JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 25, No. 12, pp. 3663-3682, Dec. 2024

Check for updates

머신러닝 기반 약물 이상사례(KARES DB)분석을 통한 인체 장기 장애 예측

권 혜 준[1] · 김 종 윤[2*] · 최 승 호[3*]
[1]이화여자대학교 휴먼바이오기계공학부 학사
[2]동덕여자대학교 약학과 조교수
[3]한성대학교 기초교양학부 조교수

Machine Learning-Based Adverse Drug Event (KARES DB) Analysis to Predict Human Organ Dysfunction

Hae-Jun Kwon[1] · Jongyoon Kim[2*] · Seoung-Ho Choi[3*]

[1]Graduate student, College of Engineering, Department of Mechanical and Biomedical Engineering, Ewha Womans University, Seoul 03760, Korea
[2]Assistant Professor, College of Pharmacy, Dongduk Women's University, Seoul 02748, Korea
[3]Assistant Professor, College of Liberal Arts Faculty of Basic Liberal Art, Hansung University, Seoul 02876, Korea

[요 약]

사람이 약을 복용하면 약에 대한 부작용으로 인해 인체에 장기 기능 장애가 발생한다. 이러한 인체 내 약물 부작용으로 인한 장기 기능 장애 현상은 정상 상태에 비해 불균형한 상태로 데이터를 생성한다. 불균형 데이터를 이용하여 분석할 때, 머신러닝 모델에서는 클래스 불균형이라는 문제가 발생한다. 인체의 장기 기능 장애를 분석하기 위한 기계 학습 방법을 최초로 제안한다. 8개의 머신러닝 모델을 사용하여 분석한다. 두 가지 신체 질환에 대해 데이터 샘플링을 수행한 횟수와 샘플링 적용 여부에 따른 머신러닝 모델의 성능을 분석한다. 데이터 샘플링을 수행하지 않은 경우에 비해 데이터 샘플링을 적용한 경우 머신러닝 모델의 성능이 향상되는 것을 확인했다. 제안한 방법이 장기 기능 장애 발생 예측이 개선됨을 확인했다.

[Abstract]

When taking medication, organ dysfunction can occur in the human body due to adverse reactions to the drug. This organ dysfunction phenomenon generates data in an unbalanced state compared to the normal state. When analyzing unbalanced data using a machine learning model, the class imbalance problem occurs. We propose a machine learning method for analyzing organ dysfunction in the human body for the first time. We considered eight machine learning models and analyzed their performance according to the amount of data sampling and whether sampling was applied for two body diseases. We confirmed that the performance of the machine learning model improved when data sampling was applied. We confirmed that the proposed method improves the prediction of organ dysfunction occurrence.

색인어 : 인체 내 장기, 머신러닝, 클래스 불균형, KAERS DB, 약물 이상사례
Keyword : Human Organs, Machine Learning, Class Imbalance, KAERS DB, Adverse Drug Cases

http://dx.doi.org/10.9728/dcs.2024.25.12.3663

Received 29 September 2024; **Revised** 11 November 2023
Accepted 27 November 2024

***Corresponding Author; Jongyoon Kim, Seoung-Ho Choi**
Tel: ※ 개인정보 표시제한
E-mail: jyoonkim@dongduk.ac.kr, jcn99250@naver.com

Ⅰ. 서 론

약물 이상 사례는 환자의 건강에 심각한 영향을 미칠 수 있는 중요한 의료 문제 중 하나이다. 약물 이상사례는 예상치 못한 반응으로 인해 약물 치료의 목적을 방해하며, 심각한 경우 입원 연장, 추가적인 치료, 심지어 사망에도 이를 수 있다. 이러한 약물 이상 사례는 환자뿐만 아니라 의료 시스템 전체에도 막대한 비용을 초래할 수 있다. 따라서 약물 이상사례를 조기에 예측하고 관리하는 것은 환자 안전과 의료 비용 절감에 중요한 역할을 수행한다. 이러한 약물 이상 사례에 대해서 사전에 아는 것은 중요하며 약물 이상 사례 중에서도 인체 내 장기의 장애 현상을 분석하는 것은 더욱 중요하다[1].

사람의 장기 장애 현상에서 소화기와 신장에 대해서 분석하는 것은 특히 약물 흡수와 배설 과정에서 중요한 역할을 수행하는 장기이기 때문에 매우 중요한 중요성을 가지고 있다[2]. 또한 장애 현상을 분석할 때에는 발생 빈도가 적기 때문에 불균형한 상태로 발생되는 경우가 많다. 머신러닝 모델을 이용하여 분석을 진행할 때에는 빈도가 높은 데이터에 대해서만 학습을 잘하기 때문에 빈도가 불균형한 상태로 발생된 데이터에 대해서는 모델의 성능이 잘 나오지 않는다는 문제가 있다. 즉, 많은 데이터에 대해서만 학습이 잘 이뤄지게 되는 문제점이 존재한다[3]. 따라서 약물 이상 사례에서 드물게 발생하는, 즉 발생 빈도가 적은 인체 내 장기의 장애 현상을 머신러닝 성능의 개선을 통하여 높은 정밀도를 지니게끔 분석할 필요성이 있다고 판단했다. 이에 기존의 통계적 기법 및 머신러닝 모델들의 한계를 극복하기 위해서 다양한 샘플링 기법을 적용하여 불균형 데이터를 효과적으로 처리하고, 예측 성능을 향상시키는 방법을 분석한다. 이어지는 제안 방법 섹션에서는 우리가 사용한 데이터와 제안된 머신러닝 모델에 대해서 구체적으로 설명한다. 본 논문의 공헌도는 아래와 같다.

처음으로 인체 내 장기의 장애 현상을 분석하는 머신러닝 방법을 제안한다.

장기의 장애 현상이 불균형하게 발생함을 고려하여, 이를 개선하기 위한 머신러닝 분석 방법을 제안한다. 이를 위해서 본 연구에서는 자발적 약물 이상사례보고시스템인 한국의약품부작용보고원시자료 시스템 데이터베이스(Korea Adverse Event Reporting System Database, KAERS DB)를 사용한다.

8가지 머신러닝 방법을 이용해서 성능을 분석한다. 머신러닝 모델의 평가 지표로는 Accuracy, F1score, Precision, Recall, 그리고 ROC AUC (Receiver and Operator Curve Area Under the Curve)를 이용하여 분석한다.

Ⅱ. 머신러닝 기반 약물 이상 사례 예측 연구

약물 이상사례 예측을 위한 연구는 오랜 기간 동안 지속되어 왔다. 초기의 연구들은 주로 통계적 모델링과 규칙 기반 시스템을 이용한 예측 방법에 초점을 맞추었다. 이러한 접근법은 특정 약물 또는 질환에 대한 예측에 유용할 수 있었으나, 복잡한 환자 데이터의 다양한 상호작용을 충분히 반영하지 못했다는 한계가 있다.

전통적인 통계적 접근법은 로지스틱 회귀(logistic regression)와 같은 회귀 모델을 주로 사용하였다. 이러한 모델은 특정 변수들이 약물 이상사례 발생에 미치는 영향을 평가하는 데 유용하였으나, 복잡한 비선형 관계를 처리하는 데 한계가 있다. Bate et al.[4]은 통계적 기법을 사용하여 약물 부작용을 분석하였으나, 복잡한 다변량 관계를 설명하는 데 어려움을 겪었다. 이는 특히 약물 간 상호작용이나 환자의 다변량 임상 특성들을 포함하는 상황에서 더 두드러졌다. 이러한 한계는 약물 이상사례 예측 모델의 성능을 제약하며, 실질적인 임상 적용에 어려움을 겪게 한다.

최근에는 머신러닝 기법이 이러한 한계를 극복하기 위한 대안으로 등장했다. Schuemie et al.[5]은 Electronics Health Records(EHR) 데이터를 활용하여 머신러닝 모델을 통해 약물 이상 사례를 예측하는 연구를 수행하였으며, 이러한 모델들이 통계적 모델보다 우수한 예측 성능을 보인다는 것을 입증했다. 특히, 앙상블 기법(예: Random Forest 그리고 AdaBoost)은 복잡한 데이터 구조를 효과적으로 처리할 수 있으며, 다양한 변수 간의 상호작용을 반영하여 예측 성능을 높일 수 있는 것으로 보고했다.

Liu et al.[6]은 약물-단백질 상호작용 정보를 활용하여 약물-약물 상호작용으로 인한 이상 약물 반응(Adverse Drug Reaction, ADR)을 예측하는 모델을 개발했다. 이 연구에서는 약물-단백질 상호작용 정보를 기반으로 약 800개의 약물에 대한 상호작용 프로파일을 구축하여 약물 조합이 ADR을 유발할 가능성을 평가했다. 그 결과 약물-약물 상호작용으로 인한 ADR을 예측하는 모델의 평균 정확도는 89%였으며, 현재 임상적으로 보고되지 않은 잠재적인 ADR도 예측할 수 있는 것으로 나타났다. 이 연구는 약물 상호작용으로 인한 ADR을 예측하고 이를 최소화할 수 있는 잠재적 약물 조합을 제시함으로써 임상 약물 사용의 안전성을 높이는 데 기여할 수 있음을 보여주었다.

Scholl et al.[7]의 연구에서는 약물 부작용(ADR)을 예측하기 위한 새로운 통계 모델을 개발하여 기존 모델보다 향상된 예측 성능을 달성했다. 이 연구는 자발적 보고 시스템에서 수집한 ADR 데이터를 기반으로 통계적 신호 감지 성능을 개선하기 위한 예측 모델을 개발하였다. 주요 예측 인자로는 보고 건수, 불균형성(Reporting Odds Ratio at the 2.5% lower confidence limit, ROR025), 나란조 점수, 제조업체 보고율(Marketing Authorization Holder Reporting Rate, MAH), 의료 전문가 보고율(Healthcare Professional Reporting Rate, HCP)이 포함되었다. 이 모델은 새로운 신호를 감지하는 데 도움이 되도록 약물과 ADR 간의 연관성을 우선시하였으며, 이전 모델에 비해 상위 800개 신호에서 신

호 감지율이 58.2% 증가하는 결과를 보였다. 또한 해당 모델은 이전 모델보다 더 높은 AUC score (Area Under the Curve, 0.740)로 예측 성능이 향상되는 것으로 나타났다. Scholl et al.[7]의 연구는 ADR 예측 모델에 다양한 변수를 통합하여 신호 탐지 효율을 높일 수 있음을 보여주며, 다른 국가나 데이터베이스에 적용할 수 있음을 시사한다. 이 연구는 기존의 통계적 접근법을 보완하고, 머신러닝 기법을 활용하여 약물-약물 상호작용으로 인한 ADR을 보다 정교하게 예측하는 데 기여할 수 있다는 점에서 유용한 참고 자료가 될 수 있다.

심층 신경망(Deep Neural Networks, DNN)과 같은 딥러닝 기법은 비선형적인 데이터 패턴을 학습하는 데 강점을 보였다. Miotto et al.[8]은 DNN을 사용하여 대규모 EHR 데이터를 분석하였으며, 이 모델이 기존의 통계적 기법보다 약물 이상사례를 더 잘 예측할 수 있음을 보여주었다. 그러나 이러한 모델들은 종종 많은 데이터와 계산 자원을 요구한다. Qureshi et al.[9]의 연구는 인공지능을 활용하여 약물 개발 및 약물 이상사례 예측에서 임상적 적용 가능성을 높이는 데 초점을 맞추고 있다. 이 연구는 다양한 머신러닝 기법을 통합하여 약물의 안전성을 평가하고, 약물 부작용 발생 가능성을 예측하는 데 기여하고 있으며, 특히 신약 개발 과정에서 약물 부작용 예측 모델의 활용 가능성을 제시하고 있다.

Ⅲ. 약물 이상 사례에서 클래스 불균형 문제를 해결하기 위한 제안 방법

그림 1은 제안된 모델에서 사용하는 샘플링 기법에 대해 시각화 한 그림이다. 제안된 방법에서는 샘플링 횟수(수행하지 않은 건, 1회 수행한 건, 5회 수행한 건)와 사용된 모델의 종류(RandomOverSampler, BorderlineSMOTE, SMOTE, 그리고 SVMSMOTE) 에 따라 결과를 비교하게 된다.

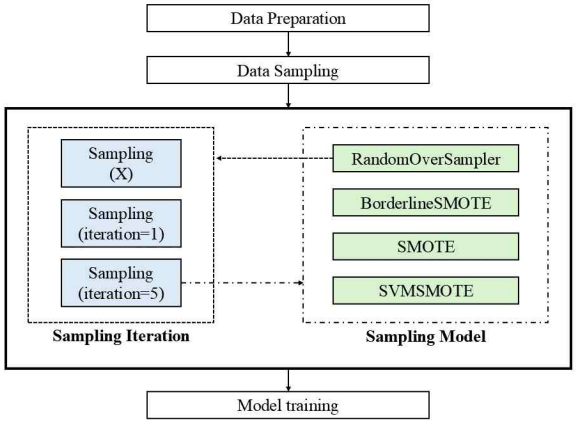


그림 1. 제안된 모델의 샘플링 과정 구성도
Fig. 1. Block-diagram of sampling process of proposed model

그림 2는 전체 시스템의 구성을 시각화 한 것이다. 순서는 크게 데이터 전처리와 모델 학습 및 성능 시각화로 나눠지며, 자세한 과정은 다음과 같다.

