



# 약물대사 안정성 예측

딥러닝 개인 프로젝트  
강종범



# 주제 설명

딥러닝 개인 프로젝트  
강종범

# 목차

---

## 1. 주제 선정 이유

## 2. 주제 설명

- ADME/T
- 다양한 머신러닝 기법 적용 가능
- QSAR
- 데이터 셋

# 소개

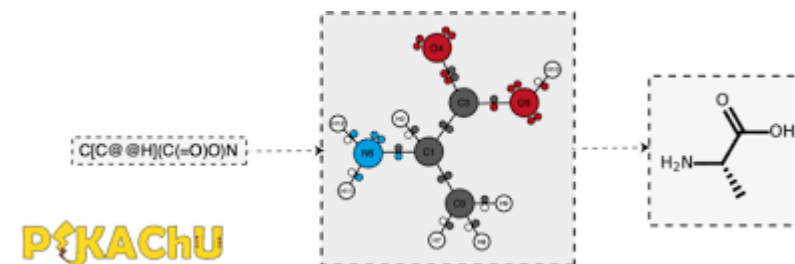
## 주제 선정 이유

- 여러가지 화학 관련 라이브러리 및 화학 데이터에 대한 이해가 가능하다.
- 여러가지 머신러닝 기법을 적용할 수 있다.
- Drug의 주요 특성(ADME/T)중 하나인 metabolism의 값을 예측 함으로써 약물 개발시간을 획기적으로 낮출 수 있다.
- 아쉬워서



Open-Source Cheminformatics  
and Machine Learning

Vanillin		<chem>O=Cc1ccc(OC)c(OC)c1</chem> <chem>COc1cc(C=O)ccc1O</chem>
----------	--	---

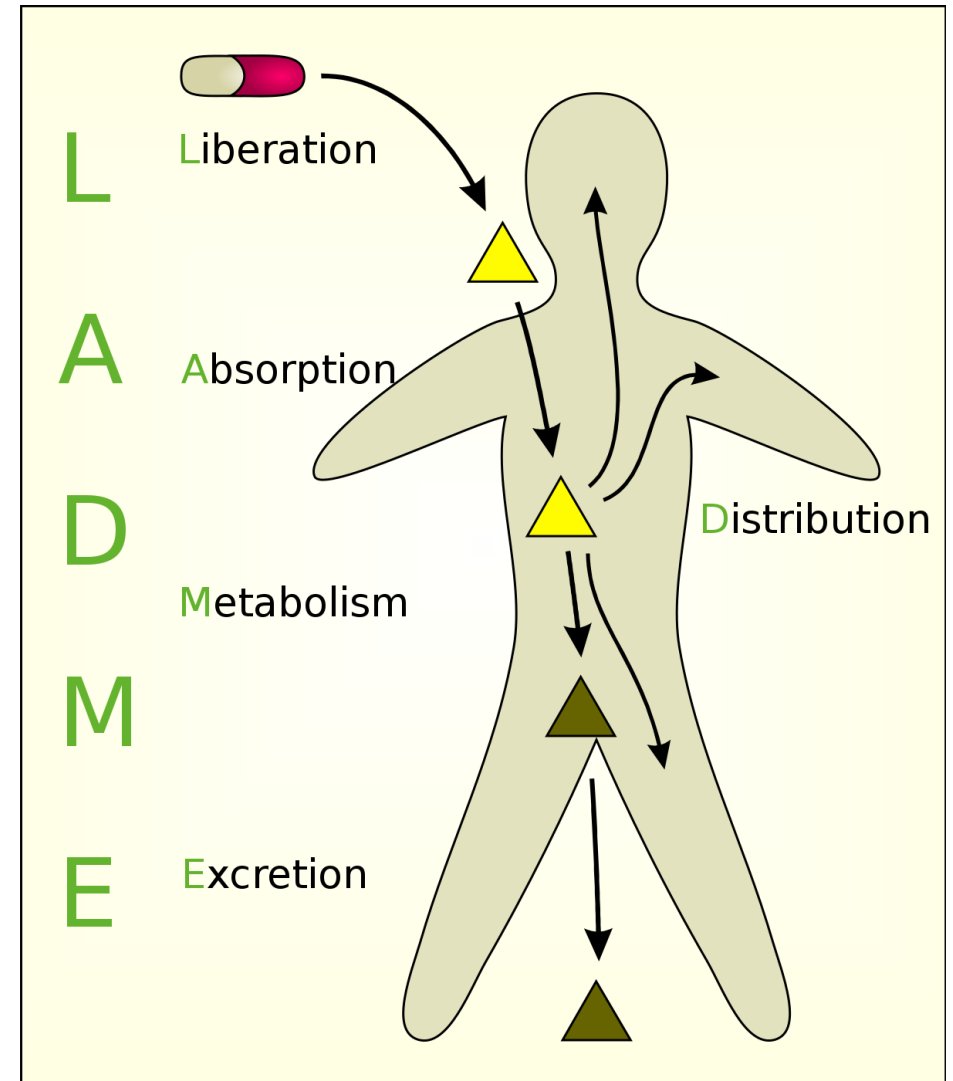


# 주제 설명

## ADME/T

- 체내 흡수 / 체내 분포 / 신진대사 / 배출 / 독성
- Metabolism은 약물을 신진대사를 의미
- 낮은 대사 안정성은 신약 개발의 실패를 야기한다. 실험을 통해 알아내는 것은 상당히 번거롭고 비싼 과정이므로 만약 구조적, 분자의 다른 특징을 통해 계산 할 수 있다면, 상당히 유용할 것으로 기대됨.

Ex) 타이레놀 섭취 후 10분만에 다 대사 되어 버리면?



# 주제 설명

## 여러가지 머신러닝 기법 적용 가능

화합물	분자량	Log P	HBD	HBA	...	대사값
C=OC...N	...	...	...	...	...	47.1
CC...OH	...	...	...	...	...	99.9
CC...C#C	...	...	...	...	...	0.01



RandomForest ,XGBOO  
ST, LIGHTGBM 등등

인공신경망 모델도 가능

여러가지 Feature을 계산을 통해 직접  
추가하는 과정 진행

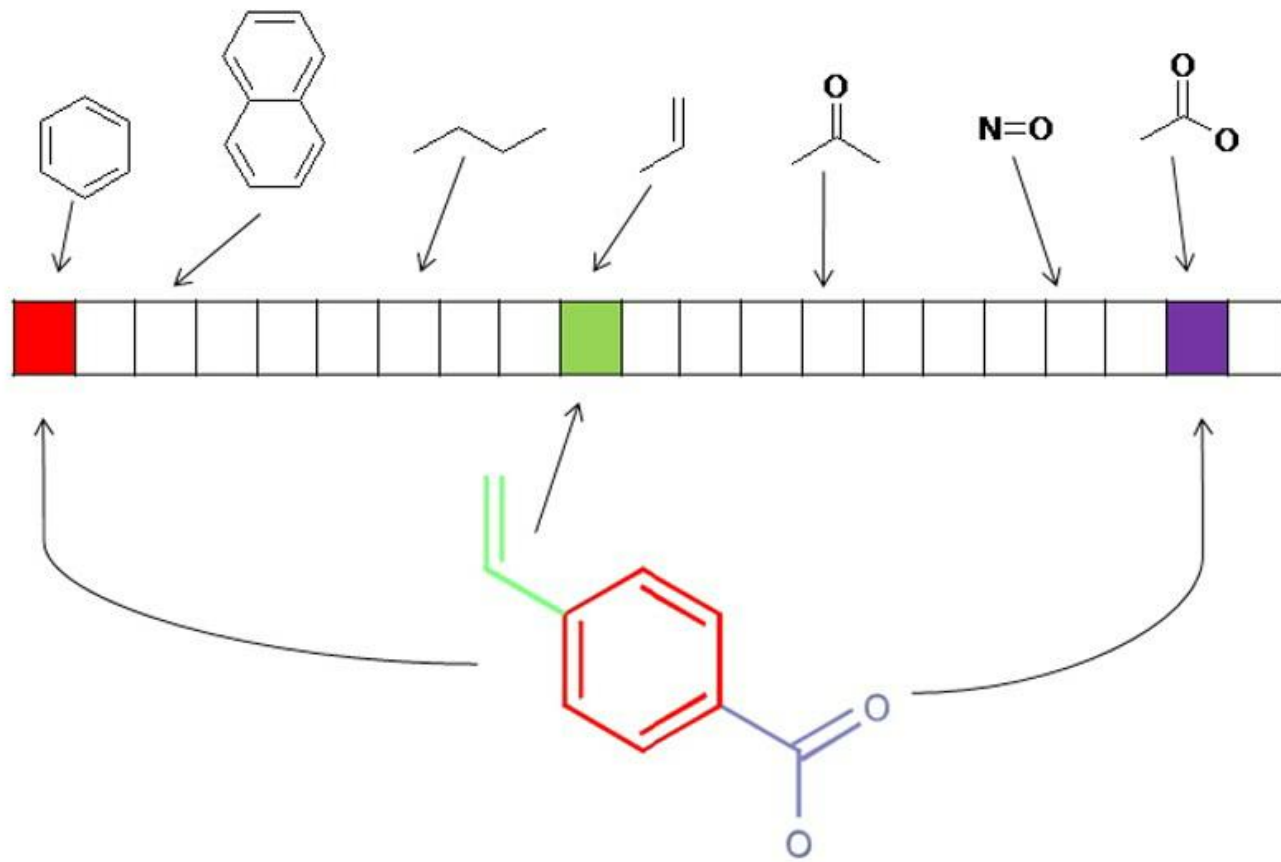


# 주제 설명

## 여러가지 머신러닝 기법 적용 가능

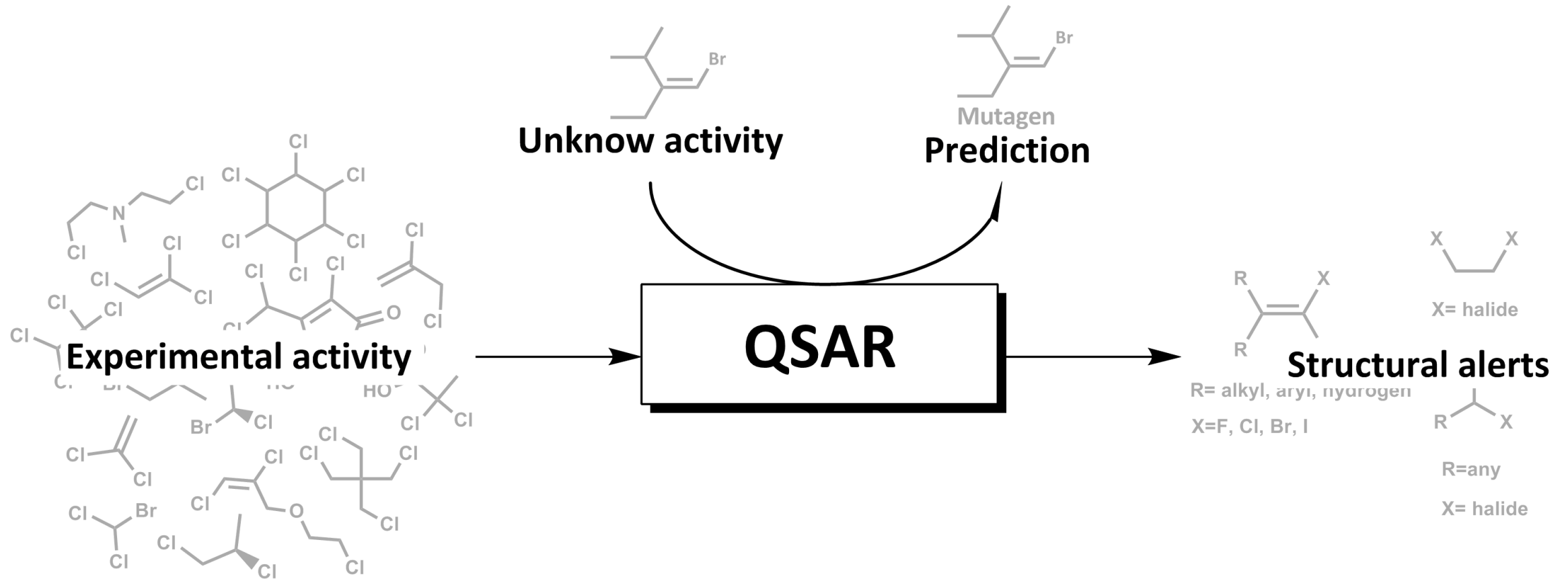
### - Fingerprint

좀 더 분자 구조에 기반한 구조기반  
특징을 추출 할 수도 있다.



