|  |  |
| --- | --- |
| Research title | 주요 독성 데이터를 활용한 Binary Classification 모델 구축 및 평가 |
| Name | 고강빈 |

**Abstract**

**Background**

심독성 등 치명적인 독성을 가진 약물이 시장에 공급될 경우, 심각한 결과를 초래할 수 있으며, 신약의 임상실험 단계에서 발견될 시 약품이 전량 폐기 될 수도 있기에, 독성여부는 약물 연구에서 주요하게 다루는 요인 중 하나이다. 따라서 초기에 화합물의 구조로부터 독성을 알아낼 수 있다면 이러한 위험성을 줄임과 동시에 신약 개발의 성공 확률을 높일 수 있을 것이다.

이 글에서는 여러 기계학습 모델을 활용하여 DILI(간독성) 외 3종의 주요 독성의 화학구조로부터 독성을 예측하고, 이를 평가하여 더 나은 예측 방식을 찾아보고자 한다.

학습데이터에는 각 독성과 연관된 화합물에 대하여, 화학구조(Canonical SMILES)와 독성 Label을 사용하게 될 것이며, 이후 새로운 화학 구조로부터 각 독성에 대한 레이블을 예측하게 될 것이며, 해당 모델에 대한 평가는 Sensitivity, Specificity, Balanced Accuracy 총 세 종류의 평가지표로 진행될 것이다.

**Results**

**Conclusions**

**Methods**

Dataset

본 연구에서는 총 4종의 약물 독성 데이터(DILI, Acute Oral Toxicity, AMES Mutagenicity, hERG)를 이용해 진행되었으며, 각각의 수집은 관련 논문, 공개된 데이터베이스 등에서 이루어졌다.

1. DILI

실험에 사용된 데이터들은 최신 논문을 포함하여 다수의 Drug-Induced Liver Injury 논문으로부터 수집되었다. 각각의 데이터는 Less-DILI-Concern을 제외한 후, Most-DILI-Concern을 Positive로, No-DILI-Concern을 Negative로 Re-labeling되었으며[[1]](#footnote-1), Compound의 SMILES 구조가 공개되어 있지 않은 경우에는 PubChemPy API를 이용하여 SMILES를 수집하였다. 이 과정에서 실험의 정확도를 위해 Compound name에 대응되는 SMILES가 하나만 존재하는 화합물만을 채택[[2]](#footnote-2)하여 Dataset을 구성하였다. 전체 Dataset에서 Feature로써 활용하기 위해 SMILES와 DILI Class를 분리하였으며, 각 데이터의 출처를 구분하기 위해 ID를 추가하였다.

수집된 데이터의 SMILES는 RDKit 라이브러리의 StandardizeSmiles 메서드에 의해 Canonicalize되었으며, 처리가 완료된 Canonical SMILES를 기준으로 중복 처리를 진행했다. 하나의 Canonical SMILES에 대응되는 DILI Class가 한 종류인 경우 제일 위에 오는 데이터 하나만을 남기고 나머지를 제거하였으며[[3]](#footnote-3), DILI Class가 같은 Canonical SMILES에 대해 여러 종류로 나타나는 경우에는 평균을 확인하여 처리하였다. 0과 1의 Label이 동일한 비율로 있는 경우는 잘못된 실험 데이터로 간주하여 제거하였으며, 나머지는 반올림을 적용하였다.

최종적으로 수집된 데이터는 2,261개로, 1,037개의 DILI Negative 데이터와 1,224개의 DILI Positive 데이터를 포함한다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **DILI Positive** | **DILI Negative** | **Total** |
| [Ma et al, 2020](https://pubs.acs.org/doi/full/10.1021/acs.chemrestox.0c00322) | 197 | 282 | 479 |
| [DILIst](https://www.sciencedirect.com/science/article/pii/S1359644619303824) [[4]](#footnote-4) | 768 | 511 | 1,279 |
| [Kotsampasakou et al, 2017](https://www.sciencedirect.com/science/article/pii/S0300483X17301701) | 398 | 413 | 811 |
| [Williams et al, 2019](https://pubs.acs.org/doi/abs/10.1021/acs.chemrestox.9b00264) | 44 | 21 | 65 |
| [Kim et al, 2017](https://link.springer.com/article/10.1186/s12859-017-1638-4) | 197 | 280 | 477 |
| [Zhang et al, 2016](https://onlinelibrary.wiley.com/doi/full/10.1002/minf.201500055) | 785 | 532 | 1,317 |
| **Total** | **1,212** | **1,014** | **2,226** |

Table 1. Overview of the datasets (DILI)

1. Acute Oral Toxicity (Rat)

해당 독성에 대한 실험에는 Gadaleta et al, 2019에서 공개한 rat acute oral systemic toxicity (LD50) 관련 Dataset을 사용하였다. 해당 Dataset은 8,994개의 Training set과 2,895개의 Evaluation set으로 이루어져 있으며, NICEATM과 NCCT의 Cheminformatics community에서 수집할 수 있다.

전체 데이터에 대하여 SMILES의 Canonicalization을 진행하였고, 이 과정에서 소실되는 데이터는 없었다. 해당 데이터는 LD50(mg/kg) 값에 기반하여 4종의 Category로 분류되어있었는데, 이 중 Binary label을 가지는 vt와 nt는 데이터의 설명력을 위해 제외하였으며, 나머지 Category에 해당하는 LD50(mg/kg) 값이 없는 데이터의 신뢰성의 확인을 위해 LD50값이 있는 데이터 8,908개에 대하여 해당 값과 매칭되는 class간의 consistency를 확인하였다. 결과적으로 전체 데이터에 대해서 각 Category의 class가 LD50값에 근거한 분류와 일치함을 확인하였고, 나머지 데이터에 대해서도 신뢰성이 높다고 판단, 실험에 사용하기로 하였다. EPA 혹은 GHS category에서의 누락값은 cross check를 통해 경계값에 있는 label을 제외하고 전부 추가를 해주었다.

원본 데이터에서 LD50 값의 단위가 mg/kg으로, 수치의 범위가 넓으며, LD50은 높을 수록 독성이 적은 것이기 때문에 적절한 예측 결과값을 얻기 위해 mg을 g으로 변환 후 -log 처리를 해주었다. 이후 중복처리를 위해 LD50값이 NaN값인 데이터를 제외한 8,908개에 대하여 Standard Deviation을 확인하였고, 이 중 std가 1 이상인 경우 이상 데이터라고 판단하여 제거하였으며, 그 외의 경우 평균으로 처리해주었다. 결과적으로 총 8,309개의 LD50 데이터를 얻을 수 있었다.

