

데이터셋 정보 확인 및 문제 정의

본 프로젝트는 Tox21 데이터셋을 활용하여 화학 물질의 독성을 예측하는 모델을 개발하는 것을 목표로 한다.

분석에 앞서, `df.info()` 와 `df.describe()` 함수를 사용하여 데이터셋의 전반적인 구조를 파악했다.

```
In [10]: import pandas as pd

df = pd.read_csv('tox21.csv.gz', compression = 'gzip')

# 데이터 정보 확인
df.info()

# 데이터 통계량 요약
df.describe()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7831 entries, 0 to 7830
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   NR-AR            7265 non-null    float64
 1   NR-AR-LBD        6758 non-null    float64
 2   NR-AhR           6549 non-null    float64
 3   NR-Aromatase     5821 non-null    float64
 4   NR-ER             6193 non-null    float64
 5   NR-ER-LBD        6955 non-null    float64
 6   NR-PPAR-gamma    6450 non-null    float64
 7   SR-ARE           5832 non-null    float64
 8   SR-ATAD5          7072 non-null    float64
 9   SR-HSE            6467 non-null    float64
 10  SR-MMP           5810 non-null    float64
 11  SR-p53            6774 non-null    float64
 12  mol_id            7831 non-null    object 
 13  smiles            7831 non-null    object 
dtypes: float64(12), object(2)
memory usage: 856.6+ KB
```

Out[10]:

	NR-AR	NR-AR-LBD	NR-AhR	NR-Aromatase	NR-ER	NR-ER-LBD
count	7265.000000	6758.000000	6549.000000	5821.000000	6193.000000	6955.000000
mean	0.042533	0.035070	0.117270	0.051538	0.128048	0.050324
std	0.201815	0.183969	0.321766	0.221110	0.334170	0.218627
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

분석 결과 요약 및 인사이트

`df.info()` 결과, 총 7831개의 데이터와 14개의 컬럼이 존재함을 확인. 특히, `smiles` 와 `mol_id` 컬럼을 제외한 12가지 독성 관련 컬럼의 Non-Null Count 가 전체 데이터 개수인(7831개)보다 훨씬 적어, 상당한 양의 결측치(NaN)가 존재함을 파악. 이는 모델 학습 이전에 필수적으로 해결해야 할 문제점으로 정의할 수 있음.

또한, `df.describe()` 결과, 12가지 독성 컬럼의 최솟값은 0, 최댓값은 1임을 확인하였고, 이에 더해 25%, 50%, 75% 지점의 값이 모두 0.000000 으로 나와 이 프로젝트는 이진 분류(Binary Classification) 문제임을 알 수 있다. 하지만 평균(mean) 값이 매우 낮고, 75% 지점의 값도 0으로 나타나는 것으로 보아, 대부분의 화합물이 비독성(0)으로 분류되는 심각한 데이터 불균형(Data Imbalance) 문제가 있음을 파악.

데이터의 의미 추론

데이터셋에 대한 별도의 설명이 없어, 컬럼의 값(0과 1)이 무엇을 의미하는지 논리적으로 추론하는 과정을 거쳤음.

`df.describe()` 결과, 12가지 독성 컬럼의 평균값이 매우 낮아 대부분의 값이 0에 집중되어 있음을 확인했음. 일반적인 독성 검사에서는 독성 물질(양성)이 비독성 물질(음성)보다 훨씬 적다는 과학적 배경 지식을 바탕으로, '0'은 비독성(비활성)을, '1'은 독성(활성)을 의미한다고 판단.

