

---

# 신용카드 고객 세그먼트 분류 AI 경진대회

DACON

1. 데이터 소개


2. EDA

3. 전처리

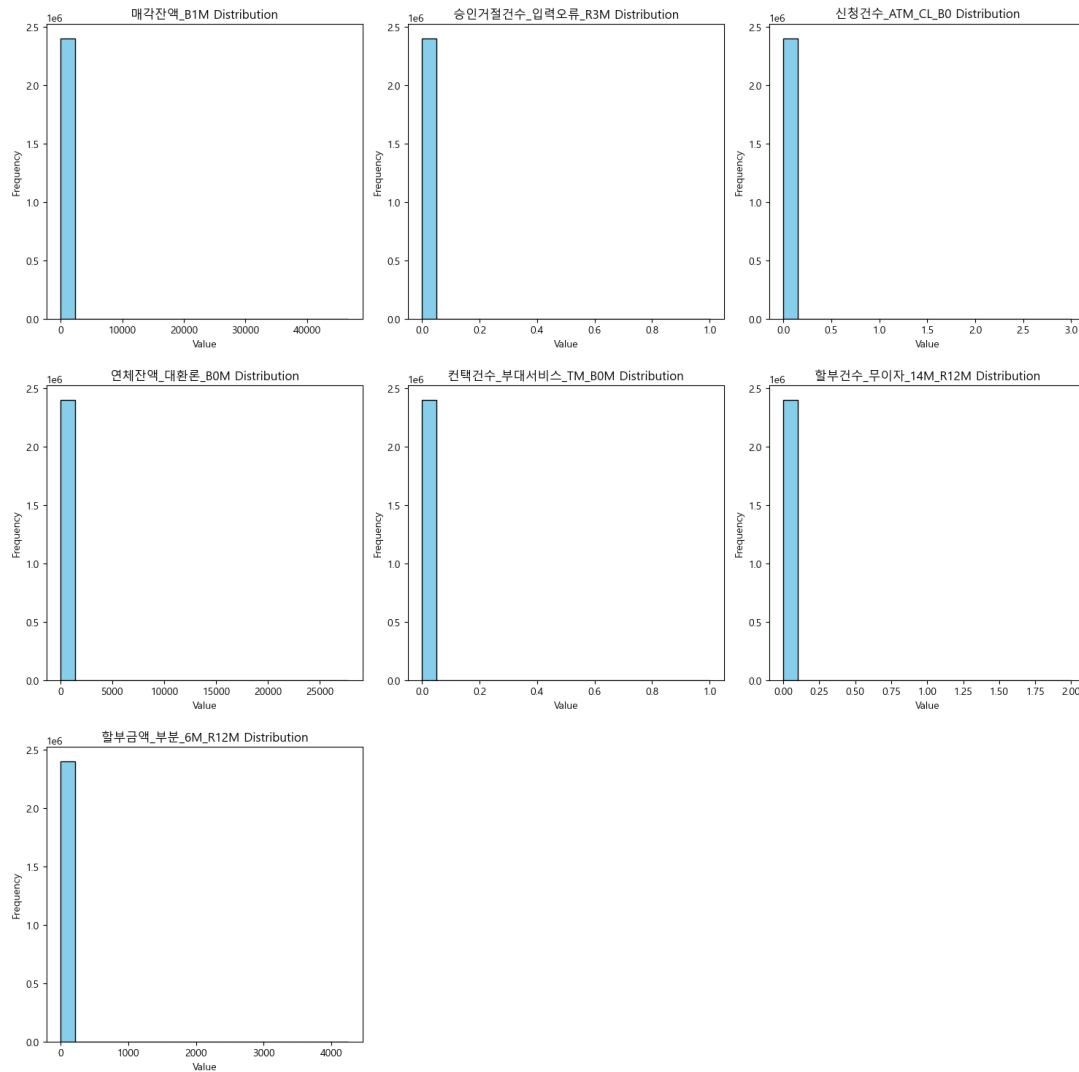
4. 모델링

---

## 데이터 소개

	Train		Test
1. 회원정보	(2400000, 78)	 77 features Segment(Target)	(600000, 77)
2. 신용정보	(2400000, 42)		(600000, 42)
3. 승인매출정보	(2400000, 406)		(600000, 406)
4. 청구입금정보	(2400000, 46)		(600000, 46)
5. 잔액정보	(2400000, 82)		(600000, 82)
6. 채널정보	(2400000, 105)		(600000, 105)
7. 마케팅정보	(2400000, 64)		(600000, 64)
8. 성과정보	(2400000, 49)		(600000, 49)

