DBpia

논문 21-46-06-08 The Journal of Korean Institute of Communications and Information Sciences '21-06 Vol.46 No.06
https://doi.org/10.7840/kics.2021.46.6.993

XGBoost 기반의 조기 중지를 활용한 광고 클릭 예측 방안

한 영 진◆, 조 인 휘°

Prediction of Ad Clicks Using Early Stop Based on XGBoost

Young-Jin Han◆, In-Whee Joe°

요 약

웹 사이트 및 소셜 미디어 플랫폼에 특정 사용자가 광고를 클릭하는지 예측하는 머신러닝 알고리즘으로 데이터 훈련을 지속적으로 시킬수록 트레이닝 데이터 셋은 성능이 향상되는 반면에 테스트 데이터 셋은 고정된 횟수의 학습 반복 후에 성능이 향상되지 않는 과적합 문제가 발생한다. 이 논문에서 과적합을 피하기 위해 기존 알고리즘보다 XGBoost 알고리즘[1]을 기반으로 학습 프로세스의 조기 중지를 제안한다. XGBoost는 복잡한 데이터 모델을 훈련시켜 과적합[2]을 피하는 방법이며, 별도의 테스트 데이터 군집에서 학습되는 모델의 성능을 모니터링하고 고정된 횟수의 교육 반복 후에 테스트 데이터 집합의 성능이 향상되지 않은 경우 교육 절차를 중지한다. 테스트 데이터 셋의 성능이 감소하기 시작하는 변곡점을 자동으로 선택하여, 모델의 과적합에 따라 트레이닝 데이터 셋의 성능이 계속 향상되는 과적합을 피하면서 정확도를 구현하였다. 마지막으로 실험 결과 Logistic Regression 알고리즘, Decision Tree 알고리즘과 비교하여 XGBoot 알고리즘 기반의 성능 향상을 확인할 수 있었다.

키워드 : 머신러닝, 익스트림 그라디언트 부스팅, 그라디언트 부스팅, 과적합, 조기중지
Key Words : Machine Learning, XGBoost, Gradient Boosting Machine, Overfitting, Early Stopping

ABSTRACT

Continuous data training on websites and social media platforms with machine learning algorithms that predict if a particular user clicks on ads results in better performance for training datasets, while test datasets experience overfitting problems that do not improve after a fixed number of learning iterations. In this paper, we propose an early stop of the learning process based on the XGBoost algorithm rather than the existing algorithm to avoid overfitting. XGBoost is a method to avoid overfitting by training complex data models, monitoring the performance of the models learned in a separate cluster of test data and stopping the training procedure if the performance of the test dataset has not improved after a fixed number of training iterations. We automatically select inflection points where the performance of the test dataset begins to decrease, thus implementing accuracy while avoiding overfitting, which continues to improve the performance of the training dataset according to the model's overfitting. Finally, the experimental results showed performance improvements based on the XGBoot algorithm compared to the Logistic Regression algorithm and the Decision Tree algorithm.

◆ First Author : Hanyang University Department of Computer Science, sni94@hanyang.ac.kr, 정회원
° Corresponding Author : Hanyang University Department of Computer Science, iwjoe@hanyang.ac.kr, 정회원
논문번호 : 202101-023-C-RN, Received January 24, 2021; Revised March 16, 2021; Accepted May 18, 2021

www.dbpia.co.kr

Ⅰ. 서 론

학습 데이터는 실제 데이터의 일부분이며, 빅 데이터에서 전체를 수집해서 학습을 시킨다는 건 측정 불가능에 가깝다. 또한 실제 데이터에서 학습 데이터로 오차가 증가하는 포인트를 예측하는 지점을 찾기란 아주 어렵다고 본다. 이러한 문제를 해결하기 위한 연구로 인공신경망(Artificial neural network, ANN)은 뛰어난 알고리즘으로 평가되었으며 대표적인 최적화 기법이다. 본 논문은 속도와 성능에 최적화되어 있는 트리 기반 모델 중에서 XGBoost로 마케팅 대행사의 쇼핑몰의 광고 데이터를 사용하여 과적합을 피하기 위한 조기 중지 연구로 광고 클릭 예측의 성능을 높이고자 한다.