데이터 전처리 (Data Preprocessing)

1. 결측치 (NaN) 처리

데이터 셋의 12가지 독성 관련 컬럼에 결측치가 다수 존재하는 것을 확인. 모델 학습의 정확성을 높이기 위해, 해당 컬럼에 결측치가 포함된 모든 행을 삭제하는 방법을 선택.

```
In [12]: # 결측치가 있는 행을 모두 제거
df_cleaned = df.dropna()

# 결측치 제거 후 데이터의 행 개수를 확인
df_cleaned.info()
```

#	Column	Non-Null Count	Dtype
0	NR-AR	3079	float64
1	NR-AR-LBD	3079	float64
2	NR-AhR	3079	float64
3	NR-Aromatase	3079	float64
4	NR-ER	3079	float64
5	NR-ER-LBD	3079	float64
6	NR-PPAR-gamma	3079	float64
7	SR-ARE	3079	float64
8	SR-ATAD5	3079	float64
9	SR-HSE	3079	float64
10	SR-MMP	3079	float64
11	SR-p53	3079	float64
12	mol_id	3079	object
13	smiles	3079	object

dtypes: float64(12), object(2)
memory usage: 360.8+ KB

2. 전처리 결과 확인

결측치를 제거한 결과, 총 7831개였던 데이터가 3079개로 줄어듦. 모든 컬럼의 Non-Null Count 가 3079로 동일해져, 데이터 전처리 과정이 성공적으로 완료되었음을 확인. 이제 깨끗해진 데이터를 사용하여 본격적인 분석을 진행할 수 있음.

```
In [13]: # NR-AR 컬럼의 0과 1의 개수 세기
df_cleaned['NR-AR'].value_counts()
```

```
Out[13]: NR-AR
0.0    3020
1.0     59
Name: count, dtype: int64
```

데이터 불균형 문제 해결

df.info() 와 df.describe() 분석을 통해 데이터셋의 심각한 '데이터 불균형' 문제를 발견. 특히 **12가지 독성 컬럼 모두** 유사한 결측치 및 불균형 문제를 가지고 있어, 이 중 NR-AR 컬럼을 대표로 선택하여 분석을 진행. 이 과정을 통해 얻은 해결책은 다른 컬럼에도 동일하게 적용 가능하다는 가설을 세움.

value_counts() 함수를 사용하여 NR-AR 컬럼의 0과 1의 개수를 세어본 결과, 비독성 샘플(0)이 3020개, 독성 샘플(1)이 59개로, 독성 샘플의 비율이 전체의 약 1.9%에 불과함을 확인. 이는 모델이 독성 물질(1)을 '0'으로 잘못 예측하더라도 높은 정확도를 얻을 수 있어, 독성 물질을 제대로 학습하지 못하는 심각한 문제를 야기.

아래 시각화를 통해 `NR-AR` 컬럼의 데이터 분포를 명확하게 볼 수 있음.

언더샘플링(Undersampling) 진행

이제 이 심각한 데이터 불균형 문제를 해결하기 위해 **언더샘플링**을 진행. 언더샘플링은 비독성(0) 샘플의 수를 독성(1) 샘플의 수와 동일하게 맞춰 데이터의 균형을 맞추는 방법임. 이 과정은 모델이 독성 물질(1)을 '희귀한 데이터'로 인식하지 않고 균형 있게 학습할 수 있게 함.

```
In [14]: # NR-AR 컬럼의 0과 1의 개수 확인
count_toxic = df_cleaned[df_cleaned['NR-AR'] == 1].shape[0]

# 독성(1) 샘플과 동일한 수의 비독성(0) 샘플을 무작위로 추출
df_nontoxic_under = df_cleaned[df_cleaned['NR-AR'] == 0].sample(count_toxic, rand

# 추출된 비독성 샘플과 모든 독성 샘플을 합쳐 새로운 데이터프레임 생성
df_balanced = pd.concat([df_nontoxic_under, df_cleaned[df_cleaned['NR-AR']==1]],

# 새로운 데이터프레임의 NR-AR 컬럼 분포 확인
df_balanced['NR-AR'].value_counts()
```

```
Out[14]: NR-AR
0.0    59
1.0    59
Name: count, dtype: int64
```

언더샘플링 결과

언더샘플링을 통해 비독성(0) 샘플과 독성(1) 샘플의 비율을 59개로 동일하게 맞추는 데 성공했음. 이로써 모델이 특정 클래스에 편향되지 않고 균형 있게 학습할 수 있는 데이터셋을 확보함.

