캐글 AdTracking Fraud Detection 데이터 EDA

대회 주제

- 스토리라인
  - 온라인 광고를 하는 기업들에게는 클릭 사기가 엄청난 양으로 발생할 수 있어, 잘못된 클릭 데이터는 돈 낭비로 이어질 수 있음.
  - 중국 최대의 독립 빅데이터 서비스 플랫폼 TalkingData는 하루에 30억 개의 클릭을 처리하며, 그 중 90%가 잠재적으로 사기성 클릭일 수 있음.
  - 클릭 사기 방지의 현재 접근 방식:
    - 사용자의 클릭 여정을 포트폴리오 전반에 걸쳐 측정
    - **많은 클릭을 발생시키지만 앱 설치로 이어지지 않는 IP 주소를 관찰.**

      → 이 정보를 통해 IP 블랙리스트와 기기 블랙리스트를 구축.
  - 성공적이었지만, 사기범들보다 항상 한 발 앞서기 위해 TalkingData는 해결책을 더욱 발전시키기 위해 Kaggle 커뮤니티의 도움을 받고자 함.

- 문제 목표

  모바일 앱 광고 클릭 후 사용자가 앱을 다운로드할지 예측하는 알고리즘을 개발하는 것

- 제공 데이터
  - 모델링을 위해 약 4일간의 약 2억 개 클릭 데이터셋을 제공
- 평가 지표
  - AUROC(area under ROC curve)

File descriptions

- **train.csv**
- **train_sample.csv**
  - train.csv에서 10만개의 랜덤 행

- **test.csv**
- **sampleSubmission.csv**
  - 제출 포맷
- **test_supplement.csv**
  - 대회 시작 전 의도치 않게 나온 test.csv보다 더 큰 data
  - official test data는 **test_supplement.csv 에 포함됨**

data features

- ip
  - 클릭 ip 주소
- app
  - 마케팅용 앱 ID
- device
  - 유저 모바일 폰의 device 타입 ID
- os
  - 유저 모바일 폰의 os 버전 ID
- channel
  - 모바일 광고 게시자의 채널 ID
- click_time
  - click 시간 (UTC)
- attributed_time
  - 유저가 광고를 클릭한 이후 앱을 다운로드 할 때, 그 시간
- is_attributed
  - 타겟 피쳐
  - 앱이 다운로드 되었는지 0/1

EDA

```
# 값의 빈도수 및 비율 계산
value_counts = train['is_attributed'].value_counts(normalize=True)

# 비율을 백분율로 변환
percentage = value_counts * 100

print(percentage)
```

```
is_attributed
0    99.773
1     0.227
Name: proportion, dtype: float64
```

- 날짜: 2017/11월
  - 훈련 데이터 - 6~9일
  - 테스트 데이터 - 10일

1등 솔루션

TalkingData AdTracking Fraud Detection Challenge
Can you detect fraudulent click traffic for mobile app ads?
https://www.kaggle.com/competitions/talkingdata-adtracking-fraud-detection/discussion/56475

I. 다운 샘플링

- is_attributed 의 불균형 0/1 → 99.8 % / 0.2%
- is_attributed 이 0인 데이터 샘플을 축소시켜도 성과에서 큰 문제가 없었다는 점 관찰.

II. 피쳐 엔지니어링

- 다운 샘플링 대신 이 전략을 통해 모든 데이터를 사용

- 사용한 변수
  - 기본 제공된 변수: ip, os, app, channel, device
    - 31 (=(2^5) - 1)개의 조합을 만들고, 각각의 조합에 대해 “click 시리즈 기반”의 특성 세트를 생성

    ex) 다음 1시간 or 6시간 내의 click 수, 앞뒤 click 시간 간격, **과거 클릭들의 평균 attributed 비율**

  - 시간 관련 변수: day, hour

  → 따로 피쳐 selection는 안하고, 위의 변수를 모두 넣어 학습. 이 때 LGB 모델의 스코어는 0.9808

III. LDA/NMF/LSA를 이용한 categorical feature embedding

- categorical feature embedding → categorical feature를 머신러닝에서 모델이 학습할 수 있도록 변수를 적절하게 가공하는 작업?

- LDA (latent Dirichlet allocation)
  - 차원 축소 기법

```
apps_of_ip = {}
for sample in data_samples:
  apps_of_ip.setdefault(sample['ip'], []).append(str(sample['app']))
ips = list(apps_of_ip.keys())
apps_as_sentence = [' '.join(apps_of_ip[ip]) for ip in ips]
apps_as_matrix = CountTokenizer().fit_transform(apps_as_sentence)
topics_of_ips = LDA(n_components=5).fit_transform(apps_as_matrix)
```

  5개의 기초 변수(ip, os, app, channel, device)으로 가능한 모든 조합(5 * (5 - 1) = 20가지)에 대해 LDA를 사용하여 주제를 추출하고, 각 주제의 크기를 5로 설정 → 총 100개의 피쳐 생성

- NMF, PCA도 비슷한 방식으로 피쳐 생성 → 총 300개의 피쳐 생성

- 이후 app 변수만 남기고 나머지 4개 기초 변수 제거 → LGBM 모델 기준, 성과 향상 (0.9821 → 09828)

IV. NN 모델

- 3층의 모델
- LGB 모델보다 조금 낮은 성과
- “성과를 올리기 위해서는 is_attributed 가 0인 데이터가 더 필요해보였다”

V. 최종 제출

- rank-based weighted averaging - 성과에 따라 랭킹을 부여하고 가중치를 부여
  - 7개의 LGBM 모델 + 1개의 NN 모델 (0.98343)