나머지 카테고리에 대해서도 동일한 처리를 진행하였고, 결과적으로 얻어진 데이터의 수는 다음 표와 같다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Source | Category | # | class 1 | class 2 | class 3 | class 4 | class 5 |
| Gadaleta et al, 2019 | LD50\_minLog | 8,309 | - | - | - | - | - |
| EPA\_cateogry | 11,744 | 980 | 2,632 | 5,861 | 2,271 | - |
| GHS\_category | 11,834 | 228 | 752 | 1,597 | 4,175 | 5,082 |

1. AMES Mutagenicity

NIHS 및 AMES Mutagenicity 관련 논문에서 데이터를 수집하였다. Source를 구분하기 위해 추가적으로 ID를 생성하였으며, Mutagenicity의 경우 Positive(class A)를 1, Negative를 0으로 Labeling하였다. 데이터의 통일성을 위해 전체 데이터에서 extraordinary / inorganic molecules 데이터와, DEREK 또는 MultiCASE 내부 데이터와 모순된 결과를 보이는 데이터를 제거하였으며[[5]](#footnote-5), 타 독성 데이터와 같은 기준으로 중복처리를 진행하였다. 중복처리는 Canonical SMILES를 기준으로 진행하였으며, Canonicalize 과정과 중복제거 과정에서 각각 660개와 7,590개의 데이터가 제거되었고, 중복된 데이터 중 Mutagenicity의 Variation이 큰 데이터 30개를 추가로 제거하였다.

최종적으로 수집된 데이터는 8,634개로, 3,800개의 AMES Negative 데이터와 4,834개의 AMES Positive 데이터를 포함한다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **AMES Positive** | **AMES Negative** | **Total** |
| [Hansen et al, 2009](https://pubs.acs.org/doi/full/10.1021/ci900161g) | 3,503 | 3,009 | 6,512 |
| [Li et al, 2021](https://pubmed.ncbi.nlm.nih.gov/33506363/) | 703 | 680 | 1,383 |
| [NIHS](http://www.nihs.go.jpdgmamesqsar.html) | 672 | - | 672 |
| [Xu et al, 2012](https://pubs.acs.org/doi/10.1021/ci300400a) | 4,866 | 3,482 | 8,348 |
| **Total** | **4,828** | **3,792** | **8,620** |

Table 3. Overview of the datasets (Ames Mutagenicity)

1. hERG

해당 데이터는 BindingDB, ChEMBL, PubChem과 Cai et al, 2019에서 수집되었다. 각 Source를 구분하기 위해 마찬가지로 ID가 추가되었으며, IC50의 값이 10uM인 경우를 기준으로, 이보다 적을 경우 1, 이외의 경우 0으로 hERG labeling을 진행하였다[[6]](#footnote-6). IC50은 단일 농도에서 hERG 단백질의 활성 억제정도를 측정한 hERG inhibition을 concentration을 높여가며 수차례 시행하여 얻어진 결과로, 억제 정도가 50%가 되는 시점을 계산한 것이므로 신뢰도가 높은 activity 척도에 속한다. hERG label이 없는 4,111개의 데이터를 제거 후, 나머지 데이터 17,056개에 대하여 Canonicalization을 진행하였으며, 이 과정에서 Canonical 할 수 없는 SMILES를 가지는 20개의 데이터가 제거되었다. 이를 기준으로 3,092개의 중복을 제거하였으며, 중복값에 대해 class의 Variance가 큰 데이터 42개를 추가로 제거하여 최종적으로, 5,790개의 Negative와 8,112개의 Positive를 가지는, 총 13,902개의 데이터를 얻었다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **hERG Positive** | **hERG Negative** | **Total** |
| BindingDB | 633 | 290 | 923 |
| ChEMBL | 3,954 | 1,741 | 5,695 |
| PubChem | 2,205 | 1,922 | 4,127 |
| [Cai et al, 2019](https://pubs.acs.org/doi/full/10.1021/acs.jcim.8b00769)(Training set) | 3,485 | 2,826 | 6,311 |
| **Total** | **8,112** | **5,789** | **13,901** |

Table 4. Overview of the datasets (hERG)

**Feature Engineering**

Dataset의 SMILES에 대하여 Feature Engineering을 수행하여 SMILES를 실수 기반 벡터인 Descriptor와 Binary 벡터인 Topological Fingerprints로 변환 후, 두 가지 벡터를 활용하여 Feature를 생성, 모델 학습에 이용하였다. 해당 과정은 RDkit의 Chem.MolFromSmiles를 이용해 Canonical SMILES로부터 Mol Object를 가져왔으며, AllChem.MorganFingerprintAsBitVect를 이용해 Binary Vector를 가져왔다. 또한 Descriptor.\_descList를 통해 RDkit에서 제공하는 전체 Descriptor 총 208개의 Descriptor 리스트를 가져왔으며, 이를 토대로 MoleculeDescriptor의 MolecularDescriptorCalculator 메서드를 통해 인스턴스를 생성, CalcDescriptor로 Descriptor의 tuple을 가져와 저장하였다. 또한 전체 데이터 중, Feature에 NaN값이 많은 데이터와 모든 Feature가 같은 데이터 등을 제거하는 cleaning 과정을 거쳤고, 해당 과정에 의해 약 30개 내외의 데이터가 제거되었다. 최종적으로 처리된 데이터는 1,024개의 Fingerprints와 208개의 Descriptor를 포함하여 총 1,232개의 Feature를 가진다.

Feature 개수 대비 다소 적은 개수의 데이터(DILI 기준 2,226개)를 실험에 활용하고 있기 때문에 Overfitting이 우려되어 Feature Selection을 진행하였다. Feature Selection은, 관련 없거나 중복되는 Feature들을 필터링하여 간결한 subset을 생성하는 과정으로, 불필요한 Feature의 개수를 줄여 Input의 차원을 축소시킴으로써 훈련에 소요되는 시간을 줄이고, Curse of Dimension을 방지하며, Overfitting을 줄여 좀더 Generalized된 학습을 진행할 수 있게 하여 ML알고리즘의 전반적인 Performance를 향상시킬 수 있는 방법 중 하나이다.

Feature Selection은 보통 Standard deviation을 이용하여 전체 데이터에서 std가 작은 feature를 제거하거나, t-test 등 통계적인 방식을 활용하는 Model-Free 방식과, RandomForest, SVM 등에 fitting하여 feature importance를 확인하고 이러한 importance가 높은 feature만을 선택하는 Model-Based 방식이 존재한다. 일반적으로는 Model-Free 방식을 먼저 적용한 후에 Model-Based 방식을 추가로 적용시키는 등의 방식을 사용한다.

