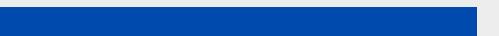


난임 환자 대상 임신 성공 여부 예측 AI 오프라인 해커톤



엄마손은 AI손

* CONTENTS

1. 문제 및 데이터 소개

2. 전체 흐름 소개

3. 데이터 전처리

- EDA & PREPROCESSING
- FEATURE ENGINEERING
- ENCODING

4. 모델 학습 및 최적화

- K-FOLD CROSS VALIDATION
- MODEL 고려 및 SEED ENSEMBLE
- HYPERPARAMETER TUNING

5. 최종 선정

* 문제 및 데이터 소개

주제 및 배경

난임 시술을 진행하는 환자들에게 시술 부담을 줄이는 동시에, 의료 기관이 차별화된 서비스를 제공할 수 있도록 난임 환자 데이터를 활용하여 **임신 성공 여부를 예측하고 임신을 결정짓는 최적의 특성을 탐색하는 AI 모델 개발**

데이터 소개

train data : 샘플별 고유 ID (256,351행)의 임신 성공 여부를 포함한 난임 환자 시술 데이터 (68개 컬럼)

test data : 샘플별 고유 ID (90,067행)의 난임 환자 시술 데이터 (67개 컬럼)

시술 당시 나이, 시술 유형, 총 임신 횟수, 총 시술 횟수,
이식된 배아 수, 정자 출처 등..

* 전체 흐름 소개

main 함수 흐름

```
def main(config_params, model_param_dict):
    start_time = time.time() # 시작 시간 기록
    y_probs = []
    for seed in config_params['seed_list']:
        print('\n'*30)
        print(f"Seed Number : {seed}")
        print("*"*30)
        for model_name, model_params in model_param_dict.items():
            print(f"<Model : {model_name}>")
            # seed 고정
            set_seed(seed)

            # 데이터셋 읽기
            df_train, df_test, df_submit = read_data(config_params['data_path'])

            # 데이터 전처리
            df_train = feature_engineering(df_train)
            df_test = feature_engineering(df_test)
            base_features, cat_features, num_features, removal_features = make_feature_lists(df_train)

            # 범주형 변수 인코딩
            df_train, df_test = encoding_cat_features(df_train, df_test,
                cat_features, config_params, model_name, seed)

            # 결측 처리
            df_train = filling_missing_values(df_train, cat_features, num_features, model_name)
            df_test = filling_missing_values(df_test, cat_features, num_features, model_name)

            # train 데이터 학습
            models = model_kfold(df_train, base_features, cat_features, config_params,
                model_params, model_name, seed)

            # 제출 파일 생성
            y_prob = predict_test(models, df_test, base_features)
            y_probs.append(y_prob)

make_submission(y_probs, df_submit)
```

시드 고정 (`set_seed`)

데이터셋 읽기 (`read_data`)

데이터 전처리 및 파생변수 생성 (`feature_engineering`)

범주형 변수 인코딩 (`encoding_cat_features`)

결측값 처리 (`filling_missing_values`)

train 데이터 학습 및 k-fold 교차검증 (`model_kfold`)

예측값 생성 및 제출파일 출력 (`predict_test`)