# 주제 설명

## QSAR





# 주제 설명

## 데이터 셋

- KRICT 등에서 주관한 제 1회 신약개발 AI 경진 대회 : 대사 안정성 예측모델에서 제공한 데이터셋 활용

### 1. train.csv

- 3498개 화합물 정보
- id: 샘플 고유 id
- SMILES: 화합물의 분자 구조
- 화합물의 물성 관련 정보 (LogD의 경우 pH값 7.4에서 계산)

- MLM(Mouse Liver Microsome) 포함
- HLM(Human Liver Microsome) 포함
- HLM 및 MLM은 간 및 마우스의 간 대사효소와 화합물을 30분 동안 반응시킨 후,
- 대사되지 않고 남아있는 화합물의 양을(%) LC-MS/MS로 측정함으로써 화합물의 간 대사효소에 대한 안정성을 평가한 데이터

### 2. test.csv

- 483개 화합물 정보
- id: 샘플 고유 id
- SMILES: 화합물의 분자 구조
- 화합물의 물성 관련 정보 (LogD의 경우 pH값 7.4에서 계산)



# 중간 과정

딥러닝 개인 프로젝트  
강종범

# 목차

## 1. 진행 상황

- 전처리 / EDA
- Feature Engineering
- 데이터셋 분리
- 각 접근 모델
- 모델 단순 평균

## 2. 개선방안



Open-Source Cheminformatics  
and Machine Learning



# 진행 상황

## 전처리 / EDA

SMILES	MLM	HLM	AlogP	Molecular_Weight	Num_H_Acceptors	Num_H_Donors	Num_RotatableBonds	LogD	Molecular_PolarSurfaceArea
[2)cc1OCC	26.01	50.68	3.259	400.495	5	2	8	3.259	117.37
cccc32)s1	29.27	50.59	2.169	301.407	2	1	2	2.172	73.47

# 필요없는 특성 분리 (ID <- SMILES로 대체 가능)

### 1. 전처리

결측치 / 중복값 / 이상치 확인

# 진행 상황

▶	1 train.isnull().sum()
📄	SMILES 0
	MLM 0
	HLM 0
	AlogP 2
	Molecular_Weight 0
	Num_H_Acceptors 0
	Num_H_Donors 0
	Num_RotatableBonds 0
	LogD 0
	Molecular_PolarSurfaceArea 0
	dtype: int64
▶	1 test.isnull().sum()
	SMILES 0
	AlogP 1
	Molecular_Weight 0
	Num_H_Acceptors 0
	Num_H_Donors 0
	Num_RotatableBonds 0
	LogD 0
	Molecular_PolarSurfaceArea 0
	dtype: int64

## 전처리 / EDA

### 1. 결측치 존재

- train df 에서 AlogP 2개 / test df 에서 AlogP 1개 존재.

- RDKit을 이용해 계산된 AlogP값으로 대체하여 결측치 해결
- LogD 값으로 대체 가능 상관성 매우 높음)



Open-Source Cheminformatics  
and Machine Learning

# 진행 상황

## 전처리 / EDA

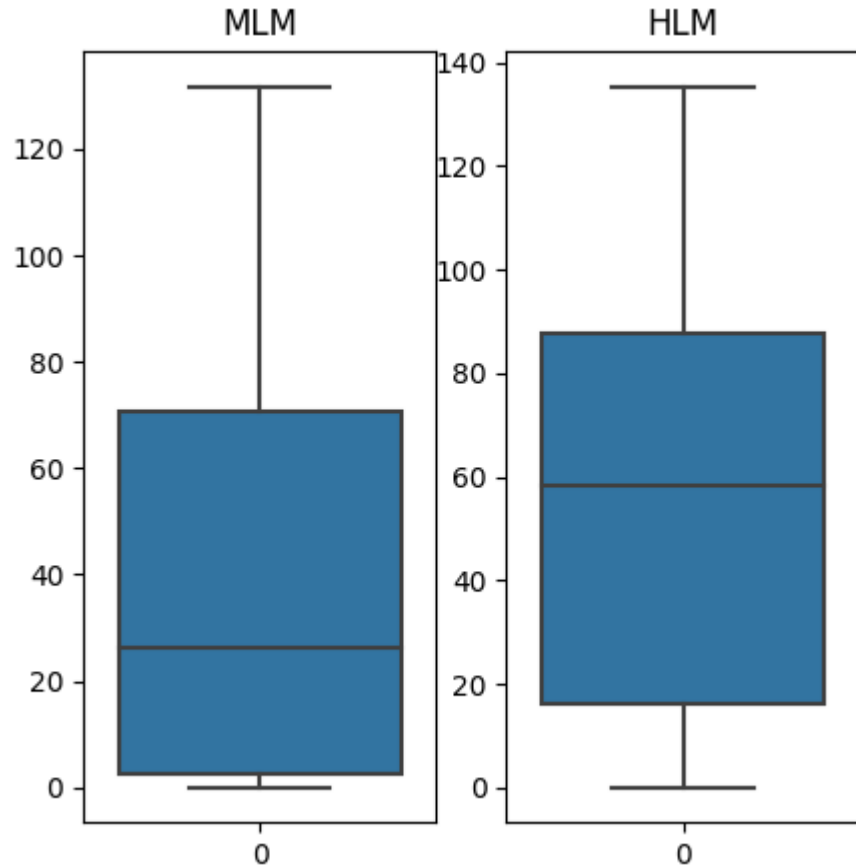
겹치는 분자는 총 27.0쌍

	SMILES	MLM	HLM	AlogP	Molecular_Weight	Num_H_Acceptors	Num_H_Donors	Num_RotatableBonds	LogD	Molecular_PolarSurfaceArea
2276	<chem>C(=C/c1nnn(Cc2ccccc2)n1)Wc1ccccc1</chem>	1.535	31.453	3.556	262.309	3	0	4	3.556	43.60
451	<chem>C(=C/c1nnn(Cc2ccccc2)n1)Wc1ccccc1</chem>	0.310	24.670	3.556	262.309	3	0	4	3.556	43.60
2891	<chem>CC(=O)Nc1ccc(N2N=C(c3ccc(O)cc3)C(C)CC2=O)cc1</chem>	55.950	69.950	2.172	337.372	4	2	3	2.169	82.00
543	<chem>CC(=O)Nc1ccc(N2N=C(c3ccc(O)cc3)C(C)CC2=O)cc1</chem>	68.485	85.872	2.172	337.372	4	2	3	2.169	82.00
837	<chem>CC(=O)Nc1nc2ccc(-c3nn(C(C)C)c4nc(N)ncc34)cc2s1</chem>	63.522	62.488	2.293	367.428	5	2	3	2.307	139.85
366	<chem>CC(=O)Nc1nc2ccc(-c3nn(C(C)C)c4nc(N)ncc34)cc2s1</chem>	73.740	66.850	2.293	367.428	5	2	3	2.307	139.85

## 2. 중복된 데이터 존재

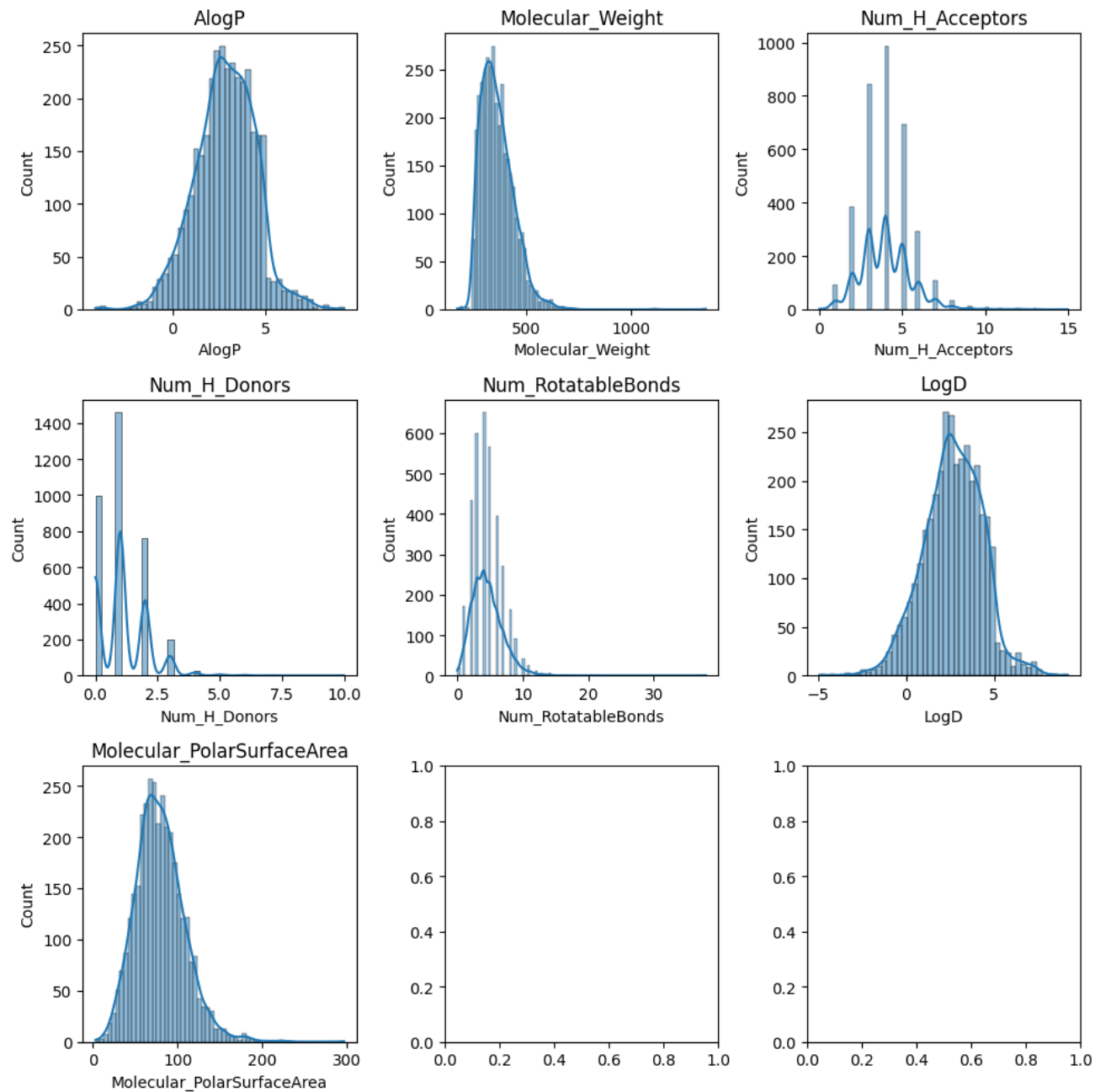
- 27쌍 (54개) 존재
- 문제점 : 단순히 중복제거 처리하기엔 결과값이 다름, 일부는 상당히 차이가 있음
- HLM , MLM에서 하나라도 40 이상의 차이가 있으면 해당 분자는 모두제거 (54개중 18개 제거)
- 남은 데이터는 MAX기준으로 하나만 남김. (남은 중복 데이터 36개중 18개만 남김)

## 전처리 / EDA



### 3. 비정상 데이터 존재

- 상식적으로는 결과값이 0- 100 사이에 있어야함.
- 결과값이 100을 넘는 데이터가 일부 존재
- HLM , MLM에서 100이 넘는 결과값은 모두 100으로 바꿔줌
- 그냥 놔두는게 모델 점수는 더 좋을 수도 있음.