# EDA



매각잔액\_B1M

승인거절건수\_입력오류\_R3M

신청건수\_ATM\_CL\_B0

신청건수\_ATM\_CL\_B0

연체잔액\_대환론\_B0M

컨택건수\_부대서비스\_TM\_BOM

할부건수\_무이자\_14M\_R12M

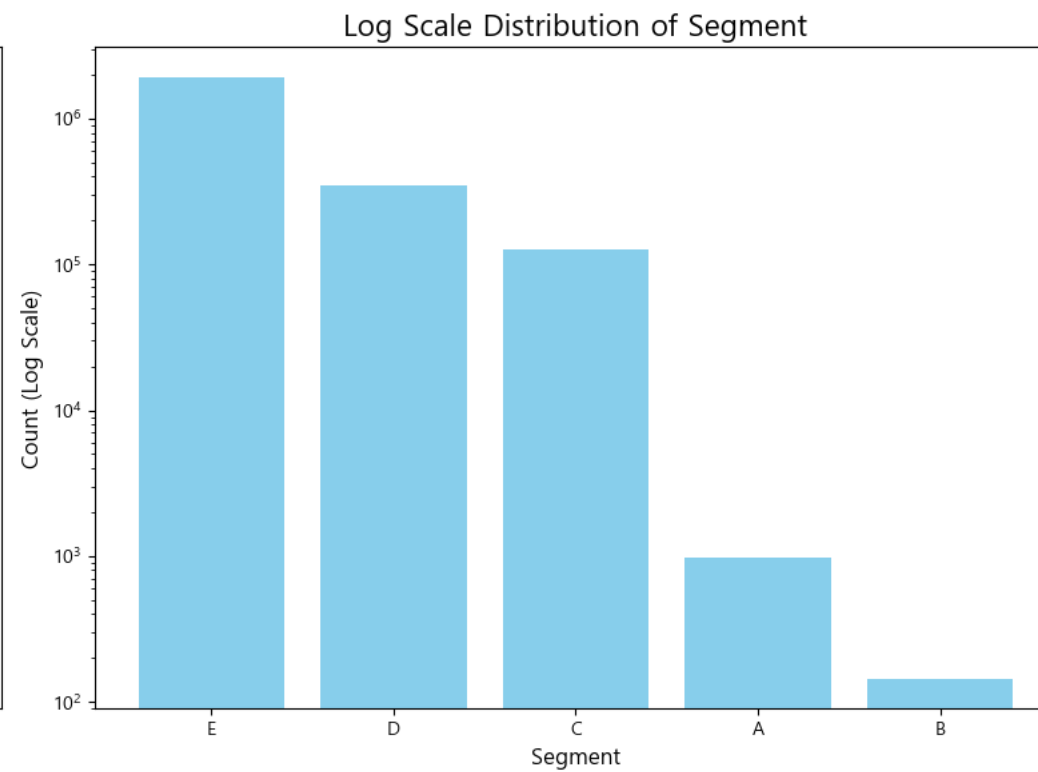
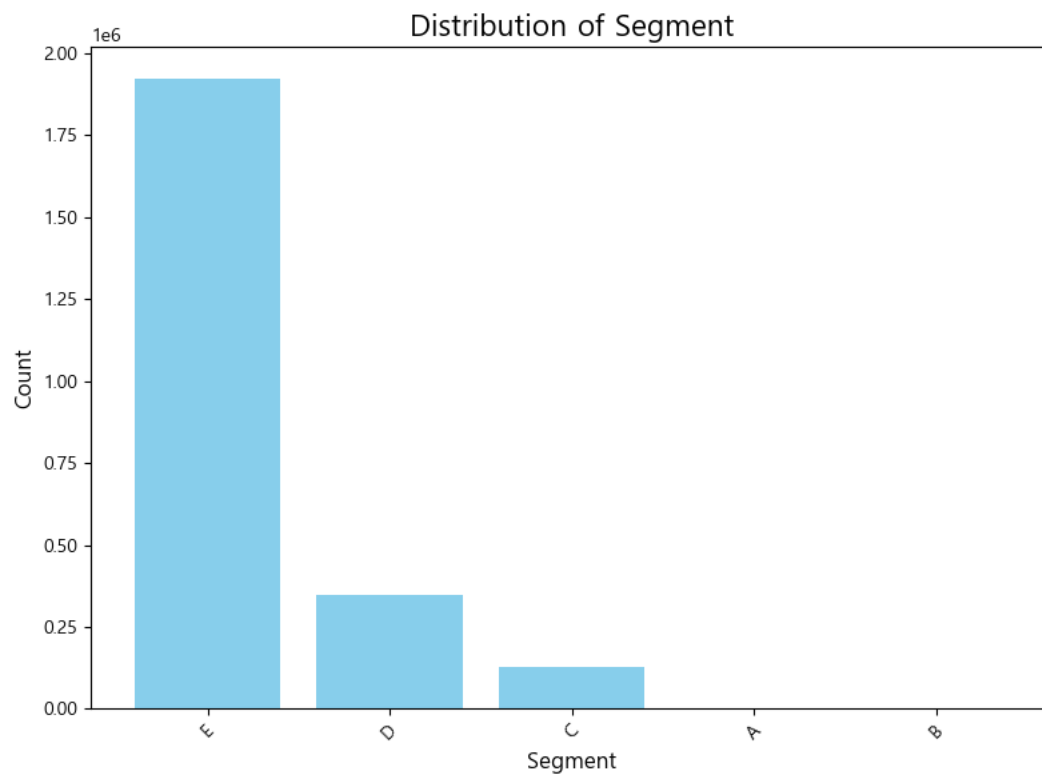
할부금액\_부분\_6M\_R12M