XGB 모델 채택 →

5 seed ensemble + hyperparameter tuning

* 전체 흐름 소개 _ Main Function Definition

set_seed

여러 라이브러리의 Seed 고정

read_data

train, test, sample_submission 데이터 파일 불러오기

feature_engineering

데이터 EDA에 따른 피처 엔지니어링

make_feature_lists

수치형, 범주형, 제외 피처 등 필요에 따른 피처 리스트 생성

make_submission

시드별 ‘임신 성공 확률’의 평균으로 최종 probability 계산 후 제출 파일 생성

encoding_cat_features

모델별 encoder 선택

filling_missing_values

결측치 처리

model_kfold

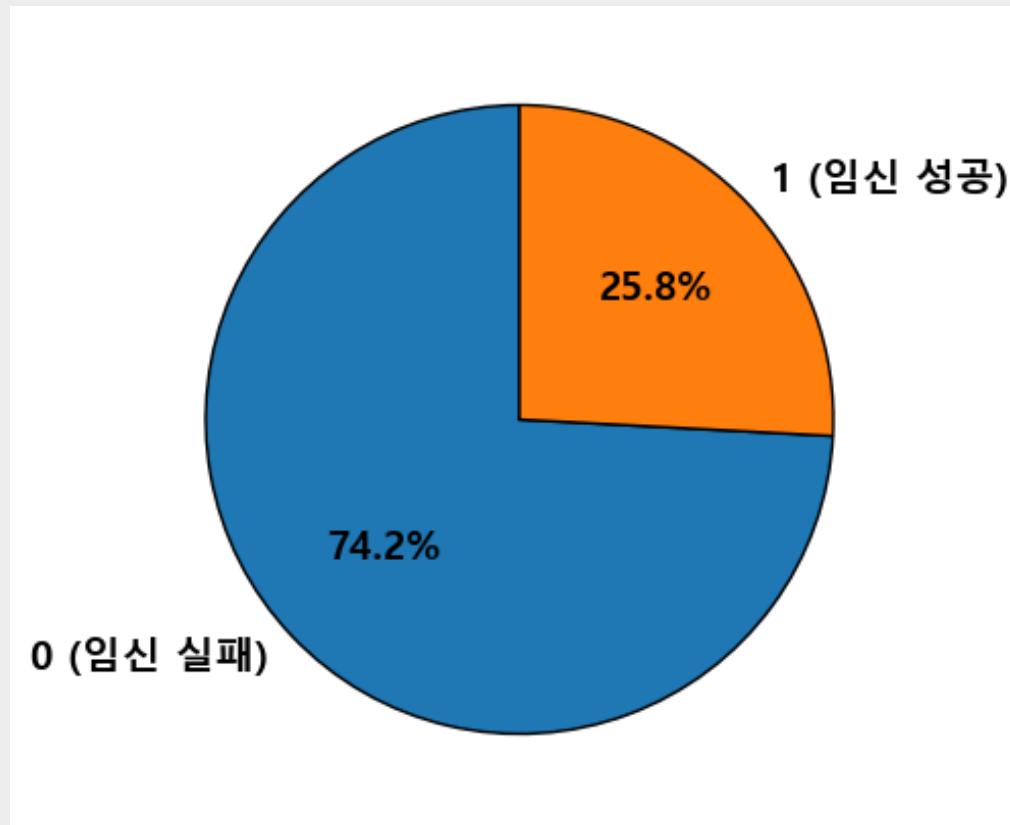
k-fold cross-validation으로 모델을 학습

predict_test

test 데이터를 fold별 모델로 추론, 평균을 예측값으로 사용

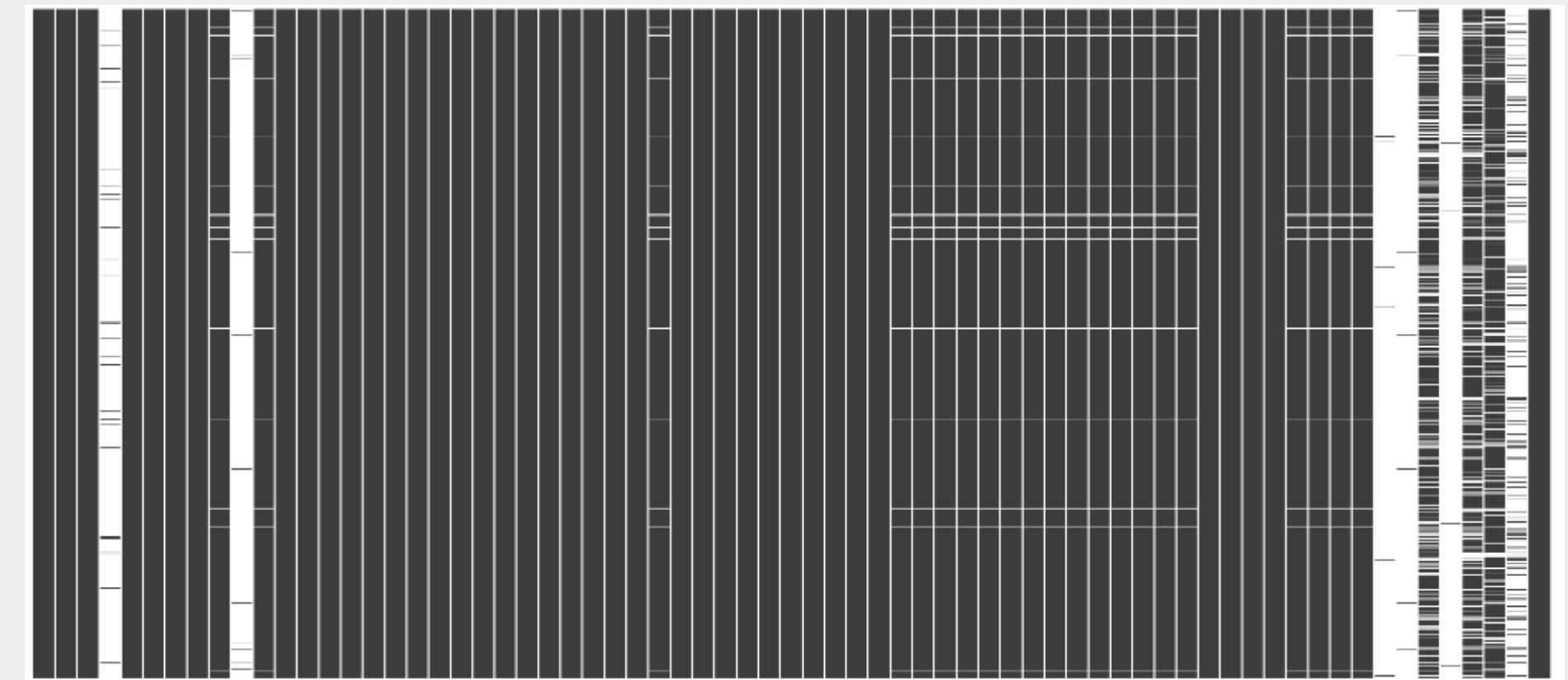
* EDA & PREPROCESSING

데이터 불균형



Target 변수의 0, 1 분포가 불균형한 것 확인

결측치 분포 확인



시술 유형에 따라 결측치 분포가 다른 것 확인
(난자 채취 및 혼합, 배아 이식 등 IVF 관련 변수의 경우,
DI 시술에서는 진행되지 않는 과정이므로 결측)

* EDA & PREPROCESSING

결측치 처리

DI 데이터의 IVF 관련 변수

DI 시술에서는 수집하지 않기에 발생한 결측으로 간주하여 -1로 대체