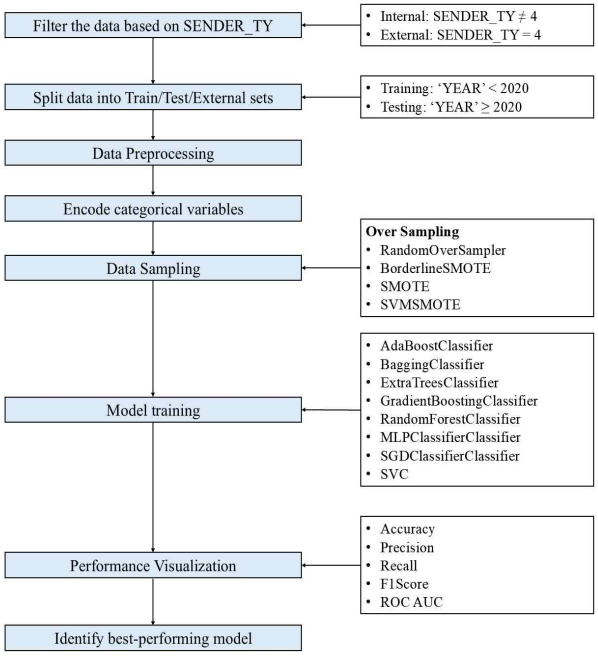


그림 2. 전체 시스템 구성도
Fig. 2. Overall system block-diagram

먼저 표 1은 논문에서 사용된 용어를 정의하는 표이다. 이는 '0600' 그리고 '1300'이 무엇을 의미하는지 나타낸다. 표 2는 실험에서 사용된 변수들의 특징을 설명하는 표이다. 이 표는 사용된 변수들의 정보와 데이터 형식에 대한 정보를 제공하여준다.

표 1. 사용 용어 색인 표
Table 1. Index of terms used

Term	Definition
0600	Refers to the disease code set as the target Label(1). For 0600, it represents a case for digestive disorder data.
1300	Refers to the disease code set to the target Label(1). 1300 indicates a case for kidney disorder data.

표 2. 사용된 변수들의 특징
Table 2. Characteristics of variables used

Varianles	Information	Type
DOSAGE_ROUTE_ID	An identifier indicating the route of administration of the medication	str
INGR_CD	The drug's ingredient code.	str
PTNT_OCCURSYMT_AGE	A variable representing the age at which the adverse drug event or symptom occurred.	int
PTNT_SEX	Patient sex	str
SENDER_TY	A variable indicating the type of entity that sent the data.	str

머신 러닝을 수행하기 전, 먼저 사용할 데이터에 대한 전처리를 수행한다. 먼저, KAERS DB 데이터를 사용할 열을 선택한 뒤 결측치를 0으로 대체한다. 이후 'INGR_CD' 변수를 기준으로 약물을 4개의 그룹(group1 ~ group4)으로 분류하고, 분류되지 않은 데이터는 제거한다. 약물 투여 시작일과 종료일을 날짜 형식으로 변환한 후, 두 날짜의 차이를 계산하여 약물 투여 기간을 의미하는 새로운 'DOSAGE PERIOD' 열을 생성한다. 그 후 'WHOART ARRN'과 'WHOART SEQ' 값을 매칭하여 새로운 'WHOART soc1' 열을 생성한다. 추가적으로, 머신 러닝 수행을 위하여 'INGR CD', 'PTNT SEX', 'DOSAGE ROUTE ID', 그리고 'EFFICACY MEDDRA ENG NM' 등의 범주형 데이터를 숫자로 매핑한다. 이후에는 'SENDER TY' 값이 4인 데이터를 외부 검증 데이터로 분리하고, 나머지 데이터를 내부 검증 데이터로 나눠 데이터셋을 생성한다. 머신 러닝 학습 시 사용할 데이터의 생성을 위하여 연도를 뜻하는 'YEAR' 값에 따라 내부 데이터를 학습용 데이터(2018-2019)와 테스트용 데이터(2020-2022)로 분리한다. 이때, 특정 인체 내 장기 장애에 대해 분석하기 위해서 'WHOART soc1' 값을 기준으로 특정 질병 코드에 해당하는 경우 1, 아닌 경우 0으로 설정하여 타겟 Label을 정의한다. 우리가 제안한 방법에서는 소화기 장애를 의미하는 '0600'과 신장 장애를 의미하는 '1300'의 질환 코드를 이용한다. 따라서 Label 1에 해당하는 값이 '1300'인 경우, 그리고 '0600'인 경우 총 두 가지에 대해서 머신러닝이 수행된다. 마지막으로, 타겟 변수('WHOART soc1')를 제거한 후, Train, Internal validation, 그리고 External validation dataset을 구성하여 머신 러닝을 위한 준비를 마친다.

다음으로, 앞선 전처리 과정에서 획득한 Train, Internal validation, 그리고 External validation 데이터를 이용하여 머신 러닝을 수행한다. 이때 사용한 모델은 Ada Boost Classifier[10], Bagging Classifier [11], Extra Trees Classifier [12], Gradient Boosting Classifier [13], Random Forest Classifier [14], MLP Classifier [15], SGD Classifier [16], 그리고 SVC [17]의 8개의 분류기 모델이다. 학습을 수행할 때에 있어서 샘플링을 이용하여 불균형 데이터를 처리한다. 사용된 샘플링 기법은 Over Sampling이며 모델은 RandomOverSampler, BorderlineSMOTE, SMOTE, 그리고 SVMSMOTE의 4종이다. 이때, 샘플링 실행 여부와 그 횟수에 따라 모델 성능을 비교하기 위하여 조건을 설정한다. 첫 번째로는, 샘플링 수행 횟수에 따른 성능을 비교하기 위해 동일하게 RandomOverSampler 모델을 사용하여 샘플링을 수행하지 않은 건, 샘플링을 1회 수행한 건, 샘플링을 5회 수행한 건에 대한 결과를 비교한다. 두 번째로는, 샘플링 모델 종류에 따른 결과를 비교하기 위하여 앞서 언급한 4종의 샘플링 모델을 이용하여 훈련 데이터를 샘플링한 뒤 모델의 예측 성능을 비교한다. 모델 성능의 비교분석은 Accuracy, Precision, Recall, F1 Score, ROC AUC의 평가 지표를 시각화하여 수행한다.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

$$\mathrm{Precision} = \frac{TP}{TP+FP} \tag{2}$$

$$Recall = \frac{TP}{TP+FN} \tag{3}$$

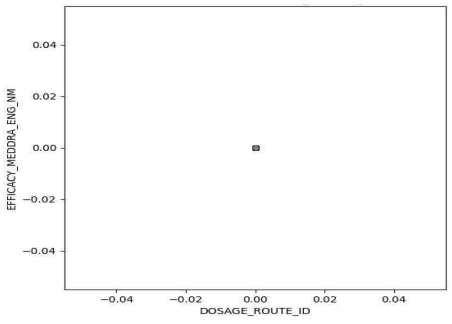
$$F1score = 2 \times \frac{\mathrm{Precision} \times Recall}{\mathrm{Precision} + Recall} \tag{4}$$

결과 분석에 사용된 평가 지표는 수식 (1)부터 (4)까지에 나와 있다. TP (True Positive)는 모델에서 실제 참인데 분류 모델 예측 결과 참으로 예측한 경우를, TN (True Negative)는 실제로 거짓인 경우를 정확히 거짓으로 예측한 경우를, FP (False Positive)는 실제로 거짓인데 참으로 예측한 경우를, FN (False Negative)는 실제로 True인데 예측은 거짓이라 판단한 경우를 뜻한다. 수식 (1)은 양수 및 음수 모델 예측에서 계산된 Accuracy 값을 설명한다. Accuracy가 높을수록 샘플을 분류하는 모델의 전반적인 성능이 더 우수하다는 것을 나타낸다. 수식 (2)는 분류 모델에 의한 양성 예측의 정확도를 정량화하는 지표인 Precision 값에 대해 설명하며, 이 값이 높을수록 오탐 예측이 적다는 것을 뜻한다. 수식 (3)은 Recall에 대해 설명하며, 이는 데이터 세트의 진양성 샘플 중 진양성 예측의 백분율로 계산된다. 이는 정탐률의 민감도를 나타내며, Recall이 높을수록 오탐 예측이 적다는 것을 나타낸다. 수식 (4)는 F1 score에 대한 식이며, 이는 Precision과 Recall로 계산된다. 이는 Precision과 Recall을 동시에 고려하는 분류 모델의 성능을 측정하는 척도를 제공한다. F1 점수가 높을수록 정확도와 회수율 모두에서 모델의 전반적인 성능이 더 우수하다는 것을 나타낸다.

Ⅳ. 실험 결과

4-1 Correlation Analysis Results

그림 3은 Label 1, 즉 '1300' 및 '0600'의 신장 장애와 소화기 장애에 대한 데이터와 'DOSAGE_ROUTE_ID',



(a)

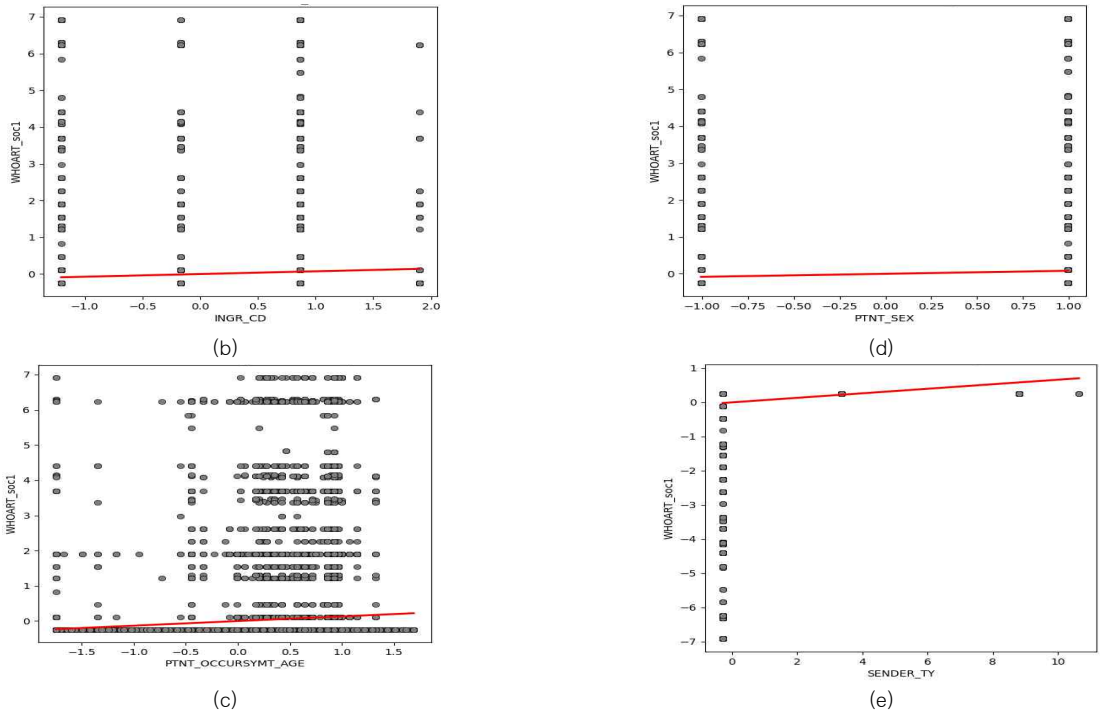


그림 3. 샘플링을 적용하지 않은 '1300' 및 '0600' 데이터의 CCA scatter plot: (a) 'DOSAGE_ROUTE_ID', (b) 'INGR_CD', (c) 'PTNT_OCCURSYMT_AGE', (d) 'PTNT_SEX', 그리고 (e) 'SENDER_TY'

Fig. 3. Visualization of CCA scatter plot for '1300' and '0600' data without sampling, (a) 'DOSAGE_ROUTE_ID', (b) 'INGR_CD', (c) 'PTNT_OCCURSYMT_AGE', (d) 'PTNT_SEX', and (e) 'SENDER_TY'

'INGR_CD', 'PTNT_OCCURSYMT_AGE', 'PTNT_SEX', 그리고 'SENDER_TY'의 특징들에 대한 scattor plot을 시각화 한 그림이다. 그림 3에 (a)의 'DOSAGE_ROUTE_ID' 산점도 그래프에서는 모든 데이터가 동일한 위치에 집중되어 있어, 'DOSAGE_ROUTE_ID' 변수에서의 값이 거의 동일하거나 제한된 범위 내에서만 분포하고 있음을 확인했다. 또한, 그림 3에 (b)의 'INGR_CD'의 산점도 그래프에서는 데이터가 산재하여 다른 변수 간의 관계가 명확하지 않거나, 데이터가 불균형하게 분포되어 있음을 확인했다. 그림 3에 (c)의 'PTNT_OCCURSYMT_AGE'의 산점도 그래프에서는 환자 발생 증상 연령의 분포가 제한적임을 확인했다. 또한, 그림 3에 (d)의 'PTNT_SEX'의 산점도 그래프에서는 성별에 따른 데이터 분포가 거의 균일하게 나타나며, 성별이 다른 변수와의 관계에서 큰 영향을 미치지 않음을 확인했다. 마지막으로, 그림 3에 (e)의 'SENDER_TY'의 산점도 그래프에서는 'SENDER_TY' 변수에 따라 데이터가 일부 증가하는 경향을 보임을 확인했다.