## 전처리 / EDA

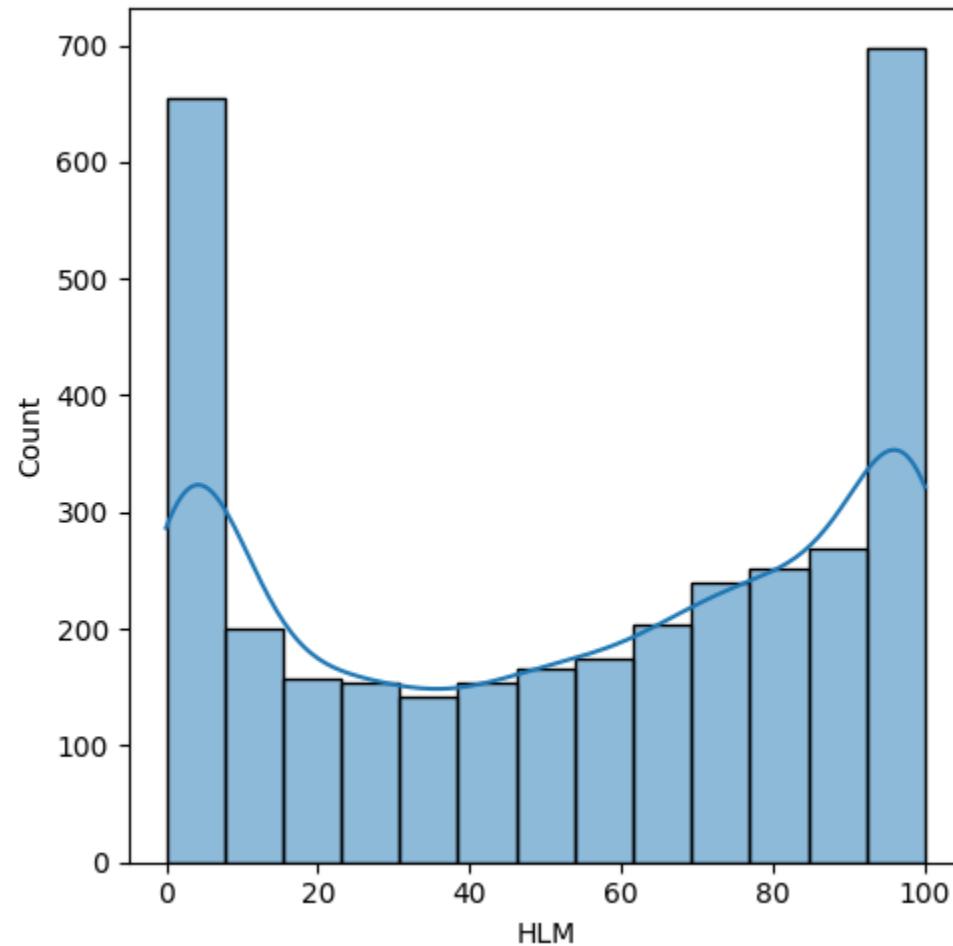
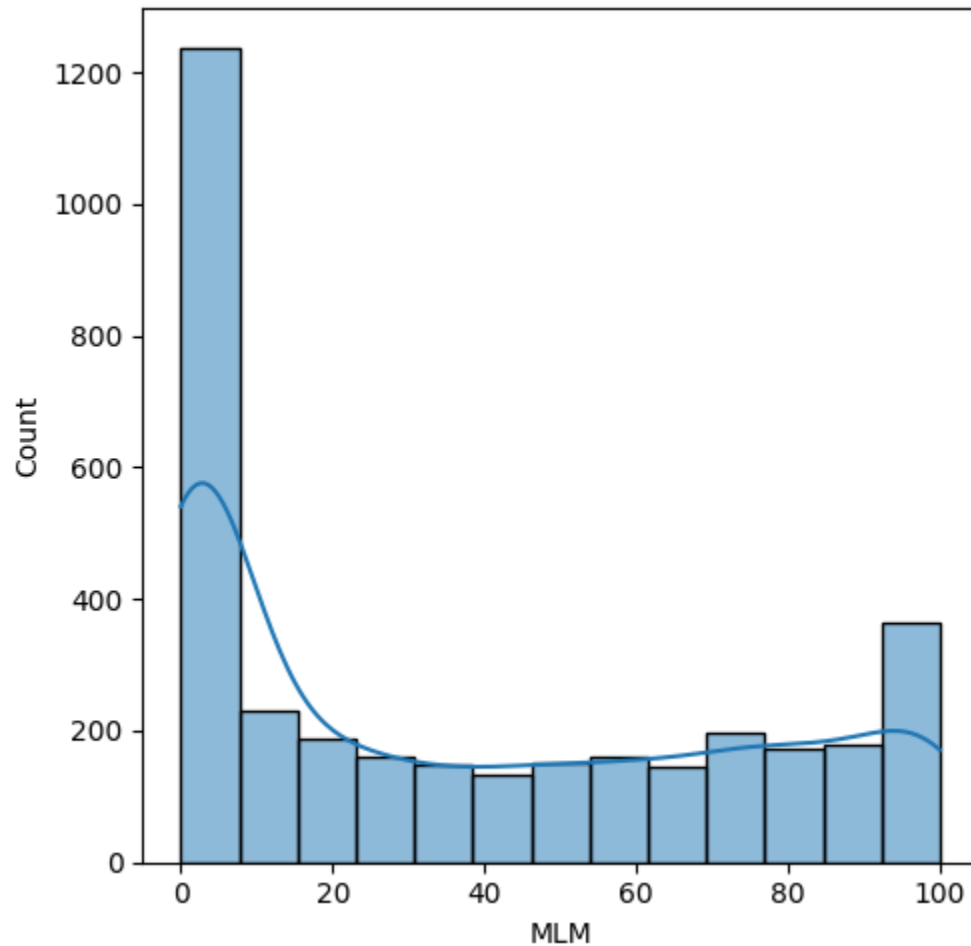
### 기본 특성 시각화(EDA)

- 적당히 정규분포를 따르는 것으로 보이고 특히 AlogP와 LogD값의 분포는 거의 비슷함.

- Rule of five를 대체로 따름



## 전처리 / EDA

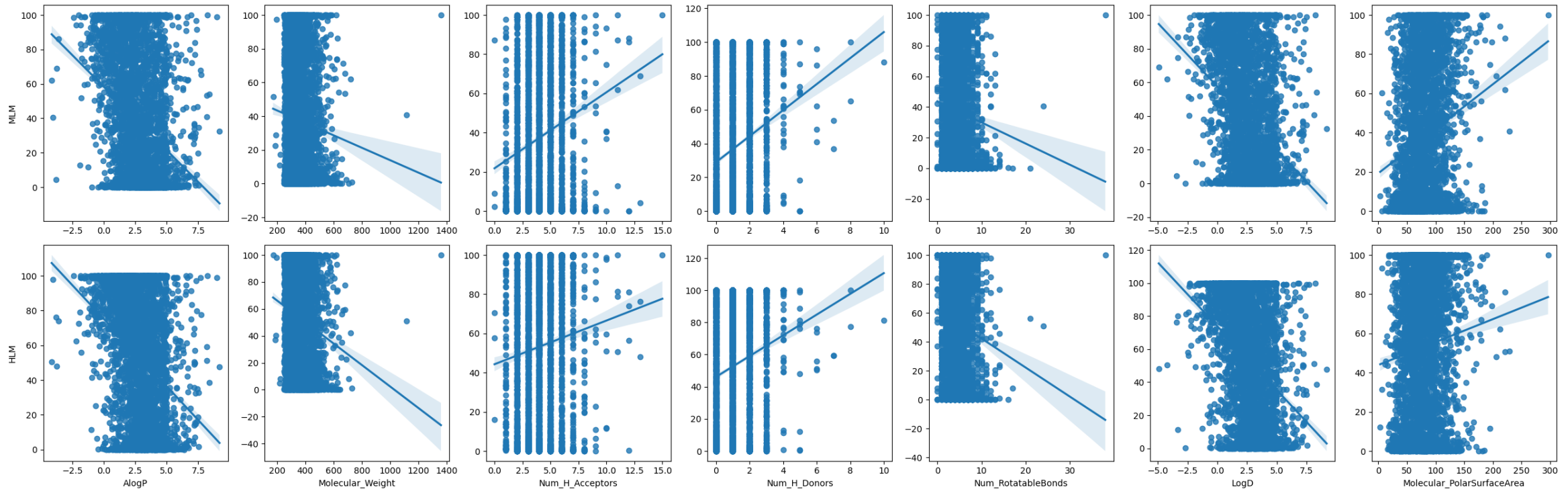


### 3. 결과 분포

- 데이터 증강
- 다운 샘플링

# 진행 상황

## 전처리 / EDA



- 특성과 결과값 간의 correlation (0.3 이상 기준)

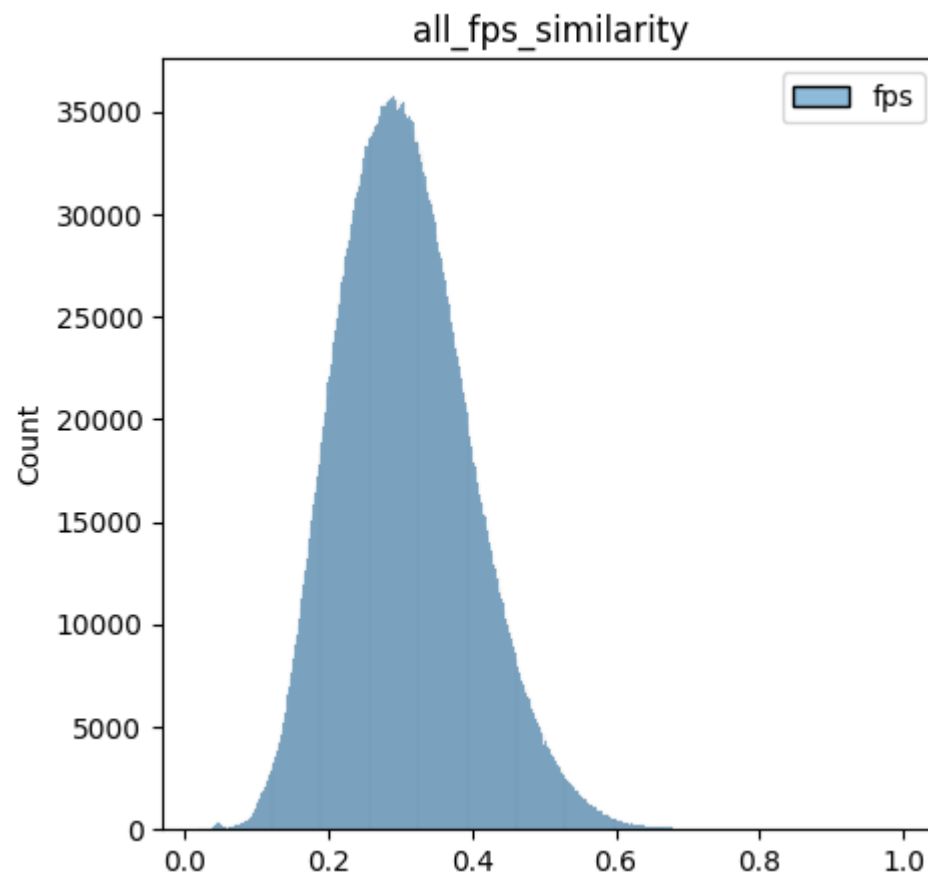
MLM와 AlogP의 corr 값은 -0.33    MLM와 LogD의 corr 값은 -0.35