XGBoost는 Gradient Boosting 트리 알고리즘에서 효율적인 오픈 소스 구현이다. Gradient Boosting은 더욱 단순하고 약한 모델 세트의 추정치의 앙상블을 결합하여 대상 변수를 정확하게 예측하려 시도하는 지도 학습 알고리즘이다. 다양한 데이터 유형, 관계 및 분산을 강력하게 처리하기 때문이며 개선된 결과를 위해 수정 및 최적화할 수 있는 다량의 Hyper Parameter 때문이며, 이러한 유연성 덕분에 XGBoost는 회귀, 분류 및 순위 관련 문제에 있어 안정적인 선택이 된다고 본다.

Ⅱ. 본 론

2.1 부스팅 개요

XGBoost는 기본적으로 부스팅(boosting)이라 불리는 기술을 사용한다. 약한 분류기를 묶어서 정확도를 예측한다. 여러 가지 물질에 서로 다른 종류의 개체를 조합하고 분류하는 과정과 비슷하다.

학습기 A에 대해 Z를 예측할 확률은 아래와 같다.

$$Z = A(x) + Error \quad (1)$$

'Error'에 대해 정밀하게 분류할 학습기 B가 있다고 가정하면(Error > Error2)

$$Error = B(x) + Error\ 2 \quad (2)$$

'Error 2'를 더 정밀하게 분리할 수 있는 학습기 C가 있다고 가정하면(Error 2 > Error 3)

$$Error2 = C(x) + Error\ 3 \quad (3)$$

(1)에 (2). (3)을 적용할 경우

$$Z = A(x) + B(x) + C(x) + Error\ 3 \quad (4)$$

학습기 A를 단독으로 사용했을 때보다 정확도가 높지만 A, B, C 분류기는 성능이 다르다. 모두 똑같은 비율(1*A + 1*B + 1*C)을 하고 있기 때문에 임의의 x에 대해 간섭하여 오류가 상승하는 결과가 나올 수 있다. 각 모델 앞에 가중치를 주고, 기계 학습으로 최적의 비중을 찾는다. (4)의 모델보다 성능(Error 3 > Error 4)이 좋은 분류기가 된다.

$$Z = \alpha*A(x) + \beta*B(x) + \gamma*C(x) + Error\ 4 \quad (5)$$

적응형 부스팅(Adaptive boosting)은 모든 관측치가 올바른 클래스에 속할 때까지 계속 반복, 최종 목표는 모든 데이터 포인트가 올바른 코스로 분류되도록 하는 것이다.[3]

그라디언트 부스팅(Gradient boosting)은 순차적 및 기호 학습 모델을 기반으로 한다. 기본 학습자는 현재 학습자가 항상 이전 학습자보다 더 효과적인 방식으로 순차적으로 생성되며, 전체 모델은 각 반복마다 순차적으로 향상된다.[4]

XGBoost는 결측값 을 채우는 경향이 있으며 극단적인 기울기 향상을 의미한다. 그라디언트 부스팅의 고급 버전이다. 주요 목표는 속도를 높이고 경쟁의 효율성을 높이는 것이다.[5]

[그림 1]의 모델 복잡도는 편향 분산 트레이드오프(Bias-Variance Tradeoff)[6]를 하는 함수이다. 회귀 및 분석트리(Classification And Regression Trees, CART)[7] 와 분산 처리로 많은 트리처리 중에 필요한 부분만 만든다.