분자 특징 추출 (Molecular Feature Engineering)

이제 깨끗하고 균형 잡힌 데이터를 바탕으로, `smiles` 문자열 데이터를 모델이 이해할 수 있는 숫자 형태로 변환할 것. 이런 과정을 **분자 특징 추출(Molecular Feature Engineering)**이라 부름

이를 위해 화학정보학 라이브러리인 **RDKit**을 사용. RDKit을 사용하면 각 화합물의 분자량을 포함한 217가지의 물리화학적 속성인 **분자 디스크립터(Molecular Descriptors)**를 추출할 수 있음. 이러한 디스크립터는 모델의 입력 데이터로 사용됨.

```
In [15]: # RDKit 설치
!pip install rdkit
```

Collecting rdkit

```
  Downloading rdkit-2025.3.5-cp313-cp313-win_amd64.whl.metadata (4.2 kB)
Requirement already satisfied: numpy in c:\users\dbozgm\anaconda3\lib\site-packages (from rdkit) (2.1.3)
Requirement already satisfied: Pillow in c:\users\dbozgm\anaconda3\lib\site-packages (from rdkit) (11.1.0)
  Downloading rdkit-2025.3.5-cp313-cp313-win_amd64.whl (23.5 MB)
----- 0.0/23.5 MB ? eta ---:--
----- 6.0/23.5 MB 263.9 kB/s eta 0:01:07
----- ...
```

In []:

```
from rdkit import Chem
from rdkit.Chem import Descriptors
import pandas as pd

# smiles의 문자열 데이터를 RDKit 문자 객체로 변환하는 함수
def smiles_to_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None

    # 217가지 문자 디스크립터 추출
    descriptor_list = Descriptors.descList
    descriptor_names = [x[0] for x in descriptor_list]
    descriptors = [x[1](mol) for x in descriptor_list]

    return pd.Series(descriptors, index=descriptor_names)

# df_balanced 데이터셋의 smiles 컬럼에서 문자 디스크립터 추출
descriptors_df = df_balanced['smiles'].apply(smiles_to_descriptors)

# 결측값이 포함된 행 제거
descriptors_df.dropna(inplace=True)

# 결측값 제거 후 데이터프레임 정보 확인
descriptors_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 118 entries, 3730 to 7828
Columns: 217 entries, MaxAbsEStateIndex to fr_urea
dtypes: float64(217)
memory usage: 201.0 KB
```

결과를 통해 문자 특징 추출이 성공적으로 완료된 것을 알 수 있음

- 118 entries: 언더샘플링을 통해 균형을 맞춘 데이터셋(`df_balanced`)의 행 수와 정확히 일치. 즉, 모든 데이터가 문제없이 변환되었음을 의미.
- 217 entries: `smiles` 문자열 데이터가 217가지의 새로운 컬럼(문자 디스크립터)으로 성공적으로 변환됨. 이제 모델이 이해 가능한 숫자 형태의 데이터가 준비됨.
- `float64` dtypes: 217개의 모든 컬럼의 데이터 타입이 모델 학습에 적합한 `float64`임을 확인.

In [18]:

```
# df_balanced와 descriptors_df를 인덱스를 기준으로 병합
final_df = pd.merge(df_balanced[['NR-AR']], descriptors_df, left_index=True, right_index=True)

# 최종 데이터프레임 정보 확인
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 118 entries, 3730 to 7828
Columns: 218 entries, NR-AR to fr_urea
dtypes: float64(218)
memory usage: 206.0 KB
```

데이터 병합 결과

`df.info()` 결과를 보니, 데이터 병합이 성공적으로 완료됨.