HLM와 AlogP의 corr 값은 -0.35    HLM와 LogD의 corr 값은 -0.36

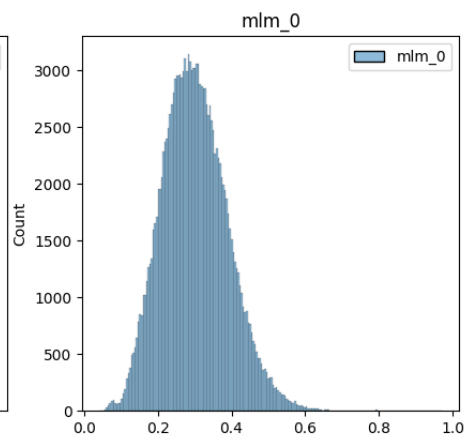
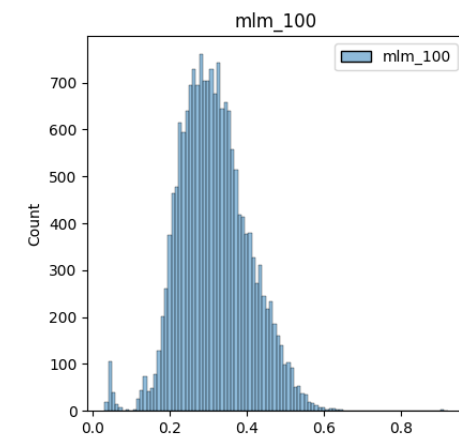
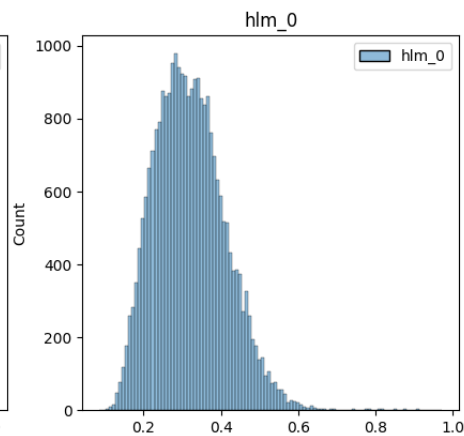
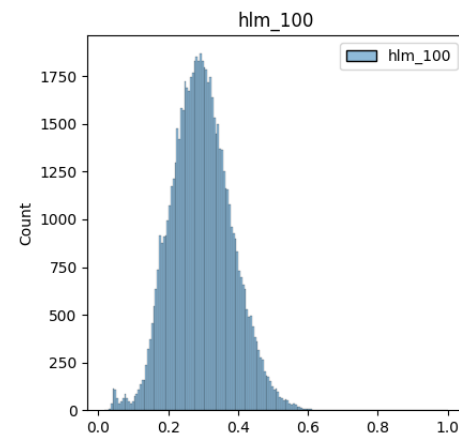
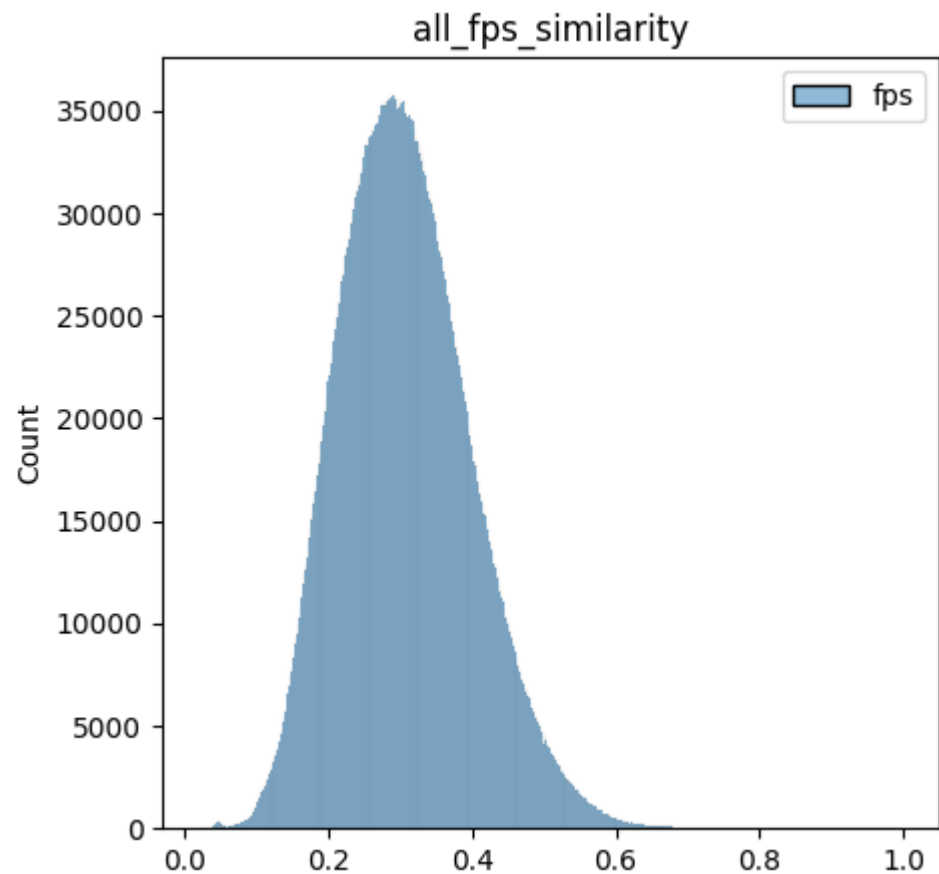
## 전처리 / EDA

### Train data 분자 유사도 시각화(EDA)

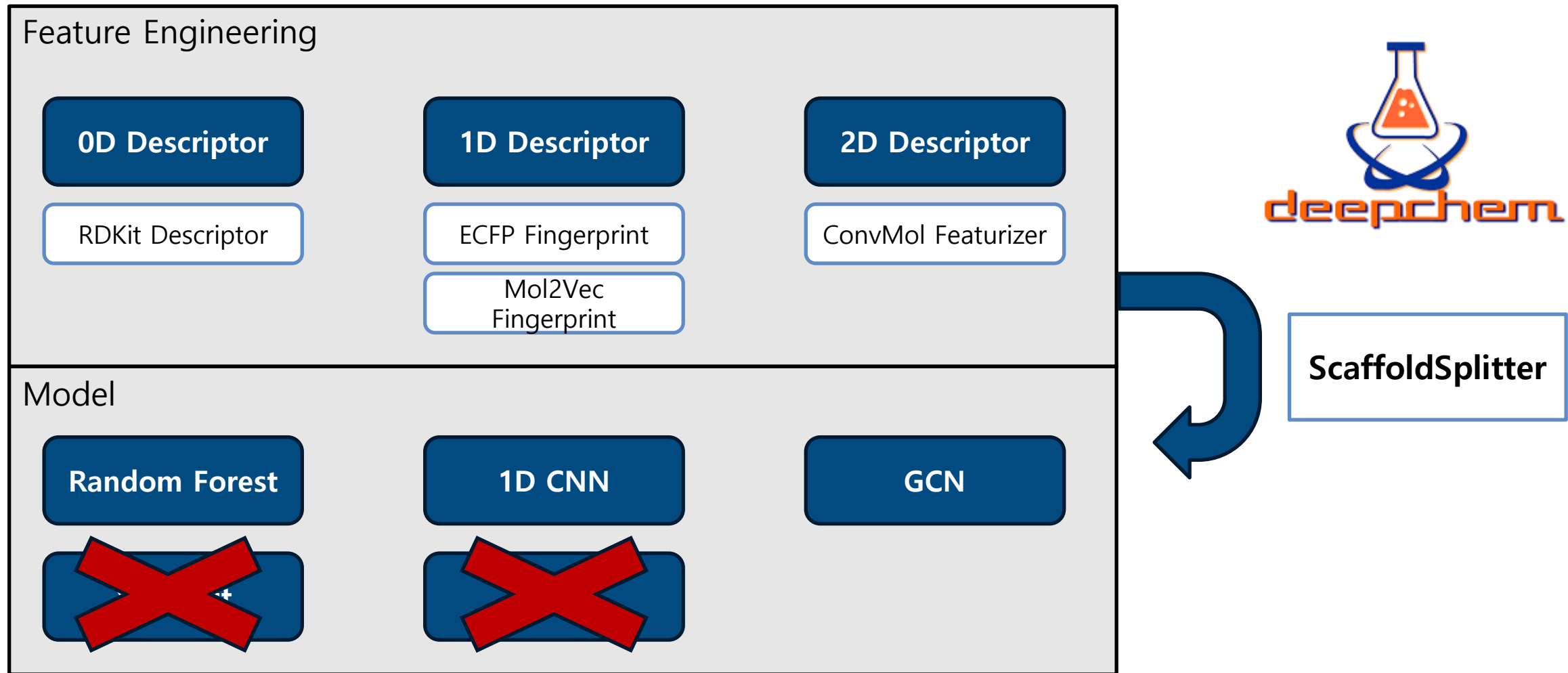
- 극단치 (0/100) 들끼리는 유사도가 높을까?
- 각 SMILES를 통해 RDKit으로 fingerprint 생성
- Train data에 대해 전체 분자에 대한 combination 2개 조합으로 생성
- Tanimoto 유사도 계산
- 전체 분자의 평균 유사도는 0.307



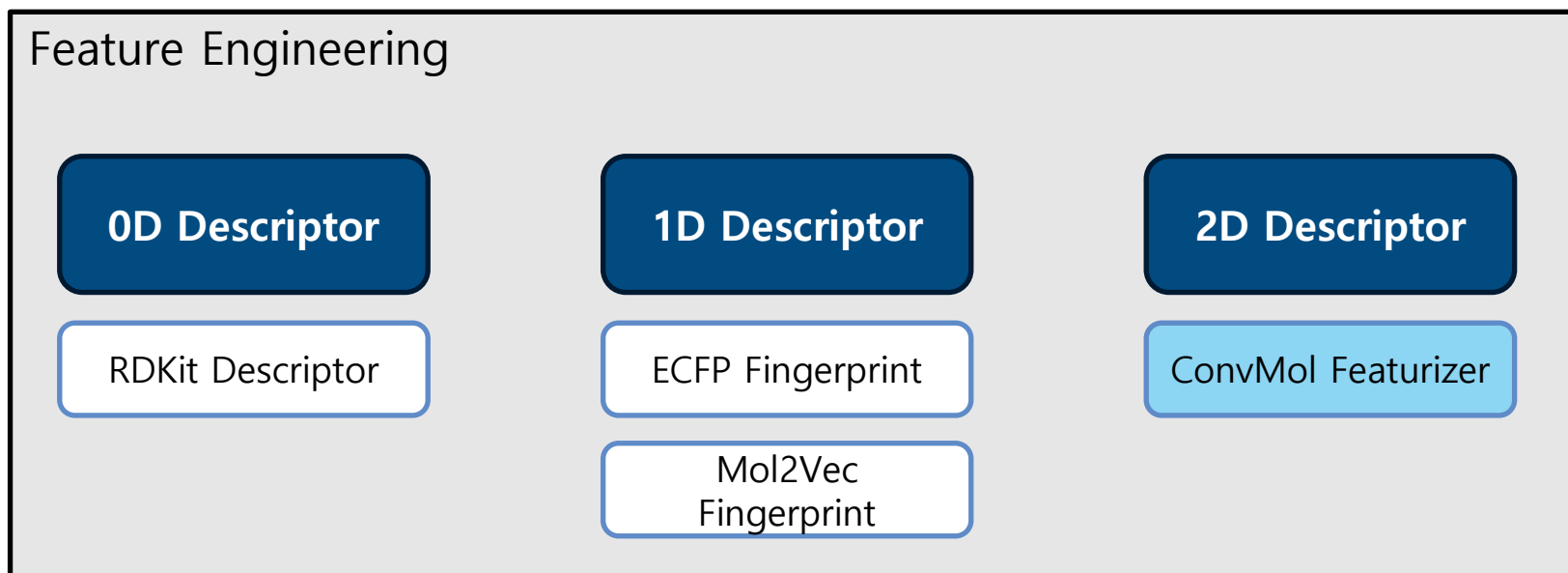
## 전처리 / EDA



# 진행 상황



## Feature engineering



데이터프레임 반환

데이터 셋 반환

## Feature engineering

```
class FpsGenerator:
    def __init__(self, size=2048, radius=4):
        self.featurizer = dc.featurizer.CircularFingerprint(size=size, radius=radius)

    def generate_fingerprints(self, df):
        list_fps = df['SMILES'].apply(self.featurizer.featurize)
        fps_features = np.vstack(list_fps)
        fps_df = pd.DataFrame(fps_features)
        fps_df.columns = fps_df.columns.astype(str)
        return fps_df
```

```
class Mol2Vec:
    def __init__(self):
        self.featurizer = dc.featurizer.Mol2VecFingerprint()

    def generate_mol2vecDataFrame(self, df):
        list_vec = df['SMILES'].apply(self.featurizer.featurize)
        vec_features = np.vstack(list_vec)
        vec_df = pd.DataFrame(vec_features)
        vec_df.columns = vec_df.columns.astype(str)
        return vec_df
```

```
# ConvMolFeaturizer 객체 생성
class ConvMol:
    def __init__(self):
        self.featurizer = dc.featurizer.ConvMolFeaturizer()

    def generate_Convmoldata(self, df):
        mols = [Chem.MolFromSmiles(s) for s in df.SMILES]
        conv_list = self.featurizer.featurize(mols)
        conv_df = pd.DataFrame({'conv': conv_list})
        return conv_df
```

Rdkit Descriptor는 길어서 생략.

