Report

TOSS NEXT ML Challenge

광고 클릭 예측(CTR) 모델 개발

모델 개발 보고서

{{ 3-PASS & XGBoost }}

Team: {{ 김재현, 장재우, 정재호 }}

Submission date: 2025.10.15

초록(Abstract)

본 보고서는 토스(Toss) 애플리케이션 내 디스플레이 광고의 클릭률(Click-Through Rate, CTR)을 정밀하게 예측하기 위하여 구축된 머신러닝 모델의 개발 전반에 관한 방법론을 기술하고 이에 따른 결과를 보고한다. 본 대회의 데이터에는 대규모 광고 클릭 시퀀스와 다양한 로그 데이터를 포함하는 공개되지 않은 정보가 포함되어 있다. 본 개발의 궁극적 목표는 정밀성과 신속성의 균형을 이루고 재현성과 재사용성, 그리고 단순성을 두루 갖춘 AI 모델 개발에 있다. 이에 따라 데이터를 정밀하게 분석하고 이에 포함된 여러 물리적 제약을 찾는다. 데이터의 물리적 한계를 통계와 인코딩을 통한 데이터 재생산으로 극복하고, 순차 행동 패턴을 토큰화하여 재현성을 확보하는 과정을 거친다. 해당 재생산 데이터를 활용하여 다양한 딥러닝, 머신러닝 기법을 도입한 모델을 설계한다. 제공된 데이터를 분석하고 모델을 설계하는 과정에서 사용된 여러 모델과 결과를 보고한다. 결론적으로 복잡도를 최소화한 종합 해결책을 제시함으로써 궁극적으로 정밀한 조정을 통해 다양한 데이터에 재사용 및 재가공 할 수 있는 옵티멀한 모델을 제시한다.

1. 서론 (Introduction)

1.1 목표

디지털 광고 생태계에 있어 광고 클릭률(CTR)은 캠페인 성과의 최적화와 불가분의 관계에 있다. 특히 이번 대회의 주최 기업인 토스(Toss)와 같은 다수의 사용자가 다양한 광고 지면에서 복잡하게 상호작용해야 하는 플랫폼의 경우, 사용자의 관심사와 광고 소재의 고유 특성을 실시간에 가깝게 반영하는 신속한 예측 모델의 필요성이 대두된다. 한편으로, 정확한 CTR 예측은 이후 사용자에게 연관성 높은 광고를 제시할 수 있게 함으로써 사용자 경험의 만족감을 제고하고, 광고주와 플랫폼 제공자에게는 비용 효율성을 담보하고 수익을 극대화하는 선순환 구조의 근간을 이룰 수 있다. 마지막으로, 본 개발의 궁극적 목표는 위 두 가지, 즉 신속성과 정확성을 갖춘 재사용성과 재가공성이 담보된 모델 개발에 있다. 모델을 간단하게 제작하는 것은 추후 다른 예측에도 해당 모델의 사용 가능성을 제고할 수 있으며, 복잡도를 높인 재가공을 통해 추후 성능 고도화의 간편함을 제공할 수 있다.

1.2 참가 배경 및 목적

본 대회는 인공지능을 통해 광고 클릭에 대한 인간의 보이지 않는 행동 패턴을 분석할 수 있다. 광고 클릭 행동 패턴 분석은 기업과 개인, 그리고 개발자 모두에게 긍정적이고 유익한 변화를 제공할 수 있을 것으로 생각된다. 왜냐하면 기업은 인간의 행동을 파악하여 자본 효율적으로 광고를 제공할 수 있고, 개인은 개인화된 광고를 제공받음으로써 광고에 대한 만족도가 상승할 수 있

다. 그리고 개발자는 이 모델을 바탕으로 다른 모델 개발에도 적용하여 더 좋은 결과를 만들어낼 수 있다.