표 3은 Label 1에 대한 데이터와 각 특징들 간의 상관 분석을 수행한 결과를 나타낸다. Pearson correlation, Spearman correlation, Kendall correlation 총 3 종을 이용하여 'DOSAGE_ROUTE_ID', 'INGR_CD', 'PTNT_OCCURSYMT_AGE', 'PTNT_SEX', 그리고 'SENDER_TY'의 특징들에 대한 상관 분석을 수행한 결과, 각 feature가 Label에 미치는 영향을 확인했다. 'DOSAGE_ROUTE_ID'의 경우 상관계수와 p-value값이 nan으로 나타나며 결측치로 인해 상관 분석이 제대로 수행되지 않음을 확인했다. 'INGR_CD'의 경우 Pearson, Spearman, 그리고 Kendall 상관 계수 모두에 대해서 상관관계가 거의 없는 것으로 나타났다. 'PTNT_SEX'과 'SENDER_TY'에서도 역시 동일하게 상관 계수 값이 매우 작아 상관관계가 거의 없는 것으로 나타났다. 'PTNT_OCCURSYMT_AGE'의 경우 선택된 5종의 특징들 중 가장 상관관계가 높은 것으로 확인되었다. Pearson 상관 분석을 통해 타겟 Label에 대해 약한 양적 선형 관계를 나타냄을 확인할 수 있었고, Spearman과 Kendall에서는 양의 비선형 관계를 갖고 있음을 확인했다. 이 값은 Label 1으로 설정된 '0600'과 '1300'의 신장 장애, 소화기 장애 데이터에 대해 모두 동일하게 나타났다.

표 3. '0600' 그리고 '1300' 데이터와 특징 간의 상관관계 분석

Table 3. Correlation analysis of features and '0600' and '1300' data

Feature	Pearson correlation		Spearman Correlation		Kendall correlation	
	correlation value	p-value	correlation value	p-value	correlation value	p-value
DOSAGE_ROUTE_ID	nan	nan	nan	nan	nan	nan
INGR_CD	0.074	0.000	0.096	0.000	0.092	0.000
PTNT_OCCURSYMT_AGE	0.132	0.000	0.161	0.000	0.132	0.000
PTNT_SEX	0.082	0.000	0.093	0.000	0.091	0.000
SENDER_TY	0.066	0.000	0.084	0.000	0.082	0.000

4-2 Model Evaluation of Performance Metrics

표 4는 샘플링을 수행하지 않은 8종의 모델의 Label 1, 즉 '1300'과 '0600'의 코드로 분류되는 신장 장애 및 소화기 장애 데이터에 대한 모델 성능을 비교한 표이다. '1300'의 경우 Accuracy는 대부분의 모델에서 매우 높은 값을 보였다. 하지만 F1-score, Precision, 그리고 Recall 값이 0에 가까운 경우가 많았다. AUC score 값은 0.5에 가까운 값부터 0.9 이상의 값을 보였다. '0600' 역시 Accuracy는 대부분의 모델에서 매우 높은 값을 보이며, F1-score, Precision, 그리고 Recall 값이 작게 나타났다. 또한, AUC score 값 역시 '1300'과 같이 높은 값을 보였다.

표 4. 샘플링을 적용하지 않은 모델의 Label 1에 대한 성능 비교
Table 4. Comparison of model performance for Label 1 without sampling

Label 1	Classifier	Dataset	Accuracy	F1score	Precision	Recall	AUC score
'1300'	AdaBoost Classifier	Train	0.997	0.000	0.000	0.000	0.902
		Internal validation	1.000	0.000	0.000	0.000	0.669
		External validation	1.000	0.000	0.000	0.000	0.541
	Bagging Classifier	Train	0.997	0.015	1.000	0.008	0.986
		Internal validation	1.000	0.000	0.000	0.000	0.528
		External validation	1.000	0.000	0.000	0.000	0.483
	ExtraTrees Classifier	Train	0.997	0.015	1.000	0.008	0.986
		Internal validation	1.000	0.000	0.000	0.000	0.474
		External validation	1.000	0.000	0.000	0.000	0.497
	Gradient Boosting Classifier	Train	0.771	0.017	0.009	0.667	0.807
		Internal validation	0.604	0.000	0.000	0.257	0.347
		External validation	0.986	0.000	0.000	0.000	0.625
	Ranfom Forest Classifier	Train	0.997	0.000	0.000	0.000	0.940
		Internal validation	1.000	0.000	0.000	0.000	0.762
		External validation	1.000	0.000	0.000	0.000	0.258
	MLP CLassifier	Train	0.997	0.000	0.000	0.000	0.743
		Internal validation	1.000	0.000	0.000	0.000	0.790
		External validation	1.000	0.000	0.000	0.000	0.815
	SGD Classifier	Train	0.997	0.000	0.000	0.000	0.500
		Internal validation	1.000	0.000	0.000	0.000	0.499
		External validation	1.000	0.000	0.000	0.000	0.499
	SVC	Train	0.997	0.000	0.000	0.000	0.483
		Internal validation	1.000	0.000	0.000	0.000	0.400
		External validation	1.000	0.000	0.000	0.000	0.705
'0600'	AdaBoost Classifier	Train	0.978	0.000	0.000	0.000	0.860
		Internal validation	0.998	0.000	0.000	0.000	0.764
		External validation	1.000	0.000	0.000	0.000	0.754
	Bagging Classifier	Train	0.979	0.286	0.576	0.190	0.978
		Internal validation	0.997	0.000	0.000	0.000	0.566
		External validation	0.994	0.000	0.000	0.000	0.458
	ExtraTrees Classifier	Train	0.979	0.283	0.577	0.188	0.978
		Internal validation	0.998	0.000	0.000	0.000	0.537
		External validation	1.000	0.000	0.000	0.000	0.486
	Gradient Boosting Classifier	Train	0.852	0.091	0.052	0.339	0.714
		Internal validation	0.856	0.011	0.005	0.314	0.536
		External validation	0.906	0.000	0.000	0.000	0.465
	Ranfom Forest Classifier	Train	0.978	0.000	0.000	0.000	0.910
		Internal validation	0.998	0.000	0.000	0.000	0.724
		External validation	1.000	0.000	0.000	0.000	0.662
	MLP CLassifier	Train	0.978	0.000	0.000	0.000	0.714
		Internal validation	0.997	0.000	0.000	0.000	0.760
		External validation	1.000	0.000	0.000	0.000	0.480
	SGD Classifier	Train	0.978	0.000	0.000	0.000	0.500
		Internal validation	0.997	0.000	0.000	0.000	0.500
		External validation	1.000	0.000	0.000	0.000	0.500
	SVC	Train	0.978	0.000	0.000	0.000	0.529
		Internal validation	0.998	0.000	0.000	0.000	0.452
		External validation	1.000	0.000	0.000	0.000	0.339

표 5는 RandomOverSampler를 이용하여 샘플링을 1회 수행한 8종의 모델의 Label 1, 즉 '1300'과 '0600'의 코드로 분류되는 신장 장애와 소화기 장애 데이터에 대한 모델 성능을 비교한 표이다. 두 표 모두 External validation 데이터에 대해서는 대부분의 모델에서 F1score, Precision, 그리고 Recall 값이 0에 가깝게 나타났으며 Accuracy 역시 낮게 나타났다. Internal validation 데이터에 대해서도 비슷한 양상을 보였으나, Train 데이터의 경우 대부분의 모델이 높은 성능을 보이는 것으로 나타났다.

표 6은 RandomOverSampler를 이용하여 샘플링을 5회 수행한 8종의 모델의 Label 1, 즉 '1300'과 '0600'의 코드로 분류되는 신장 장애와 소화기 장애 데이터에 대한 모델 성능

표 5. RandomOverSampler를 이용하여 샘플링을 1회 적용한 모델의 Label 1에 대한 성능 비교
Table 5. Comparison of model performance for Label 1 across 1 iterations of RandomOverSampler sampling

Label 1	Classifier	Dataset	Accuracy	F1score	Precision	Recall	AUC score
'1300'	AdaBoost Classifier	Train	0.794	0.762	0.903	0.660	0.903
		Internal validation	0.971	0.004	0.002	0.229	0.669
		External validation	1.000	0.000	0.000	0.000	0.541
	Bagging Classifier	Train	0.972	0.972	0.946	1.000	0.986
		Internal validation	0.972	0.000	0.000	0.000	0.483
		External validation	0.999	0.000	0.000	0.000	0.495
	ExtraTrees Classifier	Train	0.972	0.972	0.946	1.000	0.986
		Internal validation	0.975	0.000	0.000	0.000	0.473
		External validation	1.000	0.000	0.000	0.000	0.497
	Gradient Boosting Classifier	Train	0. 972	0.972	0.946	1.000	0.986
		Internal validation	0.975	0.000	0.000	0.000	0.709
		External validation	1.000	0.000	0.000	0.000	0.053
	Ranfom Forest Classifier	Train	0.888	0.890	0.877	0.903	0.953
		Internal validation	1.000	0.000	0.000	0.000	0.466
		External validation	1.000	0.000	0.000	0.000	0.508
	MLP CLassifier	Train	0.704	0.682	0.736	0.636	0.815
		Internal validation	0.872	0.002	0.001	0.629	0.842
		External validation	1.000	0.000	0.000	0.000	0.726
	SGD Classifier	Train	0.649	0.731	0.592	0.955	0.645
		Internal validation	0.550	0.001	0.001	0.943	0.747
		External validation	0.735	0.000	0.000	0.500	0.785
	SVC	Train	0.591	0.686	0.556	0.895	0.713
		Internal validation	0.545	0.001	0.001	0.943	0.829
		External validation	0.445	0.000	0.000	1.000	0.641
'0600'	AdaBoost Classifier	Train	0.814	0.833	0.756	0.928	0.860
		Internal validation	0.992	0.000	0.000	0.000	0.769
		External validation	1.000	0.000	0.000	0.000	0.752
	Bagging Classifier	Train	0.955	0.957	0.924	0.992	0.978
		Internal validation	0.998	0.000	0.000	0.000	0.493
		External validation	0.999	0.000	0.000	0.000	0.481
	ExtraTrees Classifier	Train	0.955	0.957	0.924	0.992	0.978
		Internal validation	0.988	0.000	0.000	0.000	0.484
		External validation	1.000	0.000	0.000	0.000	0.488
	Gradient Boosting Classifier	Train	0.955	0.957	0.924	0.991	0.978
		Internal validation	0.988	0.000	0.000	0.000	0.631
		External validation	1.000	0.000	0.000	0.000	0.670
	Ranfom Forest Classifier	Train	0.827	0.843	0.771	0.931	0.907
		Internal validation	0.996	0.000	0.000	0.000	0.674
		External validation	1.000	0.000	0.000	0.000	0.745
	MLP CLassifier	Train	0.658	0.739	0.598	0.967	0.738
		Internal validation	0.501	0.009	0.005	0.924	0.736
		External validation	1.000	0.000	0.000	0.000	0.508
	SGD Classifier	Train	0.648	0.662	0.637	0.688	0.677
		Internal validation	0.690	0.011	0.006	0.714	0.721
		External validation	0.936	0.000	0.000	0.000	0.558
	SVC	Train	0.608	0.704	0.566	0.934	0.662
		Internal validation	0.534	0.010	0.005	0.924	0.789
		External validation	0.417	0.000	0.000	0.333	0.474

표 6. RandomOverSampler를 이용하여 샘플링을 5회 적용한 모델의 Label 1에 대한 성능 비교
Table 6. Comparison of model performance for Label 1 across 5 iterations of RandomOverSampler sampling