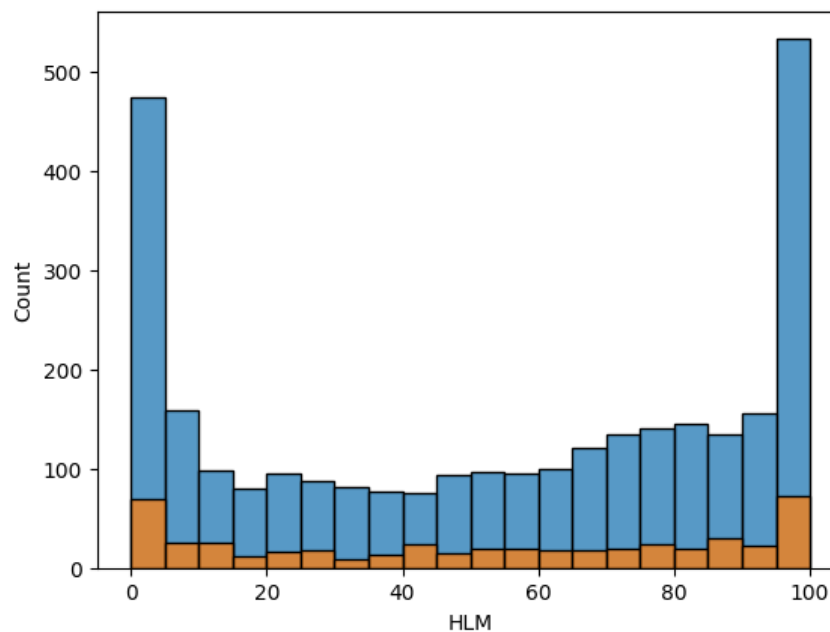
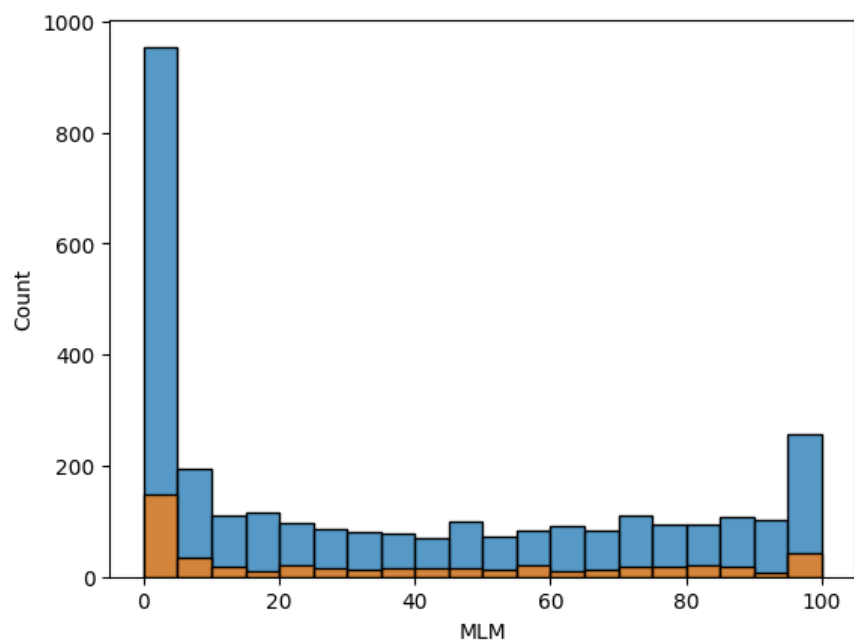
- Rdkit Descriptor는 분자 수준의 특성을 첨가하는 것이기에 수동적으로 확인하면서 추가. 중복 피쳐같은 것은 함수 내에 전처리 과정 삽입.
- 이후 사용된 피쳐는 따로 저장(TEST용)

# 진행 상황

## 데이터셋 분리

### 3. ScaffoldSplitter

- 분자 유사도를 바탕으로 훈련 / 검증 데이터를 고르게 분배
- 양끝에 결과가 너무 몰려있다.

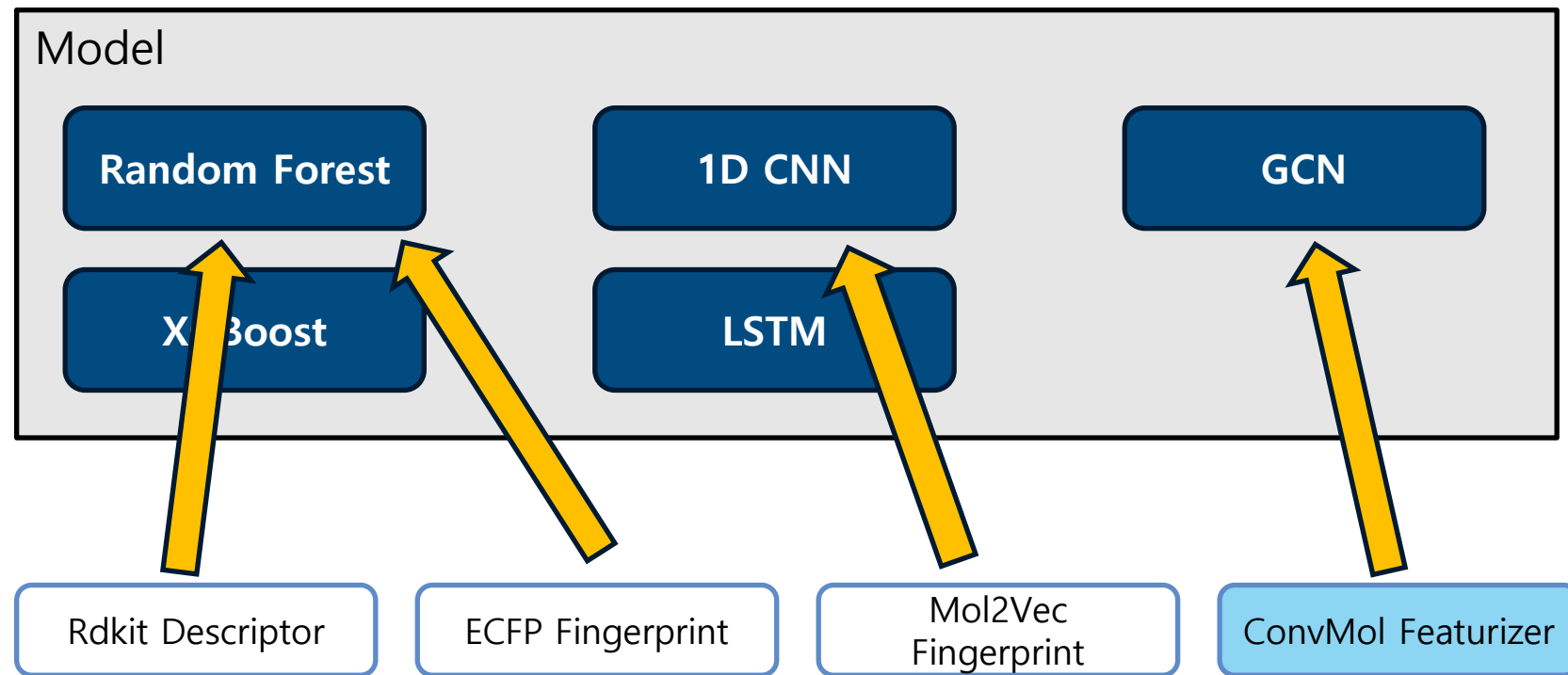




# 진행 상황

- 아직 하이퍼파라미터 튜닝 안함
- 각 특성에 어울리는 모델을 찾기 위해 하나씩 돌려봄
- **특이점** : 랜덤 포레스트는 트리형 모델이다 보니 그냥 피처가 많으면 점수가 조금씩 더 잘 나온다. 아직은 **피처 특성별**로 모델링을 했지만 이후 합쳐서도 돌려볼 예정

## 각 접근 모델



## 각 접근 모델

RMSE 값		
Random Forest	1D CNN	GCN
Rdkit Feature 31.55	Mol2Vec 32.75	ConvMol 33.03
ECFP 32.99		

# 진행 상황

## 모델 평균

### RMSE 값

Random  
Forest

Rdkit  
Feature  
31.55

ECFP  
32.99

1D CNN

Mol2Vec  
32.75

GCN

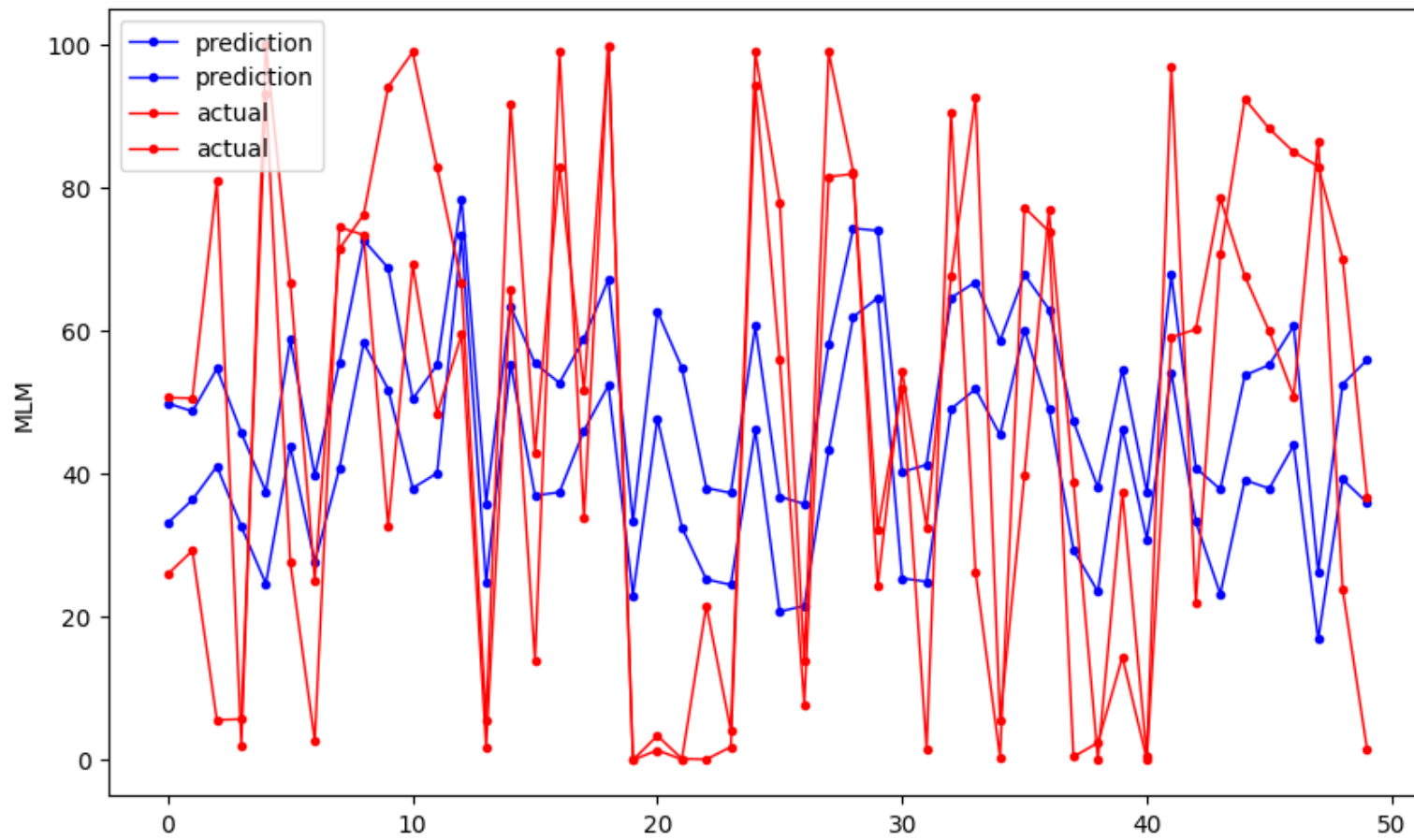
ConvMol  
33.03



모델 결과값 평균  
RMSE : 31.16

# 진행 상황

## 모델 평균



# 개선 방안

## 추가할 점

- RandomForest의 경우는 생성한 피쳐(Graph 데이터 제외)들을 합쳐서 모델링
- 검증데이터 분리시 반드시 데이터셋으로 바꿔줘야하는 등 기존과 달라 K-Fold 아직 못 해봤음 그냥 Sklearn의 데이터 분리 방법을 쓸까 고민중
- 하이퍼 파라미터 수정 안함 혹은 아직 기초 모델(특히 GCN)이라 공부 좀 필요하다.
- 이후 최종적으로 데이터 수가 적기에 모델 검증이 끝나면 정해진 하이퍼 파라미터 그대로 훈련셋 + 검증셋을 합쳐 모델 훈련 뒤 test를 할 예정.
- 단순 평균을 냈을때 검증 데이터셋에서는 비교적 좋은 점수가 나왔는데 test에서는 별로임
- Fp특성으로 돌린 모델의 설명력이 좋지 않아 분산이 작은 결과가 나왔는데 아마도 이부분때문에 평균을 낼시에 더 안좋은 영향만 준듯하다. + GCN을 쓴 경우 좋은 점수가 안 나왔다는 말이 많음.