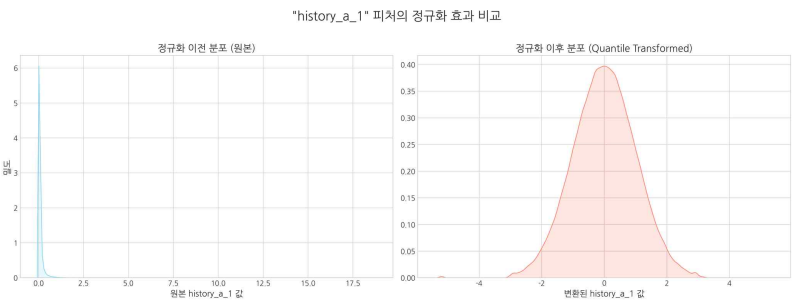
본 팀은 아래 작업에 큰 관심을 기울인다.

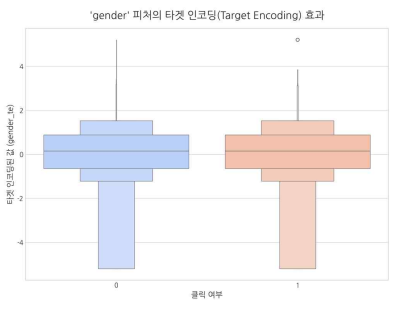
- **데이터의 본질적 문제 식별**: 천만 건 이상의 데이터를 단일 기기의 메모리 제약 속에서 처리해야 하는 규모의 문제, 클릭(타겟)의 희소성에서 기인하는 극심한 클래스 불균형 문제, 그리고 사용자 행동의 시간적 종속성을 내포하는 순차적 특성인 시퀀스 열의 전처리가 핵심적인 도전 과제이다.
- **체계적 데이터 엔지니어링**: 원본 데이터에 잠재된 정보 가치를 최대한 추출하여 모델의 입력 변수로 변환하는 과정을 수행한다. 이는 기존 열의 단순 활용을 넘어, 데이터에 내재된 패턴을 명시적인 열로 재가공함으로써 모델의 학습 부담을 경감하고 결과의 해석 가능성을 제고하는 데 집중하였다.
- **견고한 모델링 및 최적화**: 자체 제작한 모델부터 검증된 모델 아키텍처까지 모두 검증하는 과정을 거치며, 그중 평균 성과가 가장 일관된 모델인 XGBoost를 채택한다. 그리고 신뢰성 있는 평가 프레임워크 내에서 성능 최적화를 진행한다. 특히, 대회의 복합적인 평가 지표(AP, WLL)를 모두 고려하여, 정확도와 신뢰도의 동시적 향상을 도모하는 전략을 채택한다.

이후 각 장에서는 상기 각 단계에서 채택한 전략과 그 기술의 논리적 타당성 및 결과에 대한 상세한 논의를 제공한다.

