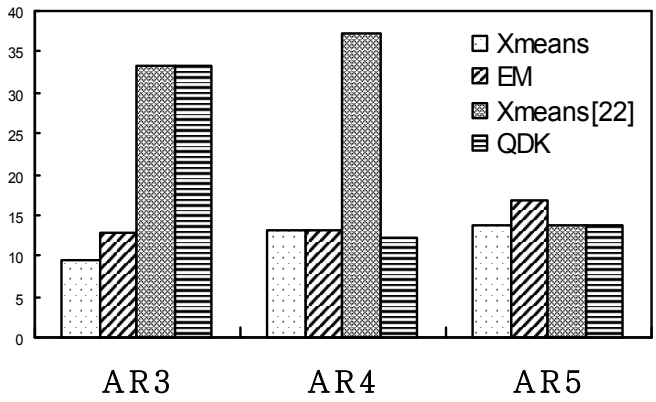

그림 6. 실험Ⅱ 클러스터링 결과 (TER)

[그림 6]은 TER 측면에서 모델들의 결과이다. AR3에서는 X-means의 결과가 X-means[22], QDK의 성능보다 뛰어나며 A-EM 알고리즘보다

조금 더 나은 성능을 보임을 알 수 있다. AR4에서는 X-means[22]는 TER은 매우 높은 데 비해 FNR은 매우 낮다. 이를 제외한 나머지 알고리즘은 비슷한 결과를 보인다. 또한, AR5는 EM의 성능이 가장 낮게 평가된다. 따라서 X-means의 성능은 X-means[22]와 QDK에 비해 매우 좋으며 A-EM과 비교하면 조금 더 좋거나 비슷하다고 할 수 있다.

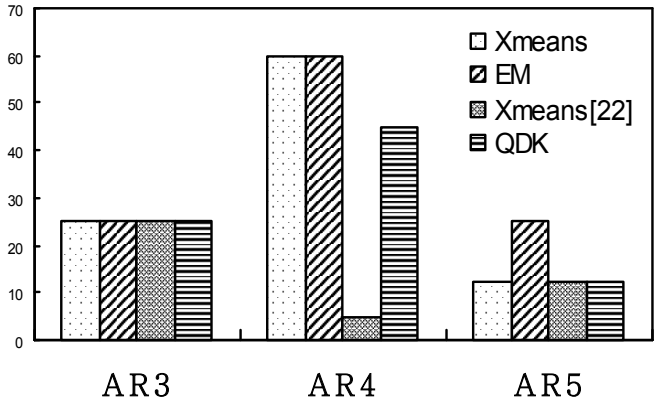

그림 7. 실험Ⅱ 클러스터링 결과 (FNR)

[그림 7]은 FNR 측면에서 모델들의 결과를 나타낸다. AR3에서는 모두 같은 결과를 보였다. AR4에서는 QDK가 X-means와 A-EM보다 좋았고 AR5에서는 A-EM이 가장 좋지 않았으며 나머지는 비슷한 결과를 보였다. 전체적으로 X-means가 AR4 데이터 집합에서 FNR 성능을 제외하고는 X-means[22], A-EM, QDK보다 좋은 성능을 보인다고 할 수 있다.

Ⅴ. 결론 및 향후 연구

지난 수십 년간 매우 많은 연구가 결함 예측 분야에서 수행되었지만 대부분 감독형 모델에 관한 연구이다. 하지만 결함 예측 모델의 중요성이 점점 증가하면서 비감독형 모델의 필요성이 대두되고 있다. 대부분의 개발 집단들은 훈련 데이터 집합을 보유하고 있지 않아 모델의 훈련 과정에 이들을 필요로 하는 감독형 모델을 사용할 수 없기 때문이다. 그러나 모델 제작의 어려움 등의 이유로 인하여 비감독형 모델에 관한 연구들은 극소수에 불과하다. 모델을 제작하기 어려운 이유 중 하나는 대부분의 클러스터링 알고리즘은 예측 모델의 성능에 많은 영향을 미치는 클러스터 수가 휴리스틱 하게 선택되어야 하기 때문이다.

