Undersampling
& Autogluon 활용

대회 명:

웹 광고 클릭률 예측 AI 경진대회

팀원 :

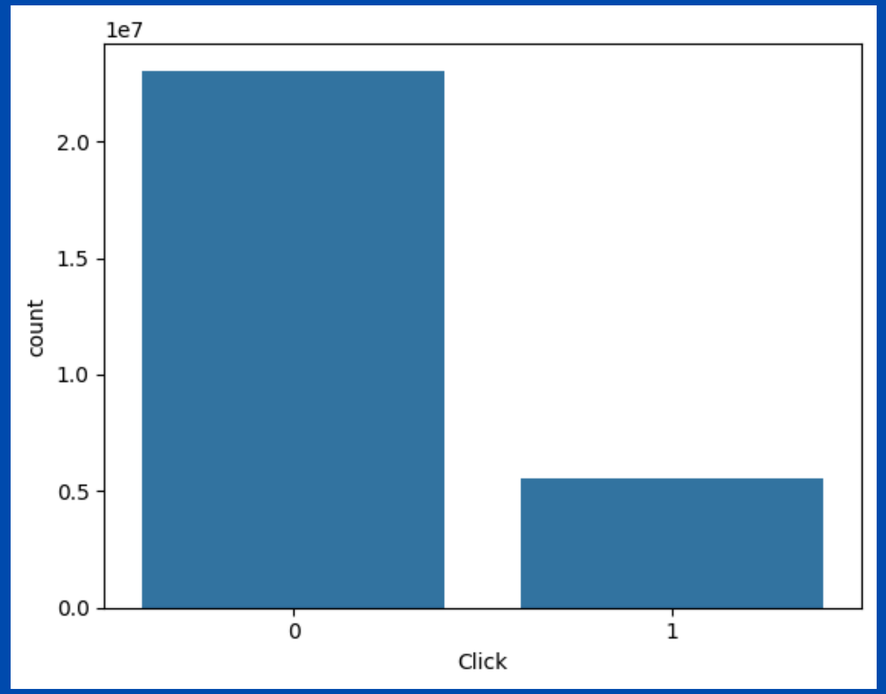
yoru

DACON

목차

1. Pre-Processing

1. Target 분포 확인



Click의 비율이

0 : 80.53 %

1 : 19.47 %

으로 "0" 인 비율이 높음

1. Pre-Processing

2. Under Sampling

언더샘플링은 불균형한 데이터셋에서 과다한 클래스의
데이터 수를 줄여 데이터의 균형을 맞추는 기법입니다.

장점 : 데이터셋의 크기를 줄여 학습 속도를 향상

단점 : 데이터의 다양성이 감소

1. Pre-Processing

2. Under Sampling

```python
def create_balanced_dataframe(df, fraction=0.1):
    # 'Click' 칼럼의 0과 1의 데이터를 분리
    df_0 = df[df['Click'] == 0]
    df_1 = df[df['Click'] == 1]

    # 각 클래스의 최소 개수로 맞추기
    min_size = min(len(df_0), len(df_1))

    target_size = int(len(df) * fraction / 2)
    target_size = min(target_size, min_size)

    # 각 클래스에서 동일한 개수의 샘플을 무작위로 추출
    df_0_balanced = df_0.sample(n=target_size, random_state=42)
    df_1_balanced = df_1.sample(n=target_size, random_state=42)

    # 두 클래스의 데이터프레임을 합쳐서 새로운 데이터프레임 생성
    df_balanced = pd.concat([df_0_balanced, df_1_balanced]).sample(frac=1, random_state=42).reset_index(drop=True)

    return df_balanced
```

1. Pre-Processing

2. Under Sampling

```python
# 새로운 1:1 비율의 데이터프레임 생성
df = create_balanced_dataframe(df, 1)

# 새로운 데이터프레임의 'Click' 칼럼 비율 확인
ratio_df_balanced = df['Click'].value_counts(normalize=True)

print(df.shape)
print(ratio_df_balanced)
```

```
(11139720, 41)
Click
1    0.5
0    0.5
Name: proportion, dtype: float64
```

"0"과 "1"을 같은 비율로 Under Sampling

2. Modeling

1. AutoGluon

AutoGluon은 Amazon Web Services(AWS)에서 개발한 자동 기계 학습(AutoML) 라이브러리입니다.