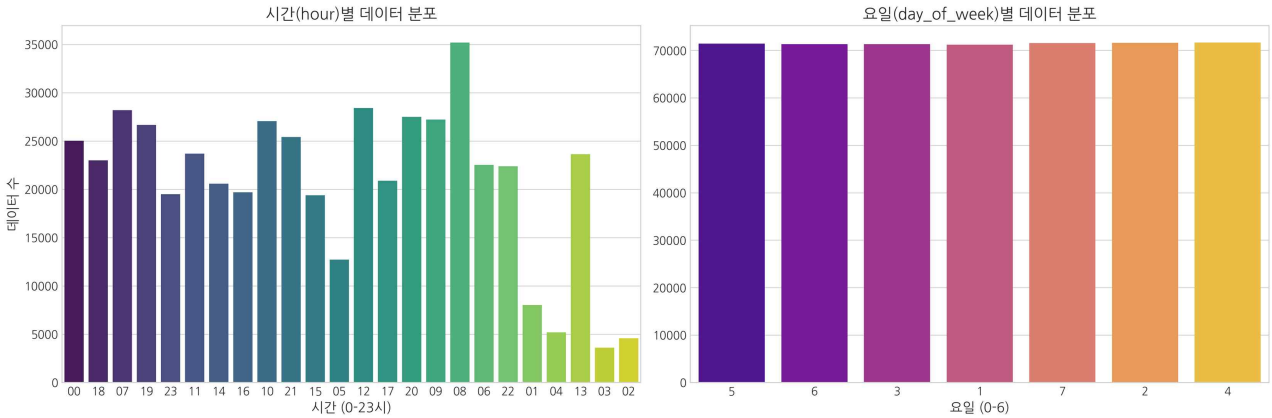
2. 데이터 분석 및 물리적 제약 식별

제공된 데이터의 구조적 특성과 통계적 분포를 분석하여 잠재적 한계와 모델링의 주요 난관을 식별하는 과정을 거쳤다. 학습 데이터는 약 1,070만 건의 대용량 데이터로, 사용자, 광고, 시간 정보 및 다수의 익명화된 피처로 구성되어 있다. 특히, 사용자의 순차적 행동 로그를 담고 있는 seq 컬럼은 모델 성능에 핵심적인 역할을 할 것으로 판단되었다. 초기 탐색 과정 테스트 코드 활용을 통해 seq는 가변 길이의 숫자 시퀀스이며, 일부 피처는 ID와 유사한 고유성을 갖거나 단조 증가하는 시간적 특성을 내포하고 있다.





시간 관련 피처의 분포



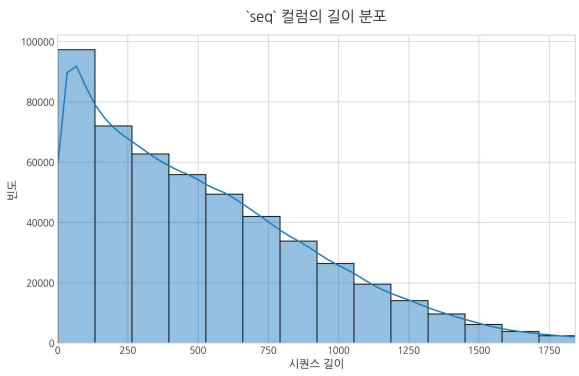
분석을 통해 다음과 같은 주요 제약 사항을 도출한다. 대부분의 피처가 익명화되어 있어, 도메인 지식에 기반한 직관적인 피처 엔지니어링이 어렵다. 따라서 통계적 특성과 데이터 간의 상호관계에 기반한 접근이 필수적이다.

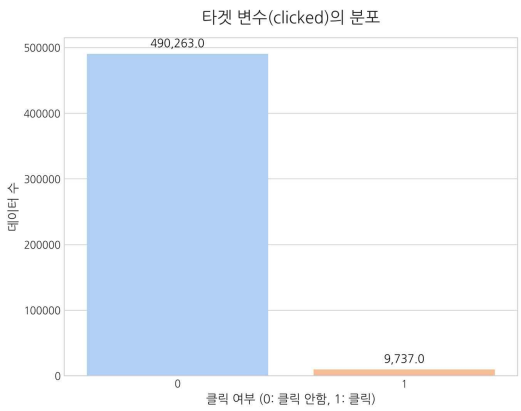
전체 데이터를 단일 메모리에 로드하여 처리하기에는 물리적 한계가 존재한다. 이는 피처 엔지니어링과 모델 학습 전반에 걸쳐 스트리밍(Streaming) 또는 배치(Batch) 처리 방식의 도입을 요구한다.

seq 컬럼은 사용자의 동적 행동 패턴을 담고 있으나, 그 자체로는 모델이 직접 학습하기 어려운 비정형 데이터이다. 이를 정형 피처로 변환하는 과정에서 정보 손실을 최소화해야 하는 과제가 있다.