- `118 entries` : 언더샘플링으로 균형을 맞춘 118개의 행이 그대로 유지
- `218 columns` : 이전에 추출한 217개의 문자 디스크립터에 `NR-AR`이라는 결과값 (Target) 컬럼 1개가 추가되어 총 218개의 컬럼이 됨.
- `Non-Null Count` : 모든 컬럼의 `Non-Null Count` 가 118로, 결측치 없이 모든 데이터가 온전함.

==> 모델을 학습시킬 준비 완료.

모델 학습

데이터를 모델 학습에 사용할 수 있도록 준비하고, 로지스틱 회귀(Logistic Regression)모델을 사용하여 독성 여부를 예측.

모델 선택: 로지스틱 회귀

본 프로젝트는 특정 화학 물질의 독성 여부(0 또는 1)를 예측하는 이진 분류 문제임. 이에 가장 적합하고 해석하기 쉬운 모델인 **로지스틱 회귀**를 선택. 이 모델은 결과에 대한 설명력이 높고, 학습 속도가 빠르며, 초기 성능을 확인하기에 효율적.

1. 데이터셋 분리

학습용 데이터셋을 `x`(입력 데이터)와 `y`(결과값)로 분리하는 과정. `x`는 217가지의 문자 디스크립터, `y`는 `NR-AR` 컬럼이 됨.

2. 학습/테스트 데이터 분할

모델의 성능을 평가하기 위해 전체 데이터셋을 학습용(train)과 테스트용(test) 데이터로 분할. 데이터의 불균형 문제를 고려하여 `stratify` 파라미터를 사용 일반적으로 학습용 데이터셋은 70~80% 비율로 해당 프로젝트에서는 8:2 비율로 분할.

```
In [32]: from sklearn.model_selection import train_test_split
import pandas as pd

# 최종 데이터 프레임의 NR-AR 컬럼을 y(결과값)로 설정
y = final_df['NR-AR']

# NR-AR 컬럼을 제외한 나머지 컬럼을 x(입력 데이터)로 설정
x = final_df.drop('NR-AR', axis = 1)

# 학습과 테스트 데이터로 분할
# test_size = 0.2는 전체 데이터의 20%를 테스트용으로 사용한다는 의미
# random_state = 42는 결과를 재현 가능하게 함
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state = 42)

# 분할된 데이터셋의 크기 확인
print("학습 데이터셋 크기:", x_train.shape)
print("테스트 데이터셋 크기:", x_test.shape)
print("학습 라벨셋 크기:", y_train.shape)
print("테스트 라벨셋 크기", y_test.shape)

# 학습용 데이터의 0과 1 개수 확인
print("학습용 데이터셋의 클래스별 개수:")
print(pd.Series(y_train).value_counts())

# 테스트용 데이터의 0과 1 개수 확인
print("\n테스트용 데이터셋의 클래스별 개수:")
print(pd.Series(y_test).value_counts())
```

학습 데이터셋 크기: (94, 217)

테스트 데이터셋 크기: (24, 217)

학습 라벨셋 크기: (94,)

테스트 라벨셋 크기 (24,)

학습용 데이터셋의 클래스별 개수:

NR-AR

1.0 47

0.0 47

Name: count, dtype: int64

테스트용 데이터셋의 클래스별 개수:

NR-AR

1.0 12

0.0 12

Name: count, dtype: int64

- `train_test_split` : 전체 데이터를 8:2 비율로 분할.
- `stratify = y` : 원본 데이터셋의 클래스 비율(`y`의 0과 1 비율)을 학습용 및 테스트용 데이터셋에 동일하게 유지. 이는 모델 학습에 사용되지 않는 테스트 데이터셋이 실제 데이터 분포를 대표하도록 보장.
- `random_state = 42` : 분석의 재현성 확보

이어서 각각 포함된 0과 1의 개수를 직접 확인하여 비율이 잘 유지되었는지 검증.