본 논문은 감독형 모델의 문제점을 해결하기 위해 대표적인 클러스터링 알고리즘인 K-means, EM, DBSCAN 및 클러스터 수를 직접 설정함으로써 야기되었던 어려움을 해결하기 위해 알고리즘 자체적으로 클러스터 수를 결정하는 X-means, EM을 이용한 비감독형 소프트웨어 품질 예측 모델을 제작하여 기존 연구들에서 사용한 모델과 성능을 비교하였다. 그 결과 전체 오류율 측면에서는 H-EM이 K-means보다 아주 약간 나은 결과를 보였으며 DBSCAN은 두 모델보다 좋지 않은 결과를 보였다. 또 다른 중요한 오류 측정치인 FNR 측면에서는 어느 모델이 낫다고 할 수 없었다. 또한, 전체적으로 X-means와 A-EM의 알고리즘은 거의 비슷한 성능을 보이며 FNR 측면에서는 X-means의 성능이 A-EM의 성능보다 조금 더 좋거나 같은 결과를 보였다.

결과적으로, 기존 연구들이 사용한 K-means 대신 EM이나 X-means 모델을 사용할 수 있으며 클러스터링 및 분석 단계를 반복하는 대신 자동으

로 클러스터 수가 결정되는 X-means 혹은 EM 모델을 사용하면 더욱 쉽게 비감독형 모델을 구축할 수 있다는 결론을 얻었다.

본 논문의 의의는 다음과 같다. 첫째, 많이 사용하는 클러스터링 기법이지만 소프트웨어 결함 예측 모델에서는 적용되지 않았던 대표적인 클러스터링 알고리즘을 사용하여 모델을 제작하고 성능 평가를 통해 효용성을 보였다. 둘째, 클러스터 수를 자동으로 결정하는 알고리즘을 사용하여 비감독형 모델을 구축하는 데 있어 가장 많은 비용과 노력이 드는 클러스터링 및 분석 단계의 반복을 줄여 예측 모델의 효율적인 수행이 가능하다는 점을 보였다. 마지막으로 임의로 만들어진 데이터 집합이 아닌 실제 PROMISE 데이터 저장소의 데이터들을 이용하여 결함 예측의 일반적인 결론을 검증했다.

소프트웨어 결함 예측 모델을 통해 소프트웨어 프로세스에서 구현이 완료된 후에 나타날 문제를 미리 예측하여 찾아냄으로써 적절한 자원 할당, 테스트 프로세스 개선, 최적의 리팩토링 후보 결정 등을 통하여 고품질의 높은 신뢰성을 갖춘 시스템 구축 등을 가능하게 할 것이다.

향후 연구로는 결함경향성 이외의 유지 보수성 등의 품질 인자를 예측하는 품질 예측 모델들을 제작하여, 고품질의 소프트웨어 제작이 가능하도록 할 것이다. 또한, 세미 감독형 알고리즘, 앙상블 알고리즘을 이용한 품질 예측 모델을 제작하여 이들 모델들과 기존의 감독형, 비감독형 모델들의 성능을 비교하는 실험을 통해 예측 성능을 향상할 수 있도록 할 것이다.

참고 문헌

[1] ISO/IEC 9126-1: Software engineering-product quality-part 1: Quality model. (2001). Geneva, Switzerland: International Organization for Standardization.

[2] Ebert, C. (1996). Fuzzy classification for software criticality analysis. Expert Systems with Applications, 11(3), 323-342.

[3] Catal, C. (2011). Software fault prediction: A literature review and current trends. Expert Systems with Applications, 38(4), 4626-4636.

[4] Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). A systematic literature review on fault prediction performance in software engineering. Software Engineering, IEEE Transactions on, 38(6), 1276-1304.

[5] Malhotra, R. (2015). A systematic review of machine learning techniques for software fault prediction. Applied Soft Computing, 27, 504-518.

[6] Gondra, I. (2008). Applying machine learning to software fault-proneness prediction. Journal of Systems and Software, 81(2), 186-195.

[7] M. K., Park, & E. S., Hong. (2014). Software fault prediction model using clustering algorithms determining the number of clusters automatically. International Journal of Software Engineering & its Applications, 8(7).

[8] T. Y., Kim, Y. K., Kim, & H. S., Chae. (2009). An experimental study of generality of software defects prediction models based on object oriented metrics. The KIPS Transactions: PartD, 16(3), 407-416.

[9] Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. Software Engineering, IEEE Transactions on, 34(4), 485-496.

[10] Song, Q., Jia, Z., Shepperd, M., Ying, S., & Liu, S. Y. J. (2011). A general software defect-proneness prediction framework. Software Engineering, IEEE Transactions on, 37(3), 356-370.

