Multi-Label Classification

with an Enzyme Substrate Dataset

이다운, 유예송

Table of contents

00

Competition 소개

Explore Multi-Label Classification with an Enzyme Substrate Dataset

Playground Series - Season 3, Episode 18

Overview Data Code Models Discussion Leaderboard Rules Team

Submission File

For each `id` in the test set, you must predict the value for the targets `EC1` and `EC2`. and have the following format:

```
id,EC1,EC2
14838,0.22,0.71
14839,0.78,0.43
14840,0.53,0.11
etc.
```

EC1~EC6 총 6그룹으로 분류되는 효소데이터에서 각 효소의 EC1, EC2 해당 확률을 예측하는 것이 목표

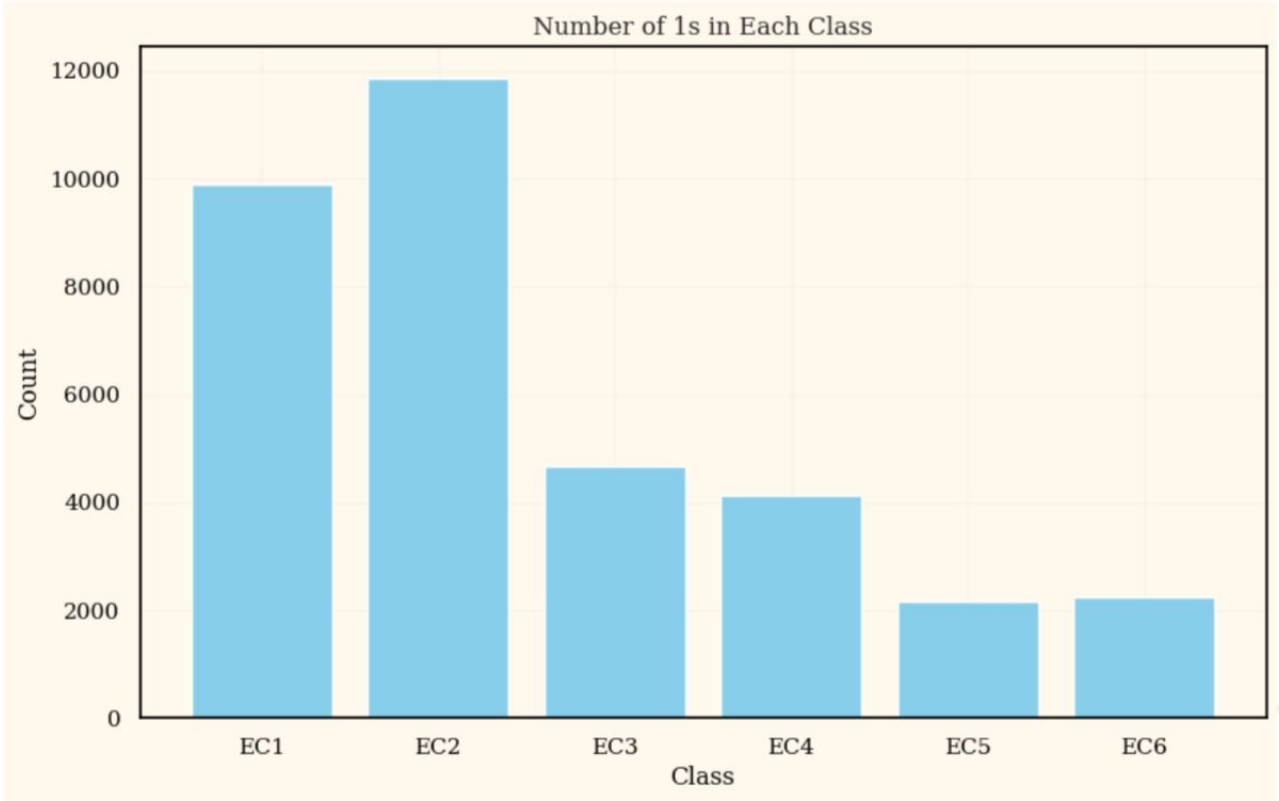
01

EDA

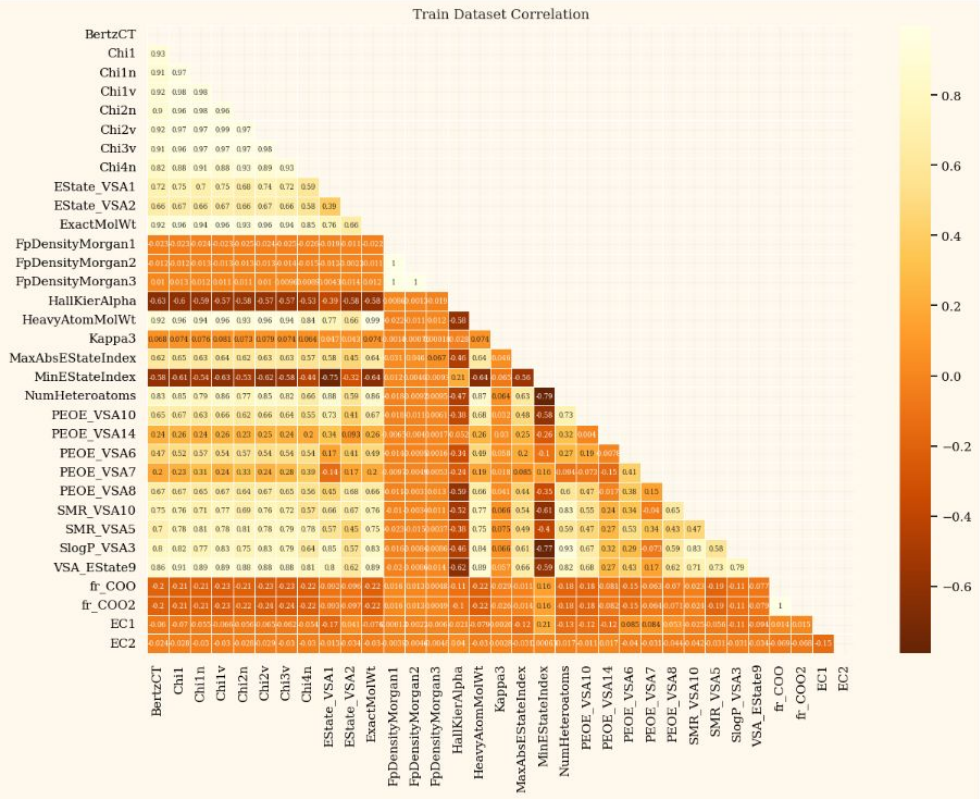
Data Summary

	data type	#missing	%missing	#unique	min	max	average	standard_deviation	first value	second value	third value
id	int64	0	0.000000	14838	0.000000	14837.000000	7418.500000	4283.505982	0.000000	1.000000	2.000000
BertzCT	float64	0	0.000000	2368	0.000000	4069.959780	515.153604	542.456370	323.390782	273.723798	521.643822
Chi1	float64	0	0.000000	1259	0.000000	69.551167	9.135189	6.819989	9.879918	7.259037	10.911303
Chi1n	float64	0	0.000000	3157	0.000000	50.174588	5.854307	4.647064	5.875576	4.441467	8.527859