모델 학습 및 훈련

데이터 준비 완료. 선택한 로지스틱 회귀 모델을 이용하여 독성 여부 예측 모델 학습 진행. 입력(`x`)과 결과(`y`) 데이터 간의 관계를 학습하여 예측 능력 습득.

모델 생성 및 훈련

모델 성능 확보를 위해 하이퍼파라미터 설정 후 모델 생성. 이후 `fit()` 메소드로 학습 데이터인 `x_train` 과 `y_train`을 이용한 훈련 진행

In [30]: `from sklearn.linear_model import LogisticRegression`

```
model = LogisticRegression(max_iter = 1000, random_state = 42)
model.fit(x_train,y_train)
```

Out[30]:

```
LogisticRegression(max_iter=1000, random_state=42)
```

- max_iter=1000: 수렴 횟수 지정. 기본값(100) 수렴 실패 경고 방지 및 학습 안정성 확보 목적.
- random_state = 42: 동일 결과 재현을 위한 난수 시드 고정. 분석 과정의 일관성 보장.

학습 결과

위 코드를 통해 모델 학습 완료. 이제 모델은 주어진 문자 디스크립터를 기반으로 독성 여부 예측 가능.

모델 성능 평가

모델 학습 완료 후, 모델이 학습에 사용되지 않은 새로운 데이터(x_test)에 대해 얼마나 정확하게 예측하는지 평가. 이는 모델의 일반화 성능을 확인하는 중요한 단계.

정확도(Accuracy) 계산

모델의 예측 성능을 측정하기 위해 정확도(Accuracy)지표를 사용. 정확도는 모델이 테스트 데이터셋에서 올바르게 예측한 샘플의 비율을 나타냄.

1. 예측값 산출: 훈련된 모델을 사용하여 테스트 입력 데이터(x_test)에 대한 예측값을 y_pred에 저장.
2. 정확도 측정: accuracy_score 함수를 사용하여 y_test와 y_pred를 비교, 정확도를 계산.

In [37]:

```
from sklearn.metrics import accuracy_score

# 1. 모델이 테스트 데이터(x_test)에 대해 예측
y_pred = model.predict(x_test)

# 2. 예측 결과(y_pred)와 실제 정답(y_test)을 비교하여 정확도 계산
accuracy = accuracy_score(y_test, y_pred)

print(f"모델의 정확도: {accuracy:.2f}")
```

모델의 정확도: 0.50

오분류표(Confusion Matrix) 확인

모델의 정확도가 50%로 낮게 나와, 오분류표를 통해 모델의 예측 오류를 더 자세히 분석. 오분류표는 모델이 예측한 값과 실제 정답이 어떻게 일치하고 불일치하는지 한눈에 보여주는 표.

- 진짜 양성 (True Positive, TP): 모델이 독성(1)이라고 예측했고, 실제로 독성인 경우.
- 가짜 양성 (False Positive, FP): 모델이 독성(1)이라고 예측했지만, 실제는 비독성(0)인 경우.
- 진짜 음성 (True Negative, TN): 모델이 비독성(0)이라고 예측했고, 실제로 비독성인 경우.
- 가짜 음성 (False Negative, FN): 모델이 비독성(0)이라고 예측했지만, 실제는 독성(1)인 경우.

이 네 가지 지표로 모델이 어떤 유형의 실수를 더 많이 하는지 파악.

```
In [38]: from sklearn.metrics import confusion_matrix

# 오분류표(Confusion Matrix) 생성
cm = confusion_matrix(y_test, y_pred)
print("오분류표(Confusion Matrix):")
print(cm)
```

오분류표(Confusion Matrix):

```
[[ 0 12]
 [ 0 12]]
```

오분류표 분석

단순 정확도(Accuracy)가 50%라는 낮은 수치를 보임에 따라, 모델이 어떤 유형의 예측 오류를 범했는지 파악하기 위해 오분류표를 확인. 이 표는 모델의 예측 결과와 실제 정답을 교차하여 보여줌.