Label 1	Classifier	Dataset	Accuracy	F1score	Precision	Recall	AUC score
'1300'	AdaBoost Classifier	Train	0.794	0.762	0.903	0.660	0.669
		Internal validation	0.971	0.002	0.002	0.229	0.903
		External validation	1.000	0.000	0.000	0.000	0.541
	Bagging Classifier	Train	0.972	0.972	0.946	1.000	0.483
		Internal validation	0.972	0.000	0.000	0.000	0.986
		External validation	0.999	0.000	0.000	0.000	0.495
	ExtraTrees Classifier	Train	0.972	0.972	0.946	1.000	0.473
		Internal validation	0.975	0.000	0.000	0.000	0.986
		External validation	1.000	0.000	0.000	0.000	0.497
	Gradient Boosting Classifier	Train	0.972	0.972	0.946	1.000	0.709
		Internal validation	0.975	0.000	0.000	0.000	0.986
		External validation	1.000	0.000	0.000	0.000	0.053
	Ranfom Forest Classifier	Train	0.888	0.890	0.877	0.903	0.466
		Internal validation	1.000	0.000	0.000	0.000	0.953
		External validation	1.000	0.000	0.000	0.000	0.508
	MLP CLassifier	Train	0.704	0.682	0.736	0.636	0.842
		Internal validation	0.872	0.002	0.001	0.629	0.815
		External validation	1.000	0.000	0.000	0.000	0.726
	SGD Classifier	Train	0.649	0.731	0.592	0.955	0.747
		Internal validation	0.550	0.001	0.001	0.943	0.645
		External validation	0.755	0.000	0.000	0.500	0.785
	SVC	Train	0.591	0.686	0.556	0.895	0.829
		Internal validation	0.545	0.001	0.001	0.943	0.713
		External validation	0.445	0.000	0.000	1.000	0.641
'0600'	AdaBoost Classifier	Train	0.000	0.833	0.756	0.928	0.860
		Internal validation	1.000	0.000	0.000	0.000	0.769
		External validation	0.992	0.815	0.000	0.000	0.752
	Bagging Classifier	Train	0.000	0.957	0.924	0.992	0.978
		Internal validation	0.999	0.000	0.000	0.000	0.493
		External validation	0.988	0.955	0.000	0.000	0.481
	ExtraTrees Classifier	Train	0.000	0.957	0.924	0.992	0.978
		Internal validation	1.000	0.000	0.000	0.000	0.484
		External validation	0.988	0.955	0.000	0.000	0.488
	Gradient Boosting Classifier	Train	0.000	0.957	0.924	0.991	0.978
		Internal validation	1.000	0.000	0.000	0.000	0.631
		External validation	0.988	0.955	0.000	0.000	0.670
	Ranfom Forest Classifier	Train	0.000	0.843	0.771	0.931	0.907
		Internal validation	1.000	0.000	0.000	0.000	0.674
		External validation	0.996	0.827	0.000	0.000	0.745
	MLP CLassifier	Train	0.000	0.739	0.598	0.967	0.768
		Internal validation	1.000	0.009	0.005	0.924	0.736
		External validation	0.501	0.658	0.000	0.000	0.508
	SGD Classifier	Train	0.000	0.662	0.637	0.688	0.677
		Internal validation	0.936	0.011	0.006	0.714	0.721
		External validation	0.690	0.648	0.000	0.000	0.558
	SVC	Train	0.000	0.704	0.566	0.934	0.662
		Internal validation	0.417	0.010	0.005	0.924	0.789
		External validation	0.534	0.608	0.000	0.333	0.474

을 비교한 표이다. Accuracy의 경우 External validation 및 Internal validation 데이터에서 낮게 나타났으며 F1-score, Precision, 그리고 Recall 등의 성능 지표 역시 낮게 나타났다. 반면 Train 데이터에서는 성능 지표 값이 높게 나타났다.

표 7은 BorderlineSMOTE를 이용하여 샘플링을 5회 수행한 8종의 모델의 Label 1, 즉 '1300'과 '0600'의 코드로 분류되는 신장 장애와 소화기 장애 데이터에 대한 모델 성능을 비교한 표이다. 일부 모델에서 Train 데이터에 대해 높은 성능을 보이지만, External validation 및 Internal validation 데이터에 대해서는 성능이 낮게 나타났다.

표 8은 SMOTE를 이용하여 샘플링을 5회 수행한 8종의 모델의 Label 1, 즉 '1300'과 '0600'의 코드로 분류되는 신장 장

표 7. BorderlineSMOTE를 이용하여 샘플링을 5회 적용한 모델의 Label 1에 대한 성능 비교
Table 7. Comparison of model performance for Label 1 across 5 iterations of BorderlineSMOTE sampling

Label 1	Classifier	Dataset	Accuracy	F1score	Precision	Recall	AUC score
'1300'	AdaBoost Classifier	Train	0.820	0.817	0.830	0.804	0.924
		Internal validation	0.945	0.001	0.001	0.171	0.533
		External validation	1.000	0.000	0.000	0.000	0.426
	Bagging Classifier	Train	0.990	0.990	0.982	0.998	0.998
		Internal validation	0.973	0.000	0.000	0.000	0.474
		External validation	0.999	0.000	0.000	0.000	0.486
	ExtraTrees Classifier	Train	0.990	0.990	0.982	0.998	0.998
		Internal validation	0.972	0.000	0.000	0.000	0.456
		External validation	1.000	0.000	0.000	0.000	0.490
	Gradient Boosting Classifier	Train	0.961	0.963	0.931	0.997	0.942
		Internal validation	0.957	0.001	0.000	0.057	0.280
		External validation	0.989	0.000	0.000	0.000	0.341
	Ranfom Forest Classifier	Train	0.937	0.937	0.942	0.932	0.976
		Internal validation	1.000	0.000	0.000	0.000	0.446
		External validation	1.000	0.000	0.000	0.000	0.522
	MLP CLassifier	Train	0.750	0.797	0.671	0.981	0.904
		Internal validation	0.618	0.001	0.001	0.886	0.776
		External validation	1.000	0.000	0.000	0.000	0.560
	SGD Classifier	Train	0.500	0.667	0.500	1.000	0.500
		Internal validation	0.000	0.000	0.000	1.000	0.500
		External validation	0.000	0.000	0.000	1.000	0.500
	SVC	Train	0.603	0.696	0.564	0.908	0.666
		Internal validation	0.560	0.001	0.001	0.943	0.829
		External validation	0.493	0.000	0.000	1.000	0.631
'0600'	AdaBoost Classifier	Train	0.913	0.833	0.754	0.930	0.882
		Internal validation	0.989	0.000	0.000	0.000	0.713
		External validation	1.000	0.000	0.000	0.000	0.775
	Bagging Classifier	Train	0.971	0.972	0.956	0.989	0.990
		Internal validation	0.984	0.015	0.009	0.051	0.526
		External validation	0.996	0.000	0.000	0.000	0.488
	ExtraTrees Classifier	Train	0.971	0.972	0.956	0.988	0.990
		Internal validation	0.991	0.000	0.000	0.000	0.505
		External validation	1.000	0.000	0.000	0.000	0.482
	Gradient Boosting Classifier	Train	0.963	0.964	0.942	0.986	0.974
		Internal validation	0.969	0.000	0.000	0.003	0.443
		External validation	0.980	0.000	0.000	0.000	0.086
	Ranfom Forest Classifier	Train	0.835	0.850	0.778	0.938	0.923
		Internal validation	0.998	0.000	0.000	0.000	0.708
		External validation	1.000	0.000	0.000	0.000	0.781
	MLP CLassifier	Train	0.706	0.762	0.640	0.941	0.749
		Internal validation	0.551	0.010	0.005	0.898	0.711
		External validation	1.000	0.000	0.000	0.000	0.538
	SGD Classifier	Train	0.641	0.703	0.599	0.852	0.663
		Internal validation	0.609	0.010	0.005	0.804	0.727
		External validation	0.907	0.000	0.000	0.000	0.525
	SVC	Train	0.624	0.721	0.573	0.969	0.627
		Internal validation	0.532	0.010	0.005	0.924	0.786
		External validation	0.410	0.000	0.000	0.333	0.472

애와 소화기 장애 데이터에 대한 모델 성능을 비교한 표이다. External validation, Internal validation, Train 데이터에서 보다 균형 잡힌 성능을 보이나 일부 모델의 경우 External validation, Internal validation 데이터에 대해서 성능 저하가 있는 것으로 나타났다.

표 9는 SVMSMOTE를 이용하여 샘플링을 5회 수행한 8종의 모델의 Label 1, 즉 '1300'과 '0600'의 코드로 분류되는 신장 장애와 소화기 장애 데이터에 대한 모델 성능을 비교한 표이다. Train data에 대해서는 높은 성능을 나타내며 External validation, Internal validation 데이터에 대해서 성능이 더 개선되어 나타나나 여전히 성능 저하가 존재했다.

표 10은 표 4~9까지의 결과에 대한 평균값을 나타낸 표이

표 8. SMOTE를 이용하여 샘플링을 5회 적용한 모델의 Label 1에 대한 성능 비교
Table 8. Comparison of model performance for 'Label 1 across 5 iterations of SMOTE sampling

Label 1	Classifier	Dataset	Accuracy	F1score	Precision	Recall	AUC score
'1300'	AdaBoost Classifier	Train	0.884	0.884	0.882	0.886	0.945
		Internal validation	0.969	0.002	0.001	0.143	0.693
		External validation	1.000	0.000	0.000	0.000	0.523
	Bagging Classifier	Train	0.980	0.980	0.962	1.000	0.994
		Internal validation	0.975	0.000	0.000	0.000	0.473
		External validation	0.998	0.000	0.000	0.000	0.487
	ExtraTrees Classifier	Train	0.980	0.980	0.962	1.000	0.994
		Internal validation	0.971	0.000	0.000	0.000	0.456
		External validation	1.000	0.000	0.000	0.000	0.490
	Gradient Boosting Classifier	Train	0.979	0.980	0.960	1.000	0.993
		Internal validation	0.972	0.000	0.000	0.000	0.628
		External validation	0.999	0.000	0.000	0.000	0.257
	Ranfom Forest Classifier	Train	0.939	0.939	0.936	0.942	0.971
		Internal validation	1.000	0.000	0.000	0.000	0.528
		External validation	1.000	0.000	0.000	0.000	0.415
	MLP CLassifier	Train	0.775	0.782	0.758	0.807	0.872
		Internal validation	0.813	0.002	0.001	0.686	0.836
		External validation	1.000	0.000	0.000	0.000	0.646
	SGD Classifier	Train	0.603	0.708	0.560	0.964	0.626
		Internal validation	0.436	0.001	0.000	0.943	0.688
		External validation	0.433	0.000	0.000	1.000	0.721
	SVC	Train	0.593	0.688	0.558	0.896	0.714
		Internal validation	0.547	0.001	0.001	0.943	0.829
		External validation	0.447	0.000	0.000	1.000	0.641
'0600'	AdaBoost Classifier	Train	0.805	0.819	0.764	0.883	0.875
		Internal validation	0.992	0.000	0.000	0.000	0.777
		External validation	1.000	0.000	0.000	0.000	0.729
	Bagging Classifier	Train	0.962	0.963	0.932	0.996	0.983
		Internal validation	0.991	0.000	0.000	0.000	0.515
		External validation	0.999	0.000	0.000	0.000	0.481
	ExtraTrees Classifier	Train	0.962	0.963	0.932	0.996	0.983
		Internal validation	0.991	0.000	0.000	0.000	0.514
		External validation	1.000	0.000	0.000	0.000	0.483
	Gradient Boosting Classifier	Train	0.957	0.958	0.928	0.991	0.978
		Internal validation	0.985	0.000	0.000	0.000	0.624
		External validation	0.997	0.000	0.000	0.000	0.598
	Ranfom Forest Classifier	Train	0.841	0.858	0.778	0.956	0.923
		Internal validation	0.995	0.000	0.000	0.000	0.726
		External validation	1.000	0.000	0.000	0.000	0.725
	MLP CLassifier	Train	0.679	0.747	0.617	0.946	0.737
		Internal validation	0.576	0.010	0.005	0.878	0.738
		External validation	1.000	0.000	0.000	0.000	0.506
	SGD Classifier	Train	0.497	0.013	0.339	0.007	0.597
		Internal validation	0.987	0.000	0.000	0.000	0.702
		External validation	1.000	0.000	0.000	0.000	0.495
	SVC	Train	0.609	0.705	0.566	0.935	0.665
		Internal validation	0.537	0.010	0.005	0.924	0.789
		External validation	0.420	0.000	0.000	0.333	0.474

다. 샘플링 수행 횟수를 기준으로 비교하였을 때, 샘플링을 전혀 적용하지 않았을 때의 성능 지표는 상당히 높은 Accuracy (0.975)를 기록했지만, F1 score (0.015), Precision (0.067), Recall (0.041) 등에서는 매우 낮은 값을 보였다. 샘플링을 1회 적용한 RandomOverSampler의 경우 Accuracy는 0.853으로 감소하였지만, F1 score (0.278), Precision (0.263), 그리고 Recall (0.450)에서 개선되는 모습을 보였다. 반면, 샘플링을 5회 적용했을 때, RandomOverSampler의 Accuracy는 0.721로 더욱 낮아졌고, F1 score (0.412)와 Recall (0.450)에서의 개선은 보였으나, Precision에서는 달라지는 값이 없었다. 따라서 샘플링을 1회 적용한 건과 5회 적용한 건 간의 평균적인 변화는 Ac

