



[9주차] CatBoost: unbiased boosting with categorical features

0. Abstract

📌 범주형 데이터를 처리하기에 적합한 CatBoost 알고리즘을 소개

- CatBoost는 기존의 부스팅 구현체들보다 다양한 데이터셋에서 더 나은 성능을 보임
- Catboost에서 소개된 두 가지 핵심 알고리즘 기술은 다음과 같음

1 Ordered Boosting

- 기존 부스팅 알고리즘의 대안으로, 순열 기반의 고전적 알고리즘을 대체함
- 이를 통해 **Target leakage**(타깃 누출)로 인한 **Prediction Shift**(예측 편향) 문제를 해결할 수 있음

2 범주형 특성 처리 알고리즘

- 타깃 누출로 인한 예측 편향 문제를 해결하기 위해 고안됨
- 기존 부스팅 알고리즘의 한계를 극복하고자 개발됨

➡ 결과적으로 CatBoost의 제안된 알고리즘들이 타깃 누출 문제를 효과적으로 해결하여 우수한 실험 결과를 달성함

1. Introduction

- 기존 그래디언트 부스팅의 통계적 문제
 - 부스팅 과정에서 얻은 예측 모델 F 는 모든 학습 데이터의 타깃 값에 의존하게 됨
 - 학습 데이터의 $F(x_k) \mid x_k$ 분포와 테스트 데이터의 $F(x) \mid x$ 분포 사이에 편향이 발생

➡ **Target Leakage**, 학습된 모델의 예측 편향을 초래(**Prediction Shift**)

- Target Leakage: 종속변수 y 를 예측할 때 독립변수인 x 만을 활용하지 않고, y 에 대한 정보가 x 에 포함되어 있는 문제
- 또한 범주형 특성을 처리하는 표준 알고리즘에서도 유사한 문제가 발생할 수 있음
 - 범주형 특성을 타깃 통계량으로 변환하는 방법은 자체적으로 target leakage을 일으킬 수 있음
- 해결 방안
 - ordering principle을 활용하여 문제 해결 >> 이를 바탕으로 ordered boosting이라는 표준 그래디언트 부스팅 알고리즘의 변형을 개발
 - 범주형 특성 처리를 위한 새로운 알고리즘을 제안
 - 이러한 기술들이 통합된 오픈소스 라이브러리인 CatBoost를 개발했으며, 기존 최첨단 구현체인 XGBoost와 LightGBM을 능가하는 성능을 보임

2. Background

문제 설정

- 데이터셋 $D = \{(x_k, y_k)\} (k=1 \dots n)$ 이 주어짐
 - $x_k = ((x_k)^1, \dots, (x_k)^m)$ 는 m 개의 특성으로 구성된 랜덤 벡터
 - y_k 는 타깃 값으로, 이진 분류 또는 수치 예측 문제일 수 있음
- 이 데이터는 알 수 없는 분포 $P(\cdot, \cdot)$ 에서 독립적으로 동일하게 분포된 것으로 가정
- 목표는 기대 손실 $L(F) := E[L(y, F(x))]$ 을 최소화하는 함수 $F: R^m \rightarrow R$ 을 학습하는 것
 - m 차원 입력 벡터 x 에 대해 실수 값 y 의 기대 손실 $L(F)$ 를 최소화하는 함수 F 를 학습
 - 즉 F 는 R^m (m 차원 유클리드 공간)에서 R (실수 집합)로 매핑되는 함수

그래디언트 부스팅 알고리즘

- 그래디언트 부스팅은 순차적으로 근사 함수 F^t 를 구축하는 방식
- $F^t = F^{(t-1)} + \alpha h^t$ 와 같이 이전 근사 함수에 새로운 base predictor h^t 를 더해 업데이트
 - α : step size
 - $h^t: R^m \rightarrow R$, 손실을 최소화하기 위한 함수

- h^t 는 다음과 같이 선택됨:

$$h^t = \arg \min_{h \in H} \mathcal{L}(F^{t-1} + h) = \arg \min_{h \in H} \mathbb{E} L(y, F^{t-1}(\mathbf{x}) + h(\mathbf{x})).$$