이러한 Feature Selection에는 크게 세 가지 방법이 존재한다.

1) Wrapper method: 유용성 측정

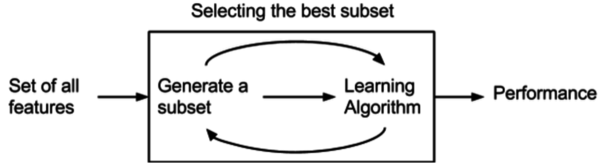


Figure 2. Wrapper method

Model-Based의 Feature Selection의 일종으로, 예측 모델을 사용하여 Feature의 subset을 반복하여 test하는 기법이다. 이 경우 기존 데이터에서 test를 진행할 hold-out set을 따로 두어 cross-validation을 진행한다. 해당 과정의 결과로, 어떤 feature가 모델 학습에 필요한지 알 수 있다. 다만 이와 같은 방법은 상당히 큰 Computing Power를 필요로 하는 NP-Problem의 일종이므로, random hill-climbing과 같은 Heuristic한 방법론을 사용한다.

2) Filter method: 관련성 탐색



Figure 3. Filter method

Filter method는 통계적 측정 방법을 사용하는 방법으로, 계산 속도가 빠르고 feature 간 상관관계를 알아내는데에 적합하므로 Wrapper method를 사용하기 전에 사용하는 방법이다.

해당 실험에서는 two-sample t-test의 일반화된 버전인, one-way ANOVA(Analysis of Variance) 방식을 활용하여 각 feature별 p-value를 계산하고, 이 중 유의미한 데이터만을 선별 후, 나머지에 대해 오름차순 정렬하여 순서에 따라 wrapper method base feature selection flow에 투입하였다.

3) Embedded method: 유용성 측정 - 내장 metirc 사용

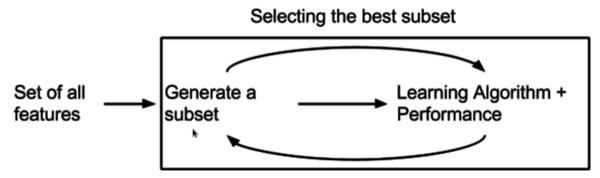


Figure 4. Embedded method

Embedded method는 모델의 정확도에 기여하는 feature를 학습한다. 좀 더 작은 계수를 가지는 회귀식을 찾는 방향으로 제약조건을 주어 이를 제어한다.

Task가 Regression인 경우 Elastic Regression이나 LASSO Regression 등을 통해서 Feature selection을 진행한다.

Hashing 방식으로 만들어지는 Fingerprints의 경우, 잘못된 Feature를 골라내면 임의의 Molecule이 입력되었을 때 On으로 표시된 bit가 의도치 않게 제거될 수 있으므로, Descriptor에 대해서만 Feature Selection을 진행하였다. 먼저 Standard Deviation을 이용한 Model-Free 방식을 적용하여 Feature Selection을 진행하였으며, 이후 단계를 높여 Model-Based 방식의 Selection을 진행하였다.

Model-Based 방식으로는 Categorical/Continuous Feature에 대해 gain이 가장 크도록 노드를 두 개로 분할하는 Decision Tree 기반의 Random Forest 알고리즘을 사용하였다. Classification 시에는 Gini impurity 를 사용하여 노드를 분할하고, Regression 시에는 MSE를 사용하여 Information Gain이 가장 높게끔 분기하게 된다.

|  |  |  |  |
| --- | --- | --- | --- |
| Impurity | Task | Formula | Description |
| Gini impurity | Classification |  | 는 label i의 노드 frequency, C는 unique label의 개수 |
| Entropy | Classification |  | 는 label i의 노드 frequency, C는 unique label의 개수 |
| Variance / MSE | Regression |  | 는 instance의 label, N은 instance의 개수, 는 로부터 얻어진 mean |
| Variance / MAE | Regression |  | 는 instance의 label, N은 instance의 개수, 는 로부터 얻어진 mean |

Information Gain이란 Decision Tree가 특정한 Feature로 분기함으로써 얻는 성능적 이득을 총칭하며, 이를 기반으로 각 Feature별 importance를 구할 수 있다.

1) Model-Free

먼저, Descriptor를 형태별로 분류하여 Binary 형태와 float 형태를 분리하였다. Binary 형태의 경우 class의 개수를 파악하여, 한 쪽 class로 과하게 치중된 경우(전체 데이터의 90%이상이 하나의 class인 경우)는 타겟에 대한 충분한 설명력을 가지지 못한다고 가정하여 제거하였다. Float 형태의 Descriptor의 경우, Scikit-learn의 MinMaxScaler를 이용하여 0~1사이 값으로 Scaling을 진행 후 Standard Deviation이 특정 threshold보다 작은 Descriptor를 제거하였다. threshold값은 Model-Free방식에 의해 제거되는 Descriptor의 개수가 65개 내외가 될 수 있도록 설정하였으며, 각 데이터에 대한 threshold와, Model-Free Feature Selection 이후의 Descriptor 개수는 다음과 같다.

|  |  |  |
| --- | --- | --- |
| Data | threshold | # of Descriptors |
| DILI | 0.06 | 138 (-70) |
| Acute Oral Toxicity: LD50 | 0.05 | 143 (-65) |
| Acute Oral Toxicity: EPA | 0.03 | 145 (-63) |
| Acute Oral Toxicity: GHS | 0.03 | 145 (-63) |
| AMES Mutagenicity | 0.035 | 143 (-65) |
| hERG | 0.07 | 145 (-63) |

2) Model-based

2-1) Binary Classification: DILI, AMES Mutagenicity, hERG

Binary Classification은 데이터를 2개의 class(0,1)로 분류하는 문제이다. 따라서 K를 2로 설정하여 Gini impurity를 계산할 수 있다.

각 샘플이 두 class 0,1에 속할 확률을 각각 이라고 정의할 때, 노드 에서의 Gini impurity는 다음과 같이 정의된다.

해당 노드에서 샘플들이 균등하게 분포되어 있을 경우 Gini impurity가 높아지게 되며, 분류 시에는 이 Gini impurity가 최대한 감소하는 방향으로 노드를 분기하게 된다. 이후 각 노드의 Importance Gain을 계산하게 되는데, 이는 노드룰 분기함으로써 얻는 성능 이득이므로, 현재 노드의 Gini impurity에서 자식 노드의 Gini impurity를 샘플 수의 비율만큼 가중치를 곱해서 뺀 값이 되고, 해당 식은 다음과 같다.