표 9. SVMSMOTE를 이용하여 샘플링을 5회 적용한 모델의 Label 1에 대한 성능 비교
Table 9. Comparison of model performance for Label 1 across 5 iterations of SVMSMOTE sampling

Label 1	Classifier	Dataset	Accuracy	F1score	Precision	Recall	AUC score
'1300'	AdaBoost Classifier	Train	0.840	0.840	0.840	0.840	0.937
		Internal validation	0.968	0.002	0.001	0.114	0.466
		External validation	1.000	0.000	0.000	0.000	0.444
	Bagging Classifier	Train	0.990	0.990	0.982	0.997	0.998
		Internal validation	0.485	0.001	0.000	0.943	0.728
		External validation	0.731	0.000	0.000	1.000	0.831
	ExtraTrees Classifier	Train	0.990	0.990	0.982	0.997	0.998
		Internal validation	0.969	0.000	0.000	0.000	0.581
		External validation	1.000	0.000	0.000	0.000	0.745
	Gradient Boosting Classifier	Train	0.953	0.955	0.921	0.991	0.928
		Internal validation	0.578	0.001	0.000	0.771	0.669
		External validation	0.729	0.000	0.000	1.000	0.864
	Ranfom Forest Classifier	Train	0.915	0.915	0.913	0.916	0.976
		Internal validation	1.000	0.000	0.000	0.000	0.420
		External validation	1.000	0.000	0.000	0.000	0.447
	MLP CLassifier	Train	0.500	0.000	0.000	0.000	0.681
		Internal validation	1.000	0.000	0.000	0.000	0.824
		External validation	1.000	0.000	0.000	0.000	0.684
	SGD Classifier	Train	0.577	0.686	0.546	0.923	0.579
		Internal validation	0.429	0.001	0.000	0.943	0.686
		External validation	0.399	0.000	0.000	1.000	0.704
	SVC	Train	0.594	0.692	0.557	0.913	0.632
		Internal validation	0.529	0.001	0.000	0.943	0.831
		External validation	0.406	0.000	0.000	1.000	0.641
'0600'	AdaBoost Classifier	Train	0.812	0.826	0.767	0.896	0.878
		Internal validation	0.983	0.019	0.011	0.066	0.717
		External validation	1.000	0.000	0.000	0.000	0.594
	Bagging Classifier	Train	0.971	0.972	0.953	0.992	0.992
		Internal validation	0.989	0.000	0.000	0.000	0.512
		External validation	0.994	0.000	0.000	0.000	0.476
	ExtraTrees Classifier	Train	0.971	0.972	0.953	0.991	0.992
		Internal validation	0.992	0.000	0.000	0.000	0.513
		External validation	1.000	0.000	0.000	0.000	0.482
	Gradient Boosting Classifier	Train	0.970	0.970	0.950	0.991	0.989
		Internal validation	0.989	0.000	0.000	0.000	0.591
		External validation	0.991	0.000	0.000	0.000	0.689
	Ranfom Forest Classifier	Train	0.856	0.866	0.808	0.933	0.934
		Internal validation	0.994	0.000	0.000	0.000	0.667
		External validation	1.000	0.000	0.000	0.000	0.597
	MLP CLassifier	Train	0.713	0.697	0.738	0.661	0.815
		Internal validation	0.743	0.008	0.004	0.401	0.702
		External validation	1.000	0.000	0.000	0.000	0.457
	SGD Classifier	Train	0.500	0.000	0.000	0.000	0.500
		Internal validation	0.997	0.000	0.000	0.000	0.500
		External validation	1.000	0.000	0.000	0.000	0.500
	SVC	Train	0.618	0.713	0.571	0.952	0.633
		Internal validation	0.535	0.010	0.005	0.924	0.791
		External validation	0.400	0.000	0.000	0.333	0.476

표 10. 오버 샘플링 방법에 대한 모델 성능 비교
Table 10. Comparison of model performance for over sampling methods

Over Sampling Method	Accuracy	F1score	Precision	Recall	AUC score
Without sampling (no samling)	0.975	0.015	0.067	0.041	0.633
1 iterations of RandomOverSampler	0.853	0.278	0.263	0.450	0.692
5 iterations of RandomOverSampler	0.721	0.412	0.263	0.450	0.692
5 iterations of BorderlineSMOTE	0.825	0.285	0.263	0.485	0.650
5 iterations of SMOTE	0.856	0.271	0.259	0.439	0.688
5 iterations of SVMSMOTE	0.825	0.253	0.240	0.467	0.694

curacy에서 약 0.132의 감소가 있었고, F1 score에서는 약 0.134의 증가가 있었다. 샘플링 모델의 종류에 대해 성능을 비교했을 때, BorderlineSMOTE와 SVMSMOTE는 각각 F1 score (0.285 그리고 0.253)과 Recall (0.485 그리고 0.467)에서 상대적으로 높은 성능을 보였다.

4-3 ROC Curves of Prediction Model

그림 4(a)~(c)는 샘플링을 수행하지 않은 8종의 모델의 Label 1, 즉 '1300'의 코드로 분류되는 신장 장애 데이터에 대한 ROC curve를 표현한 그림이다. (a), (b), 그리고 (c)는 순서대로 Train, Internal validation, External validation 데이터를 구분한 것이다. External validation과 Internal

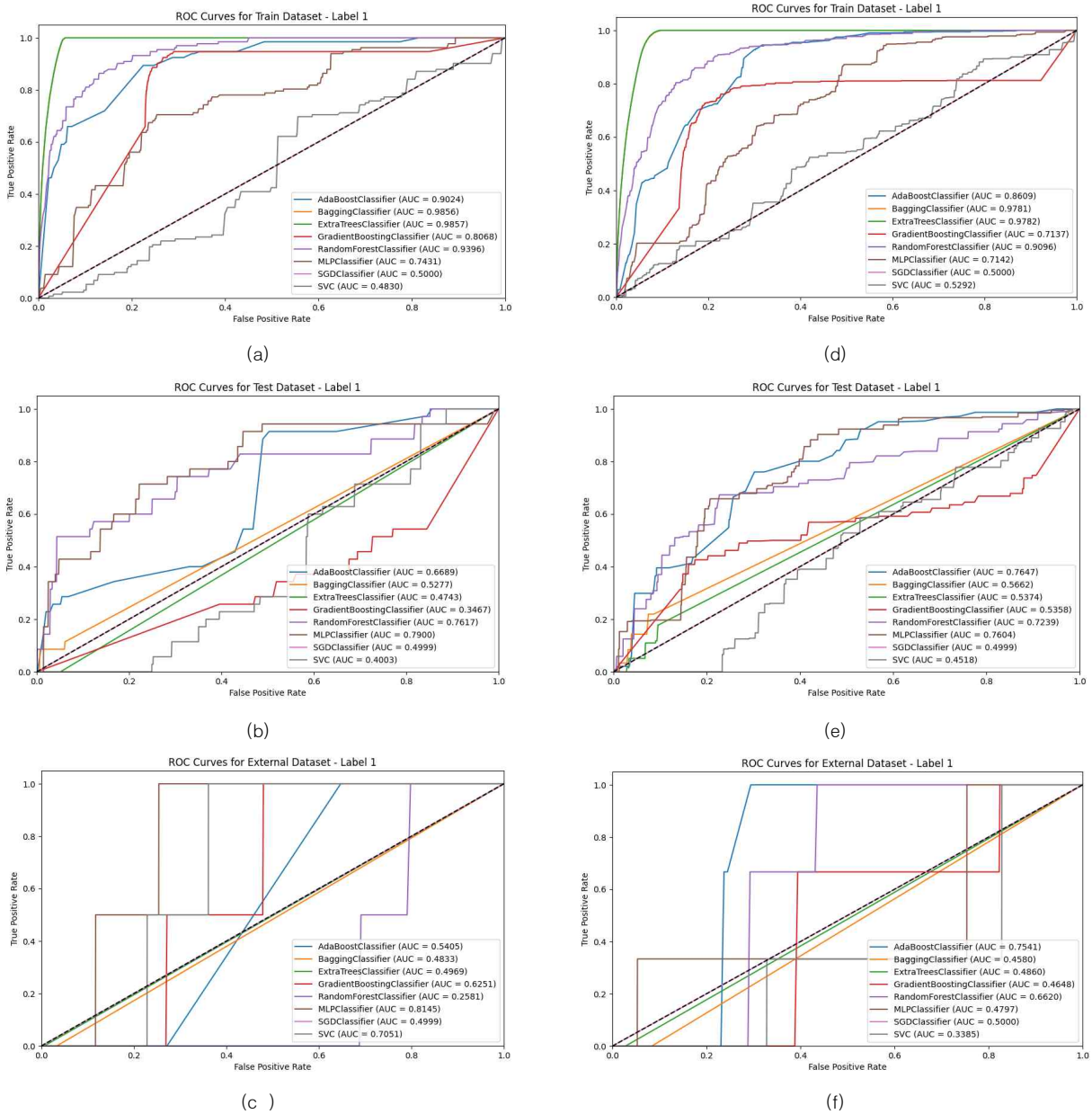


그림 4. 샘플링을 적용하지 않은 Label 1에 대한 데이터의 ROC 곡선, (a) '1300'에 대한 학습 데이터셋, (b) '1300'에 대한 내부 검증 데이터셋, (c) '1300'에 대한 외부 검증 데이터셋, (d) '0600'에 대한 학습 데이터셋, (e) '0600'에 대한 내부 검증 데이터셋, 그리고 (f) '0600'에 대한 외부 검증 데이터셋

Fig. 4. Visualization of ROC curves for Label 1 data without sampling, (a) Train dataset for '1300', (b) Internal validation dataset for '1300', (c) External validation dataset for '1300', (d) Train dataset for '0600', (e) Internal validation dataset for '0600', and (f) External validation dataset for '0600'

validation 데이터에 대해서는 MLPClassifier가, Train에서는 ExtraTreesClassifier가 모델 분류 성능이 가장 뛰어난 것으로 나타났다. 그림 4(d)~(f)는 샘플링을 수행하지 않은 8종의 모델의 Label 1, 즉 '0600'의 코드로 분류되는 소화기 장애 데이터에 대한 ROC curve를 표현한 그림이다. (d), (e), 그리고 (f)는 순서대로 Train, Internal validation, 그리고 External validation 데이터를 구분한 것이다. External validation과 Internal validation 데이터에 대해서는 AdaBoostClassifier가, Train에서는 ExtraTreesClassifier가 모델 분류 성능이 가장 뛰어난 것으로 나타났다.

그림 5(a)~(c)는 RandomOverSampler를 이용하여 샘플링을 1회 수행한 8종의 모델의 Label 1, 즉 '1300'의 코드로 분류되는 신장 장애 데이터에 대한 ROC curve를 표현한 그림이다. (a), (b), 그리고 (c)는 순서대로 Train, Internal validation, External validation 데이터를 구분한 것이다. External validation 데이터에 대해서는 SGDClassifier가,