Chi1v	float64	0	0.000000	3306	0.000000	53.431954	6.738497	5.866444	5.875576	5.834958	11.050864
Chi2n	float64	0	0.000000	3634	0.000000	32.195368	4.432570	3.760516	4.304757	3.285046	6.665291
Chi2v	float64	0	0.000000	3725	0.000000	34.579313	5.253221	4.925065	4.304757	4.485235	9.519706
Chi3v	float64	0	0.000000	3448	0.000000	22.880836	3.418749	3.436208	2.754513	2.201375	5.824822
Chi4n	float64	0	0.000000	2930	0.000000	16.072810	1.773472	1.865898	1.749203	1.289775	1.770579
EState_VSA1	float64	0	0.000000	719	0.000000	363.705954	29.202823	31.728679	0.000000	45.135471	15.645394
EState_VSA2	float64	0	0.000000	445	0.000000	99.936429	10.435316	13.651843	11.938294	0.000000	6.606882
ExactMolWt	float64	0	0.000000	1666	1.007276	2237.318490	292.623087	225.384140	222.068080	260.029719	382.131027
FpDensityMorgan1	float64	0	0.000000	556	-666.000000	3.000000	1.236774	5.491284	1.181818	1.346154	1.085714
FpDensityMorgan2	float64	0	0.000000	650	-666.000000	3.200000	1.812070	5.495565	1.727273	2.076923	1.742857
FpDensityMorgan3	float64	0	0.000000	654	-666.000000	3.400000	2.255470	5.501200	2.363636	2.769231	2.400000