[[TN FP], [FN TP]]

- 결과 분석: 오분류표 [[0, 12], [0, 12]] 분석 결과, 모델은 모든 예측을 독성 물질(1)로 진행.
 - 진짜 양성(TP): 실제 독성 물질(12개)을 모두 정확하게 독성으로 예측.
 - 가짜 양성(FP): 실제로는 비독성 물질(12개)임에도 불구하고, 모두 독성으로 잘못 예측.
 - 진짜 음성(TN) & 가짜 음성(FN): 비독성 물질(0)에 대한 예측은 단 한 건도 맞히지 못함.

이는 모델이 데이터 내에서 유의미한 패턴을 찾지 못해, 가장 단순한 전략인 '모든 것을 1로 예측'하는 방식을 택했음을 의미함. 이로 인해 정확도가 50%라는 수치에 머무르게 됨.

성능 보고서(Classification Report) 분석

오분류표를 통해 파악한 모델의 편향된 성능을 정량화하고 세부적으로 분석함. 성능 보고서는 정밀도(Precision), 재현율(Recall), F1-점수(F1-Score) 등 다양한 지표를 통해 각 클래스에 대한 모델의 성능을 종합적으로 보여줌.

```
In [39]: from sklearn.metrics import classification_report

report = classification_report(y_test, y_pred)
```

```
print("성능 보고서(Classification Report):")
print(report)
```

성능 보고서(Classification Report):

	precision	recall	f1-score	support
0.0	0.00	0.00	0.00	12
1.0	0.50	1.00	0.67	12
accuracy			0.50	24
macro avg	0.25	0.50	0.33	24
weighted avg	0.25	0.50	0.33	24

```
c:\Users\dbozgm\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:156
5: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in label
s with no predicted samples. Use `zero_division` parameter to control this behav
ior.

_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\dbozgm\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:156
5: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in label
s with no predicted samples. Use `zero_division` parameter to control this behav
ior.

_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\dbozgm\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:156
5: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in label
s with no predicted samples. Use `zero_division` parameter to control this behav
or.

_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

- 클래스 0.0(비독성):
 - 정밀도, 재현율, F1-점수 모두 0.00을 기록함. 이는 모델이 비독성 물질에 대한 예측을 단 한 번도 수행하지 않았기 때문. 이로 인해 `UndefinedMetricWarning` 경고가 발생
- 클래스 1.0(독성):
 - 정밀도(Precision): 0.5 모델이 독성이라고 예측한 것 중 절반만 실제로 독성이었음을 의미.
 - 재현율(Recall): 1.00 실제 독성 물질을 하나도 놓치지 않고 100% 모두 찾아냈음을 의미.

이 보고서는 모델이 독성 물질 예측에는 완벽했으나, 비독성 물질을 전혀 예측하지 못한 채 모두 독성으로 분류하는 편향된 행동을 보였음을 명확히 보여줌.

모델 개선: 결정 트리

로지스틱 회귀 모델이 50%라는 낮은 정확도를 보인 것을 확인. 이는 모델이 독성과 비독성을 구분할 만한 명확한 선형 패턴을 찾지 못했기 때문으로 판단됨. 따라서 더 복잡한 비선형 패턴을 학습할 수 있는 **결정 트리(Decision Tree) 모델을 사용해 성능 개선을 시도.