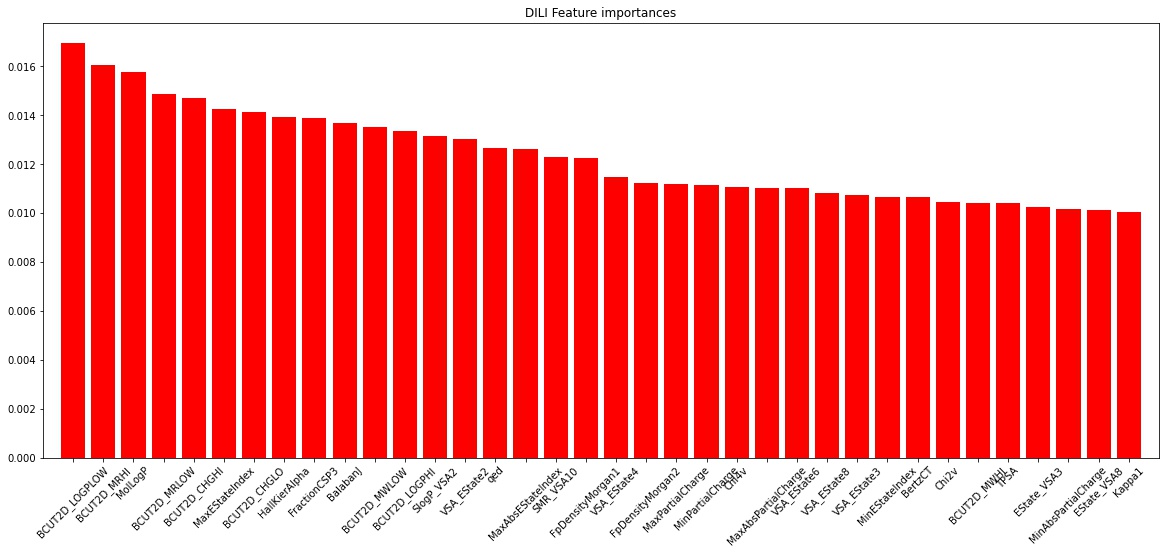
이제 i번째 Feature의 importance 를 계산할 수 있는데, 이는 전체 노드의 Importance Gain의 총합에서 에 의해 분기된 모든 노드의 Importance Gain의 비율로 정의된다.

마지막으로 모든 Feature의 importance의 합으로 나누어 Normalize한다.

여기서 사용항 앙상블 기반의 Random Forest 모델은 여러 Descision Tree가 합쳐진 것이므로, feature importance를 구하기 위해 각 Tree의 importance를 평균 내어 사용하게 되며, 이를 MDI(Mean Decrease in Impurity) 방법이라고 한다.

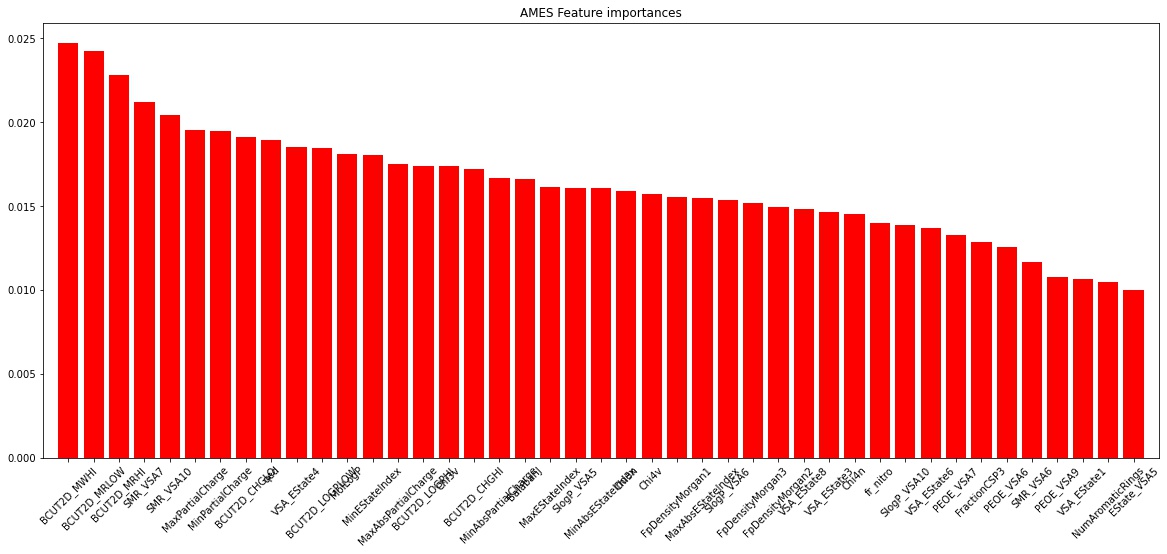
이러한 방식을 사용하면 각 Feature가 Target에 주는 영향을 고려한 Feature 각각의 importance를 확인할 수 있다. 하지만 이 방법에는 문제가 있는데, MDI 기반의 feature importance는 기본적으로 연속형 변수나 많은 범주를 가진(high-cardinality) 변수의 importance를 과대평가하는 경향이 그것이다. 이는 분기에 따른 importance gain 자체는 적더라도 이러한 변수로 노드 분기 자체가 많아지면 상대적인 importance가 증가하게 되어 노드 분기의 기준이 될 기회가 많기 때문이다. 수치형 변수를 normalize하고 범주형의 범주를 줄이는 과정을 통해 이러한 문제를 해결할 수는 있다.