최종

딥러닝 개인 프로젝트  
강종범

# 목차

## 1. 진행 상황

- 개선 방안
- 수정
- 추가 개선 방안
- 문제점 및 해결
- 추가 진행
- 추가 목표



## 2. 정리

# 개선 방안

## 추가할 점

- RandomForest의 경우는 생성한 피쳐(Graph 데이터 제외)들을 합쳐서 모델링
- 검증데이터 분리시 반드시 데이터셋으로 바꿔줘야하는 등 기존과 달라 K-Fold 아직 못 해봤음 그냥 Sklearn의 데이터 분리 방법을 쓸까 고민중
- 하이퍼 파라미터 수정 안함 혹은 아직 기초 모델(특히 GCN)이라 공부 좀 필요하다.
- 이후 최종적으로 데이터 수가 적기에 모델 검증이 끝나면 정해진 하이퍼 파라미터 그대로 훈련셋 + 검증셋을 합쳐 모델 훈련 뒤 test를 할 예정.
- 단순 평균을 냈을때 검증 데이터셋에서는 비교적 좋은 점수가 나왔는데 test에서는 별로임
- Fp특성으로 돌린 모델의 설명력이 좋지 않아 분산이 작은 결과가 나왔는데 아마도 이부분때문에 평균을 낼시에 더 안좋은 영향만 준듯하다. + GCN을 쓴 경우 좋은 점수가 안 나왔다는 말이 많음.

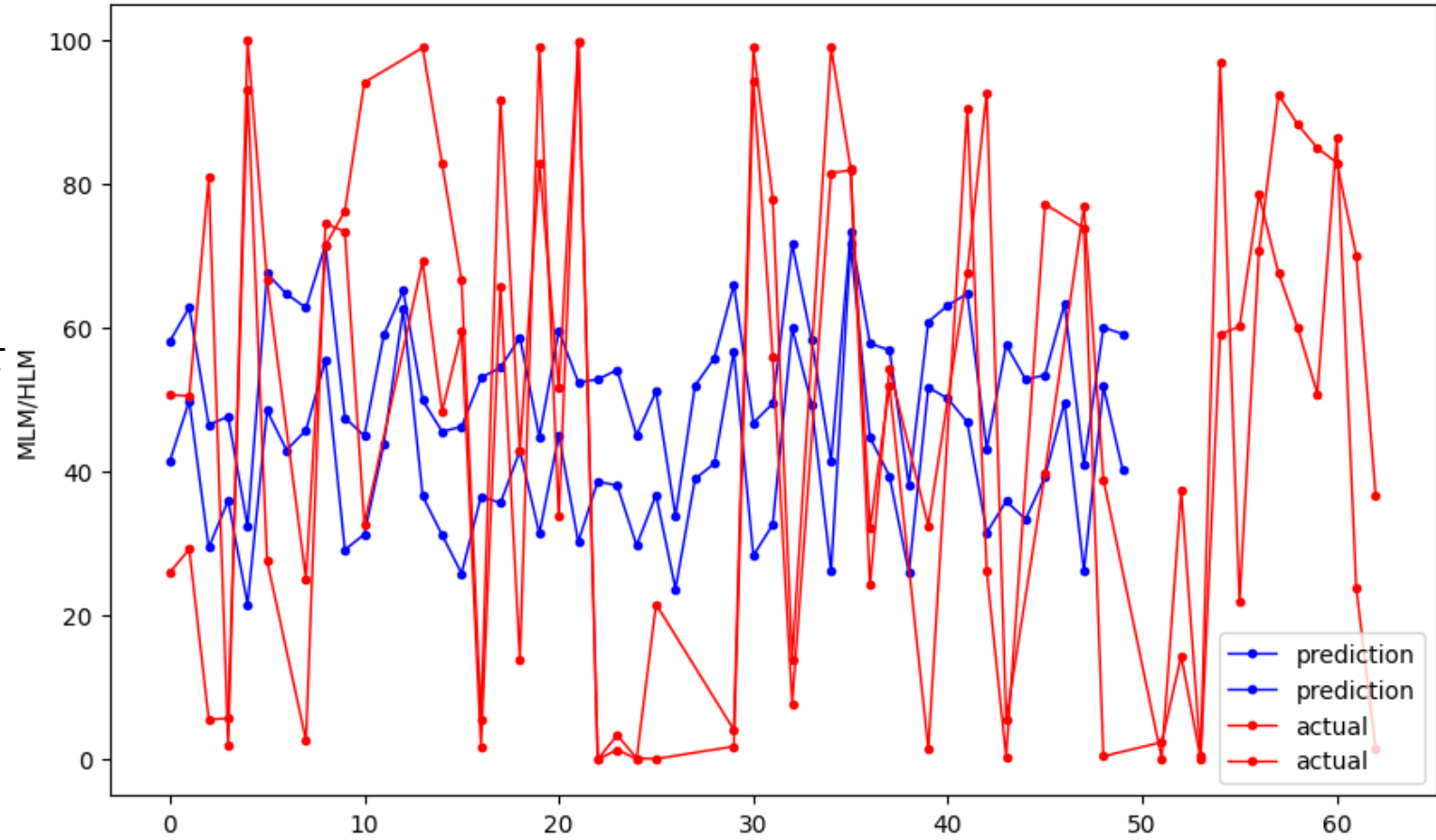


# 수정

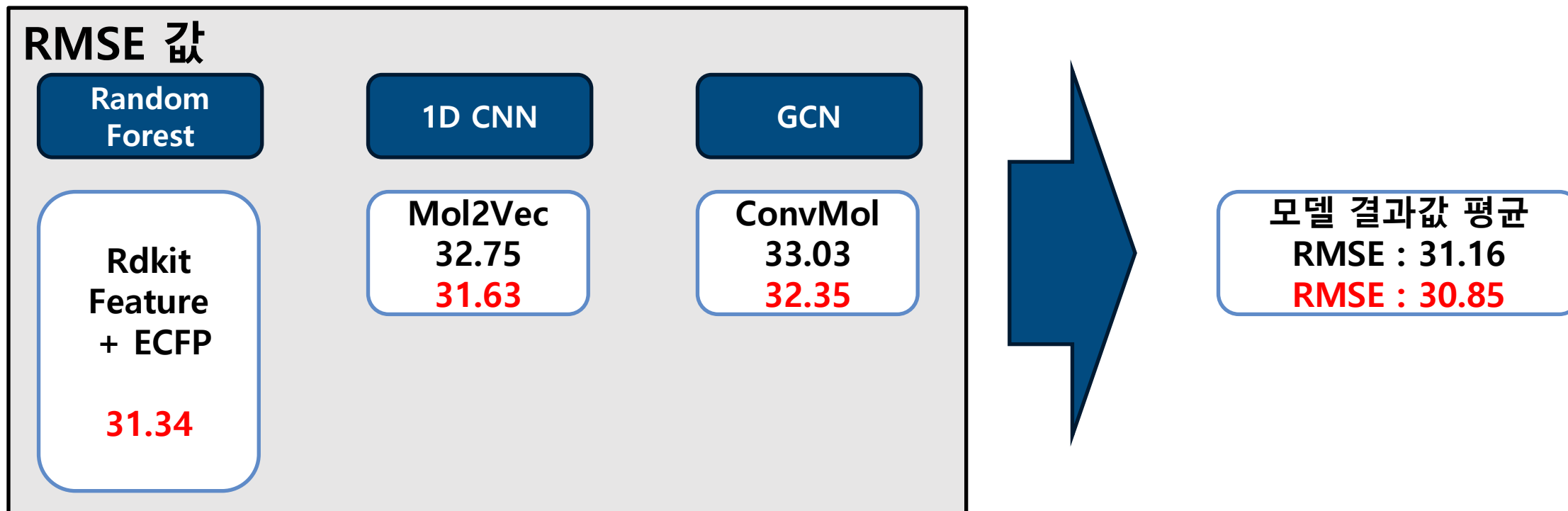
## 문제점

- ECFP의 RF모델 예측값 비교

- 모델의 성능이 낮은 것 뿐만 아니라 예측치가 평균에 거의 머물러 있어 모델을 종합해 평균할 시에 종합모델의 예측을 방해만 함.



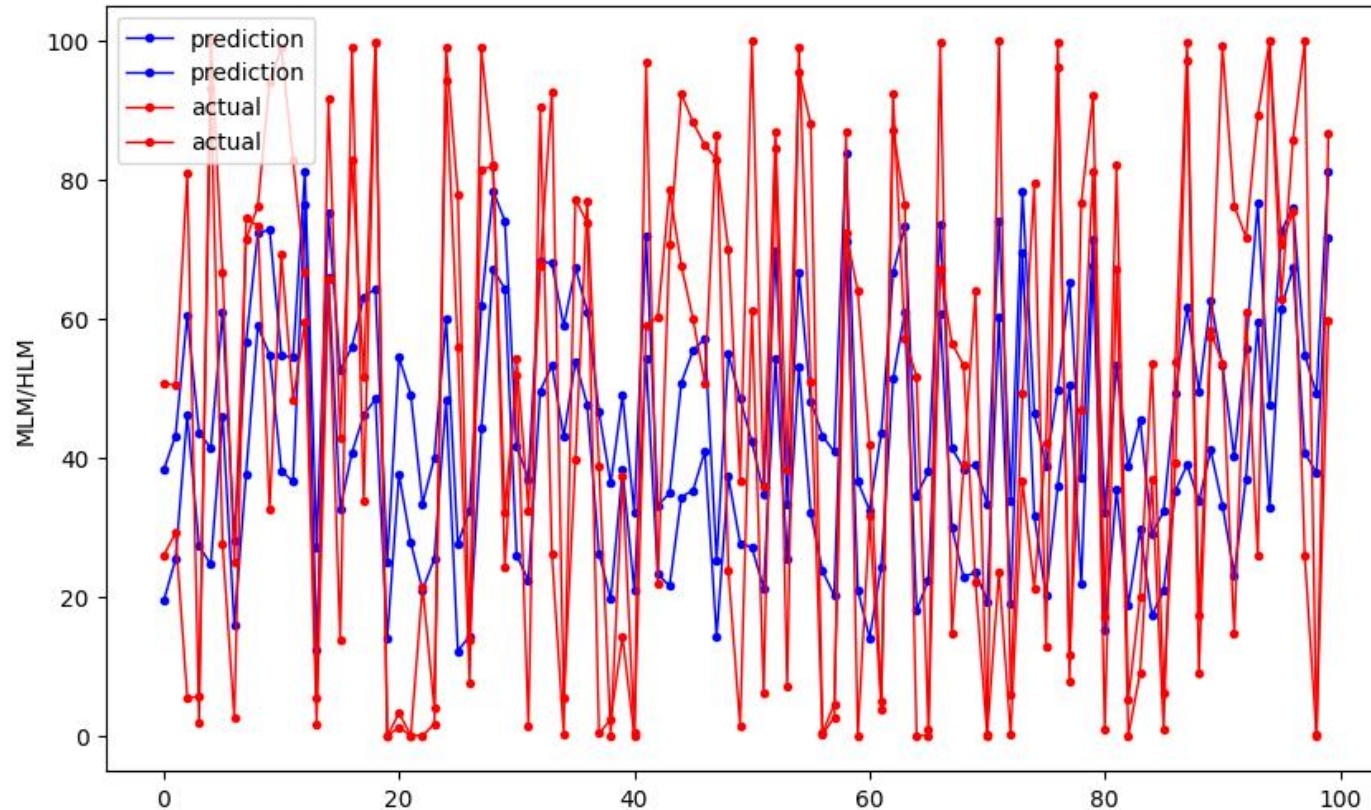
## RF 모델 Feature 통합 + 하이퍼파라미터 수정 후



# 수정

- 이전 평균모델보다 분산이 더 커지면서 모델의 설명력이 좋아진 것으로 보임
- 테스트 데이터에 적용시켜 제출했더니 RMSE 는 29.26으로 더 낮아짐.