```
In [40]: from sklearn.tree import DecisionTreeClassifier

# 결정 트리 모델을 만들고 훈련
dt_model = DecisionTreeClassifier(random_state = 42)
```

```
dt_model.fit(x_train, y_train)

# 테스트 데이터로 예측을 수행
y_pred_dt = dt_model.predict(x_test)

# 분류 보고서 출력
print("결정 트리 모델 성능 보고서:")
report_dt = classification_report(y_test,y_pred_dt)
print(report_dt)
```

결정 트리 모델 성능 보고서:

	precision	recall	f1-score	support
0.0	0.70	0.58	0.64	12
1.0	0.64	0.75	0.69	12
accuracy			0.67	24
macro avg	0.67	0.67	0.66	24
weighted avg	0.67	0.67	0.66	24

결정 트리 모델 성능 분석

로지스틱 회귀 모델의 낮은 정확도(50%)를 개선하기 위해, 비선형 패턴을 학습할 수 있는 결정 트리 모델을 사용.

- 결과 분석:

- 전체 정확도(Accuracy)가 67%로 크게 상승하여, 모델이 데이터를 통해 유의미한 패턴을 학습했음을 보여줌.
- 클래스 0.0(비독성)과 클래스 1.0(독성) 모두에서 정밀도와 재현율이 0.50 이상으로 나타나, 모든 클래스를 균형 있게 예측하고 있음을 확인.

이 결과는 로지스틱 회귀와 달리 결정 트리가 데이터 내의 복잡한 규칙을 비교적 성공적으로 찾아냈음을 의미함.

결정트리 모델 성능 분석 및 통합

결정 트리는 그 예측 과정을 시각적으로 확인할 수 있다는 장점이 있음. 이를 통해 모델이 어떤 기준으로 독성 여부를 판단했는지 명확히 파악할 수 있음. `export_graphviz` 함수를 사용하여 모델의 예측 규칙을 `.dot` 파일로 생성하고, 이를 웹사이트에서 시각화.

In [42]:

```
from sklearn.tree import export_graphviz

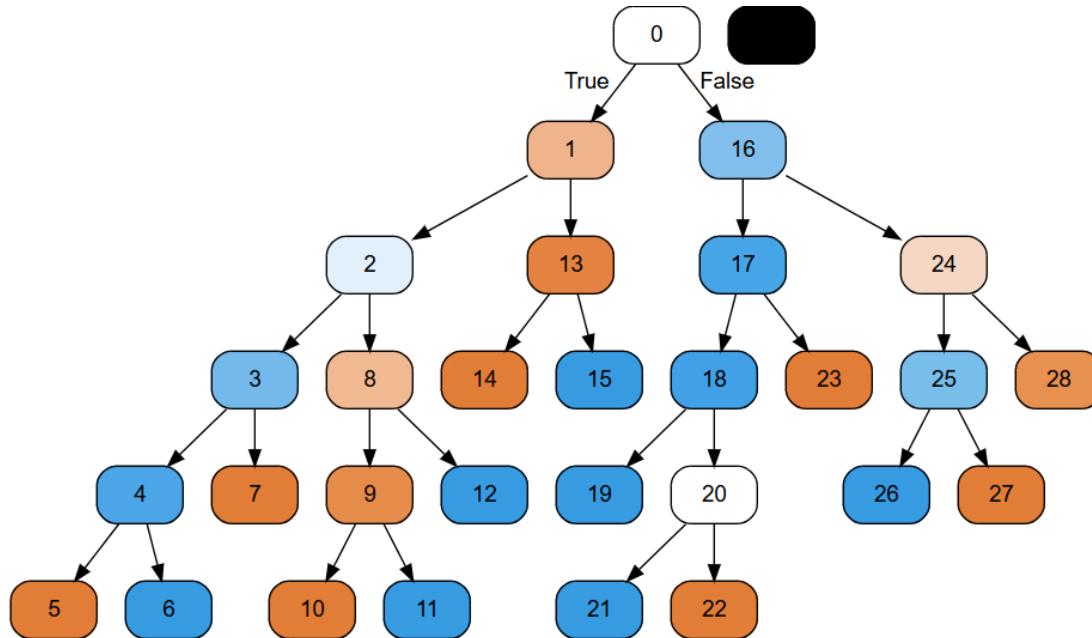
# 결정 트리 시각화를 위한 dot 파일 생성
# feature_names와 class_names를 지정하여 결과를 더 쉽게 이해할 수 있게 함
dot_data = export_graphviz(dt_model, out_file = None, feature_names=x.columns, c
```