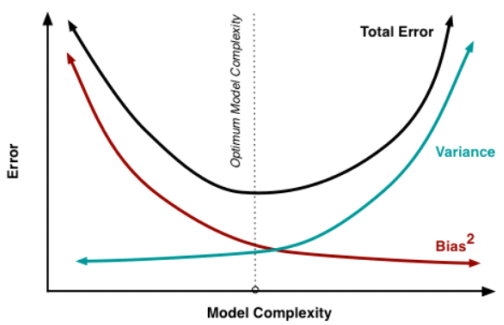

그림 1. 편향분산 트레이드오프
Fig. 1. Bias-Variance Tradeoff

www.dbpia.co.kr

Training set으로 모델을 만들면 반복을 하면 할수록 error rate는 지속적으로 줄어든다. 위 그래프에서 데이터 셋 Validation set에 적용을 하면 error rate가 줄어들다가 어느 순간부터는 반대로 증가하게 된다. 이 곡점이 과적합이 시작되는 곳이다. 변곡점을 넘어서도 지속적으로 훈련을 하게 되면 데이터 내부의 구조, 패턴, 관계를 학습해서 일반화하는 게 아니고, Validation set이 오답이 나오게 되어 Validation set의 error rate는 반대로 올라가게 되는 것이다.[8]

Training set으로 예측. 분류 모델에 검증 셋 (Validation set) 데이터를 적용해서 error를 측정한 시각화이다.[그림 2]

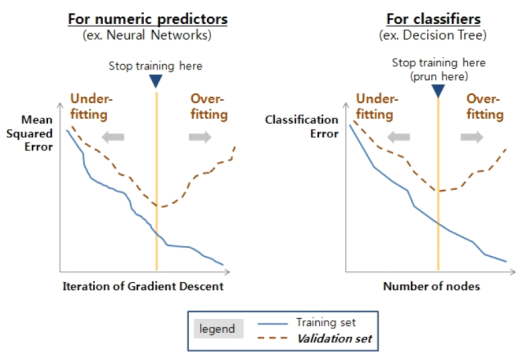

그림 2. 예측, 분류 모델에 검증 셋 데이터 에러 측정
Fig. 2. Measurement of validation set data errors in the model

2.2 교차 검증

Training set을 k 등분하고 k-1 개의 fold (=($\frac{1}{k}$)/k)는 Training set으로 사용, 남아 있는 1개의 fold ($\frac{1}{k}$)는 validation set으로 사용한다. Validation set에 해당되는 fold를 round로 바꿔주게 된다. [그림 3]의 5-fold Cross Validation 그래프를 참고한다.

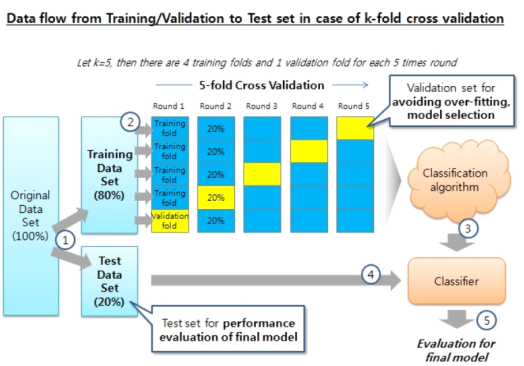

그림 3. 5겹 교차 검증
Fig. 3. 5-fold Cross Validation

4개 fold에서 Training set, Validation set의 1개 fold를 이용해서 모형 훈련을 시킨 것을 5 round 시행하고, 분류 모형을 선택한 후 Test set을 가지고 최종 모형을 평가한다.[9]

성능 측정은 모형을 훈련시키기 위해 최적화되는 손실 함수 (예 : 로그 손실) 또는 일반적으로 문제에 관심 있는 Classification Metric (예 : 분류 정확도)으로 시각화한다.

2.3 XGBoost의 장점

첫째, 정규화이다. 표준 GBM(Gradient Boosting Machine) 구현에는 XGBoost와 같은 정규화 가 없으므로 과적합을 줄이는 데 도 도움이 된다. 실제로 XGBoost는 '정규화 된 부스팅 '기법으로도 알려져 있다.