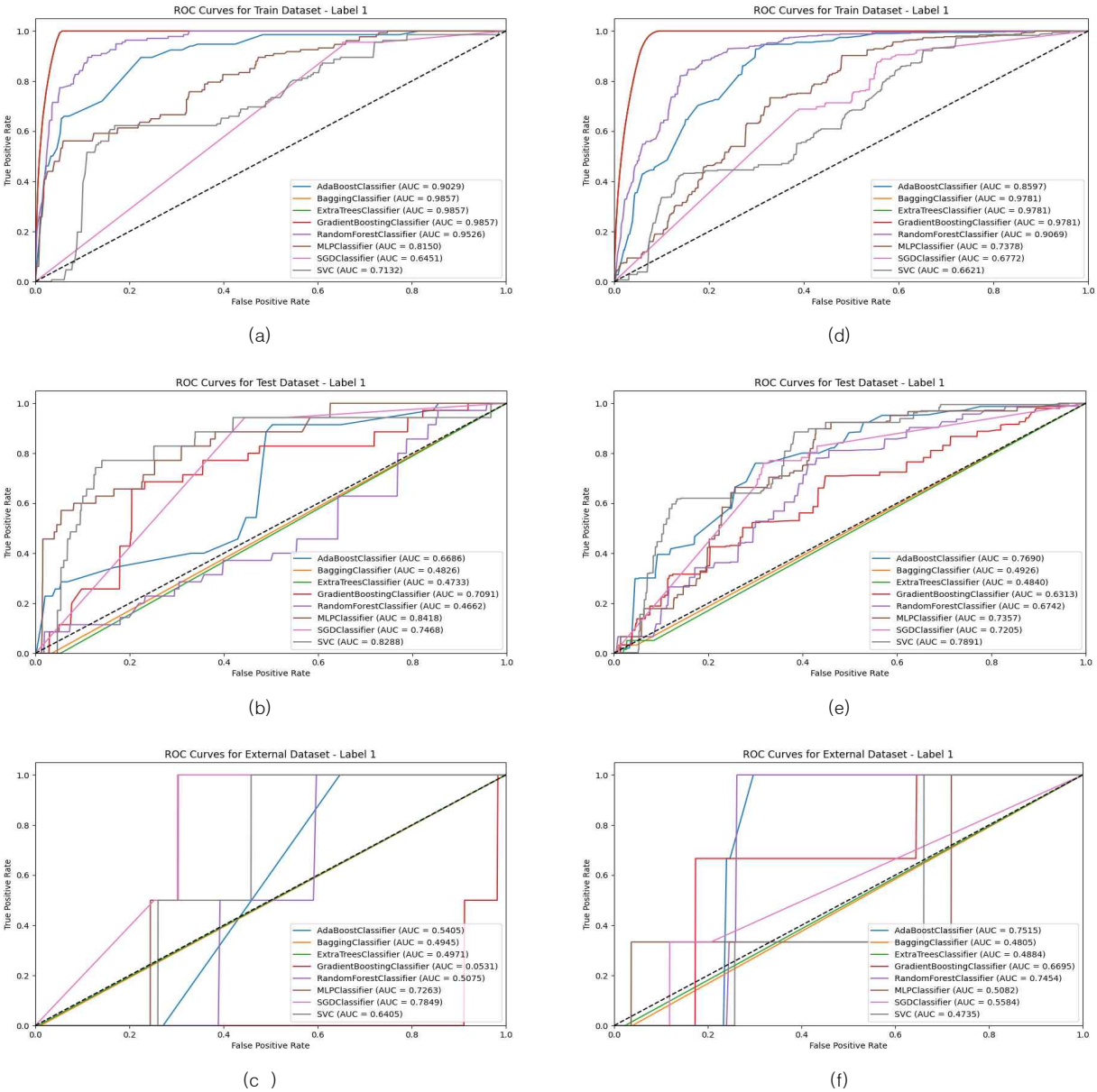


그림 5. RandomOverSampler를 이용하여 샘플링을 1회 적용한 Label 1에 대한 데이터의 ROC 곡선, (a) '1300'에 대한 학습 데이터셋, (b) '1300'에 대한 내부 검증 데이터셋, (c) '1300'에 대한 외부 검증 데이터셋, (d) '0600'에 대한 학습 데이터셋, (e) '0600'에 대한 내부 검증 데이터셋, 그리고 (f) '0600'에 대한 외부 검증 데이터셋

Fig. 5. Visualization of ROC curves for '1300' data aacross 1 iterations of RandomOverSampler sampling, (a) Train dataset for '1300', (b) Internal validation dataset for '1300', (c) External validation dataset for '1300', (d) Train dataset for '0600', (e) Internal validation dataset for '0600', and (f) External validation dataset for '0600'

Internal validation 데이터에 대해서는 MLPClassifier가, Train에서는 BaggingClassifier, ExtraTreesClassifier, GradientBoostingClassifier가 모델 분류 성능이 가장 뛰어난 것으로 나타났다. 그림 5(d)~(f)는 RandomOverSampler를 이용하여 샘플링을 1회 수행한 8종의 모델의 Label 1, 즉 '0600'의 코드로 분류되는 소화기 장애 데이터에 대한 ROC curve를 표현한 그림이다. (d), (e), 그리고 (f)는 순서대로 Train, Internal validation, 그리고 External validation 데이터를 구분한 것이다. External validation 데이터에 대해서는 AdaBoostClassifier가, Internal validation 데이터에 대해서는 SVC가, Train에서는 BaggingClassifier, ExtraTreesClassifier, 그리고 GradientBoostingClassifier가 모델 분류 성능이 가장 뛰어난 것으로 나타났다.

그림 7 (a)~(c)는 BorderlineSMOTE를 이용하여 샘플링을 5회 수행한 8종의 모델의 Label 1, 즉 '1300'의 코드로 분류되는 신장 장애 데이터에 대한 ROC curve를 표현한 그림

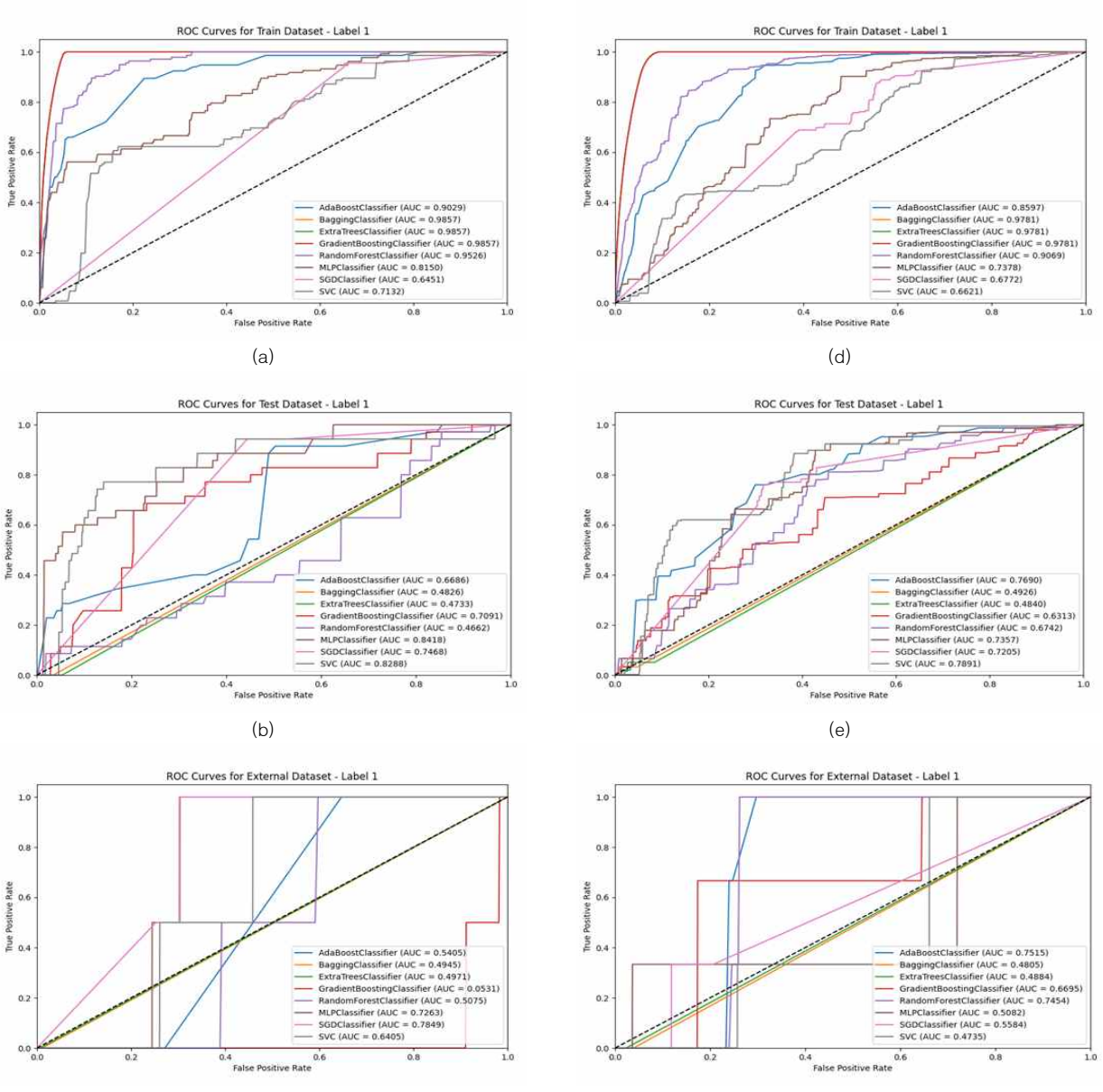


그림 6. RandomOverSampler를 이용하여 샘플링을 5회 적용한 Label 1에 대한 데이터의 ROC 곡선, (a) '1300'에 대한 학습 데이터셋, (b) '1300'에 대한 내부 검증 데이터셋, (c) '1300'에 대한 외부 검증 데이터셋, (d) '0600'에 대한 학습 데이터셋, (e) '0600'에 대한 내부 검증 데이터셋, 그리고 (f) '0600'에 대한 외부 검증 데이터셋

Fig. 6. Visualization of ROC curves for '1300' data aacross 5 iterations of RandomOverSampler sampling, (a) Train dataset for '1300', (b) Internal validation dataset for '1300', (c) External validation dataset for '1300', (d) Train dataset for '0600', (e) Internal validation dataset for '0600', and (f) External validation dataset for '0600'

이다. (a), (b), 그리고 (c)는 순서대로 Train, Internal validation, External validation 데이터를 구분한 것이다. External validation과 Internal validation 데이터에 대해서는 SVC가, Train 데이터에서는 BaggingClassifier과 ExtraTreesClassifier가 모델 분류 성능이 가장 뛰어난 것으로 나타났다. 그림 7의 (d)~(f)는 BorderlineSMOTE를 이용하여 샘플링을 5회 수행한 8종의 모델의 Label 1, 즉 '0600'의 코드로 분류되는 소화기 장애 데이터에 대한 ROC curve를 표현한 그림이다. (d), (e), 그리고 (f)는 순서대로 Train, Internal validation, 그리고 External validation 데이터를 구분한 것이다. External validation 데이터에 대해서는 RandomForestClassifier가, Internal validation 데이터에 대해서는 SVC가, Train 데이터에서는 ExtraTreesClassifier가 모델 분류 성능이 가장 뛰어난 것으로 나타났다.

그림 8 (a)~(c)는 SMOTE를 이용하여 샘플링을 5회 수행한 8종의 모델의 Label 1, 즉 '1300'의 코드로 분류되는 신

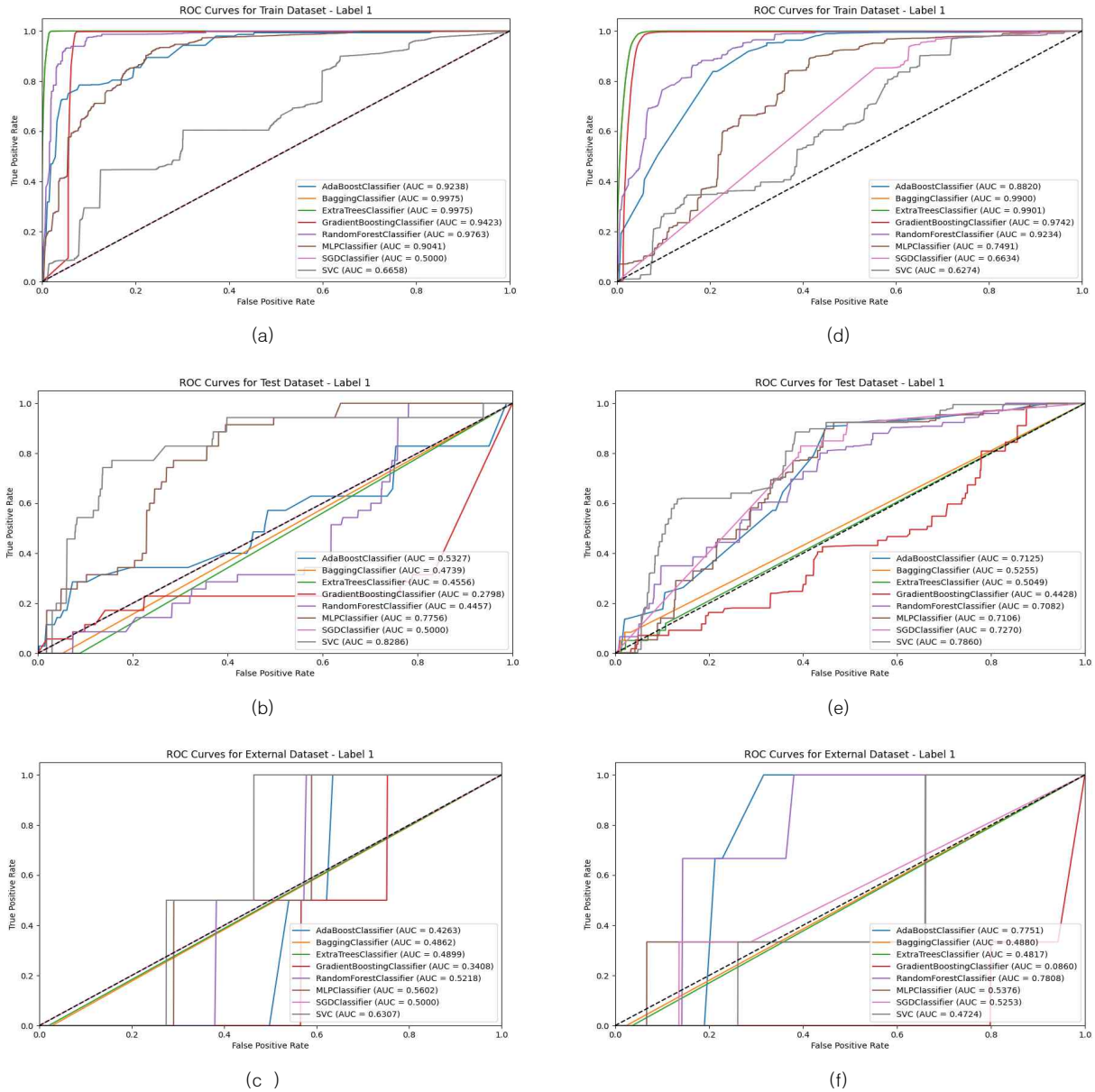


그림 7. BorderlineSMOTE를 이용하여 샘플링을 5회 적용한 Label 1에 대한 데이터의 ROC 곡선, (a) '1300'에 대한 학습 데이터셋, (b) '1300'에 대한 내부 검증 데이터셋, (c) '1300'에 대한 외부 검증 데이터셋, (d) '0600'에 대한 학습 데이터셋, (e) '0600'에 대한 내부 검증 데이터셋, 그리고 (f) '0600'에 대한 외부 검증 데이터셋