EC1~EC6

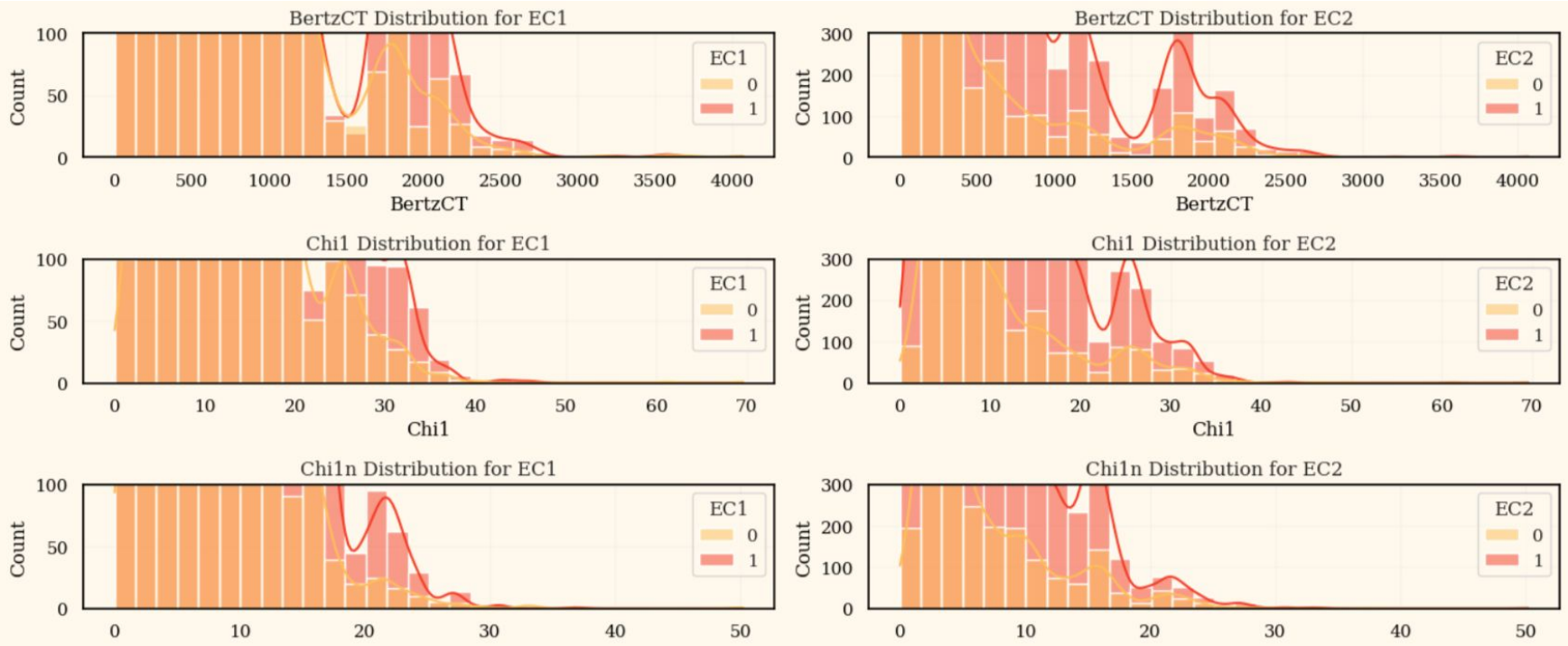

Number of 1s in Each Class
12000
10000
8000
6000
4000
2000
0
Count
EC1
EC2
EC3
EC4
EC5
EC6
Class

Correlation



1. BertzCT and the Chi's
2. ExactMolWt and the Chi's
3. VSA_Estate9 and the Chi's
4. ExactMolWt and HeavyAtomMolWt
5. FpDensityMorgan's
6. fr_COO and fr_COO2

distributions of EC1 and EC2



02

Data Preprocessing

Test, Train data

1. Input데이터와 target데이터가 모두 존재하는 Train data를 Train set과 Test set으로 분리

2. Train과 Test 데이터에서 EC1~EC6 열을 분리

```
train_X = #(11870, 31)
np.array(train_data.iloc[:,1:-6], dtype = np.float32)
train_Y = #(11870, 2)
np.array(train_data.iloc[:, -6:-4],dtype = np.float32)
```

id	BertzCT	Chi1	EC1	EC2	EC3	EC4	EC5	EC6
0	323.3907823	9.879917853	1	1	0	0	0	0
1	273.7237977	7.259037475	0	1	1	0	0	0

Data Loader, Dataset

Custom Dataset:

방대한 데이터를 한 번에 불러오는 것이 어려울 때 사용

→ 현재 데이터에서 필수적인 것은 아니다!

Data Loader:

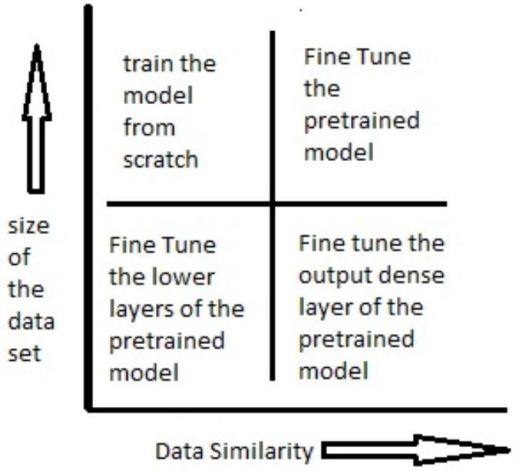
학습할 때 쉽게 이용할 수 있도록 데이터를 배치 형태로 만들어주는 라이브러리