```
# 생성된 dot 파일 내용 출력
print(dot_data)
```

```

digraph Tree {
    node [shape=box, style="filled, rounded", color="black", fontname="helvetica"] ;
    edge [fontname="helvetica"] ;
    0 [label=<HeavyAtomMolWt &le; 223.16<br/>gini = 0.5<br/>samples = 94<br/>value = [47, 47]<br/>class = 비독성>, fillcolor="#ffffff" ] ;
    1 [label=<BCUT2D_LOGPHI &le; 2.06<br/>gini = 0.42<br/>samples = 50<br/>value = [35, 15]<br/>class = 비독성>, fillcolor="#f0b78e" ] ;
    0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
    2 [label=<VSA_EState8 &le; 1.871<br/>gini = 0.497<br/>samples = 26<br/>value = [12, 14]<br/>class = 독성>, fillcolor="#e3f1fb" ] ;
    1 -> 2 ;
    3 [label=<SMR_VSA5 &le; 16.304<br/>gini = 0.355<br/>samples = 13<br/>value = [3, 10]<br/>class = 독성>, fillcolor="#74baed" ] ;
    .....길이로 인한 편집.....
    2 -> 3 ;
    27 [label=<gini = 0.0<br/>samples = 2<br/>value = [2, 0]<br/>class = 비독성>, fillcolor="#e58139" ] ;
    25 -> 27 ;
    28 [label=<NumValenceElectrons &le; 106.0<br/>gini = 0.198<br/>samples = 9<br/>value = [8, 1]<br/>class = 비독성>, fillcolor="#e89152" ] ;
    24 -> 28 ;
    29 [label=<gini = 0.0<br/>samples = 1<br/>value = [0, 1]<br/>class = 독성>, fillcolor="#399de5" ] ;
    28 -> 29 ;
    30 [label=<gini = 0.0<br/>samples = 8<br/>value = [8, 0]<br/>class = 비독성>, fillcolor="#e58139" ] ;
    28 -> 30 ;
}

```



모델 예측 과정 시각화 및 통찰

복잡한 코드를 그대로 보면 인간이 이해하기 어렵기 때문에 **시각화 도구**를 이용한다.

`dot` 코드를 시각화 웹사이트(<http://www.webgraphviz.com/>)로 변환하여 모델의 판단 기준을 명확하게 확인할 수 있었음. `dot` 코드 분석을 통해 모델이

`HeavyAtomMolWt`, `BCUT2D_LOGPHI` 같은 문자 특성들을 단계적으로 확인하며 독성 여부를 분류했음을 확인. 결정 트리를 통해 예측의 근거를 명확히 파악할 수 있음.

67%라는 정확도가 절대적인 수치로는 낮아 보일 수 있지만, 초기 모델의 무작위 예측(50%)에 비해 유의미한 개선을 이루었음. 특히 데이터 불균형이 심한 독성 물질 예측 문제에서는 모델이 유의미한 패턴을 학습했다는 중요한 지표.

추후 개선 사항

이러한 방법론은 나머지 11개 컬럼에도 동일하게 적용해 본 결과, 대부분은 로지스틱 회귀 모델보다 좋아졌는데, 몇몇 컬럼은 데이터셋의 한계 때문인지 0.49처럼 낮게 나옴. 모든 문제가 하나의 방법으로 해결되는 건 아니라는 걸 배웠고, 앞으로는 여러 종류의 모델을 동원하는 양상들이나 하나의 모델이더라도 내부가 여러층으로 설계되어 매우 깊은 딥러닝 같은 걸 추후에 시도해 볼 것.