Fig. 7. Visualization of ROC curves for '1300' data across 5 iterations of BorderlineSMOTE sampling (a) Train dataset for '1300', (b) Internal validation dataset for '1300', (c) External validation dataset for '1300', (d) Train dataset for '0600', (e) Internal validation dataset for '0600', and (f) External validation dataset for '0600'

장 장애 데이터에 대한 ROC curve를 표현한 그림이다. (a), (b), 그리고 (c)는 순서대로 Train, Internal validation, External validation 데이터를 구분한 것이다. External validation 데이터에 대해서는 SGDClassifier가, Internal validation 데이터에 대해서는 MLPClassifier가, Train 데이터에서는 ExtraTreesClassifier가 모델 분류 성능이 가장 뛰어난 것으로 나타났다. 그림 8의 (d)~(f)는 SMOTE를 이용하여 샘플링을 5회 수행한 8종의 모델의 Label 1, 즉 '0600'의 코드로 분류되는 소화기 장애 데이터에 대한 ROC curve를 표현한 그림이다. (d), (e), 그리고 (f)는 순서대로 Train, Internal validation, 그리고 External validation 데이터를 구분한 것이다. External validation 데이터에 대해서는 AdaBoostClassifier가, Internal validation 데이터에 대해서는 SVC가, Train 데이터에서는 ExtraTreesClassifier가 모델 분류 성능이 가장 뛰어난 것으로 나타났다.

그림 9의 (a)~(c)는 SVMSMOTE를 이용하여 샘플링을 5

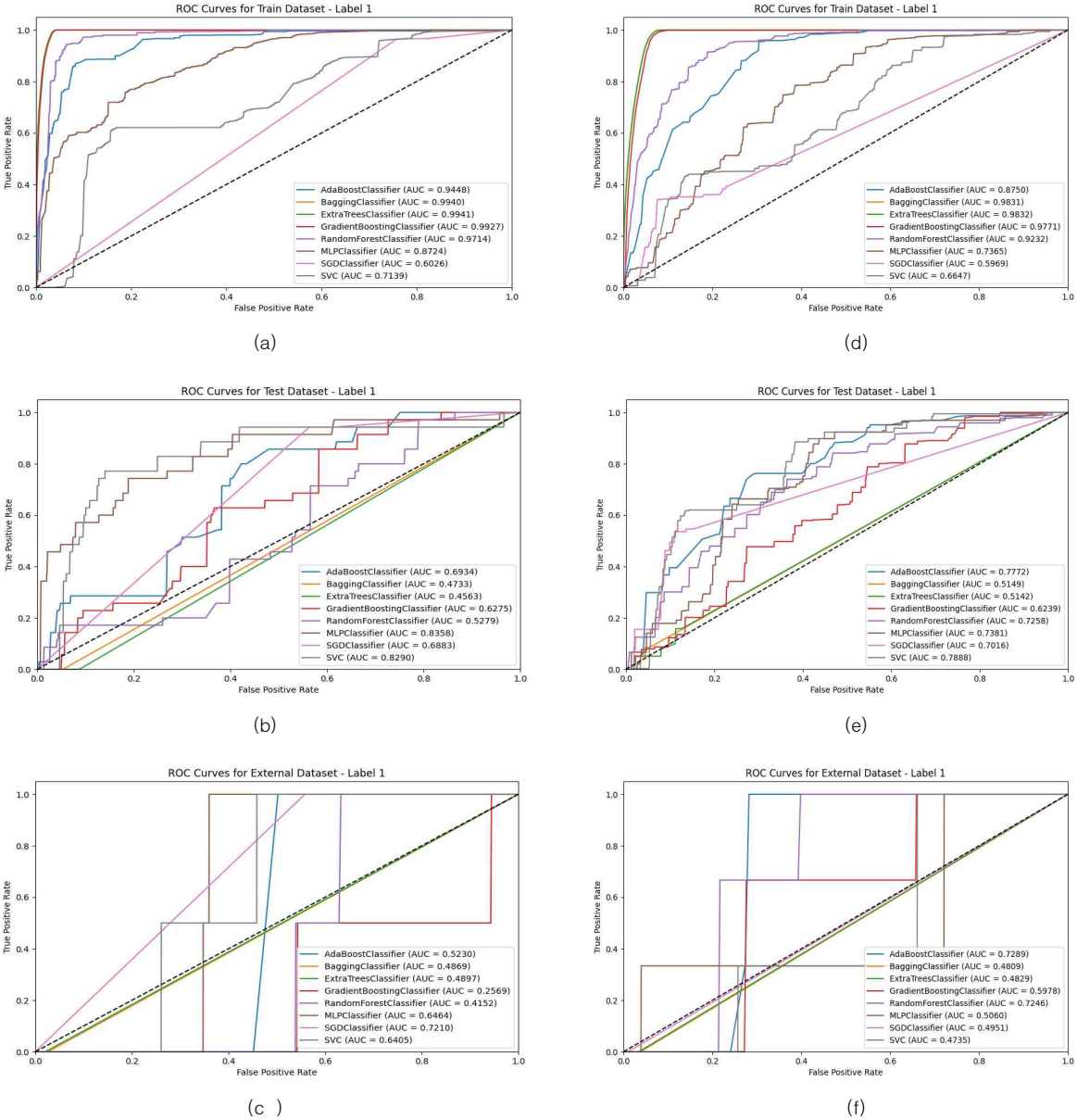


그림 8. SMOTE를 이용하여 샘플링을 5회 적용한 Label 1에 대한 데이터의 ROC 곡선, (a) '1300'에 대한 학습 데이터셋, (b) '1300'에 대한 내부 검증 데이터셋, (c) '1300'에 대한 외부 검증 데이터셋, (d) '0600'에 대한 학습 데이터셋, (e) '0600'에 대한 내부 검증 데이터셋, 그리고 (f) '0600'에 대한 외부 검증 데이터셋

Fig. 8. Visualization of ROC curves for '1300' data across 5 iterations of SMOTE sampling (a) Train dataset for '1300', (b) Internal validation dataset for '1300', (c) External validation dataset for '1300', (d) Train dataset for '0600', (e) Internal validation dataset for '0600', and (f) External validation dataset for '0600'

회 수행한 8종의 모델의 Label 1, 즉 '0600'의 코드로 분류되는 신장 장애 데이터에 대한 ROC curve를 표현한 그림이다. (a), (b), 그리고 (c)는 순서대로 Train, Internal validation, 그리고 External validation 데이터를 구분한 것이다. External validation 데이터에 대해서는 GradientBoostingClassifier가, Internal validation 데이터에 대해서는 SVC가, Train 데이터에서는 BaggingClassifier와 ExtraTreesClassifier가 모델 분류 성능이 가장 뛰어난 것으로 나타났다. 그림 9의 (d)~(f)는 SVMSMOTE를 이용하여 샘플링을 5회 수행한 8종의 모델의 Label 1, 즉 '0600'의 코드로 분류되는 소화기 장애 데이터에 대한 ROC curve를 표현한 그림이다. (d), (e), 그리고 (f)는 순서대로 Train, Internal validation, 그리고 External validation 데이터를 구분한 것이다. External validation 데이터에 대해서는 AdaBoostClassifier가, Internal validation 데이터에 대해서는 SVC가, Train 데이터에서는 ExtraTreesClassifier가

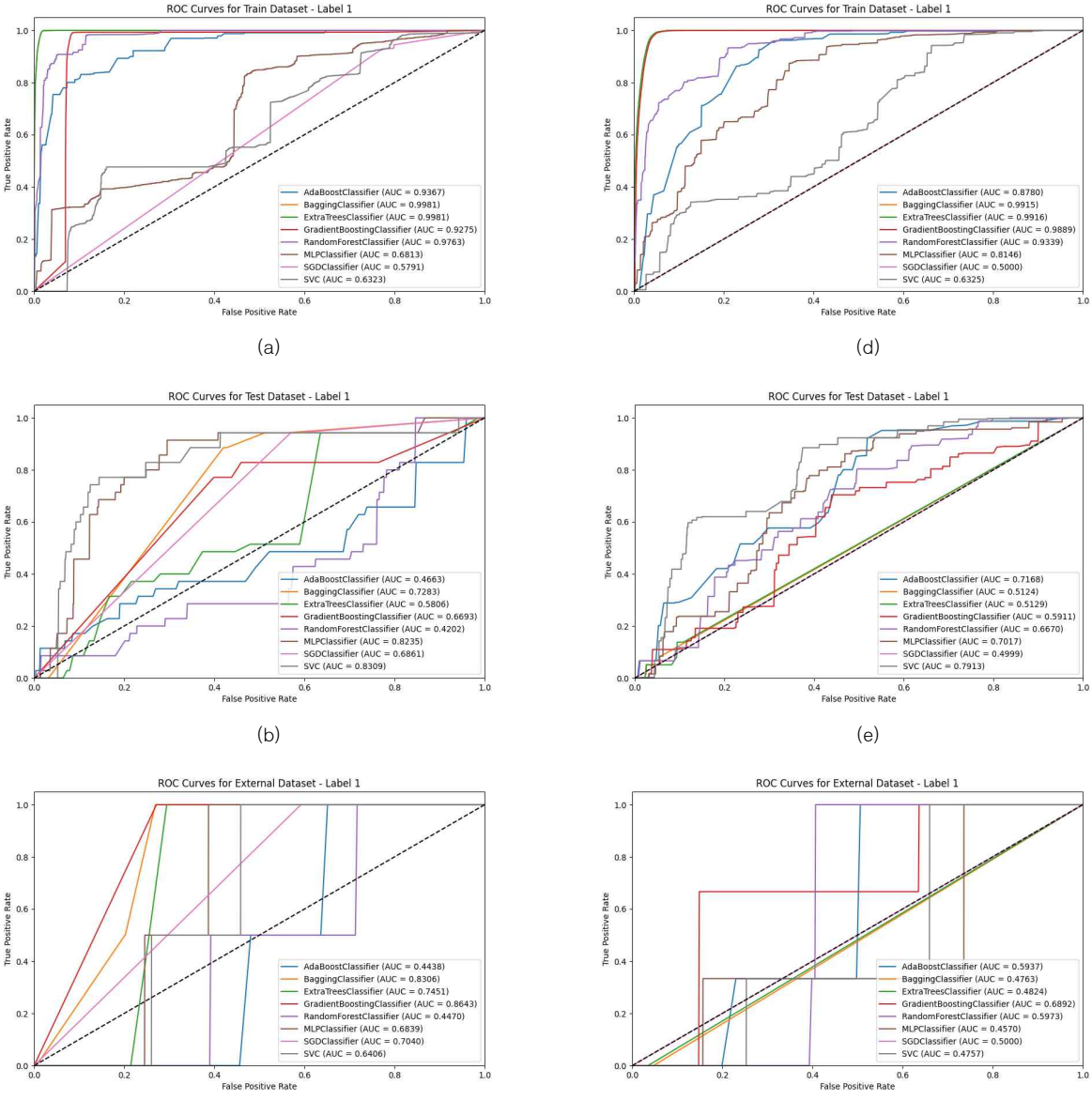


그림 9. SVMSMOTE를 이용하여 샘플링을 5회 적용한 Label 1에 대한 데이터의 ROC 곡선, (a) '1300'에 대한 학습 데이터셋, (b) '1300'에 대한 내부 검증 데이터셋, (c) '1300'에 대한 외부 검증 데이터셋, (d) '0600'에 대한 학습 데이터셋, (e) '0600'에 대한 내부 검증 데이터셋, 그리고 (f) '0600'에 대한 외부 검증 데이터셋

Fig. 9. Visualization of ROC curves for '1300' data across 5 iterations of SVMSMOTE sampling (a) Train dataset for '1300', (b) Internal validation dataset for '1300', (c) External validation dataset for '1300', (d) Train dataset for '0600', (e) Internal validation dataset for '0600', and (f) External validation dataset for '0600'

모델 분류 성능이 가장 뛰어난 것으로 나타났다.