## 수정 후 평균



# 추가 개선 방안

## 추가할 점

- ~~• RandomForest의 경우는 생성한 피쳐(Graph 데이터 제외)들을 합쳐서 모델링~~
- 검증데이터 분리시 반드시 데이터셋으로 바꿔줘야하는 등 기존과 달라 K-Fold 아직 못 해봤음 그냥 Sklearn의 데이터 분리 방법을 쓸까 고민중
- -> 기존 ScaffoldSplitter 의 성능이 좋고 앙상블 모델 검증을 위해 K-Fold 사용 X
- 하이퍼 파라미터 수정 안함 혹은 아직 기초 모델(특히 GCN)이라 공부 좀 필요하다.
- -> GCN에서 문제점 발견 및 수정
- 이후 최종적으로 데이터 수가 적기에 모델 검증이 끝나면 정해진 하이퍼 파라미터 그대로 훈련셋 + 검증셋을 합쳐 모델 훈련 뒤 test를 할 예정.
- -> X
- ~~• 단순 평균을 냈을때 검증 데이터셋에서는 비교적 좋은 점수가 나왔는데 test에서는 별로임~~
- ~~• Fp특성으로 돌린 모델의 설명력이 좋지 않아 분산이 작은 결과가 나왔는데 아마도 이부분때문에 평균을 낼시에 더 안좋은 영향만 준듯하다. + GCN을 쓴 경우 좋은 점수가 안 나왔다는 말이 많음.~~

# 문제점 및 해결

## GraphConvModel

1. GraphConvModel의 경우 여러 시도를 해봐도 시드 고정이 되지 않아 결과가 계속 바뀜.
2. 다른 경우는 계산을 돌리면 그대로 결과값이 재현이 되었을 뿐더러 GCN에서 Random\_state 설정을 해주었을때 오류가 나지 않아서 늦게 알아차림
3. 랜덤 시드 고정은 포기하고 하이퍼 파라미터 수정 뒤 해당 모델을 저장 시도
4. ( checkpoint / mode.save() / pickle / joblib) 등등 안됨
5. Github issue 와 Stackoverflow 로 해결,,



chstem commented on 2020년 7월 24일 Contributor ...

Deepchem automatically creates checkpoints during training in `model_dir`, which you can reload later. To explicitly create a checkpoint you can call `model.save_checkpoint()`. These checkpoints can be restored with `model.restore()`.

But for GraphConvModel there are currently some problems, see [#2013](#) and [#1943](#).

`model.save()` exists, but is not implemented. I this actually used somewhere?

```
deepchem/deepchem/models/models.py
Lines 123 to 128 in 1b7083b

123     def save(self) -> None:
124         """Dispatcher function for saving.
125
126         Each subclass is responsible for overriding this method.
127         """
128         raise NotImplementedError
```

👍 1

# 문제점 및 해결

## 이후 결과들

1. 비록 시드 고정은 못했지만 여러 하이퍼 파라미터 수정 및 시도들을 거쳐 그나마 오차값이 적은 모델을 Best\_gcn\_model로 저장했다. (기존 RMSE : 32.35 -> **32.38**)
2. 달라진 GCN 모델로 다시 모델 검증, **검증셋**에서는 평균시 더 RMSE가 낮아짐.

### RMSE 값

Random  
Forest

Rdkit  
Feature  
+ ECFP  
  
31.34

1D CNN

Mol2Vec  
31.63

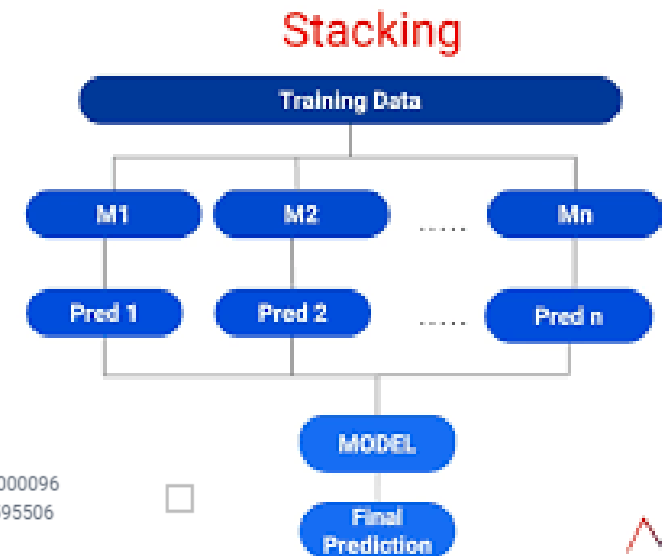
GCN

ConvMol  
32.35  
**32.38**

모델 결과값 평균  
RMSE : 30.85  
**RMSE : 30.73**

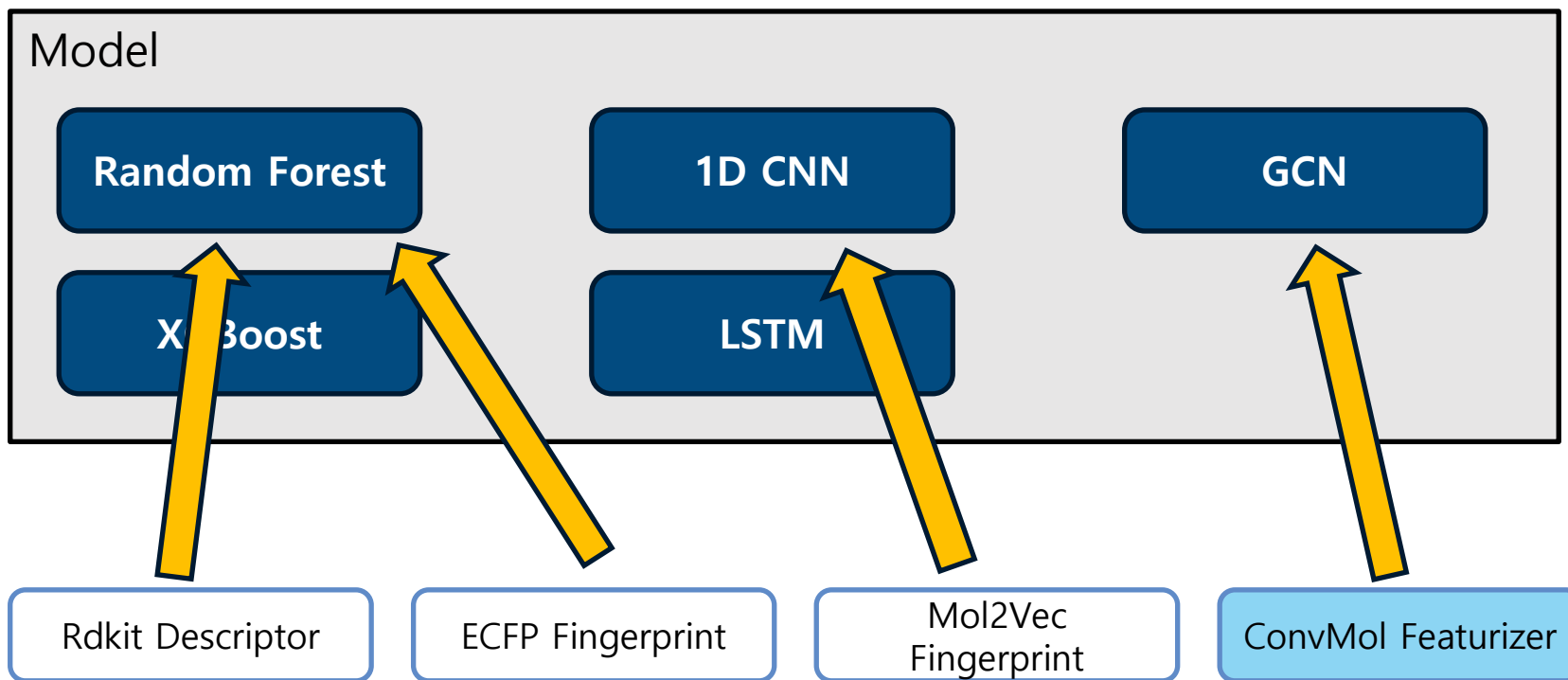
## 이후 결과들

1. 다만 **test csv**로 직접 제출해보았을때는 RMSE 값 29.42로 기존 제출 결과 29.27 보다 높은 오차를 기록함.
1. 복구시킬수 없는 모델로 아쉬워 하기보다 모델결과 단순 평균보다 더 괜찮은 앙상블 방법을 적용시켜보고자 **Stacking Ensemble** 시도
2. 스택킹 시도시에는 과적합 우려가 있어 K-fold로 과적합 확인.
3. 스택킹을 위한 모델로는 **RF모델** - > **LinearRegression** 모델
4. 결과 29.42 -> **29.24** 로 기존 평균보다도 낮은 오차, 복구가 불가능 했던 이전 모델보다도 조금 더 낮은 오차를 기록.



# 추가 진행

## XGBoost Model

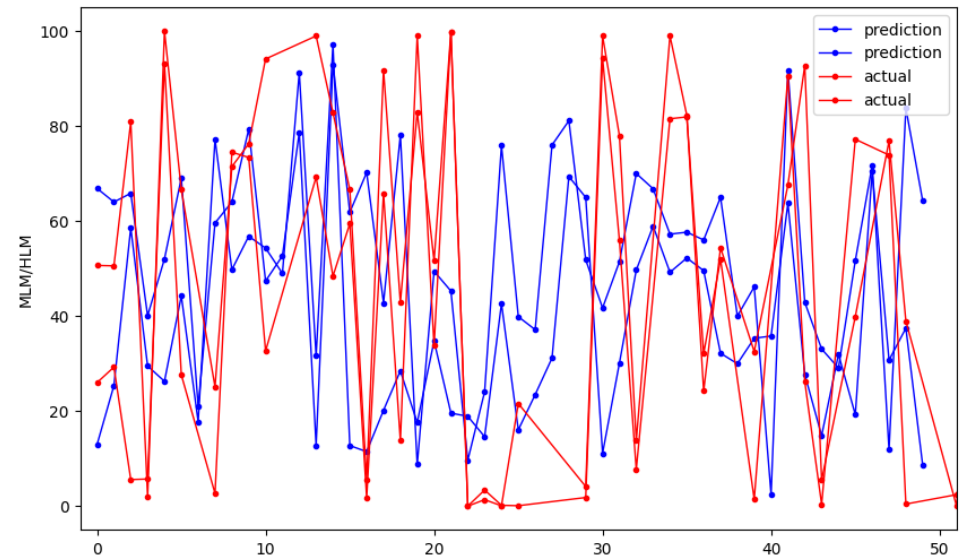




# 추가 진행

## XGBoost Model 하이퍼파라미터 조정

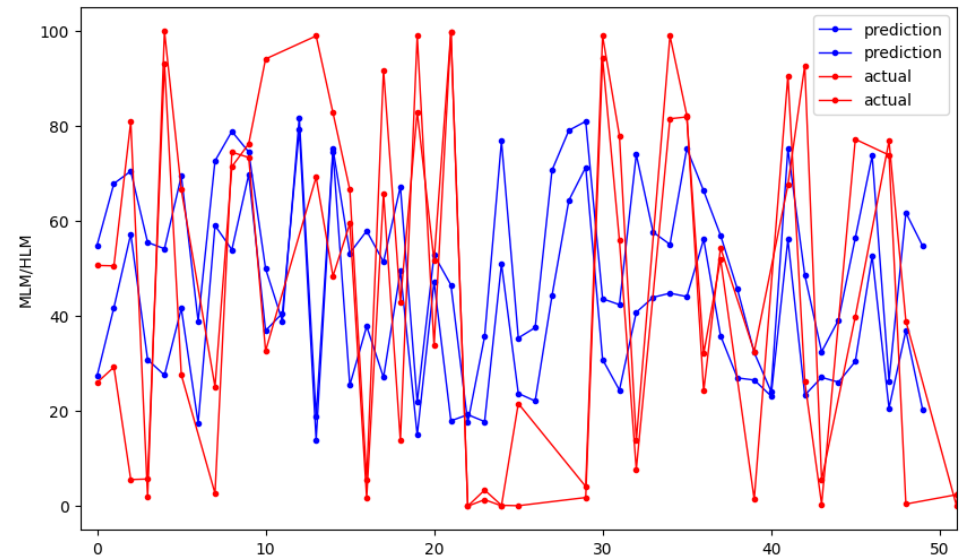
1. RF 모델에서는 하이퍼 파라미터를 크게 건드리지 않음.
2. 기존의 XGBoost 모델은 그냥 바로 쓰면 RF와 비교 했을때 큰 차이가 났다, 그래서 안 썼었다  
(RF Model RMSE : 31.46 / XGB Model RMSE : 33.24)
3. 다만 XGBoost 모델의 경우 Boosting 모델이기에 learning rate와 early\_stop을 설정해 주는 건 필요하다고 생각