해당 피쳐들의 분포가

하나의 값으로 매우 치우쳐져 있어

학습에 악영향을 미칠 가능성 있음

## Segment(Target) 분포



Target variable: Imbalanced classes

### Step 1.) 데이터 병합

- 카테고리별 각 201807~201812 데이터를 병합하여 train.parquet, test.parquet 데이터 파일을 생성함.
- train.parquet: (2400000, 858)
- test.parquet: (2400000, 857)

### Step 2.) 단일 값만 가지는 컬럼들을 제거

- 병합된 train.parquet, test.parquet을 각각 불러들여 train.parquet에서 단일 값만 가지는 컬럼들을 추출(총 108개 추출)
  - 추출된 컬럼들을 기반으로 train.parquet, test.parquet에서 drop 하였음.
  - train.parquet: (2400000, 750)
  - test.parquet: (2400000, 749)
  - train\_filtered.parquet, test\_filtered.parquet으로 다시 저장함.
-

### Step 3.) 메모리 사용량 최소화

- train\_filtred.parquet, test\_filtered.parquet를 데이터 프레임으로 다시 불러옴.
- train\_columns\_type() 함수(코드 참고)로부터 데이터 프레임의 각 컬럼들의 데이터 타입을 변환하여 메모리 사용량 최소화
  - int64 → int32, float64 → float32

### Step 4.) Null 값을 가지는 컬럼들 모두 제거

- Null 적어도 한 개 이상을 가지는 컬럼이면 제거.
  - train\_filtred.parquet: (2400000, 719)
  - train\_filtred.parquet: (2400000, 718)
  - [참고] 평균 대체, 0 대체 등의 여러 기본 결측치 처리 기법을 활용하였으나, drop 하는 것이 성능이 좋았음.
-

# 전처리

## Step 5.) 불균형 분포를 가진 컬럼들 제거

매각잔액_B1M	
0	2399999
46708	1

승인거절건수_입력오류_R3M	
0	2399990
1	10

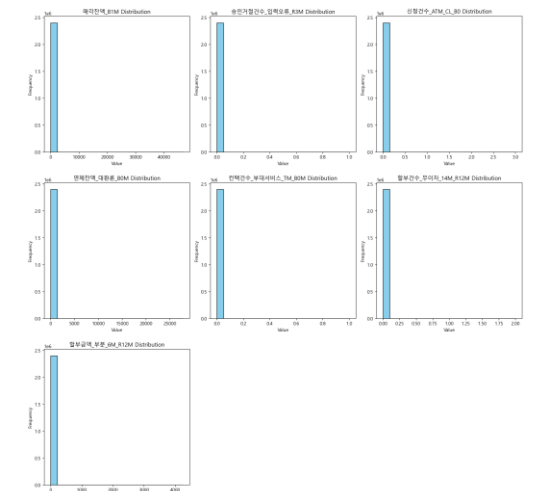
신청건수_ATM_CL_B0	
0	2399999
3	1

컨택건수_부대서비스_TM_B0M	
0	2399999
1	1

할부건수_무이자_14M_R12M	
0	2399997
2	3

할부건수_무이자_14M_R12M	
0	2399999
4249	1

연체잔액_대환론_B0M	
0	239994
27479	1
27348	1
27752	1
27018	1
27108	1
27663	1



- train\_filtred.parquet: (2400000, 712)
- train\_filtred.parquet: (2400000, 711)