03

ResNet18

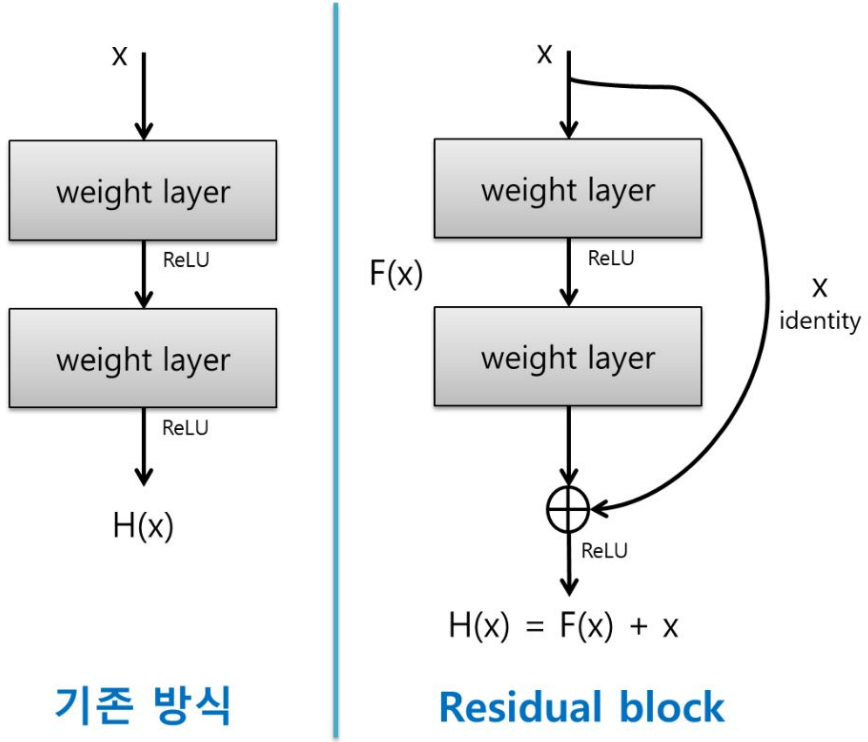
ResNet 선정 이유

1. **Gradient Vanishing** 문제 해결
2. **Pre-trained model** 제공 **(transfer learning)**
3. 이미지 뿐만 아니라 다양한 형태의 데이터에서 적용 가능

