Project 04

01. [광고 클릭률 예측 Catboost]

7일간의 웹 로그를 기반으로 하루 동안의 광고 클릭률을 예측하는 AI 알고리즘 개발

1. 기본정보

- 담당역할
  - Chunk라이브러리로 대용량 데이터 로드
  - 대용량 데이터를 효율적으로 처리 할 수 있는 전처리 작업
  - Optuna로 하이퍼파라미터 튜닝

---

2. 프로젝트 진행 배경

- 데이콘에서 주최하는 예측 AI 경진대회 참여
- 대용량, 클래스 불균형, 고차원(High Cardinality)데이터를 활용한 실무경험을 쌓기 위함

---

3. 결과 및 직무에 적용할 점

- **클릭률 예측(Cick-Through Rate, CTR) 모델을 개발**
  - 실제 비즈니스 시나리오에서의 적용을 통해 매출 증대와 마케팅 효율성을 극대화
- **웹 로그 데이터를 적절하게 처리할 수 있는 AI 모델을 개발**
  - 복잡한 로그 데이터 작업을 통해 대용량 데이터 핸들링 스킬 향상

---

4. 주요 액션

Chunk 데이터 로드 → 데이터 전처리 작업 → 모델 학습 & 평가 → 하이퍼파라미터 튜닝

상세

① 데이터 로드

- 자세한 내용과 코드는 GitHub Link를 참고
- 데이콘에서 제공된 훈련데이터를 사용

    - **train.csv [파일]**
    - 시간 순으로 나열된 7일 동안의 웹 광고 클릭 로그
    - ID: train 데이터 샘플 고유 ID
    - Click: 예측 목표인 클릭 여부
    - 0: 클릭하지 않음, 1: 클릭
    - F01 ~ F39 : 각 클릭 로그와 연관된 Feature
    - 개인정보 보호를 위해 상세 정보는 비식별 처리됨

---

② 데이터 로드

- 자세한 내용과 코드는 GitHub Link를 참고
- 대용량 데이터를 제한된 RAM메모리 상 로드하기 위해 Chunk형태로 로드

```
import glob

# 청크 파일이 저장된 디렉토리 경로
directory_path = '/content/'

# 디렉토리 내에서 특정 패턴에 맞는 모든 파일 경로를 가져오기
chunk_files = sorted(glob.glob(directory_path + 'processed_chunk_*.csv'))

# 가져온 파일 리스트 출력
print(f"Found {len(chunk_files)} chunk files.")
```

```
Found 287 chunk files.
```

    - 6.97GB -> 287 chunk files
        - glob 모듈을 활용해 대량의 파일을 효율적으로 검색하고 필터링하며, 이를 통해 파일 정리 및 관리 프로세스를 자동화

③ 데이터 전처리

- 자세한 내용과 코드는 GitHub Link를 참고
- EDA결과에 따라 아래와 같이 전처리를 수행함
    - **데이터 결측치 제거**
        - 수치형 데이터의 결측치는 0 , 범주형 데이터의 결측치는 'Missing'으로 처리
    - **범주형 변수 인코딩**
        - catboost는 따로 범주형 데이터를 인코딩 할 필요는 없지만 RAM사용량을 최대한 낮춰보고자 인코딩 진행
    - **고차원 컬럼(High cardinality Columns) 제거**
        - 모델이 high cardinality 컬럼의 정보를 잘 활용하지 못하거나, 성능 향상에 기여하지 않는 경우가 있어 학습 시간과 자원 소모 절약을 위해 제거

③ 모델 학습 & 평가

- 자세한 내용과 코드는 GitHub Link를 참고
- 1차 Catboost 머신러닝 모델 학습을 통한 최적의 성능

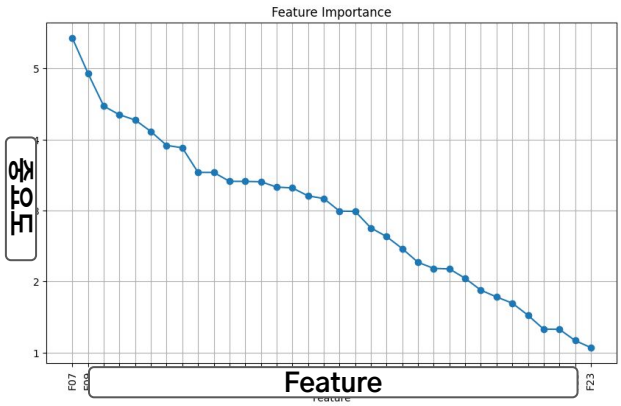
```
Accuracy: 0.8127314183539739
Precision: 0.5801578549195106
Recall: 0.13837008470589926
F1 Score: 0.22344709815976269
ROC AUC Score: 0.5570791134729078
```