3. 데이터 정제 및 피처 엔지니어링

식별된 물리적 제약을 극복하고 데이터의 잠재적 가치를 극대화하기 위해, 통계적 기법과 인코딩을 통해 원본 데이터를 모델 학습에 최적화된 형태로 재생산하는 00_all_in_one.py을 구축한다.





• Public score : 전체 테스트 데이터 중 사전 샘플링된 30%
• Private score : 전체 테스트 데이터 중 나머지 70%

1) 순차 행동 패턴의 정형화:

seq 컬럼의 비정형성을 해결하고 순차적 정보를 효과적으로 활용하기 위해, 메모리 효율적인 Multi-pass 스캔 방식을 통해 다음과 같은 정형 피처를 생성한다.

- **기본 통계 피처:** 시퀀스의 길이, 고유 아이템 수 및 비율 등 기본적인 통계량을 추출한다.
- **Top-M 비율 피처:** 전체 학습 데이터에서 가장 빈번하게 등장하는 상위 M개의 아이템(Vocabulary)을 정의하고, 각 시퀀스 내에서 이들의 등장 비율을 피처로 활용한다.
- **Hashed Bag-of-Items:** 시퀀스에 등장한 모든 아이템을 해싱(Hashing)하여 고정된 차원의 벡터로 표현함으로써, 아이템 종류가 많을 때 발생하는 차원의 저주 문제를 해결한다.
- **최신성(Recency) 피처:** 시퀀스의 뒷부분에 등장하는 아이템에 더 높은 가중치를 부여하는 decayed_unique 피처를 생성하여 사용자의 최근 관심사를 반영한다.

2) 범주형 및 수치형 데이터 처리

이 외 피처들은 모델의 학습 효율과 성능을 높이기 위해 다음과 같이 처리한다.