- 즉, 기대 손실을 최소화하는 h^t 를 H 함수군에서 찾음
- 이때 $h^t(\mathbf{x})$ 는 $-g^t(\mathbf{x}, y)$ 를 근사하도록 선택됨

$$g^t(\mathbf{x}, y) := \left. \frac{\partial \bar{L}(y, s)}{\partial s} \right|_{s=F^{t-1}(\mathbf{x})}.$$

- 즉, 손실 함수 L 의 F^{t-1} 에 대한 gradient를 근사
- 보통 최소 제곱 근사를 사용함:


$$h^t = \arg \min_{h \in H} \mathbb{E} \left(-g^t(\mathbf{x}, y) - h(\mathbf{x}) \right)^2.$$

의사결정 트리 기반 구현

- CatBoost는 이러한 그래디언트 부스팅 알고리즘을 구현한 것
- 여기서 base predictor h 는 이진 의사결정 트리로 사용됨
- 의사결정 트리 $h(\mathbf{x})$ 는 다음과 같이 표현됨:

$$h(\mathbf{x}) = \sum_{j=1}^J b_j \mathbb{1}_{\{\mathbf{x} \in R_j\}},$$

- R_j : 트리의 리프 노드에 해당하는 특성 공간의 disjoint 영역
- b_j : 각 리프 노드에 할당된 예측값

 이와 같이 CatBoost는 그래디언트 부스팅 프레임워크를 사용하면서, 범주형 특성 처리와 타깃 누출 문제 해결에 초점을 맞추고 있음

3. Categorical features

3.1 Related work on categorical features

- 범주형 특성(Categorical Features)의 처리
 - 범주형 특성은 서로 비교할 수 없는 이산적인 값들의 집합으로 구성됨
 - 범주형 특성을 다루는 대표적인 방법: **원-핫 인코딩**
 - 각 범주에 대해 새로운 이진 특성을 추가하는 방식
 - 하지만 범주의 개수가 많은 경우(사용자 ID 등), 특성의 개수가 너무 많아진다
는 문제가 있음
 - **타깃 통계(Target Statistics, TS)**를 이용한 방법
 - 범주를 **제한된 수의 클러스터로 그룹화**하고, **각 클러스터의 타깃 평균값**을 새로운 **수치형 특성**으로 사용하는 방법
 - 원-핫 인코딩보다 정보 손실이 적고, 계산 및 메모리 효율성이 높음
 - 타깃 통계 특성은 클릭률 예측 등 다양한 문제에서 중요한 역할을 함
 - **LightGBM**의 접근법
 - LightGBM은 각 단계에서 범주형 특성의 gradient 통계를 계산하여 사용함
 - 이 방법은 계산 시간과 메모리 사용량이 많은 단점이 존재
 - LightGBM은 꼬리 범주를 하나의 클러스터로 그룹화하여 이 문제를 해결하려 했지만, 정보 손실이 발생함
 - LightGBM은 여전히 범주형 특성을 수치형 특성으로 변환하는 것이 좋다고 제안함
- ➡ 타깃 통계(TS)를 이용한 방법이 정보 손실을 최소화하면서 효율적임
- ➡ 본 논문에서는 **타깃 통계 기반 방법**에 초점을 맞추고, 원-핫 인코딩과 gradient 통계 방법은 다루지 않음
- ➡ 다만 논문에서 제안하는 ordering 원리는 gradient 통계 방법에도 효과적일 것으로 보임

3.2 Target statistics

Greedy TS

- 가장 간단한 방법은 각 범주의 평균 타깃값을 사용하는 것

- 범주형 변수 x_i 를 수치형 변수 \hat{x}_{ik} 로 변환하는 식

$$\hat{x}_{ik}^i = \frac{\sum_{j=1}^n \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + a p}{\sum_{j=1}^n \mathbb{1}_{\{x_j^i = x_k^i\}} + a},$$