2-1-1) DILI



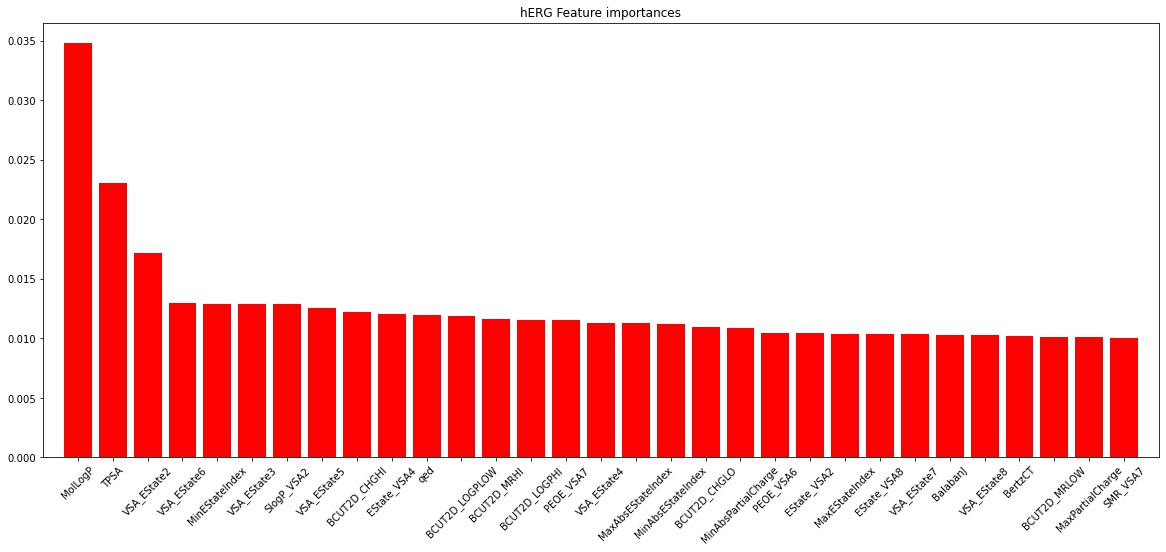
위는 DILI 데이터에 대한 각 Feature의 importance를 기준으로 큰 값부터 정렬하여 시각화한 결과이다. 수치형 feature의 normalize와 범주형 변수의 범주를 줄이는 과정을 거쳤으며, 실행결과의 일관성을 위하여 RandomForestClassifier의 random state는 999로 설정하였다. 0.01보다 큰 importance를 가지는 Feature만을 골라냈고, 총 36개의 Feature가 Select 되었다.

2-1-2) AMES Mutagenicity



마찬가지로, AMES Mutagenicity 데이터에 대한 각 Feature의 importance를 기준으로 큰 값부터 정렬하여 시각화한 결과이다. 0.01보다 큰 importance를 가지는 Feature만을 골라냈으며, 총 43개의 Feature가 Select 되었다.

2-1-3) hERG

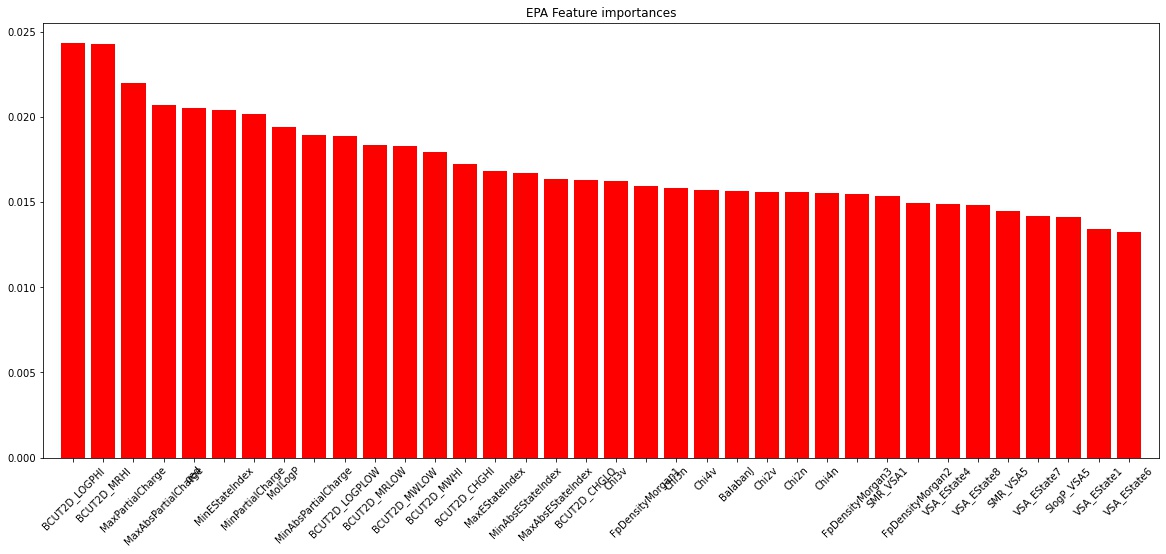


앞의 결과에 이어서, hERG 데이터에 대한 각 Feature의 importance를 기준으로 큰 값부터 정렬하여 시각화한 결과이다. 0.01보다 큰 importance를 가지는 Feature만을 골라냈으며, 총 31개의 Feature가 Select 되었다.

2-2) Multi-Class Classification: Acute Oral Toxicity(EPA\_category), Acute Oral Toxicity(GHS\_category)

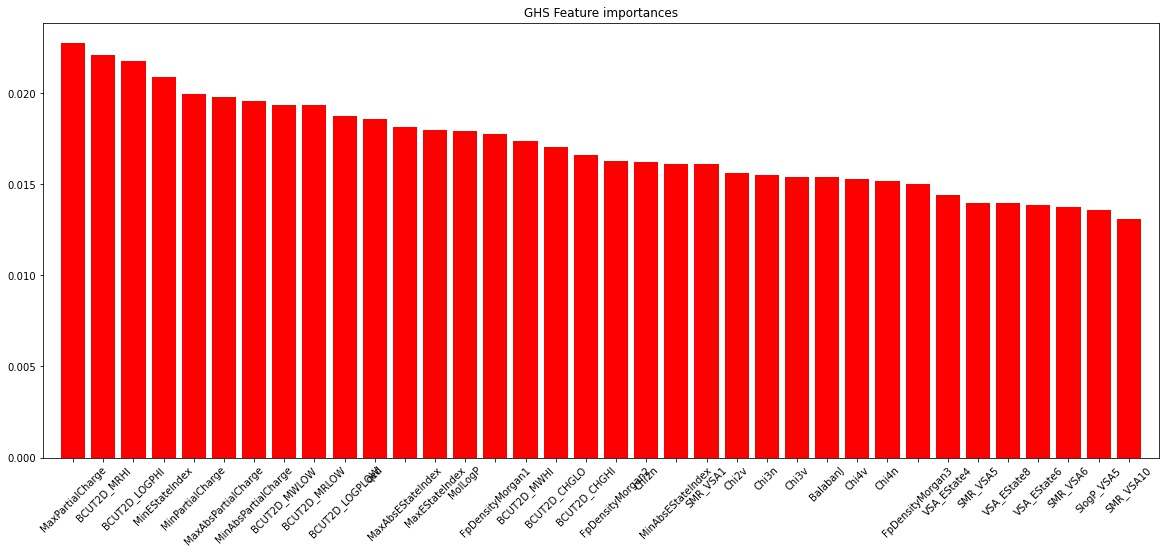
기본적인 개념은 Binary와 동일하나, 이 경우 K의 값이 Class의 개수로 달라진다. 따라서, EPA\_category의 경우 K=4이고, GHS\_category인 경우 K=5가 된다.