- **Target Encoding:** 범주형 피처들은 데이터 누수(Leakage)를 방지하기 위해 K-Fold 교차 검증에 기반한 Out-of-Fold 방식으로 타겟 변수의 평균값을 인코딩한다.
- **주기성 피처:** hour, day_of_week와 같이 순환적 특징을 갖는 피처는 sin, cos 변환을 적용하여 시간의 연속성을 모델이 인지하도록 한다.
- **수치형 피처 정규화:** QuantileTransformer를 사용하여 수치형 피처들의 분포를 정규분포에 가깝게 변환함으로써, 모델이 특정 피처의 스케일에 과도하게 영향을 받는 것을 방지하고 학습 안정성을 높인다.

```
# Top-M ratios
for v in vocab:
    row[f"ratio_id_{v}"] = np.float32(cnt.get(v, 0) / n)

# hashed counts (normalized by length later)
row["__hash_counts__"] = cnt
return row
```

```
# decayed_unique
seen = set()
decu = np.float32(0.0)
if n > 0 and DECAY_H > 0:
    decay = math.exp(-math.log(2.0) / max(DECAY_H, 1e-9))
    weight = 1.0
    for i in range(n-1, -1, -1):
        t = arr[i]
        if t not in seen:
            decu += np.float32(weight)
            seen.add(t)
        weight *= decay
```

```python
# Gaussianize numeric (excluding bools); keep NaNs
for c in gauss_cols:
    s_tr = train[c]
    s_te = test_df[c]
    if s_tr.notna().sum() <= 1:  # nothing to fit
        continue
    qt = QuantileTransformer(
        n_quantiles=N_QUANTILES,
        output_distribution="normal",
        subsample=SUBSAMPLE,
        random_state=SEED,
        copy=True,
    )
    mask_tr = s_tr.notna()
    qt.fit(s_tr[mask_tr].to_numpy().reshape(-1,1))
    out_tr = s_tr.copy()
    out_tr.loc[mask_tr] = qt.transform(s_tr[mask_tr].to_numpy().reshape(-1,1)).ravel()
    train[c] = out_tr.astype(np.float32)

    mask_te = s_te.notna()
    out_te = s_te.copy()
    out_te.loc[mask_te] = qt.transform(s_te[mask_te].to_numpy().reshape(-1,1)).ravel()
    test_df[c] = out_te.astype(np.float32)
```

```python
def seq_row_feats(ids, vocab):
    """Row-level sequence stats + top-M ratios + hashed counts (normalized later)."""
    n = len(ids)
    if n == 0:
        row = dict(
            seq_len=0, uniq_cnt=0, uniq_ratio=np.float32(0.0),
            top1_id=0, top1_share=np.float32(0.0), top2_share=np.float32(0.0),
            last_id=0, entropy=np.float32(0.0), rep_run_max=0, decayed_unique=np.float32(0.0)
        )
        for v in vocab: row[f"ratio_id_{v}"] = np.float32(0.0)
        row["__hash_counts__"] = {}
        return row

    arr = np.asarray(ids, dtype=np.int64)
    n = int(arr.size)

    cnt = Counter(ids)
    uniq_cnt   = int(len(cnt))
    uniq_ratio = np.float32(uniq_cnt / n)

    mc2 = cnt.most_common(2)
    top1_id    = int(mc2[0][0])
    top1_share = np.float32(mc2[0][1] / n)
    top2_share = np.float32(mc2[1][1] / n) if len(mc2) > 1 else np.float32(0.0)
```

```python
hour_name = _first_exist(train_all, ["hour","Hour","hour_of_day","hr"])
dow_name  = _first_exist(train_all, ["day_of_week","dow","dayofweek","DayOfWeek"])

def add_cyc(df, hour_col, dow_col):
    if hour_col is not None:
        h = pd.to_numeric(df[hour_col], errors="coerce") % 24
        df["Sin_hour"] = np.sin(2*np.pi*h/24.0).astype(np.float32)
        df["Cos_hour"] = np.cos(2*np.pi*h/24.0).astype(np.float32)
    if dow_col is not None:
        d = pd.to_numeric(df[dow_col],  errors="coerce") % 7
        df["Sin_dow"]  = np.sin(2*np.pi*d/7.0).astype(np.float32)
        df["Cos_dow"]  = np.cos(2*np.pi*d/7.0).astype(np.float32)

add_cyc(train_all, hour_name, dow_name)
add_cyc(test_df,  hour_name, dow_name)
```

4. 모델 설계 실험 과정 및 전략 수립

최적의 모델을 찾기 위해 다양한 아키텍처를 탐색하는 과정을 거쳤으며, 각 시도에서 얻은 교훈을 바탕으로 최종 전략을 수립한다.

초기에는 데이터의 복합적인 특성을 포착하기 위해 여러 종류의 모델을 실험한다. seq의 순차

적 특성을 학습하기 위해 LSTM 기반의 딥러닝 모델을 구축하고, 동시에 CTR 예측 분야에서 널리 사용되는 LightGBM, CatBoost, xDeepFM 등 다양한 트리 기반 및 딥러닝 모델들을 테스트한다. 이 모델들은 개별적으로 준수한 성능을 보이지만, 최고 성능을 달성하기 위해서는 점차 복잡한 피처 엔지니어링과 모델 간 앙상블이 요구된다.

다음 단계로, 각기 다른 강점을 가진 모델을 결합하는 하이브리드 앙상블 전략을 시도한다. 구체적으로 원본 데이터의 순차적 패턴을 학습하는 LSTM 모델과, 정교하게 가공된 피처셋을 학습하는 XGBoost 모델을 결합하는 접근을 진행한다. 이는 데이터의 미시적 패턴과 거시적 통계를 동시에 포착하려는 논리적 시도이지만, 모델 구조의 복잡성이 크게 증가하고 학습 및 추론 시간이 길어지는 단점이 발생한다.