- 분자: k번째 범주에 속하는 **예제들의 타깃값의 합** + smoothing 효과
- 분모: k번째 범주에 속하는 **예제들의 수** + smoothing 파라미터
- \hat{x}_{ik} : k번째 예제의 i번째 범주형 변수를 수치형 변수로 변환한 값
- $\mathbb{1}_{\{x_{ij}=x_{ik}\}}$: j번째 예제의 i번째 변수 값이 k번째 예제의 i번째 변수 값과 같으면 1, 아니면 0
- y_j : j번째 예제의 타깃값
- a : smoothing 파라미터 (0보다 큰 값)
 - smoothing(분자) ➡ 노이즈를 줄이고 일반화 성능을 향상시킴
- p : 전체 데이터셋의 평균 타깃값
- 문제점: **Target Leakage**
 - 위 방법의 문제점은 \hat{x}_{ik} 를 계산할 때 y_k (k번째 예제 타깃값)을 사용한다는 것
 - 이로 인해 **학습 데이터와 테스트 데이터의 \hat{x}_{ik} 분포가 달라지는 문제 발생 (Conditional Shift)**
 - 이 문제는 모델의 일반화 성능을 크게 저하시킬 수 있음
- 해결책: 타깃 누출 방지
 - 타깃 누출 방지를 위해 **\mathbf{x}_k 를 제외한 나머지 데이터**로 \hat{x}_{ik} 를 계산

$$\hat{x}_{ik}^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + a p}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}.$$

- \mathcal{D}_k : \mathbf{x}_k 를 제외한 나머지 데이터
- 나머지 변수는 위와 동일

- 위 수식과 같이 타깃 누출을 방지하면 **학습 데이터와 테스트 데이터의 \hat{x}^i_k 분포가 유사**해져 모델의 일반화 성능이 향상됨

Holdout statistics

- 훈련 데이터셋 D 를 두 부분 D^0 와 D^1 로 나눔
 - D^0 부분을 사용하여 TS(Target Statistics)를 계산
 - D^1 부분을 사용하여 모델을 학습
- P1 속성(**TS가 타깃 누수(target leakage)가 없어야 한다**)은 만족하지만, "**모든 훈련 데이터를 TS 계산과 모델 학습에 효과적으로 사용해야 한다**"는 P2 속성을 위반하게 됨
- Holdout TS 방식은 D^0 와 D^1 로 데이터를 나누기 때문에, **모든 데이터를 효과적으로 사용하지 못함**

Leave-one-out TS

- 각 훈련 예제 x_k 를 제외한 나머지 데이터 $D_k = D \setminus x_k$ 를 사용하여 TS를 계산
- 전체 데이터 D 를 사용하여 모델을 학습
- 그러나 여전히 타깃 누수 문제가 발생
 - 상수 카테고리컬 특성 $\mathbf{x}^i_k = \mathbf{A}$

$$\hat{x}_k^i = \frac{n^+ - y_k + a p}{n - 1 + a}$$

- n^+ 는 $y = 1$ 인 예제의 수, y_k 는 k 번째 예제의 타깃 값
- k 번째 예제의 TS 값을 계산하는 식
- 이 식을 이용하면 훈련 데이터셋을 완벽하게 분류할 수 있는 임계값 t 를 찾아낼 수 있음

$$t = \frac{n^+ - 0.5 + a p}{n - 1 + a}$$

Ordered TS

- CatBoost는 기존 gradient boosting 알고리즘의 타깃 누수 문제를 해결하기 위해 "ordered TS"라는 새로운 접근법을 제안함
- 이 방식은 Online 학습 알고리즘에서 영감을 받았으며, Online 학습에서는 **훈련 예제가 시간 순서대로 제공됨**
- 따라서 각 예제의 TS 값은 **이전에 관찰된 데이터(history)에만 의존함**

Online 학습

- 새로운 데이터가 실시간으로 들어올 때 이를 활용하여 **모델을 점진적으로 업데이트**하는 방식
- 기존의 배치 학습(Batch Learning)과 달리, 온라인 학습은 전체 데이터를 한 번에 학습하는 것이 아니라 **데이터가 순차적으로 들어올 때마다 모델을 점진적으로 업데이트함**