2-2-1) Acute Oral Toxicity(EPA\_category)



4개의 Class를 가지는 EPA\_category를 타겟으로 하여 각 Feature의 importance를 계산하고, 이를 시각화 한 결과이다. 최종 갯수를 위 Binary에서와 비슷하게 맞추기 위해 0.013보다 큰 importance를 가지는 Feature만을 골라냈으며, 총 36개의 Feature가 Select되었다.

2-2-2) Acute Oral Toxicity(GHS\_category)

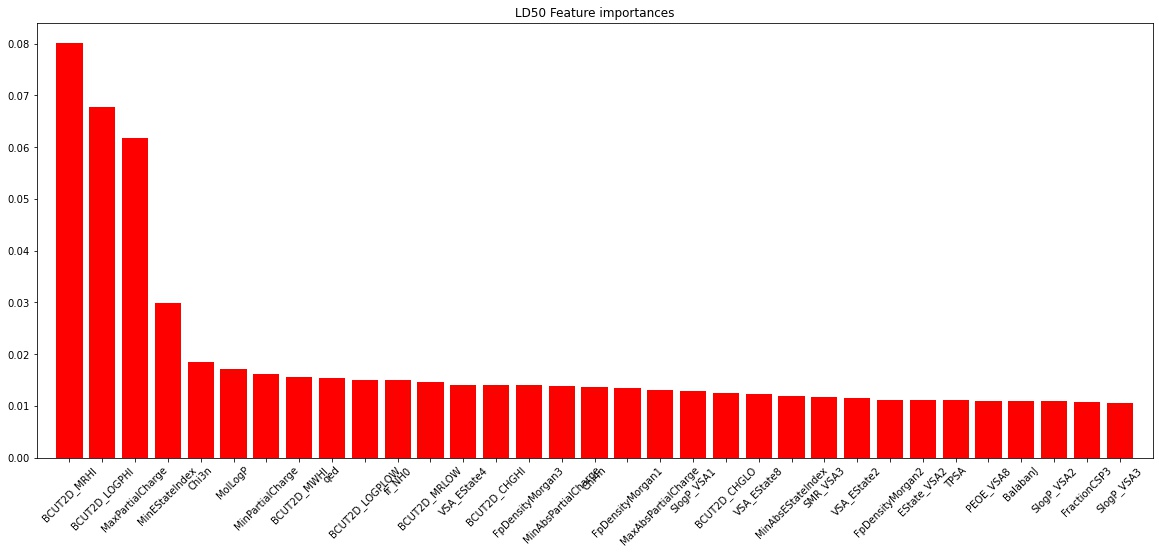


5개의 Class를 가지는 GHS\_category를 타겟으로 하여 각 Feature의 importance를 계산하고, 이를 시각화 한 결과이다. EPA\_category에서와 마찬가지로 0.013보다 큰 importance를 가지는 Feature만을 골라냈으며, 총 36개의 Feature가 Select되었다.

2-3) Regression: Acute Oral Toxicity(LD50)

위에서 진행한 Classification과 달리, Regression에서는 MSE를 사용하여 Information Gain이 가장 높게끔 분기하게 된다. Random Forest Regressor를 이용하면 회귀에도 Random Forest방식을 사용할 수 있으므로 동일한 과정을 적용하여 Feature importance를 계산하였다.

2-3-1) Acute Oral Toxicity(LD50)



연속 타겟 변수인 LD50 값에 대한 Feature들의 importance를 계산하기 위해 Random Forest Resgressor를 사용하였다. 전체 Feature 중 0.01 보다 큰 importance를 가지는 feature만을 선별하였고, 총 33개의 feature가 select되었다.

**Classifier evaluation**

총 다섯 종류의 Classifier[[7]](#footnote-7)를 사용하여 Classification을 진행 후, 각각의 성능을 분석하고 비교하였다. Generalize된 classification performance를 측정하기 위해 각 Classifier에 대해 Startified 3-Fold Cross Validation procedure이 수행되었으며, 이 과정은 매번 다른 random fold division을 활용하여 3번씩 수행되었다. 최종 classification performance는 전체 수행결과의 평균값으로 측정되었다.

Classification Evaluation metric으로는 서로 다른 5가지의 classification performance indicator가 사용되었으며 각각에 대한 정의는 다음과 같다.

1) Accuracy (True prediction Rate)

Classification metrics 중 가장 기본적인 metric이라고 할 수 있다. 전체 예측값 중 정확하게 예측한 예측값의 비율을 나타내는 것으로, 식은 다음과 같다.

Class Imbalance가 없는 데이터에 대하여 Classification을 수행할 때 채택하며, 반대의 경우 무의미한 지표가 될 수 있으므로 주의해야 한다.

2) Balanced Accuracy

위에서 설명하였듯이, Accuracy는 Imbalance한 Class를 가지는 Data에 대해서는 성능이 좋지 않을 수 있다. 따라서 Acute Oral Toxicity (EPA\_category, GHS\_category)에 대한 Classification을 수행할 때는 Accuracy의 의미가 다소 약해질 것으로 우려되므로 이 경우에는 다음 평가지표를 활용할 예정이다.

이는 이어서 설명할 Recall과, Predicted Negative 대비 True Negative의 비율인 Specificity의 산술평균으로, Dataset의 Class가 Imbalance할 때 Accuracy보다 더 나은 성능을 보인다.

3) Recall (True Positive Rate)

Sensitivity라고도 하며, Observed Positive 대비 True Positive의 rate를 계산한다. 식은 다음과 같다.

Positive를 Positive로 예측한 True Positive(TP)와 Positive를 Negative로 예측한 False Negative(FN) 중 TP의 비율로, 모델이 얼마나 정확하게 Positive 값을 찾느냐를 나타내는 지표이다.

4) Precision (True Positive Rate)

Predictied Positive 대비 True Positive의 rate를 계산한다. 식은 다음과 같다.

모델이 Positive로 예측한 값들 중 True Positive(TP)의 비율로, 모델의 Positive 예측 정확도를 나타낸다.