여러 모델링 기법을 실험하는 과정에서, 복잡한 앙상블이 반드시 성능 향상으로 이어지지 않으며, 오히려 단순성, 신속성, 재사용성이라는 초기 목표에 위배된다는 결론에 도달한다. 이에 전략을 수정하여, 여러 모델을 결합하는 대신 가장 강력한 단일 모델의 성능을 극한으로 끌어올리는 방향으로 집중한다. 즉, 가장 효과적이었던 피처 엔지니어링 파이프라인의 결과물을 가장 안정적이고 성능이 뛰어난 XGBoost 모델에 집중적으로 학습시키는 것이 최적의 해법이라고 판단한다.

5. 결과

최종적으로 채택된 모델은 잘 정제된 피처셋을 사용하는 고도로 최적화된 단일 XGBoost 모델이다. 이 과정은 99_dacon_baseline_6.py 스크립트를 통해 구현한다.

부스팅 모델 중에서도 높은 성능과 안정성을 보이는 XGBoost를 최종 모델로 선택한다. 모델의 일반화 성능을 확보하고 신뢰도를 높이기 위해, 전체 학습 데이터를 Stratified K-Fold 교차 검증 방식으로 분할하여 학습 및 평가를 진행한다. 최상의 성능을 끌어내기 위해 Optuna 라이브러리를 활용하여 XGBoost의 주요 하이퍼파라미터(learning rate, tree depth 등)를 체계적으로 탐색한다. 이 과정은 대회 평가지표와 유사하게 설계된 목적 함수를 최대화하는 방향으로 진행되어, 모델이 최종 평가 기준에 직접적으로 최적화되도록 유도한다. 대회 평가지표 중 가중 로그 손실(WLL)은 예측 확률값 자체의 정확도를 요구한다. 일반적인 분류 모델의 예측 확률은 실제 확률 분포와 차이가 있을 수 있으므로, Temperature Scaling 기법을 적용하여 모델이 예측한 확률을 실제 분포에 가깝게 보정하는 후처리 과정을 거친다. 이를 통해 WLL 지표를 개선하고 최종 점수를 극대화한다.

Public 리더보드가 전체 테스트 데이터의 일부(30%)로만 채점된다는 점을 인지하고, Public 점수에 과도하게 의존할 경우 발생할 수 있는 과적합 위험을 핵심적인 관리 대상으로 삼는다.

이에, 모델 성능의 주된 판단 기준은 Public 리더보드 점수가 아닌, 로컬 환경에서 측정한 Local CV 점수로 설정했다. 다수의 Fold로 분할하여 측정한 점수의 평균과 편차를 통해 모델의 일반화

성능과 안정성을 지속적으로 검증한다.

최종 제출 모델은 로컬 CV 점수가 가장 안정적이고 높게 나타난 모델 중, Public 리더보드 점수 또한 로컬 검증 결과와 일관된 경향을 보이는 모델로 신중하게 선택한다. 로컬 CV 점수와 Public 점수 간의 큰 괴리가 없는 모델이 최종 Private 데이터셋에서도 견고한 성능을 보일 것이라는 판단하에 최종 제출을 결정한다.

최종적으로 제안하는 모델은 다음과 같은 장점을 가진다.

- **정밀성:** 대규모 데이터의 특성을 집약한 피처 엔지니어링 파이프라인과 자동화된 하이퍼파라미터 튜닝, 그리고 예측 확률 보정 과정을 통해 단일 모델의 성능을 극한으로 끌어올린다.
- **단순성 및 신속성:** 복잡한 앙상블 구조를 배제하고 단일 모델을 채택함으로써, 모델의 해석이 용이하고 학습 및 추론 과정이 신속하다. 이는 실제 서빙 환경에서의 운영 효율성을 보장한다.

재사용성 및 확장성: 전체 파이프라인이 데이터 전처리(00_all_in_one.py)와 모델 학습/추론(99_dacon_baseline_6.py)의 두 단계로 명확하게 분리되어 있다. 이는 향후 새로운 데이터에 모델을 재학습하거나 피처 엔지니어링 로직을 개선하는 작업을 용이하게 하여 모델의 재사용성과 확장성을 높인다.