장점

- 실시간으로 새로운 데이터를 반영할 수 있어 **빠르게 변화하는 환경에 적응**할 수 있음
- 전체 데이터를 한 번에 학습할 필요가 없어 **메모리와 계산 자원을 절약**할 수 있음
- 데이터가 지속적으로 생성되는 경우 효과적으로 대응할 수 있음

단점

- 이전에 학습된 데이터를 저장하지 않기 때문에 **과거 데이터에 대한 정보를 잃을 수 있음**
- 새로운 데이터가 잘못되었거나 관련성이 없는 경우 모델 업데이트가 잘못될 수 있어 **모니터링이 필요함**

- Offline 환경에서의 적용
 - CatBoost는 위의 아이디어를 오프라인 환경에 적용하기 위해 **"인공적인 시간"**을 도입함 ➡ **훈련 데이터의 순서를 무작위 순열 σ 로 정의함**
 - 그리고 각 예제 x_k 에 대해 이전에 관찰된 "history" $D_k = \{x_j : \sigma(j) < \sigma(k)\}$ 를 사용하여 TS를 계산함

- $\sigma(j) < \sigma(k)$ 는 j번째 데이터 포인트가 k번째 데이터 포인트보다 "인공적인 시간"상 앞서 관찰되었음을 의미
 - D_k 는 k번째 데이터 포인트 x_k 이전에 관찰된 모든 데이터 포인트 x_j 의 집합임
 - 반면 테스트 데이터의 경우 전체 데이터 D 를 사용하여 TS를 계산함
- Ordered TS의 장점
 - 이렇게 계산된 Ordered TS는 **P1 속성**(TS가 타깃 누수가 없어야 한다)을 만족함
 - 또한 전체 훈련 데이터를 사용하여 모델을 학습할 수 있어 **P2 속성**도 만족함
- Variance 문제 해결
 - 단, 만약 하나의 순열만 사용한다면 이전 예제의 TS 값이 후속 예제보다 분산이 훨씬 더 크게 됨
 - 이로 인해 이전 예제의 TS 값이 후속 예제에 큰 영향을 미치게 되어 **모델의 안정성과 일반화 성능이 저하될 수 있음**
 - 이를 해결하기 위해 CatBoost는 gradient boosting의 각 단계에서 **서로 다른 순열**을 사용함
 - 각 boosting 단계마다 새로운 순열 σ 를 생성하여 적용
 - 각 예제의 TS 값이 **이전 예제와 독립적으로 계산되어**, 분산 문제를 효과적으로 해결할 수 있음

4. Prediction shift and ordered boosting

4.1 Prediction shift

- Prediction Shift 문제
 - 일반적인 그래디언트 부스팅 알고리즘에서는 **학습 데이터의 타겟 값을 사용하여 그래디언트를 계산함**
 - 이때 **학습 데이터의 그래디언트 분포와 실제 테스트 데이터의 그래디언트 분포가 달라지며** 이를 **"Prediction Shift"**라고 함
 - 학습 데이터에 기반한 예측값과 실제 테스트 데이터의 예측값이 달라지는 문제가 발생하는 것
- Prediction shift의 원인 ➡ **"Target Leakage"에 의해 발생**

- 각 부스팅 단계에서 사용되는 그래디언트는 현재 모델 $F^{(t-1)}$ 이 학습된 동일한 데이터 포인트의 타겟 값을 사용하여 계산됨
- 따라서 학습 데이터의 $F^{(t-1)}(x_k) \mid x_k$ 분포와 실제 테스트 데이터의 $F^{(t-1)}(x) \mid x$ 분포가 달라지게 됨
- 해결책: **Ordered Boosting**
 - CatBoost는 "Ordered Boosting"이라는 방법을 사용하여 문제를 해결함
 - Ordered Boosting은 각 부스팅 단계에서 **서로 다른 순열**을 사용하여 그래디언트를 계산함
 - 이를 통해 **학습 데이터와 테스트 데이터의 그래디언트 분포 차이**를 줄일 수 있음