5) F1

위에서 계산한 Precision과 Recall의 조화평균으로, 1에 가까울수록 좋은 성능을 보인다고 할 수 있으며, Precision과 Recall이 모두 무난하게 좋은 모델을 원할 때 이용하는 지표이다. 식은 다음과 같다.

식에서 확인할 수 있듯이, Classifier의 Precision과 Recall의 균형을 유지하는 역할을 하며, 둘 중 하나라도 낮아질 경우 F1의 점수는 낮아진다.

**Binary Classification**

1) Stratified 5-fold Cross Validation

Train set과 Test set에서의 Class의 비율을 일정하게 유지하는 Stratified Cross Validation에 n\_splits를 5로 주어 5-fold CV를 수행하였다. 각 Metric 값은 5번 수행에서 나온 결과값의 평균값이다.

2) Modeling

여러 Classifier Model에 대하여 결과를 도출하였다. 각 모델별로 추가적인 처리를 해주었고, 그 중 최대의 수행 점수를 내는 경우를 도출하였다.

2-1) DILI

2-1-1) 처리 전 모델 성능 평가

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score | Balanced-Accuracy |
| Logistic Regression | 0.594 | 0.604 | 0.741 | 0.665 | 0.580 |
| LGBM Classifier | 0.715 | 0.722 | 0.776 | 0.747 | 0.709 |
| Random Forest Classifier | 0.720 | 0.721 | 0.798 | 0.755 | 0.715 |
| k-NN | 0.624 | 0.642 | 0.698 | 0.669 | 0.616 |
| SVM | 0.554 | 0.550 | 0.992 | 0.708 | 0.511 |

처리 전의 성능은 전반적으로 좋지 못한 모습을 보인다. 따라서 추가적인 처리를 가해 다시 모델의 성능을 평가하였다.

2-1-2) Normaization 이후 성능 평가

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score | Balanced-Accuracy |
| Logistic Regression | 0.677 | 0.698 | 0.718 | 0.708 | 0.673 |
| LGBM Classifier | 0.708 | 0.717 | 0.766 | 0.740 | 0.702 |
| Random Forest Classifier | 0.718 | 0.715 | 0.797 | 0.757 | 0.712 |
| k-NN | 0.668 | 0.697 | 0.691 | 0.694 | 0.666 |
| SVM | 0.709 | 0.700 | 0.818 | 0.754 | 0.699 |

MinMaxScaling을 통해 Normaization을 진행 후 모델의 성능을 평가하였다.

데이터들의 위치를 기반으로 분류하는 모델인 Logistic Regression, SVM, k-NN을 제외하면 성능의 향상이 이루어지지 않았으며, Logistc Regression과 SVM에서의 Recall 값에서는 오히려 성능이 낮아지는 모습을 보였다. 이로부터 Normalization은 데이터들을 일정한 비율로 분포할 수 있게 만들어, 거리 기반 모델에서는 전반적인 성능을 증가시킬 수 있으나, 오히려 Positive의 민감성에는 악영향을 주었다는 결론을 도출할 수 있다.

2-1-3) SMOTE를 이용한 Oversampling

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score | Balanced-Accuracy |
| Logistic Regression | 0.701 | 0.710 | 0.681 | 0.695 | 0.701 |
| LGBM Classifier | 0.741 | 0.739 | 0.746 | 0.742 | 0.740 |
| Random Forest Classifier | 0.745 | 0.734 | 0.766 | 0.757 | 0.744 |
| k-NN | 0.639 | 0.717 | 0.462 | 0.561 | 0.639 |
| SVM | 0.727 | 0.720 | 0.747 | 0.733 | 0.727 |

Normalization이 진행된 데이터에 대하여 추가적인 SMOTE 기법을 통해 Class의 비율을 동일하게 맞춰준 후 모델의 성능을 평가하였다.

대부분의 경우 약간의 성능향상이 있었으나, Recall의 경우 모든 모델에서 점수가 감소하였고, 특히k-NN에서는 전반적인 성능 자체가 하락하였다.

2-2) AMES Mutagenicity

2-2-1) 처리 전 모델 성능 평가

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score | Balanced-Accuracy |
| Logistic Regression | 0.761 | 0.780 | 0.800 | 0.790 | 0.756 |
| LGBM Classifier | 0.816 | 0.829 | 0.846 | 0.837 | 0.812 |
| Random Forest Classifier | 0.829 | 0.848 | 0.851 | 0.847 | 0.827 |
| k-NN | 0.705 | 0.725 | 0.763 | 0.744 | 0.698 |
| SVM | 0.714 | 0.706 | 0.839 | 0.767 | 0.697 |

DILI와 비교했을 때 데이터의 양이 비교적 많았기에 성능이 전반적으로 좋은 모습을 보이고는 있지만, 여전히 처리 전의 성능은 그닥 좋지 못하다. 따라서 추가적인 처리를 가해 다시 모델의 성능을 평가하였다.

2-2-2) Normaization 이후 성능 평가

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score | Balanced-Accuracy |
| Logistic Regression | 0.760 | 0.782 | 0.794 | 0.788 | 0.756 |
| LGBM Classifier | 0.815 | 0.827 | 0.846 | 0.837 | 0.811 |
| Random Forest Classifier | 0.829 | 0.853 | 0.850 | 0.847 | 0.828 |
| k-NN | 0.784 | 0.821 | 0.785 | 0.803 | 0.783 |
| SVM | 0.810 | 0.827 | 0.835 | 0.831 | 0.806 |

MinMaxScaling을 통해 Normaization을 진행 후 모델의 성능을 평가하였다.

SVM과 k-NN을 제외하고는 성능의 향상이 거의 이루어지지 않았다. 하지만 DILI에서와는 달리 Recall값 또한 균등하게 향상된 것을 확인할 수 있다.

2-2-3) SMOTE를 이용한 Oversampling

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score | Balanced-Accuracy |
| Logistic Regression | 0.771 | 0.777 | 0.761 | 0.769 | 0.771 |
| LGBM Classifier | 0.835 | 0.839 | 0.831 | 0.835 | 0.835 |
| Random Forest Classifier | 0.851 | 0.869 | 0.832 | 0.848 | 0.852 |
| k-NN | 0.786 | 0.858 | 0.687 | 0.763 | 0.786 |
| SVM | 0.831 | 0.847 | 0.809 | 0.828 | 0.831 |

Normalization이 진행된 데이터에 대하여 추가적인 SMOTE 기법을 통해 Class의 비율을 동일하게 맞춰준 후 모델의 성능을 평가하였다.