착상 전 유전 검사 사용 여부, PGS 시술 여부, PGD 시술 여부

결측치 제외 모든 행이 1인 여부 관련 컬럼의 경우, 결측치 0으로 대체 (해당 컬럼들은 IVF 관련 컬럼이므로 DI 데이터는 -1)

경과일 관련 컬럼

해당 과정을 진행하지 않아 발생한 결측으로 간주하여 -1로 대체

임신 시도 또는 마지막 임신 경과 연수, 특정 시술 유형

그 외 해석 불가능한 컬럼은 실험 후 가장 성능이 좋았던 -1, UNK로 대체

* EDA & PREPROCESSING

결측치 처리

결론적으로 IVF 데이터의 ‘PGS 시술 여부’, ‘PGD 시술 여부’, ‘착상 전 유전 검사 사용 여부’ 칼럼의 결측치만 0으로 처리
그 외 결측치는 전부 -1로 대체

```
# 확정 결측치 처리
for feat in ['PGS 시술 여부', 'PGD 시술 여부', '착상 전 유전 검사 사용 여부']:
    ivf_df[feat] = ivf_df[feat].fillna(0)

def filling_missing_values(df_input, cat_features, num_features, model):
    df = df_input.copy()

    if model == 'CB':
        for cat_feature in cat_features:
            df[cat_feature] = df[cat_feature].fillna('UNK')
            df[cat_feature] = df[cat_feature].astype(str)

    for num_feature in num_features:
        df[num_feature] = df[num_feature].fillna(-1)

    return df
```