Related work on prediction shift

- Out-of-bag 추정을 기반으로, Breiman은 "out-of-bag" 잔차 추정치를 사용하여 각 반복에서 배깅된 약 학습기를 구축하는 반복 배깅을 제안함
- ➡ 그러나 여전히 잔차 추정치가 편향되어 있으며 배깅 방식은 데이터 버킷 수만큼 학습 시간을 증가시킴
- Friedman[13]이 제안한 각 반복마다 데이터셋을 부분 샘플링하는 방법 또한 문제를 완전히 해결하지는 못함

예측 편향 문제에 대한 기존 해결책들도 완전하지 않기에, 이 문제를 해결하기 위한 보다 체계적인 접근이 필요함

Analysis of prediction shift

- 회귀 문제와 제곱 손실 함수
 - 회귀 문제에서 제곱 손실 함수 $L(y, \hat{y}) = (y - \hat{y})^2$ 를 사용하는 경우를 다룸
- 음의 경사도와 잔차

$$h^t = \arg \min_{h \in H} \frac{1}{n} \sum_{k=1}^n \left(-g^t(\mathbf{x}_k, y_k) - h(\mathbf{x}_k) \right)^2$$

- 음의 경사도 $-g_t(x_k, y_k)$ 는 이전 단계 모델 $F_{t-1}(x_k)$ 와 실제 타겟 y_k 의 차이인 잔차 $r_{t-1}(x_k, y_k) = y_k - F_{t-1}(x_k)$ 로 대체될 수 있음
- 가정
 - 2개의 독립 동일 분포(i.i.d.) 베르누이 랜덤 변수 x_1, x_2 가 존재
 - 타겟 함수는 $y = f^*(x) = c_1x_1 + c_2x_2$
 - 2단계의 경사 부스팅을 수행하며, 깊이 1의 결정 스템프(decision stump)를 사용하고 학습률 $\alpha = 1$
 - 최종 모델은 $F = F_2 = h_1 + h_2$ 이며, h_1 은 x_1 에 기반하고 h_2 는 x_2 에 기반함 ($|c_1| > |c_2|$ 인 경우 일반적)
 - 위 가정 하에서 예측 편향 문제를 분석하고자 함
- Theorem1 분석
 - 독립 데이터셋 사용 시 무편향 추정**
 - 만약 h_1 과 h_2 를 각각 독립된 데이터셋 D_1 과 D_2 를 사용해 추정한다면, 최종 모델 $F_2(x)$ 의 기댓값은 $f^*(x) + O(1/\sqrt{n})$ 로 참 함수 $f^*(x)$ 에 수렴함
 - 즉, 독립 데이터셋을 사용하면 무편향 추정이 가능함
 - 동일 데이터셋 사용 시 편향 발생**
 - 만약 동일한 데이터셋 $D = D_1 = D_2$ 를 사용해 h_1 과 h_2 를 추정한다면, 최종 모델 $F_2(x)$ 의 기댓값은 $f^*(x) - 1/(n-1)c_2(x_2 - 1/2) + O(1/\sqrt{n})$ 로 편향됨
 - 이 편향은 데이터 크기 n 에 반비례하며, 참 함수 f^* 의 계수 c_2 에 비례함
 - 편향 발생 원인
 - h_1 의 기댓값은 테스트 데이터에서 $c_1x_1 + c_2/2$ 이지만, 학습 데이터에서는 $c_1x_1 + c_2/2 - c_2*(2x_2 - 1)/n + O(2^{-n})$ 로 편향됨
 - 이로 인해 h_2 의 기댓값도 테스트 데이터에서 $c_2*(x_2 - 1/2)*(1 - 1/(n-1)) + O(2^{-n})$ 로 편향됨
 - 결과적으로 $h_1 + h_2$ 의 기댓값은 $f^*(x) - 1/(n-1)c_2(x_2 - 1/2) + O(1/\sqrt{n})$ 로 편향됨