대부분의 경우 약간의 성능향상이 있었으나, Recall의 경우 거의 모든 모델에서 점수가 감소하였다. 가장 성능 향상이 적은 모델은 k-NN이다.

2-3) hERG

2-3-1) 처리 전 모델 성능 평가

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score | Balanced-Accuracy |
| Logistic Regression | 0.679 | 0.700 | 0.788 | 0.741 | 0.657 |
| LGBM Classifier | 0.818 | 0.825 | 0.873 | 0.848 | 0.807 |
| Random Forest Classifier | 0.848 | 0.851 | 0.892 | 0.873 | 0.835 |
| k-NN | 0.698 | 0.726 | 0.777 | 0.751 | 0.683 |
| SVM | 0.585 | 0.584 | 1.000 | 0.738 | 0.502 |

Tree 기반 모델을 제외하면 전반적인 성능이 좋지는 않다. 다만 SVM에서의 Recall 값이 1인 것은 이례적이라고 보여진다. 나머지 모델 성능의 향상을 위해 추가적인 처리를 거쳐 다시 평가를 수행하였다.

2-3-2) Normaization 이후 성능 평가

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score | Balanced-Accuracy |
| Logistic Regression | 0.757 | 0.781 | 0.812 | 0.796 | 0.746 |
| LGBM Classifier | 0.817 | 0.825 | 0.873 | 0.848 | 0.806 |
| Random Forest Classifier | 0.846 | 0.853 | 0.893 | 0.873 | 0.837 |
| k-NN | 0.792 | 0.817 | 0.830 | 0.823 | 0.785 |
| SVM | 0.806 | 0.814 | 0.866 | 0.839 | 0.794 |

MinMaxScaling을 통해 Normaization을 진행 후 모델의 성능을 평가하였다.

DILI에서와 마찬가지로 거리기반 모델의 성능은 전반적으로 향상되었으나, Tree 기반의 경우 이렇다 할 향상 효과를 보이지 않았다.

2-3-3) SMOTE를 이용한 Oversampling

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score | Balanced-Accuracy |
| Logistic Regression | 0.766 | 0.770 | 0.758 | 0.764 | 0.766 |
| LGBM Classifier | 0.845 | 0.847 | 0.842 | 0.844 | 0.845 |
| Random Forest Classifier | 0.877 | 0.879 | 0.877 | 0.877 | 0.876 |
| k-NN | 0.808 | 0.865 | 0.731 | 0.792 | 0.808 |
| SVM | 0.836 | 0.846 | 0.821 | 0.833 | 0.836 |

Normalization이 진행된 데이터에 대하여 추가적인 SMOTE 기법을 통해 Class의 비율을 동일하게 맞춰준 후 모델의 성능을 평가하였다.

전반적인 성능의 향상이 관측되었다. 다만 여전히 0.9 이상으로는 정확도가 올라가지 않고있다. 이는 데이터의 개수 부족 등의 문제라고 판단된다.

**Multi-Class Classification**

1) Stratified 5-fold Cross Validation

Train set과 Test set에서의 Class의 비율을 일정하게 유지하는 Stratified Cross Validation에 n\_splits를 5로 주어 5-fold CV를 수행하였다. 각 Metric 값은 5번 수행에서 나온 결과값의 평균값이다.

2) Modeling

여러 Classifier Model에 대하여 결과를 도출하였다. 각 모델별로 추가적인 처리를 해주었고, 그 중 최대의 수행 점수를 내는 경우를 도출하였다.

2-1) Acute Oral Toxicity: EPA\_category

2-1-1) 처리 전 모델 성능 평가

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score | Balanced-Accuracy |
| Logistic Regression | 0.565 | 0.540 | 0.433 | 0.455 | 0.433 |
| LGBM Classifier | 0.637 | 0.646 | 0.552 | 0.583 | 0.552 |
| Random Forest Classifier | 0.648 | 0.671 | 0.551 | 0.578 | 0.546 |
| k-NN | 0.524 | 0.475 | 0.456 | 0.460 | 0.456 |
| SVM | 0.518 | 0.584 | 0.297 | 0.256 | 0.297 |

전반적으로 모든 모델에 대하여 성능이 좋지 않다. 이는 Class3이 Class1에 비해 5배 가량을 차지하기 때문이라고 생각이 되는데, 이것을 확인하고, 동시에 모델 성능을 향상 시키기 위해 추가적인 처리를 거쳐 다시 평가를 수행하였다.

2-1-2) Normaization 이후 성능 평가

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score | Balanced-Accuracy |
| Logistic Regression | 0.562 | 0.533 | 0.496 | 0.510 | 0.496 |
| LGBM Classifier |  |  |  |  |  |
| Random Forest Classifier |  |  |  |  |  |
| k-NN |  |  |  |  |  |
| SVM |  |  |  |  |  |

MinMaxScaling을 통해 Normaization을 진행 후 모델의 성능을 평가하였다.

DILI에서와 마찬가지로 거리기반 모델의 성능은 전반적으로 향상되었으나, Tree 기반의 경우 이렇다 할 향상 효과를 보이지 않았다.

2-1-3) SMOTE를 이용한 Oversampling

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score | Balanced-Accuracy |
| Logistic Regression | 0.683 | 0.675 | 0.683 | 0.677 | 0.683 |
| LGBM Classifier | 0.804 | 0.808 | 0.804 | 0.804 | 0.804 |
| Random Forest Classifier | 0.838 | 0.840 | 0.840 | 0.839 | 0.836 |
| k-NN | 0.751 | 0.756 | 0.751 | 0.723 | 0.751 |
| SVM |  |  |  |  |  |

Normalization이 진행된 데이터에 대하여 추가적인 SMOTE 기법을 통해 Class의 비율을 동일하게 맞춰준 후 모델의 성능을 평가하였다.

1. Ma et al, 2020: Deep Graph Learning with Property Augmentation for Predicting Drug-Induced Liver Injury [↑](#footnote-ref-1)
2. Kim et al, 2017: Prediction models for drug-induced hepatotoxicity by using weighted molecular fingerprints [↑](#footnote-ref-2)
3. Xu et al, 2021 [↑](#footnote-ref-3)
4. Thakkar et al, 2020: Drug-induced liver injury severity and toxicity (DILIst): binary classification of 1279 drugs by human hepatotoxicity [↑](#footnote-ref-4)
5. Hansen et al, 2009 [↑](#footnote-ref-5)
6. Cai et al, 2019 [↑](#footnote-ref-6)
7. Logistic Regression / LGBM Classifier / Random Forest / k-NN / SVM [↑](#footnote-ref-7)