EDA 참고.(page-4)

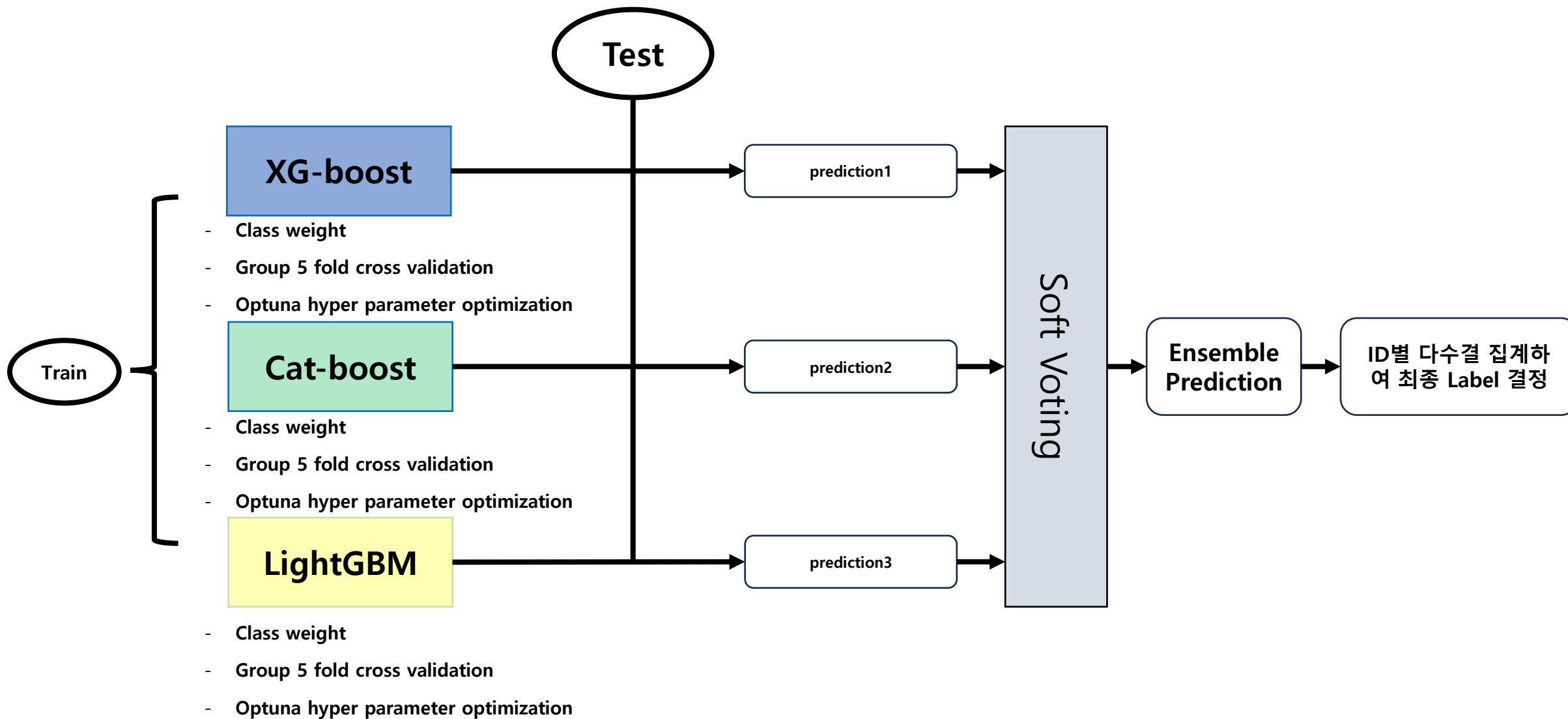


### Step 6.) Label Encoding

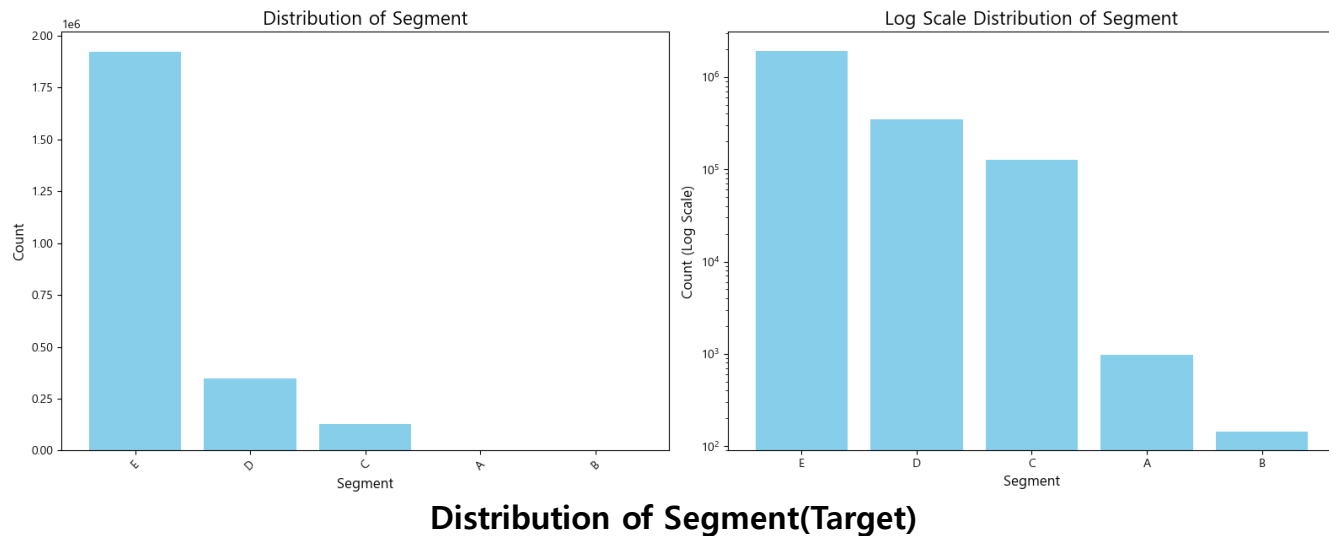
- 각 컬럼의 label encode에서 test 데이터에만 있는 범주형 변수 존재 시, label encoder에 추가하여 처리함.  
(baseline code 활용)

\* 데이터셋이 워낙 크고 high dimension이기 때문에, 불필요한 컬럼들은 최대한 줄이고자 하였고, 이후 파생변수 생성 없이 모델링 하는데 초점을 두었음.

---



## Class 불균형 문제 해결 방법 1.



- XGB, LightGBM, CatBoost 모델 각각 class weight 지정함.
- compute\_class\_weight 함수로부터 클래스의 가중치를 데이터셋에서의 불균형 정도에 따라 자동 조정함.
- Class weight 지정으로 기존 public score: 약 0.63 → 0.67로 향상

## compute\_class\_weight

`sklearn.utils.class_weight.compute_class_weight(class_weight, *, classes, y)`

Estimate class weights for unbalanced datasets.

[\[source\]](#)

### Parameters:

**class\_weight** : dict, "balanced" or None

If "balanced", class weights will be given by  $n\_samples / (n\_classes * np.bincount(y))$ . If a dictionary is given, keys are classes and values are corresponding class weights. If None is given, the class weights will be uniform.

**classes** : ndarray

Array of the classes occurring in the data, as given by `np.unique(y_org)` with `y_org` the original class labels.

**y** : array-like of shape (n\_samples,)

Array of original class labels per sample.

### Returns:

**class\_weight\_vect** : ndarray of shape (n\_classes,)

Array with `class_weight_vect[i]` the weight for i-th class.

### Class 불균형 문제 해결방법 2.

- Group 5 Fold cross validation.
- 동일 ID 내에서는 Segment 변화가 없는 것을 확인함. 데이터 누수 방지를 위해 ID 그룹 단위로 분할할 필요 있음.
- 동시에 Segment의 분포가 불균형하므로, 각 fold마다 클래스 비율이 유지되도록 할 필요 있음.
- 따라서, Stratified + Group 기준이 모두 반영된 StratifiedGroupKFold 사용.

### Optuna Hyperparameter optimization

- 각 모델별 optuna를 통해 하이퍼 파라미터 최적화 수행함.