둘째, 병렬 처리이다. XGBoost는 병렬 처리를 구현하며 GBM에 비해 빠르다. 부스팅이 순차적인 프로세스인데 병렬화할 수 있는 방법은 각 트리가 이전 트리 이후에만 빌드 될 수 있다는 것이다. 모든 코어를 사용하여 트리를 만들며, XGBoost는 Hadoop에서의 구현도 지원한다.

셋째, 유연성이다. XGBoost를 사용하면 사용자 정의 최적화 목표 및 평가 기준을 정의할 수 있다.

넷째, 결측값 처리이다. XGBoost에는 누락된 값을 처리하는 내장 루틴이 있다. 다른 관측치와 값을 제공하고 이를 매개 변수로 전달해야 한다. XGBoost는 각 노드에서 누락된 값을 발견하고 향후 누락된 값에 대해 취할 경로를 학습하므로 여러 가지 시도를 한다.

다섯째, 트리 가지치기이다. GBM은 분할에서 부정적인 손실이 발생하면 노드 분할을 중지한다. 반면 XGBoost는 지정된 max_depth까지 분할한 다음 트리를 뒤로 잘라 내고 양의 이득이 없는 분할을 제거한다. 또 다른 장점은 마이너스 손실의 분할이 -2 뒤에 플러스 손실 +10의 분할이 이어질 수 있다는 것이다. GBM은 -2를 만나면 중지된다. 그러나 XGBoost는 더 깊어지고 분할의 +8의 결합 효과를 보고 둘 다 유지한다.

여섯째, 교차 유효성 검사(Cross Validation)이다.[10] XGBoost를 사용하면 부스팅 프로세스의 각 반복에서 교차 검증을 실행할 수 있으므로 단일 실행에서 정확한 최적의 부스팅 반복 횟수를 쉽게 얻을 수 있다. 이것은 그리드 검색을 실행해야 하고 제한된 값

www.dbpia.co.kr

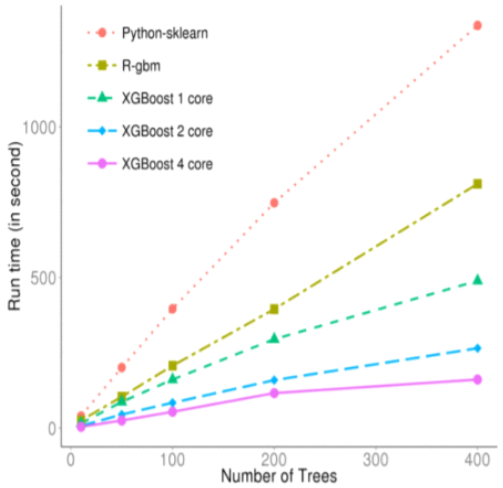

그림 4. 처리 속도 비교 (Tree 수량)
Fig. 4. Speed Comparison to Other Packages

만 테스트할 수 있는 GBM과는 다르다. 기존 모델에서 계속 사용자는 이전 실행의 마지막 반복에서 XGBoost 모델 학습을 시작할 수 있다. 이것은 특정 응용 프로그램에서 중요한 이점이다.