이어서 샘플링 반복 횟수에 따른 모델 성능에 대해서 설명한다. 먼저, 샘플링을 수행하지 않은 건에 대해서는 Accuracy가 대부분의 모델에서 매우 높은 값을 보이고 있음을 확인할 수 있었지만, F1-score, Precision, Recall 값이 0에 가까운 값으로 나타났다. 이를 통해 대부분의 샘플이 0 (negative class)에 해당되는 등, 모델이 거의 모든 샘플을 0으로 예측하며 1 (positive class)에 대해 제대로 예측하지 못함을 확인했다. AUC score 역시 높은 값을 갖으며 모델이 불균형 클래스 간의 구분 능력을 어느 정도 가지고 있음을 나타냈지만, F1-score나 Recall이 매우 낮아 positive class (1)를 제대로 예측하지 못하고 있음을 재차 확인했다. 결국, 샘플링을 수행하기 전에는 불균형 데이터의 문제가 해결되지 않는다는 것을 알 수 있었다.

RandomOverSampler를 이용하여 샘플링을 1회 수행한 건에 대해서는 External validation과 Internal validation 데이터에 대해서 대부분의 모델에서 F1 score, Precision, 그리고 Recall 값이 0으로 나타나며 외부 데이터에 대해 모델이 타겟 라벨(1)을 분류하는 데 어려움을 겪고 있음을 확인했다. 표 4와 5의 성능 지표 값을 통해 이 경우 타겟 Label(1)를 잡아내긴 했지만, 다른 성능 지표가 매우 낮아 유의미한 분류를 하지 못하며 다수의 오탐(False Positive)이 존재함을 확인했다. 이 케이스 역시 대부분의 모델에서 Recall이 0이거나 매우 낮게 나타나며, 타겟 클래스 (1)를 거의 탐지하지 못한 것으로 나타났다. 또한, AUC score 값이 샘플링을 수행하지 않은 경우보다 일반적으로 낮아지며 클래스 구분 성능이 떨어졌음을 확인했다.

RandomOverSampler를 이용하여 샘플링을 5회 수행한 건에 대해서는 Internal validation 및 Train 성능에서 일부 성능 지표가 개선되었음을 확인했다. 하지만 External validation 데이터에 대해서는 여전히 F1score, Precision, Recall 값이 0에 가깝게 나타나는 것이 많았다. 따라서 모델이 외부 데이터에 대해 과적합되었거나, 불균형 문제가 해결되지 않았음을 추정할 수 있었다. Train 데이터의 경우 대부분의 모델들이 비교적 높은 성능 지표 값을 가졌으며, 이를 통해 샘플링 기법을 통해 균형 있는 학습 수행이 가능하다 판단했다. 더하여, AUC score 값이 샘플링을 1회 수행한 건 보다 개선된 경우가 많은 것으로 미루어 보았을 때, 클래스 구분 성능이 향상됨을 알 수 있었다.

표 9의 평균 결과에 따르면, 샘플링을 수행 하였을 때에 비해 수행하지 않았을 때에 F1 score, Precision, Recall 등에서는 매우 낮은 값을 보이며 반복 횟수를 늘려감에 따라 값이 개선됨을 확인했다. 이런 결과를 통해 데이터 불균형이 모델 학습에 영향을 미치고 있으며 샘플링 반복 횟수를 증가시킴에 따라 이런 불균형이 개선될 수 있을 것이라 사료된다.

본 연구는 샘플링 모델 유형에 따른 모델 성능에 대해서 설명한다. 첫째로, RandomOverSampler를 이용하여 샘플링을 5회 반복한 결과, 성능 지표들이 External validation 및 Internal validation 데이터에 대해서 낮게 나타났으며 Train에 대해서는 더 높은 지표를 가졌다. BorderlineSMOTE를 이용한 건에 대해서는 RandomOverSampler와 마찬가지로, 성능 지표가 Train 데이터에 비해 External validation 및 Internal validation 데이터에서 더 낮게 나타났다. 특히, BorderlineSMOTE의 경우 RandomOverSampler와 비교했을 때 일부 모델에서 성능 향상을 보였다. SMOTE의 경우 데이터셋 전반에 걸쳐 균형 잡힌 성능을 보였으나, Internal validation 데이터의 성능 지표가 여전히 Train 데이터에 비해서 낮게 나타났다. SVMSMOTE의 경우 Internal validation 데이터에 대한 성능 지표가 BorderlineSMOTE 및 RandomOverSampler에 비해 향상된다는 특징을 가졌다. 하지만 Train 데이터에 대한 성능 지표가 역시 Internal validation 보다 높게 나타났다.

표 9의 평균 결과에 따르면, BorderlineSMOTE와 SVMSMOTE의 두 샘플링 모델이 데이터의 경계에 있는 샘플을 더 잘 다루어 불균형 문제를 개선하는 데 효과적임을 시사한다. 결론적으로, 샘플링 기법을 적용함으로써 데이터 불균형 문제를 어느 정도 해결할 수 있으며, 특히 BorderlineSMOTE와 SVMSMOTE가 상대적으로 더 나은 성능을 나타내어 효과적인 대안으로 평가된다. 하지만 대부분의 모델이 External validation 및 Internal validation 데이터 보다 Train 데이터에서 더 높은 성능을 보이는 것으로 보아 모델이 과적합(Over-fitting)되었을 가능성이 존재하는 것을 확인했다.

V. 결 론

약물 이상 사례 예측에서 인체 내 장기의 장애 현상을 분석하는 것은 환자의 안전과 의료 비용 절감에 중요한 역할을 수행한다. 이에 균형하게 발생하는 약물 이상 사례를 다루기 위해 본 연구는 인체 내 장기의 장애 현상 분석을 위한 새로운 머신러닝 접근법을 제안하였다. 특히, 데이터 불균형 문제를 해결하기 위하여 샘플링 횟수와 모델의 종류에 따라 제안한 모델의 성능을 비교하였다.

샘플링을 수행하지 않은 경우 모델의 Accuracy는 높았으나 F1-score, Precision, Recall 등의 지표가 매우 낮아 모델이 Label 1(positive class)을 제대로 예측하지 못하는 문제가 있음을 확인하였다. 이어, RandomOverSampler를 5회 반복 수행한 결과, Train 데이터에서는 1회 반복 건에 비하여 성능 평가 지표가 개선됨을 확인했다. 또한, 모델 종류에 따른 결과에 의하면 BorderlineSMOTE와 SMOTE를 사용한 경우 RandomOverSampler보다 성능이 높게 나타났으며, SVMSMOTE는 Internal validation 데이터에서 특히 우수한 성능을 보임을 확인하였다. 다만 대부분의 모델이 External validation 및 Internal validation 데이터보다

Train 데이터에서 높은 성능을 보이며 제안한 모델에게 과적합(over-fitting)의 가능성이 존재하는 것으로 나타났다. 하지만, 결론적으로 4종의 샘플링 모델을 적용한 결과 각 모델마다 다른 성능을 보였으며, 샘플링 모델의 반복 사용을 통해 불균형 데이터 문제를 완화하고 모델의 예측 성능을 향상시킬 수 있음을 확인했다.

향후 연구에서는 샘플링 기법을 성능에 맞게끔 함께 적용하고, 데이터의 품질과 양을 개선하는 등 모델의 일반화 성능을 높일 수 있는 방법을 제안하는 것이 필요할 것으로 사료된다. 추가적으로, 다양한 샘플링 기법을 조합한 앙상블 방법론을 탐색하여 각 기법의 장점을 극대화하는 방안을 모색하거나, 모델의 과적합 문제를 해결하기 위해 교차 검증 기법을 활용하고, 하이퍼파라미터 튜닝을 통해 모델의 최적화를 시도하는 등의 방법론을 적용할 수 있을 것이다.

참고문헌

[1] J.-L. Montastruc, A. Sommet, I. Lacroix, P. Olivier, G. Durrieu, C. Damase-Michel, ... and H. Bagheri, “Pharmacovigilance for Evaluating Adverse Drug Reactions: Value, Organization, and Methods,” *Joint Bone Spine*, Vol. 73, No. 6, pp. 629-632, December 2006. https://doi.org/10.1016/j.jbspin.2006.09.002

[2] R. W. L. Leong and F. K. L. Chan, “Drug-Induced Side Effects Affecting the Gastrointestinal Tract,” *Expert Opinion on Drug Safety*, Vol. 5, No. 4, pp. 585-592, 2006. https://doi.org/10.1517/14740338.5.4.585

[3] L. M. Taft, R. S. Evans, C. R. Shyu, M. J. Egger, N. Chawla, J. A. Mitchell, ... and M. Varner, “Countering Imbalanced Datasets to Improve Adverse Drug Event Predictive Models in Labor and Delivery,” *Journal of Biomedical Informatics*, Vol. 42, No. 2, pp. 356-364, April 2009. https://doi.org/10.1016/j.jbi.2008.09.001

[4] P. Bate, P. Mendel, and G. Robert, *Organizing for Quality: The Improvement Journeys of Leading Hospitals in Europe and the United States*, Abingdon, UK: Radcliffe Publishing, 2008. https://doi.org/10.1201/b20730

[5] D. Yoon, E. K. Ahn, M. Y. Park, S. Y. Cho, P. Ryan, M. J. Schuemie, ... and R. W. Park, “Conversion and Data Quality Assessment of Electronic Health Record Data at a Korean Tertiary Teaching Hospital to a Common Data Model for Distributed Network Research,” *Healthcare Informatics Research*, Vol. 22, No. 1, pp. 54-58, January 2016. https://doi.org/10.4258/hir.2016.22.1.54

[6] R. Liu, M. D. M. AbdulHameed, K. Kumar, X. Yu, A. Wallqvist, and J. Reifman, “Data-Driven Prediction of Adverse Drug Reactions Induced by Drug-Drug Interactions,” *BMC Pharmacology and Toxicology*, Vol. 18, 44, June 2017. https://doi.org/10.1186/s40360-017-0153-6

[7] J. H. G. Scholl, F. P. A. M. van Hunsel, E. Hak, and E. P. van Puijenbroek, “A Prediction Model-based Algorithm for Computer-Assisted Database Screening of Adverse Drug Reactions in the Netherlands,” *Pharmacoepidemiology & Drug Safety*, Vol. 27, No. 2, pp. 199-205, February 2018. https://doi.org/10.1002/pds.4364

[8] R. Miotto, L. Li, B. A. Kidd, and J. T. Dudley, “Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records,” *Scientific Reports*, Vol. 6, 26094, May 2016. https://doi.org/10.1038/srep26094

[9] R. Qureshi, M. Irfan, T. M. Gondal, S. Khan, J. Wu, M. U. Hadi, ... and T. Alam, “AI in Drug Discovery and Its Clinical Relevance,” *Heliyon*, Vol. 9, No. 7, e17575, July 2023. https://doi.org/10.1016/j.heliyon.2023.e17575

[10] Scikit-Learn. Ada Boost Classifier [Internet]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html.

[11] Scikit-Learn. Bagging Classifier [Internet]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html.

[12] Scikit-Learn. Extra Trees Classifier [Internet]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html.

[13] Scikit-Learn. Gradient Boosting Classifier [Internet]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html.

[14] Scikit-Learn. Random Forest Classifier [Internet]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[15] Scikit-Learn. MLP Classifier [Internet]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[16] Scikit-Learn. SGD Classifier [Internet]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html.

[17] Scikit-Learn. SVC [Internet]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

권혜준(Hae-Jun Kwon)

2021년～2024년: 이화여자대학교 (공학사-휴먼기계바이오공학부)
※관심분야 : 의료머신러닝, 생체의공학, 의료메카트로닉스 등

김종윤(Jongyoon Kim)

2009년 : 경희대학교 대학원 (의과학석사)
2010년 : 플로리다 주립대학 (팜디) / University of Florida (Pharm.D.)
2013년～2014년: 삼성서울병원
2014년～2015년: 서울대학교병원
2015년～2019년: 한국의약품안전관리원
2021년～현 재: 동덕여자대학교 약학대학 조교수
※관심분야 : 약물치료학, 항생제 안전성 및 유효성, 항암제 안전성 및 유효성

최승호(Seoung-Ho Choi)

2020년 : 한성대학교 (공학석사)
2022년～2023년: 광운대학교 초빙교수
2023년～현 재: 한성대학교 기초교양학부 조교수
※관심분야 : 의료 딥러닝