* EDA & PREPROCESSING

변수 유형 변환

나이, 횟수 관련 변수 수치형으로 변환

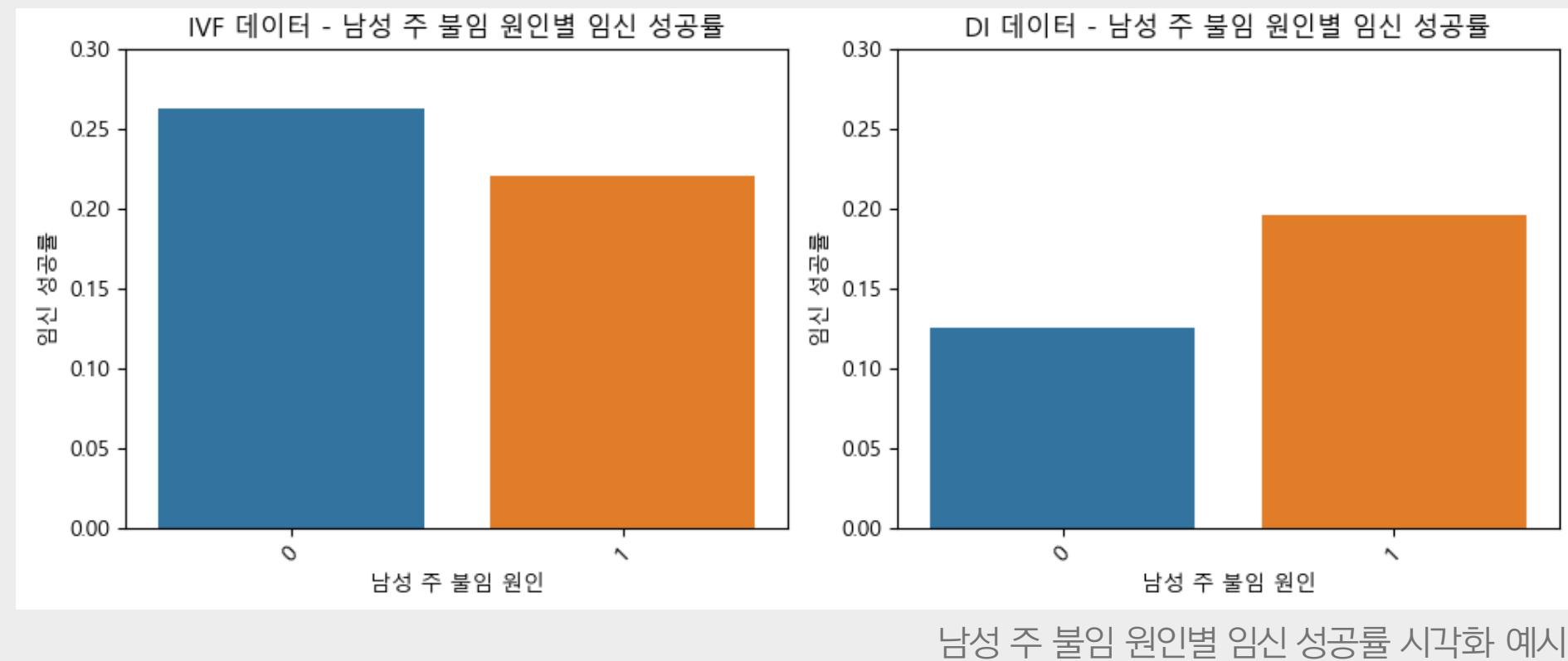
```
# '시술 당시 나이' 변수 수치형으로 변환
df['시술 당시 나이'] = df['시술 당시 나이'].map({
    '만18-34세':1, '만35-37세':2, '만38-39세':3, '만40-42세':4,
    '만43-44세':5, '만45-50세':6, '알 수 없음':7
})

# '~횟수' 변수 수치형으로 변환
count_features = [feat for feat in df.columns if '횟수' in feat]
for feat in count_features:
    df[feat] = df[feat].apply(lambda val : int(val[0])) # n회/n회 이상 (str) -> n (int)

# '정자, 난자 기증자 나이' 변수 수치형으로 변환
for feat in ['난자 기증자 나이', '정자 기증자 나이']:
    df[feat] = df[feat].apply(lambda val : int(val[1:3]) if val[0] == '만' else -1)
```

* EDA & PREPROCESSING

DI 데이터 flip



EDA 결과, 일부 불임 원인 변수에서 IVF 시술과 DI 시술의 target 분포가 다른 것을 확인.