  - AUC Score = 0.5570
  - 정확도는 높으나 정밀도와 재현율이 낮은걸로 보아 클릭한 광고(positive) 데이터의 비율이 너무 낮을 수 있어 데이터의 불균형이 일어났을 것이라 가정하고 scale_pos_weight로 데이터 불균형을 해소

---

④ 모델 최적화 & 재학습

- 자세한 내용과 코드는 GitHub Link를 참고
- **scale_pos_weight** = num_negative / num_positive `weight: 3.967463116586359`
- **feature_importances**




```
'Feature': ['F07', 'F09', 'F21', 'F13', 'F02', 'F06', 'F32', 'F37', 'F25', 'F31',
            'F28', 'F11', 'F39', 'F04', 'F29', 'F16', 'F08', 'F26', 'F24', 'F15',
            'F17', 'F27', 'F33', 'F03', 'F20', 'F19', 'F22', 'F36', 'F14', 'F18',
            'F30', 'F38', 'F35', 'F23'],
'Importance': [5.430621, 4.931369, 4.471226, 4.349468, 4.276111, 4.113677,
               3.918145, 3.884583, 3.537387, 3.537180, 3.412234, 3.411378,
               3.406357, 3.331293, 3.319156, 3.207146, 3.169901, 2.992534,
               2.989026, 2.755503, 2.633545, 2.463786, 2.271102, 2.185275,
               2.178459, 2.045781, 1.878657, 1.782823, 1.695817, 1.525557,
               1.328667, 1.327660, 1.169004, 1.069572]
```

  - 중요 피쳐에서는 특이점이 나타나지 않아, 추가적인 제거 작업은 수행하지 않음

- **Optuna 하이퍼파라미터 최적화**

```
best_params = {
    'iterations': 716,
    'depth': 6,
    'learning_rate': 0.1434452681311719,
    'random_strength': 9.636924231305613,
    'bagging_temperature': 0.8801310826837648,
    'border_count': 181,
    'l2_leaf_reg': 4.3754592812493405,
    'scale_pos_weight': 0.5297966866606383,
    'eval_metric': 'AUC',
    'task_type': 'CPU',
    'random_seed': 42
}
```

```
269: test: 0.7448622    best: 0.7450109 (265)    total: 16.5s
270: test: 0.7448969    best: 0.7450109 (265)    total: 16.6s
271: test: 0.7449389    best: 0.7450109 (265)    total: 16.6s
272: test: 0.7449237    best: 0.7450109 (265)    total: 16.7s
273: test: 0.7449494    best: 0.7450109 (265)    total: 16.8s
274: test: 0.7449919    best: 0.7450109 (265)    total: 16.8s
275: test: 0.7449547    best: 0.7450109 (265)    total: 16.9s
Stopped by overfitting detector  (10 iterations wait)

bestTest = 0.7450109046
bestIteration = 265
```

  - Best Test = 0.745 (AUC Score)
  - bestIteration = 265

※ 분석 회고

KPT 회고

- K(만족 및 이어갔음 하는 부분)
    - **알고리즘 모델 최적화**: 새로운 알고리즘 모델을 도입하면서 다양한 최적화 기법들을 학습하고 적용해볼 수 있었던 점이 매우 유익했습니다. 이 과정에서 모델 성능 향상에 실질적인 도움을 받을 수 있었으며, 이러한 최적화 방법에 대한 이해도를 높일 수 있었습니다.
    - **대용량 데이터 처리 경험**: 대용량 데이터를 전처리하는 과정을 통해 실무에서 자주 접할 수 있는 데이터 처리의 복잡성과 효율적인 관리 방법을 직접 경험할 수 있었습니다. 이는 앞으로의 프로젝트에서도 큰 도움이 될 중요한 경험이라고 생각합니다.
- P(개선이 필요한 부분)
    - **프로젝트 초기 기획 부족**: 데이터 분석을 시작하기 전에 전체 프로세스에 대한 철저한 기획이 부족했던 점이 아쉬웠습니다. 중간에 분석 과정이 복잡해지면서 예기치 않은 문제들이 발생했고, 결국 프로젝트를 처음부터 다시 시작해야 하는 상황에 직면하게 되어 시간과 리소스가 낭비되었습니다.
- T(P에 대한 해결책)
    - **알고리즘 및 프로젝트 기획 사전 학습**: 앞으로는 프로젝트를 시작하기 전에 단순히 알고리즘의 표면적인 장점에 의존하기보다는, 각 알고리즘의 핵심 개념, 최적화 방법, 그리고 중요한 요소들에 대해 사전에 충분히 학습할 계획입니다. 이를 통해 보다 구체적이고 체계적인 프로젝트 계획을 수립할 수 있을 것이며, 예기치 않은 문제를 예방할 수 있을 것입니다. 또한, 초기 단계에서 전체 프로세스의 로드맵을 명확히 설정하여, 프로젝트가 중간에 꼬이는 상황을 방지하고, 효율적인 진행을 도모하겠습니다.