[그림 4]의 수치는 Higgs Boson Competition의 데이터로 생성되었다. 단일 시스템에서 두 개의 주요 트리 부스팅 패키지와 비교하여 시스템을 벤치마킹한다. 단일 코어를 사용해도 알 수 있듯이, XGBoost는 R의 gbm보다 2배, Scickit-learn보다 4배 더 빨리 실행된다.[11]

Ⅲ. 실 험

3.1 실험 방법

실험은 마케팅 대행사 광고 데이터 A, B, C(가명) 3개 쇼핑몰 회사의 홈페이지를 대상으로 광고 데이터를 수집하였다.[표 1]

데이터의 구성은 11가지의 변수로 구성되어 있으며 데이터 전처리(Data Preprocessing)[12]로 범주 열에 패턴이 없고 고유한 요소가 많은 변수는 제외한다. 시간 매개변수(TimeStamp)는 월, 일, 주간, 시간으로 분리하여 구성하며, 광고 클릭(Click on Ad)으로는 사용자가 광고를 클릭하는 두 가지 결과(0: Don't Click, 1: Clicked)를 가지고 기계학습 알고리즘을 시작한다.[표 2]

제품을 구매할 가능성이 있는 특정 사용자가 광고 데이터를 사용하여 광고를 클릭하는 예측 확률의 알고리즘은 Logistic Regression[13]과 Decision Tree[14]와 XGBoost로 성능 향상을 목표로 수차례 비교 진행하였다.[표 3]

예측 성능을 높이기 위해 XGBoost 알고리즘으로 과적합을 줄이는 방법과 정확성이 어느 정도 향상이 되었는지와 높은 유연성, 결측값 처리시키는 과정과 결과에 대해서 기존 알고리즘과 분류 보고서로 실험

표 1. 실험 대상
Table 1. Subect of Experiment

Company	Data(Bytes)	Sales	Collection method
A	107,885	Shopping Mall	Advertising corp
B	828,676	Shopping Mall	Advertising corp
C	930,477	Shopping Mall	Advertising corp

표 2. 광고 데이터 전처리 및 타임스탬프
Table 2. Pre-processing and timestamp advertising data

Existing Category	Data Pre processing	Timestamp New Category	New Category
Daiy on Site	O		Daiy on Site
Age	O		Age
Annual Income	O		Annual Income
Daily Internet Usage	O	T_Month	Daily Internet Usage
Ad Topic Line	X	T_Day_month	Male
City	X	T_Day_week	T_Month
Male	O	T_Hour	T_Day_month
Country	X		T_Day_week
Timestamp	O		T_Hour
Click on Ad	O		Click on Ad
Spending Score	O		Spending Score

표 3. 예측 모델, 알고리즘
Table 3. Predictive Model, Algorithm

Type	Algorithm	informal content
Probability Model	Logistic Regression	A statistical technique used to predict the likelihood of an event using linear combinations
Predictive model	Decision Tree	It is mainly used in decision analysis to find strategies that can produce results closest to the goal
Boosting	XGBoost	Weighing the learning errors of weak predictive models and making strong predictive models by reflecting them sequentially on the next learning model

www.dbpia.co.kr

결과를 예측하고자 한다.

3.2 실험 결과

광고 데이터의 추세 시각화를 scatter_matrix 함수를 이용하여 분석을 수행하여 사용자의 Daily on Site(일일 사이트 사용), Age(나이), Annual Income (수입), Daily Internet usage(인터넷 사용), Spending Score(지출 점수)를 바탕으로 서로 간의 분석을 수행한다. 결과는 '지출 점수'의 scatter가 분산되어 있어 데이터가 일률적이지 않아 모델의 정확도는 낮을 것이라 판단한다.[그림 5]

XGBoost 모델을 학습하는 동안 독립형 테스트 세트 (eval_set)에서 이진 분류 오류율 (" error ")을 보고 할 수 있다. 데이터의 67 %에서 모델을 학습하고