→ 해당 결과를 보이는 불임 원인 컬럼들의 **DI 데이터 값을 flip**($0 \leftrightarrow 1$)

→ symmetric한 모델인 XGBoost에서 소수 class인 DI 시술 데이터에 대한 보정 효과

- flip 변수 : 남성 주 불임 원인, 남성 부 불임 원인, 여성 주 불임 원인, 여성 부 불임 원인, 부부 주 불임 원인, 부부 부 불임 원인, 불임 원인 - 정자 농도

* FEATURE ENGINEERING

변수 제거

신선 배아 사용 여부, 동결 배아 사용 여부

의미상 연결되는 변수인 총 생성 배아 수, 해동된 배아 수와 분포가 일치하지 않는 것 확인
⇒ 정보량이 더 많은 총 생성 배아 수, 해동된 배아 수를 남기고 변수 제거

총 생성 배아 수 > 0	신선 배아 사용 여부	개수
0	0	46161
0	1	13479
1	0	54
1	1	196657

해동된 배아 수 > 0	동결 배아 사용 여부	개수
0	0	215966
1	0	259
1	1	40126

동일한 값을 가지는 컬럼

모든 행이 동일한 값을 가지는 경우, 해당 변수 제거

```
removal_features = ['신선 배아 사용 여부', '동결 배아 사용 여부']
removal_features |= set(df.columns[df.nunique(dropna=False) == 1])
```

* FEATURE ENGINEERING

파생변수 생성

배아 생성 주요 이유

다중 선택된 경우가 있어 더 명확한 반영을 위해 더미 컬럼 생성

```
# 배아 생성 주요 이유 다중 선택 변환 (4개 더미 컬럼 생성)
categories = ['현재 시술용', '배아 저장용', '기증용', '난자 저장용']
for category in categories:
    df[f'배아 이유_{category}'] = df['배아 생성 주요 이유'].apply(lambda x: 1 if pd.notna(x) and category in x else 0)
```

출산 성공률 총 출산 횟수 / 총 임신 횟수

임신 성공률 총 임신 횟수 / 총 시술 횟수

```
# '출산 성공률', '임신 성공률' 파생변수 생성
df['출산 성공률'] = df['총 출산 횟수'] / df['총 임신 횟수']
df['임신 성공률'] = df['총 임신 횟수'] / df['총 시술 횟수']
```

* ENCODING_CATEGORY FEATURE

인코더 설정

XGB, LGBM, CatBoost 각 모델별 encoder 선택 비교 실험 후, 모델별 Encoder 채택

- label encoding
- ordinal encoding
- target encoding
- catboost encoding
- frequency encoding