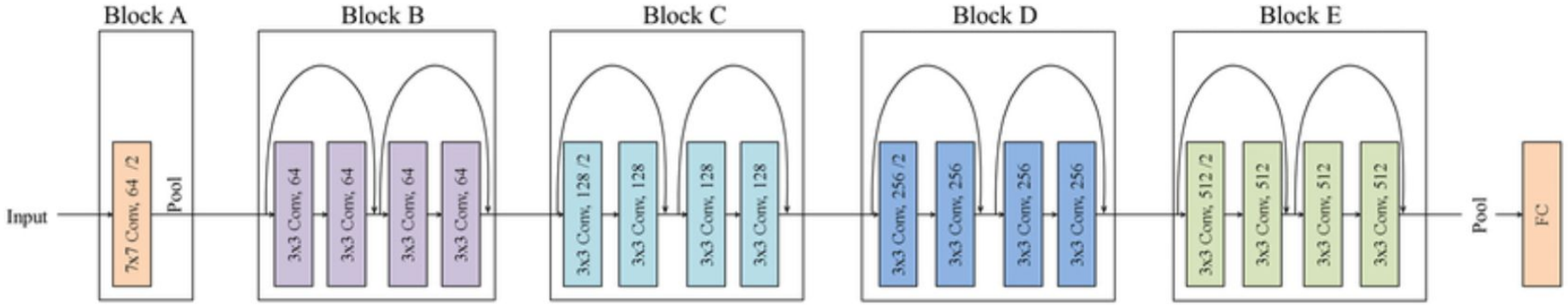


ResNet18

shortcut connection:
입력값을 출력값에 더해주기 위해 지름길 추가



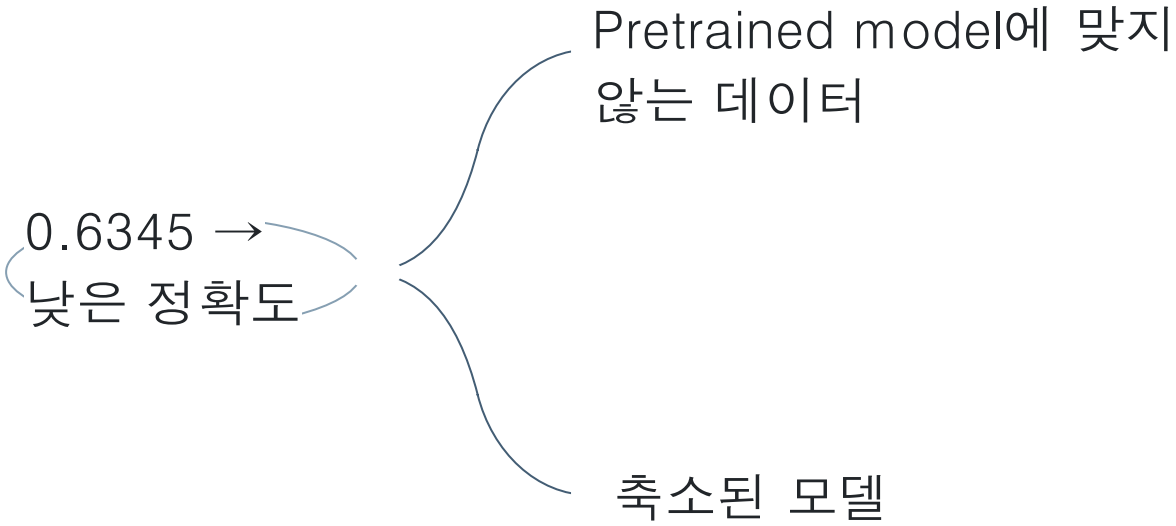
ResNet 18 구조




```
class BasicBlock(nn.Module):
.
.
out += identity  # residual connection, 입력값을 더해줌
        out = self.relu(out)

        return out
```

결과



04

XGBoost, LightGBM

XGBoost 선정 이유

1. 뛰어난 예측 성능

Regression, Classification 문제를 모두 지원하고 성능과 자원 효율 측면에서 뛰어남. Kaggle, DACON 등과 같은 분석 대회에서 좋은 성적을 내는 모델임.

2. 빠른 수행 시간

병렬 연산을 지원하여 GBM에 비해 빠른 학습 및 예측이 가능함. 빠른 수행 속도로 대규모 데이터셋이나 복잡한 모델에서도 효과적으로 사용 가능함.

3. 과적합 규제

트리의 깊이, 각 트리에서 사용되는 리프 노드의 최소 수 등을 조절하여 모델의 복잡성을 제어하고 과적합을 방지함

4. 자체 내장된 교차 검증

반복 수행시마다 내부적으로 교차검증을 수행해 평가 데이터세트의 예측값이 최적화되면 반복을 중간에 멈출 수 있는 early stopping 기능이 있음

Hyperparameter Tuning

Parameter	Description	Value
n_estimators	학습하는 트리의 개수	454
learning_rate	학습률	0.05
reg_alpha	L1 Regularization 적용값	2.062
reg_lambda	L2 Regularization 적용값	4.461
gamma	트리 분할 시 필요한 최소 손실 감소값	0.9
max_depth	각 트리의 최대 깊이	6.0
min_child_weight	리프 노드에서 필요한 최소 샘플 가중치의 합	2.0
subsample	훈련 데이터의 일부를 선택하는 비율	1.0
colsample_bytree	트리마다 사용할 특성의 비율	0.9

XGBoost 예측 결과

```
xgb_params_optimized = {
    'n_estimators': int(best['n_estimators']),
    'learning_rate': best['learning_rate'],
    'reg_alpha':best['reg_alpha'],
    'reg_lambda': best['reg_lambda'],
    'max_depth' : int(best['max_depth']),
    'gamma' : best['gamma'],
    'subsample': round(best['subsample']),
    'min_child_weight': best['min_child_weight'],
    'colsample_bytree':best['colsample_bytree']
}

model = MultiOutputClassifier(XGBClassifier(**xgb_params_optimized))
model.fit(X_train, y_train)
```