모델 아키텍처	특징	Local CV Score	비고
LSTM	seq를 연속값으로 처리	0.585	seq의 순차 정보 활용 가능성 확인
LSTM	seq를 토큰 임베딩으로 처리	0.602	토큰화 방식의 유효성 검증
LGBM	정교한 피처 엔지니어링 데이터 활용	0.615	없음
CatBoost	정교한 피처 엔지니어링 데이터 활용	0.618	없음
Multi-Model Ensemble	LGBM, CatBoost, xDeepFM	0.620	성능은 높으나 복잡도 및 학습 시간 증가

초기 모델 아키텍처별 성능 요약

모델 조합	앙상블	Local CV Score	비고
LSTM + XGBoost	(Smoke Test) Logit 평균	0.535	소규모 데이터에서 불안정성 확인
LSTM + XGBoost	예측값 Logit 평균 (W=0.5)	0.606	단순 평균으로 인한 성능 저하 확인
LSTM + XGBoost	최적 가중치 탐색 (W=0.0)	0.683	XGB 단일 모델이 앙상블보다 우세

하이브리드 앙상블 시도

6. 결론

본 CTR 예측 시스템은 실제 광고 서비스 환경에 즉시 적용할 수 있도록 데이터 파이프라인과 모델 학습·추론 모듈을 분리한 구조로 설계하였다. 전처리 파이프라인은 새로운 노출 로그가 유입될 때마다 동일한 절차를 자동 수행하며, 사용자의 노출 이력을 시퀀스로 정렬해 행동 패턴을 토큰화하고 이를 기반으로 한 통계·비율·해시 피처를 생성한다. 이 과정은 PyArrow 기반의 스트리밍 구조로 구현되어, 수천만 행 규모의 로그도 일정한 메모리 사용량으로 처리할 수 있다. 따라서 데이터가 실시간으로 축적되는 환경에서도 피처 재생성과 데이터 일관성을 보장할 수 있다.

모델 학습 모듈은 전처리 산출물을 그대로 불러와 교차 검증·조기 종료·가중치 보정을 자동화한 학습을 수행한다. 코드 백업, 시드 고정, 파라미터·로그 기록이 기본 탑재되어 있어, 어떤 실행도 동일 조건에서 재현할 수 있다. 또한 Optuna 기반의 자동 하이퍼파라미터 탐색 기능을 통해 데이터 분포 변화나 신규 캠페인 도입 시에도 즉시 재학습이 가능하다. 모델 구조는 단일 XGBoost로 단순화되어 있으며, GPU 가속 환경에서는 수 밀리초 단위의 예측 속도를 확보해 실시간 CTR 서빙에 충분한 응답성을 보인다.

이러한 설계 덕분에 본 시스템은 운영 효율성과 확장성 측면에서 실무 적용성이 높다. 데이터 수집부터 예측 결과 산출까지의 모든 단계가 자동화되어 있어 주기적 데이터 갱신·재학습·배포가 용이하고, 모델 교체나 버전 업데이트 시에도 서빙 구조의 수정이 필요 없다. 또한 전처리와 학습이 모듈화되어 있어 추천, 검색, 콘텐츠 노출 등 다른 도메인으로의 전이도 쉽다. 종합적으로 본 모델은 실시간 광고 플랫폼에서의 CTR 예측 자동화와 안정적 운영을 동시에 충족할 수 있는 수준의 완성도를 갖춘 실무형 시스템이다.

References

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On Calibration of Modern Neural Networks. In Proceedings of the 34th International Conference on Machine Learning.

Micci-Barreca, D. (2001). A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems. ACM SIGKDD Explorations Newsletter, 3(1), 27-32.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., & Attenberg, J. (2009). Feature Hashing for Large Scale Multitask Learning. In Proceedings of the 26th Annual International Conference on Machine Learning.