실험을 통해 config_params에 지정된 모델별 Encoder 사용

- CatBoost – 자체 Encoder
- XGB – ordinal Encoder
- LGBM – targetEncoder

* K-FOLD CROSS VALIDATION

K-FOLD 교차검증

K-fold cross validation을 통해 모델 학습 및 실험 진행

target imbalance를 고려하여 **Stratified K-fold**로 fold별 target 분포 동일하게 데이터 분할

k-fold 비교 실험

5, 7, 9, 11 k의 수 변화하며 비교한 결과 fold 개수 5개로 결정 \Rightarrow ‘k-fold’ : 5

데이터를 5개로 분할한 후, 각 fold를 한 번씩 valid set으로 사용

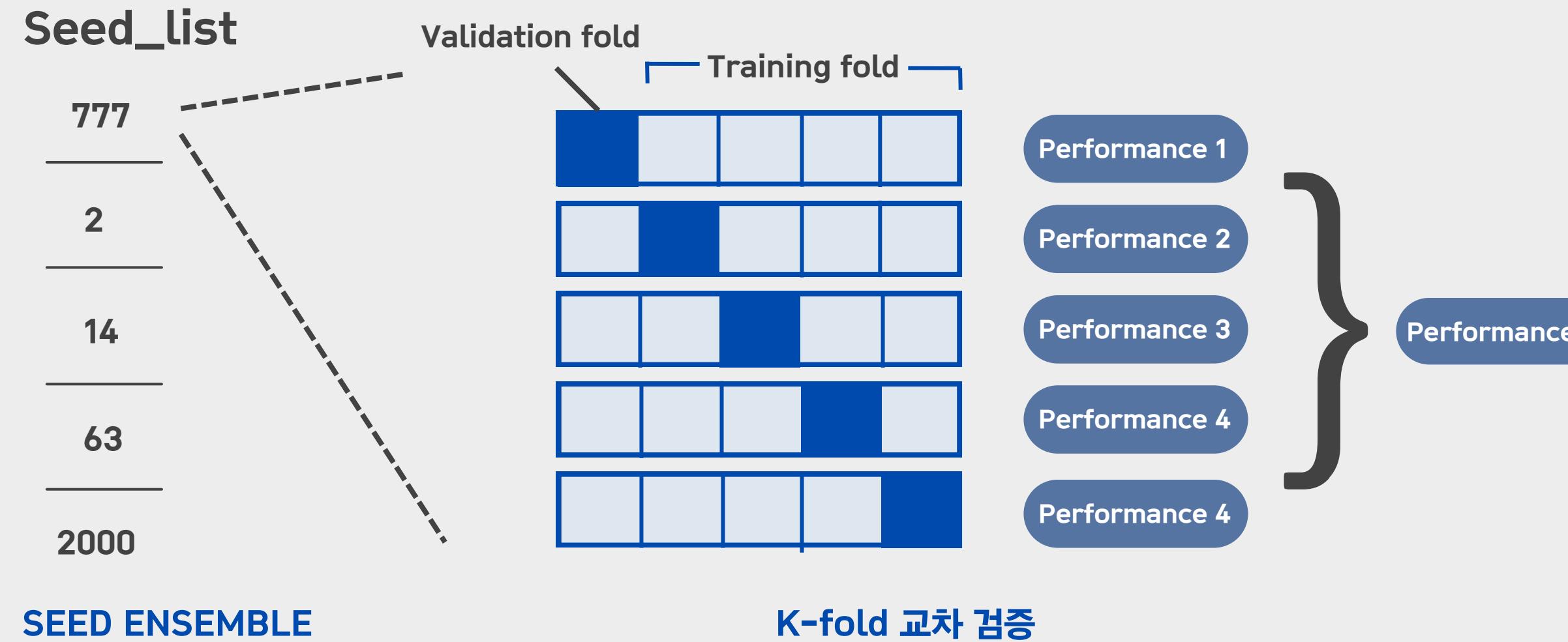
\Rightarrow 모델 성능 **일반화**와 **과적합** 방지를 위한 작업

각 fold별 valid score의 평균과 표준편차 출력 후 모든 fold의 평균값으로 해당 실험의 valid 성능 측정

```
print('Valid] ', end=' ')
y_pred = model.predict_proba(X_valid)[:,1]
score = roc_auc_score(y_valid, y_pred)
print(f'Fold #{k_fold + 1} Score: {score:.4f}')
```

```
print(f'Avg. Score of validset: {np.mean(scores)}')
print(f'Std. Score of validset: {np.std(scores)}')
print()
```

* 모델 고려 및 SEED ENSEMBLE_ 모델 구조



테스트 데이터 중 샘플링된 50%의 public 점수만 확인할 수 있기에, 모델의 과적합과 일반화에 집중
모델의 일반화, 안정성 증가를 통해 과적합을 방지를 위해 **Seed Ensemble** 사용
XGB, LGBM, CatBoost 세 가지의 모델을 이용하여 모델 양상별, 하이퍼파라미터 튜닝 등을 통해 성능 최적화 실험 진행