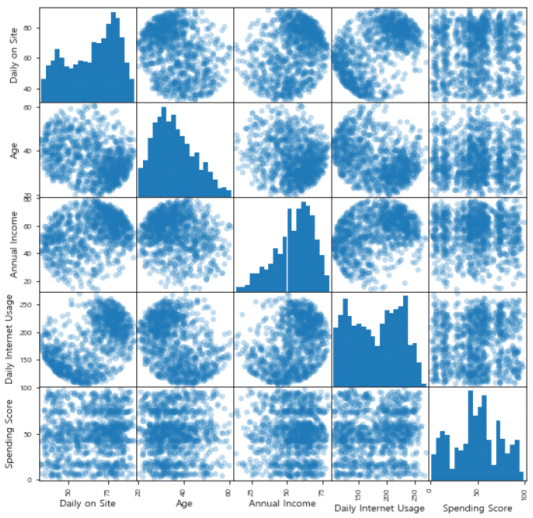

그림 5. 사용자 광고 클릭 별 인터넷 사용 구분
Fig. 5. Separate Internet usage by user advertising click

$$y'_i = \sum_{k=1}^{K} f_k(x_i)\ , f_k\ \in F$$

$$\text{Obj} = \sum_{i=1}^{n} l(y_i, y'_i) + \sum_{k=1}^{K} \Omega(f_k)$$

$$* \ where\ y'_i = \text{predict score corresponding to } x_i,$$

$$f_k = k\ th\ decision\ tree\ in\ function\ space\ F,$$

$$l = loss\ function,$$

$$\Omega = regularization\ term (= complexity\ of\ Trees)$$

그림 6. XGBoost의 객체 함수와 y hat
Fig. 6. XGBoost's Object function and y hat

33 % 테스트 데이터 세트에서 모든 교육 시대마다 모델을 평가한다.

수식의 f 함수는 알고리즘에서 생성된 Decision Tree 모델들을 의미하며, regularization term의 의미는 Tree들의 파라미터 정의이다. 모델을 다수로 학습하여 더해진 예측값들을 사용해 결정하여 과적합과 기존 모델의 취약한 부분에 대해 보완을 한다.[그림 6]

분류 오류는 각 반복마다 보고되고 마지막으로 분류 정확도가 보고 된다. [표 4]는 분류 오류가 각 학습 반복 (각 부스트 트리가 모델에 추가된 후)마다 보고되는 것을 볼 수 있다.

평가 데이터 세트에서 모델의 성능을 검색하고 이를 학습하는 동안 학습이 어떻게 전개되었는지, 각 평

표 4. 모델 이진 분류 오류율
Table 4. Model eval set error

Model eval set error (validation_0-error & validation_0-logloss)		
[0]	validation_0-error:0.219403	validation_0-logloss:0.64638
	validation_1-error:0.209091	validation_1-logloss:0.647487
[1]	validation_0-error:0.219403	validation_0-logloss:0.607855
	validation_1-error:0.209091	validation_1-logloss:0.609865
[2]	validation_0-error:0.219403	validation_0-logloss:0.575572
	validation_1-error:0.209091	validation_1-logloss:0.578715
[3]	validation_0-error:0.219403	validation_0-logloss:0.548196
	validation_1-error:0.209091	validation_1-logloss:0.55234
[4]	validation_0-error:0.219403	validation_0-logloss:0.524678
	validation_1-error:0.209091	validation_1-logloss:0.530437
...		. . .
[85]	validation_0-error:0.076119	validation_0-logloss:0.242397
	validation_1-error:0.236364	validation_1-logloss:0.378109
[86]	validation_0-error:0.073134	validation_0-logloss:0.241446
	validation_1-error:0.233333	validation_1-logloss:0.378796
[87]	validation_0-error:0.073134	validation_0-logloss:0.239848
	validation_1-error:0.236364	validation_1-logloss:0.379651
[88]	validation_0-error:0.074627	validation_0-logloss:0.238069
	validation_1-error:0.236364	validation_1-logloss:0.379949
[89]	validation_0-error:0.074627	validation_0-logloss:0.23739
	validation_1-error:0.236364	validation_1-logloss:0.379943
[90]	validation_0-error:0.074627	validation_0-logloss:0.236925
	validation_1-error:0.236364	validation_1-logloss:0.380394
[91]	validation_0-error:0.071642	validation_0-logloss:0.235484
	validation_1-error:0.233333	validation_1-logloss:0.380163
[92]	validation_0-error:0.071642	validation_0-logloss:0.234535
	validation_1-error:0.230303	validation_1-logloss:0.381327
[93]	validation_0-error:0.067164	validation_0-logloss:0.232416
	validation_1-error:0.230303	validation_1-logloss:0.38107
[94]	validation_0-error:0.068657	validation_0-logloss:0.23163
	validation_1-error:0.230303	validation_1-logloss:0.380631
[95]	validation_0-error:0.065672	validation_0-logloss:0.230923
	validation_1-error:0.230303	validation_1-logloss:0.380772
[96]	validation_0-error:0.068657	validation_0-logloss:0.23027
	validation_1-error:0.236364	validation_1-logloss:0.381381
[97]	validation_0-error:0.067164	validation_0-logloss:0.229311
	validation_1-error:0.239394	validation_1-logloss:0.381333
[98]	validation_0-error:0.065672	validation_0-logloss:0.228403
	validation_1-error:0.239394	validation_1-logloss:0.382273
[99]	validation_0-error:0.067164	validation_0-logloss:0.22802
	validation_1-error:0.239394	validation_1-logloss:0.38225