Accuracy: **0.6552**

LightGBM 선정 이유

1. 뛰어난 예측 성능

Leaf-wise 분할 방식을 사용해 손실값이 높은 노드에 대해 더 깊게 트리를 분할하며 손실값을 줄여 높은 정확도를 보임

2. 빠른 수행 시간

히스토그램 기반의 학습 방식을 사용하여 빠르게 대용량 데이터를 처리할 수 있음

3. 최적화된 분할 알고리즘

리프 중심 분할 방식과 효율적인 리프 중심 학습을 통해 트리의 분할을 최적화함. 이로써 모델의 복잡성을 줄이고 과적합을 방지함

Hyperparameter Tuning

Parameter	Description	Value
n_estimators	학습하는 트리의 개수	314
learning_rate	학습률	0.02
reg_alpha	L1 Regularization 적용값	0.048
reg_lambda	L2 Regularization 적용값	0.3
min_child_samples	리프 노드에 필요한 최소 샘플 수	5.0
subsample	훈련 데이터의 일부를 선택하는 비율	0.7
colsample_bytree	트리마다 사용할 특성의 비율	2.0
colsample_bynode	노드마다 사용할 특성의 비율	1.0

LightGBM 예측 결과

```
lgbm_params_optimized = {
    'n_estimators':int(best['n_estimators']),
    'learning_rate':best['learning_rate'],
    'reg_alpha':best['reg_alpha'],
    'reg_lambda': best['reg_lambda'],
    'min_child_samples': int(best['min_child_samples']),
    'colsample_bynode': best['colsample_bynode'],
    'colsample_bytree': best['colsample_bytree']
}

model = MultiOutputClassifier(LGBMClassifier(**lgbm_params_optimized))
model.fit(X_train, y_train)
```

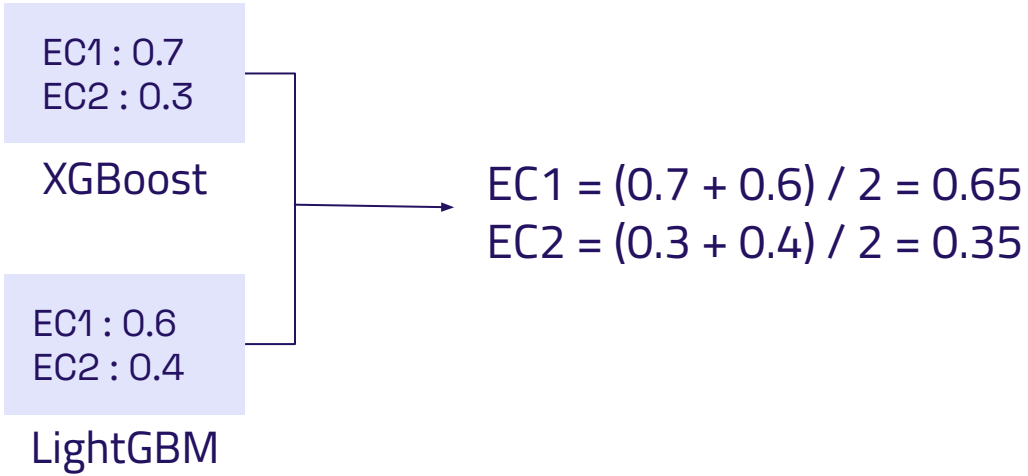
Accuracy: **0.6570**

05

Ensemble

XGBoost + LightGBM

Soft Voting



Ensemble 예측 결과

```
xgb_pred = xgb_model.predict_proba(X_test)
xgb_pred = np.array(xgb_pred)

lgbm_pred = lgbm_model.predict_proba(X_test)
lgbm_pred = np.array(lgbm_pred)

overall_pred = (xgb_pred + lgbm_pred)/2
overall_pred = np.array(overall_pred)[:,:,1].T
```

Accuracy: **0.6577**

06

Conclusion

최종 결과

Model	Accuracy
ResNet18	0.6345
XGBoost	0.6552
LightGBM	0.6570
XGBoost & LightGBM Ensemble	**0.6577**

성과 및 한계점

- 목표 점수 달성

- 최종 점수(**0.6577**) > 벤치마크 점수(0.6541)
- 딥러닝, 머신러닝 모델 등 다양한 모델과 파라미터를 시도해 정확도를 높임

- 데이터 전처리 미흡

- EDA 결과 데이터의 특성을 반영한 feature engineering 과 적절한 정규화 기법을 사용해서 모델 정확도를 개선할 것임

감사합니다