* HYPERPARAMETER TUNING

OPTUNA

```
def objective(trial: optuna.Trial, config_params, df, target: str, base_features, cat_features) -> float:

    if config_params['model'] == 'XGB':
        params = {
            'objective': 'binary:logistic',
            'eval_metric': 'auc',
            'n_estimators': trial.suggest_int('n_estimators', 2200, 2400),
            'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.03, log=True),
            'max_depth': trial.suggest_int('max_depth', 7, 8),
            'subsample': trial.suggest_float('subsample', 0.6, 0.9),
            'colsample_bytree': trial.suggest_float('colsample_bytree', 0.6, 0.9),
            'reg_lambda': trial.suggest_float('reg_lambda', 0.1, 1, log=True),
            'reg_alpha': trial.suggest_float('reg_alpha', 5, 9, log=True),
            'gamma': trial.suggest_float('gamma', 2.0, 3.0),
            'random_state': config_params['seed'],
        }
```

```
model_param_setting = {
    'XGB': {
        'n_estimators': 2357,
        'learning_rate': 0.016535403312726182,
        'max_depth': 7,
        'subsample': 0.6887677799128866,
        'colsample_bytree': 0.6879076928246035,
        'reg_lambda': 0.28211678046980465,
        'reg_alpha': 6.879354177483545,
        'gamma': 2.413362122265103,
        'objective': 'binary:logistic',
        'scale_pos_weight': 1,
        'verbosity': 0,
    },
}
```

Optuna를 활용하여 **XGBoost** 하이퍼파라미터 튜닝을 진행

→ 트리 개수, 학습률, 최대 깊이, 샘플링 비율, L1/L2 정규화, 감마 등의
하이퍼파라미터를 고려하여 범위를 좁혀가며 최적의 하이퍼파라미터 리스트를 출력받아, 이를 모델에 적용

* 최종 선정

```
=====
Seed Number : 777
=====
<Model : XGB>
[Valid] Fold #1 Score: 0.7384
[Valid] Fold #2 Score: 0.7420
[Valid] Fold #3 Score: 0.7376
[Valid] Fold #4 Score: 0.7416
[Valid] Fold #5 Score: 0.7428
Avg. Score of validset: 0.7404880968025566
Std. Score of validset: 0.002063187912857618
```

```
=====
Seed Number : 2
=====
<Model : XGB>
[Valid] Fold #1 Score: 0.7420
[Valid] Fold #2 Score: 0.7388
[Valid] Fold #3 Score: 0.7392
[Valid] Fold #4 Score: 0.7423
[Valid] Fold #5 Score: 0.7380
Avg. Score of validset: 0.7400497815215938
Std. Score of validset: 0.00176950084224428
```

```
=====
Seed Number : 14
=====
<Model : XGB>
[Valid] Fold #1 Score: 0.7408
[Valid] Fold #2 Score: 0.7394
[Valid] Fold #3 Score: 0.7401
[Valid] Fold #4 Score: 0.7402
[Valid] Fold #5 Score: 0.7406
Avg. Score of validset: 0.7402477093606528
Std. Score of validset: 0.0004824819346555312
```

```
=====
Seed Number : 63
=====
<Model : XGB>
[Valid] Fold #1 Score: 0.7394
[Valid] Fold #2 Score: 0.7384
[Valid] Fold #3 Score: 0.7413
[Valid] Fold #4 Score: 0.7407
[Valid] Fold #5 Score: 0.7416
Avg. Score of validset: 0.7402954986096031
Std. Score of validset: 0.0012158551039369349
```

```
=====
Seed Number : 2000
=====
<Model : XGB>
[Valid] Fold #1 Score: 0.7381
[Valid] Fold #2 Score: 0.7406
[Valid] Fold #3 Score: 0.7423
[Valid] Fold #4 Score: 0.7395
[Valid] Fold #5 Score: 0.7406
Avg. Score of validset: 0.7402266975981797
Std. Score of validset: 0.0013729474099695947
```

다양한 모델의 양상블, 하이퍼파라미터 튜닝, 시드 양상블, SMOTE 기반 오버샘플링 등의 여러 조합의 실험 끝에,

**최종적으로 Optuna 기반 하이퍼파라미터 최적화가 적용된
XGBoost 단일 모델의 5개 시드 양상블**이 최적 모델로 채택

#	팀	팀 멤버	최종점수	제출수
30	엄마손은 AI손	고갱 ye ss yo le	0.7422	129



감사합니다.

LG Almers 6th

엄마손은 AI손

고경준 김예진 나영은 이수민 정서윤