[11] Seliya, N., & Khoshgoftaar, T. M. (2007). Software quality analysis of unlabeled program modules with semisupervised clustering. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 37(2), 201-211.

[12] Seliya, N., & Khoshgoftaar, T. M. (2007). Software quality estimation with limited fault data: A semi-supervised learning perspective. Software Quality Journal, 15(3), 327-344.

[13] Catal, C., & Diri, B. (2009). Unlabelled extra data do not always mean extra performance for semi-supervised fault prediction. Expert Systems, 26(5), 458-471.

[14] Seliya, N., Khoshgoftaar, T. M., & Zhong, S. (2004). Semi-supervised learning for software quality estimation. Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on, 183-190.

[15] Lu, H., Cukic, B., & Culp, M. (2011). An iterative semi-supervised approach to software fault prediction. Proceedings of the 7th International Conference on Predictive Models in Software Engineering, 15.
[16] Jiang, Y., Li, M., & Zhou, Z. (2011). Software defect detection with ROCUS. Journal of Computer Science and Technology, 26(2), 328-342.
[17] Lu, H., Cukic, B., & Culp, M. (2014). A semi-supervised approach to software defect prediction. Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual, 416-425.
[18] Li, M., Zhang, H., Wu, R., & Zhou, Z. (2012). Sample-based software defect prediction with active and semi-supervised learning. Automated Software Engineering, 19(2), 201-230.
[19] Zhong, S., Khoshgoftaar, T. M., & Seliya, N. (2004). Analyzing software measurement data with clustering techniques. Intelligent Systems, IEEE, 19(2), 20-27.
[20] Zhong, S., Khoshgoftaar, T. M., & Seliya, N. (2004). Unsupervised learning for expert-based software quality estimation. Hase, 149-155.
[21] E. S., Hong. (2003). A software quality prediction model without training data set. The KIPS Transactions: PartD, 10(4), 689-696.
[22] Catal, C., Sevim, U., & Diri, B. (2009). Software fault prediction of unlabeled program modules. Proceedings of the World

Congress on Engineering, , 1 1-3.
[23] Bishnu, P. S., & Bhattacherjee, V. (2012). Software fault prediction using quad tree-based K-means clustering algorithm. Knowledge and Data Engineering, IEEE Transactions on, 24(6), 1146-1150.
[24] Gray, D., Bowes, D., Davey, N., Sun, Y., & Christianson, B. (2012). Reflections on the NASA MDP data sets. Software, IET, 6(6), 549-558.
[25] Jain, A. K., & Dubes, R. C. (1988). Algorithms for clustering data Prentice-Hall, Inc.
[26] E. S., Hong, & M. K., Park. (2014). Unsupervised learning model for fault prediction using representative clustering algorithms. KIPS Transactions on Software and Data Engineering, 3(2), 57-64.
[27] Galit, S., Nitin, R. P., & Peter, C. B. (2011). Data mining for business intelligence.
[28] E. S., Hong. (2013). Ambiguity analysis of defectiveness in NASA MDP data sets. Journal of the Korea Society of IT Services, 12(2), 361-371.
[29] Hall, M. A. (1999). Correlation-based feature selection for machine learning. The University of Waikato.
[30] Witten, I. H., & Frank, E. (2005). Data mining: Practical machine learning tools and techniques Morgan Kaufmann.

ABSTRACT

Unsupervised Quality Prediction Model Using Clustering Algorithms

Park Mikyeong
Dept. of Computer Science
The Graduate School of
Sungshin Women's University

As the software industry advances, improvement and evaluation methods of software development process are under the spotlight. In order to ensure a high quality of software, it is needed to select the core part in initial process and place the limited resources properly. Therefore, software quality prediction model is becoming more important.

Most previous studies of software fault prediction model which determines the fault-proneness of input modules have focused on supervised learning model using training data set. However, unsupervised learning model is needed in case supervised learning model cannot be applied: either past training data set is not present or even though there exists data set, current project type is changed. Building an unsupervised learning model is extremely

difficult that is why only a few studies exist. One of the difficulties is to decide the number of clusters. In this paper, we build unsupervised models using representative clustering algorithms, EM and DBSCAN, that have not been used in prior studies and compare these models with the previous model using K-means algorithm. Also, to solve the problem of selecting the number of clusters, we build unsupervised models using clustering algorithms, EM and X-means, which determine the number of clusters automatically and compare them with results of earlier studies.