4.2 Ordered Boosting

- 기존 boosting 알고리즘은 예측 편향(prediction shift) 문제에 취약했음
 - ➡ 이를 해결하기 위해 저자들은 새로운 **boosting 알고리즘을 제안**
- 이상적인 알고리즘
 - 무한한 양의 학습 데이터가 있다면, 각 boosting 단계에서 새로운 독립 데이터셋 D_t 를 샘플링하여 편향 없는 잔차를 계산할 수 있음
- 현실적인 제약
 - 실제로는 레이블된 데이터가 제한적
 - l 개의 트리를 학습할 때, 각 예제 x_k 에 대해 F_{l-1} 모델을 x_k 없이 학습해야 편향 없는 잔차 $r_{l-1}(x_k, y_k)$ 를 얻을 수 있음
 - 이는 **모든 예제에 대해 편향 없는 잔차를 계산해야** 하므로, 학습 과정이 불가능한 것으로 보임
- Ordered Boosting의 아이디어
 - 예제 순서를 **랜덤 순열 σ** 로 정하고, 각 예제 i 까지 학습한 모델 M_i 를 유지
 - 예제 j 의 잔차를 계산할 때는 M_{j-1} 모델을 사용
- 한계점
 - **n 개의 다른 모델**을 유지해야 하므로, 복잡도와 메모리 요구사항이 n 배 증가함
 - 실제 응용 분야에서는 실용적이지 않음
- CatBoost에서의 구현
 - CatBoost는 이 ordered boosting 아이디어를 기반으로 gradient boosting 알고리즘을 구현함
 - 결과적으로 예측 편향 문제를 해결하면서도 실용적인 알고리즘을 제공함

Ordered boosting with categorical features

- 본 논문에서는 TS 계산과 ordered boosting을 위해 각각 다른 랜덤 순열 σ_{cat} , σ_{boost} 를 사용
- 두 알고리즘을 결합할 때는 $\sigma_{cat} = \sigma_{boost}$ 로 해야 예측 편향(prediction shift)을 방지할 수 있음

- 이렇게 해야 타깃 y_i 가 모델 M_i 학습에 사용되지 않을 수 있음
- TS 계산이나 gradient 추정에 y_i 가 사용되지 않음

➡ TS 계산과 모델 학습에 동일한 랜덤 순열 σ 를 사용해야 예측 편향을 방지할 수 있음

5. Practical implementation of ordered boosting

- CatBoost에는 Ordered 모드와 Plain 모드라는 두 가지 부스팅 모드가 존재
 - Plain 모드는 내장된 ordered TS를 사용하는 표준 GBDT 알고리즘
 - Ordered 모드는 Algorithm 1의 효율적인 수정 버전
- 알고리즘 설명
 - CatBoost는 학습 데이터셋에 대해 $s+1$ 개의 독립적인 랜덤 순열을 생성함
 - $\sigma_1, \dots, \sigma_s$ 는 트리 구조(내부 노드)를 정의하는 분할을 평가하는 데 사용됨
 - σ_0 는 얻어진 트리의 리프 값 b_j 를 선택하는 데 사용됨
- 순열 사용의 이점
 - 특정 순열에서 역사가 짧은 예제의 경우, TS와 ordered boosting의 예측 (Algorithm 1의 $M_{\sigma(i)-1}(x_i)$)이 높은 분산을 가질 수 있음
 - 여러 개의 순열을 사용하면 이러한 효과를 줄일 수 있음
 - 즉 **Ordered Boosting** 모드는 학습 데이터셋에 대한 여러 개의 랜덤 순열을 사용하여 예측의 분산을 줄일 수 있음