간단한 코드로 다양한 데이터 유형(표 형식 데이터, 이미지, 텍스트 등)을 처리하여 고성능 모델을 생성합니다.

2. Modeling

1. AutoGluon

```
from autogluon.tabular import TabularDataset, TabularPredictor
import autogluon.core as ag

train_data = TabularDataset(df)
test_data = TabularDataset(test)

label = 'Click'
eval_metric = 'roc_auc'
```

'Click' 레이블을 예측하기 위해 데이터셋을 로드

평가 메트릭을 'roc_auc'로 설정

2. Modeling

1. AutoGluon

```python
from autogluon.tabular import TabularPredictor

# 시간 제한 설정 (예: 12 시간)
time_limit = 12 * 60 * 60

# GPU를 사용할 수 없는 모델을 제외하도록 설정
exclude_model_types = [
    'KNN',  # K-Nearest Neighbors
    'RF',   # Random Forest
    'XT',   # Extra Trees
    'LR',   # Linear Regression
    'NN'    # Tabular Neural Network
]

# TabularPredictor 객체 생성 및 학습
predictor = TabularPredictor(
    label=label,
    eval_metric=eval_metric,
    path='AutogluonModels/ag-20240518_080907'  # 모델 저장 경로
).fit(
    train_data,
    presets='best_quality',  # 'best_quality', 'medium_quality', 'good_quality' 등의 프리셋 설정
    num_stack_levels=0,  # 스택 레벨 설정
    num_bag_folds=0,  # 배깅 설정
    time_limit=time_limit,  # 시간 제한 설정
    num_gpus=1,  # GPU 사용 설정
    excluded_model_types=exclude_model_types  # 제외할 모델 유형 설정
)
```

```
Fitting model: WeightedEnsemble_L2 ... Training model for up to 1579.47s of the -242.91s of remaining time.
        Ensemble Weights: {'LightGBMXT': 0.625, 'LightGBM': 0.125, 'CatBoost': 0.125, 'NeuralNetFastAI': 0.125}
        0.7855   = Validation score   (roc_auc)
        2.6s     = Training   runtime
        0.02s    = Validation runtime
AutoGluon training complete, total runtime = 16460.9s ... Best model: "WeightedEnsemble_L2"
```

3. Result

1. Local Score

```
print(predictor.leaderboard(silent = True))
```

```
                  model  score_val eval_metric  pred_time_val      fit_time  \
0  WeightedEnsemble_L2   0.785520     roc_auc     156.671587  15571.708904
1           LightGBMXT   0.784337     roc_auc     115.354186   4819.048686
2             LightGBM   0.781697     roc_auc      36.012419   2813.206511
3             CatBoost   0.768801     roc_auc       3.881580   1493.251512
4       NeuralNetFastAI   0.762715     roc_auc       1.402467   6443.600036
5              XGBoost   0.679288     roc_auc       1.493099    269.992987

   pred_time_val_marginal  fit_time_marginal  stack_level  can_infer  \
0                0.020935           2.602160            2       True
1              115.354186        4819.048686            1       True
2               36.012419        2813.206511            1       True
3                3.881580        1493.251512            1       True
4                1.402467        6443.600036            1       True
5                1.493099         269.992987            1       True

   fit_order
0          6
1          1
2          2
3          3
4          4
5          5
```

DACON

3. Result

2. Public Score

11	yoru	yo	0.78495

DACON

3. Result

3. Private Score

10	yoru	yo	0.7851

DACON