# 추가 진행

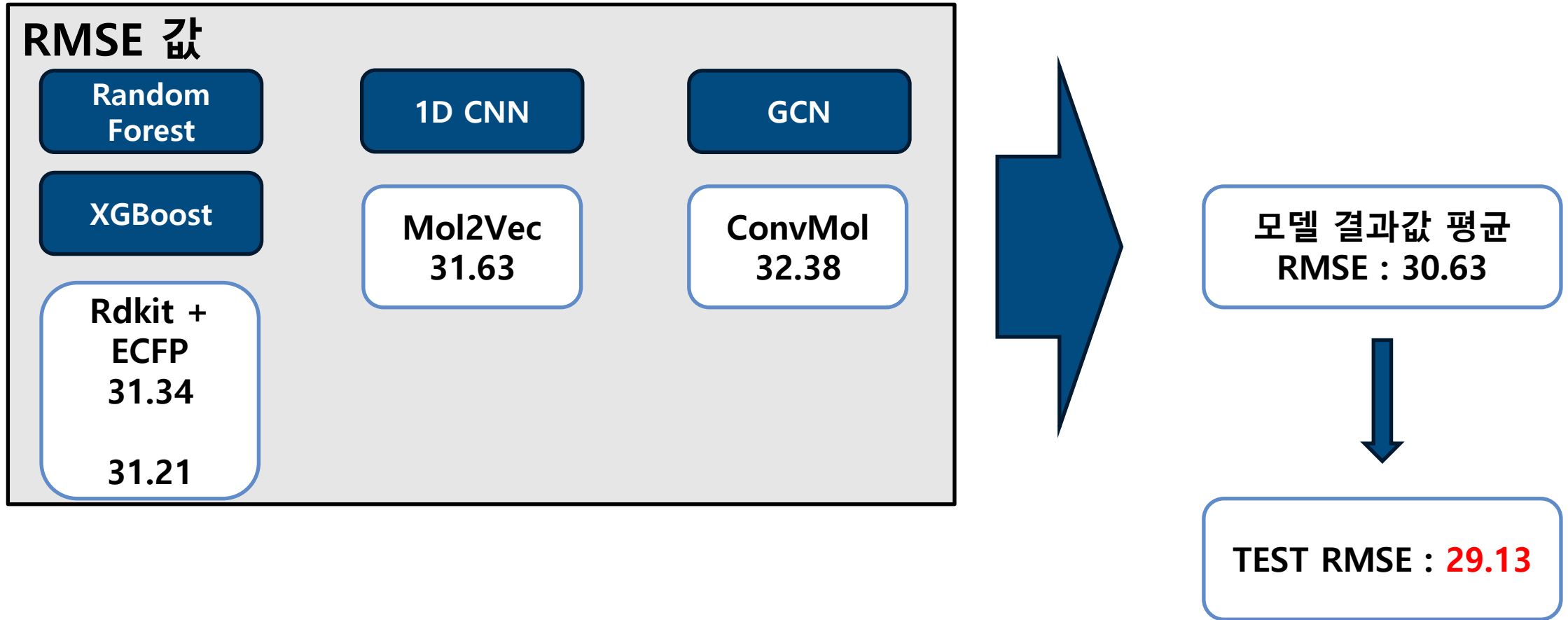
## XGBoost Model 하이퍼파라미터 조정

1. Learning rate를 기본 0.3 -> 0.05, Early\_stopping\_round : 5 로 설정해 기존 n\_estimators 횟수를 다 채우지 않아도 과적합 전에 stop
2. RMSE 값 33.24 - > **31.21**
3. RF 모델보다 더 좋은 성능(RF : 31.46)에 모델 훈련시간도 훨씬 단축됨.(2분 -> 13초)



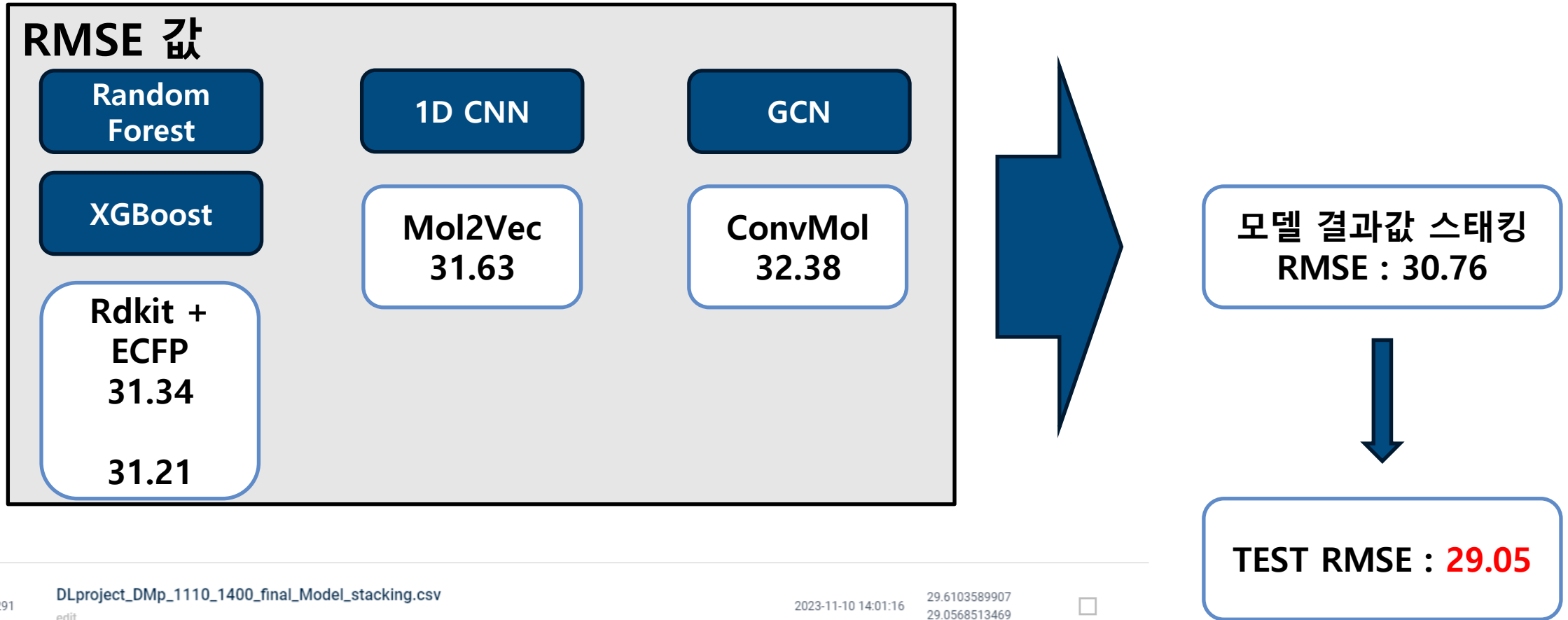
# 추가 진행

## XGBoost 모델 포함 Mean(총 4개의 결과)



# 추가 진행

## XGBoost 모델 포함 결과 Stacking (총 4개의 결과)



# 추가 진행

## 시도 했던 것들

1. RandomForest/XGBoost K-Fold 후 낮은 RMSE 기록한 모델 상위 5개로 예측하기 <- 첫 시도인데 나쁘지 않은 예측력 보임.
2. Feature Engineering에서 데이터프레임 형식으로 넣을 수 있는 특성은 전부 추가하고(약 1200개) RandomForest 로 Feature importance 확인후 상위 50개만 추려내 각각 다시 예측 <- 모델링 보다는 피쳐 엔지니어링 방법으로 쓰는데 성능도 별로고 시간이 너무 오래걸림.
3. 직접 관련 정보 찾아가면서 중요한 Feature 메모 후 머신러닝(RF,XGB,Lightgbm 1번의 방법 用) <- 모델 앙상블 전까지는 가장 좋은 점수기록(Pycaret 제외)
4. DNN, RNN, LSTM , WeaveModel <- 생각보다 별로
5. 예측 라벨이 총 2개인데 각각에 대한 최적의 모델 찾기 <- 시간이 너무 오래 걸리고 확실하지도 않음, 변수도 많아져서 그냥 2개 동시 예측
6. autoML(Pycaret) 시도 <- 가장 좋음

# 추가 진행

## AutoML 이기기

1. 대회 초기 Pycaret 이라는 AUTO ML 을 사용했었음.
2. 가능한 많은 Feature을 넣어주고 예측 실행.(사용 모델 : RF, gbr , lightgbm, xgboost)
3. 모델 Train 시간 1시간 20분 소요
4. TEST RMSE : 29.19

```
[ ] 1 def pycaret_train(df, target="MLM"):  
2     _df = df.drop(columns=["MLM", "HLM"], axis=1).copy()  
3     _df[target] = df[target]  
4     _setup = setup(data=_df.iloc[:,1:], target=target, train_size=0.8, session_id=seed, transformation=False, normalize=False, use_gpu=False)  
5  
6     _compare_models = compare_models(sort="RMSE", include=["rf", "gbr", "lightgbm", "xgboost"], n_select=2)  
7     tuned_models = [tune_model(model, optimize="RMSE") for model in _compare_models]  
8  
9     blender = blend_models(tuned_models, optimize="RMSE")  
10    save_model(blender, f"/content/drive/MyDrive/대사 안정성 예측 프로젝트/blender_{target}_{version}")  
  
[ ] 1 def pycaret_prediction(df):  
2     blender_MLM = load_model(f"/content/drive/MyDrive/대사 안정성 예측 프로젝트/blender_MLM_{version}")  
3     blender_HLM = load_model(f"/content/drive/MyDrive/대사 안정성 예측 프로젝트/blender_HLM_{version}")  
4  
5     pred_MLM = predict_model(blender_MLM, df)  
6     pred_HLM = predict_model(blender_HLM, df)  
7     return pred_MLM, pred_HLM  
  
[ ] 1 pycaret_train(train,target=mlm_target)  
  
[ ] 1 pycaret_train(train,target=hlm_target)  
  
[ ] 1 pred_MLM, pred_HLM = pycaret_prediction(test)
```

# 추가 목표

## SMILES TO METABOLIC\_RATE

1. 대사율을 알고 싶은 SMILES파일만 CSV로 넣으면 알아서 분자의 간 대사율을 예측해주도록 완성
2. 완성 시켜놓은 스크립트에서 예측에 필요한 코드는 전부 함수 / 클래스로 새로운 SMILES Test 파일에 사용이 가능하도록 만들어서 py 파일로 저장
3. 새로운 스크립트에 저장된 py 파일을 백그라운드에서 재생하고 SMILES2Metabolicrate() 함수 안에 테스트 할 csv 파일을 만들면 피처 엔지니어링 / 분리 / 데이터셋 변환 / 예측까지 한번에 실행되도록 시도
4. SMILES만 사용하기 때문에 주최측에서 제공된 Feature은 쓰지 못해서 예측력은 떨어지지만 (TEST RMSE : 29.53)단순히 SMILES만 넣어줘도 예측을 해주는 편리함에 있어 의미가 있다고 봄.
5. 400개 SMILES 기준 15초이내 결과 반환

# 추가 목표


## SMILES TO METABOLIC\_RATE


✓ [7] `!pip install deepchem`

✓  
0초 [8] `import os`  
`os.chdir('/content/drive/MyDrive/약물대사예측')`

✓ [9] `%run Metabolism_only_smiles.py`

[10] `test_file = pd.read_csv('/content/drive/MyDrive/약물대사예측/test.csv')`  
`test_file = test_file.loc[:, 'SMILES']`  
`test_file = test_file.to_frame()`

✓  
11초  `SMILES2Metabolicrate(test_file)`

 16/16 [=====] - 1s 37ms/step

	MLM	HLM
0	34.680229	55.937583
1	54.762212	70.426412
2	35.902222	57.902885
3	34.909130	55.316668
4	45.002877	64.022086





# 정리

## 고찰

1. 어떤 모델을 만들어보아도 좋은 결과를 보이지 않아 개선의 여지가 보이지 않았는데 앙상블 방법이 잘 적용되어서 다행이었다.
2. 여전히 예측이 어려운 시도인 것은 분명하다. 해당 대회 1등 RMSE값이 26점대임을 감안하면 50기준으로 up/down 이중분류를 하는 것이 더 좋은 결과를 가져올 수 도 있겠다는 생각이 들었다.
3. Research의 영역에 가까운 경진대회였는데 화학과 전공으로서 상당히 뜻깊은 대회였다. 특히 SMILES 하나로도 여러가지 Feature 들을 가져올 수 있는 것은 상당히 재밌고 더 알아보고 싶은 의욕이 생겼다.
4. 저분자 데이터의 경우 어떻게 증강을 시도하는지 더 알아보아야겠다 생각했다.



Thank you