📌 CatBoost의 Ordered Boosting 모드 구현 세부사항

1 트리 구축 과정

- CatBoost는 oblivious decision trees(또는 decision tables)를 사용함
- Ordered Boosting 모드에서는 학습 과정 중 지원 모델 $M_{r,j}$ 를 유지함
- 각 반복 단계 t 에서 랜덤 순열 σ_r 을 샘플링하고, 이를 기반으로 트리 T_t 를 구축함
- 트리 구축 시 이전 예제의 gradient $\text{grad}_{r,\sigma(i)-1}(i)$ 를 사용하여 근사화
- 리프 값 $\Delta(i)$ 는 동일한 리프에 속한 이전 예제들의 gradient 평균으로 계산됨

2 리프 값 선택

- 최종 모델 F 의 리프 값은 표준 gradient boosting 절차에 따라 계산됨

- 새로운 예제 예측 시에는 전체 학습 데이터에 대한 TS를 사용

3 복잡도 개선

- 실제 구현에서는 $Mr,j(i)$ 대신 $MOr,j(i) = Mr,2j(i)$ 만 저장하고 업데이트하여 복잡도를 줄임

4 기타 구현 세부사항

- 카테고리컬 특성 조합을 추가 특성으로 사용
- 부스팅 절차에서 Bayesian bootstrap 가중치를 적용함
- 순열의 초기 예제들은 gradient 분산이 높아 제외함

6. Experiments

- **CatBoost와 기존 라이브러리 비교**
 - CatBoost는 XGBoost, LightGBM 등 오픈소스 라이브러리와 비교했을 때 모든 데이터셋에서 더 나은 성능을 보임
 - 대부분의 데이터셋에서 CatBoost의 성능 향상이 통계적으로 유의미했음(p-value < 0.01)
- **Ordered Boosting vs Plain Boosting**
 - Ordered Boosting 모드가 특히 작은 데이터셋에서 더 좋은 성능을 보였음
 - 이는 Ordered Boosting이 편향(bias)을 줄이는 데 효과적이기 때문으로 해석됨
 - 데이터셋 크기를 줄여가며 실험한 결과, 데이터셋이 작아질수록 Plain Boosting 모드의 성능이 상대적으로 더 나빠짐
- **타겟 통계량(TS) 비교**
 - CatBoost의 ordered TS가 다른 TS 방법들보다 우수한 성능을 보임
 - 홀드아웃 TS가 두 번째로 좋은 성능을 보였는데, 이는 conditional shift 문제를 해결했기 때문
 - 그리디 TS와 leave-one-out TS는 데이터셋에 따라 성능 차이가 큼
- **특성 조합의 효과**
 - 특성 조합 개수를 1에서 2로 늘리면 로그 손실이 평균 1.86% 감소함
 - 조합 개수를 3까지 늘리면 2.04% 감소했지만, 그 이상 늘리면 성능 향상이 미미함

- 순열 개수의 효과

- 순열 개수 s 를 1에서 3으로, 다시 9로 늘리면 로그 손실이 각각 0.19%, 0.38% 감소함

➡ 종합적으로, CatBoost는 기존 라이브러리에 비해 우수한 성능을 보였으며, 특히 Ordered Boosting 모드와 ordered TS가 효과적이었음. 또한 특성 조합과 순열 개수 증가도 성능 향상에 기여함

7. Conclusion

- 예측 편향 문제 해결
 - 기존 그래디언트 부스팅 구현에서 존재하는 예측 편향(prediction shift) 문제를 확인하고 분석함
 - 이를 해결하기 위해 ordered boosting with ordered TS라는 일반적인 솔루션을 제안
- CatBoost 구현
 - 제안한 솔루션을 CatBoost라는 새로운 그래디언트 부스팅 라이브러리에 구현함
- 실험 결과
 - CatBoost가 기존 GBDT 패키지들을 능가하며, 일반적인 벤치마크에서 새로운 최신 성과를 달성했음을 실험적으로 입증

논문에 대한 의견 및 의문점(꼭지)

➡ 위 논문을 통해 CatBoost가 기존 GBDT 패키지를 능가하는 성능을 보였지만, 더 큰 데이터셋이나 복잡한 문제에서의 확장성도 고려해볼 필요가 있다고 생각함. 더 다양한 기능들을 더하여 CatBoost의 활용 범위를 넓힐 수 있을 것 같다고 봄