www.dbpia.co.kr

표 5. 반복 학습 중 최적의 중지(과접합)
Table 5. Optimal Stop During Repeated Learning

validation_0-logloss	validation_0-logloss
[0] validation_0-logloss:0.641987	[32] validation_0-logloss:0.362452
[1] validation_0-logloss:0.602536	[33] validation_0-logloss:0.360779
[2] validation_0-logloss:0.567945	[34] validation_0-logloss:0.356347
[3] validation_0-logloss:0.540007	[35] validation_0-logloss:0.355119
[4] validation_0-logloss:0.515481	[36] validation_0-logloss:0.351969
[5] validation_0-logloss:0.493885	[37] validation_0-logloss:0.351555
[6] validation_0-logloss:0.476864	[38] validation_0-logloss:0.348933
[7] validation_0-logloss:0.462742	[39] validation_0-logloss:0.347768
[8] validation_0-logloss:0.450885	[40] validation_0-logloss:0.348257
[9] validation_0-logloss:0.440005	[41] validation_0-logloss:0.347229
[10] validation_0-logloss:0.430869	[42] validation_0-logloss:0.346798
[11] validation_0-logloss:0.422323	[43] validation_0-logloss:0.344639
... . . .	[44] validation_0-logloss:0.344662
[21] validation_0-logloss:0.37872	[45] validation_0-logloss:0.345211
[22] validation_0-logloss:0.371888	[46] validation_0-logloss:0.344778
[23] validation_0-logloss:0.371272	[47] validation_0-logloss:0.345346
[24] validation_0-logloss:0.367766	[48] validation_0-logloss:0.344948
[25] validation_0-logloss:0.367471	[49] validation_0-logloss:0.344783
[26] validation_0-logloss:0.366254	[50] validation_0-logloss:0.344864
[27] validation_0-logloss:0.365353	[51] validation_0-logloss:0.345966
[28] validation_0-logloss:0.364517	[52] validation_0-logloss:0.346494
[29] validation_0-logloss:0.365057	[53] validation_0-logloss:0.346299
[30] validation_0-logloss:0.363582	Stopping. Best iteration:
[31] validation_0-logloss:0.363213	[43] validation_0-logloss:0.344639

가 세트에서 model.evals_result () 함수를 호출하여 학습 후 모델에서 저장하고 사용한다. 평가 데이터 세트 및 점수 사전을 반환한다.[표 5]

차지하는 치수 공간을 줄이고 과적합을 처리하기 위해 형상 선택 수행. 그리드 검색 교차 유효성 검사는 동일한 작업을 수행하는 데 사용되고 L1 정규화는 기능 수와 F-1 정확도 점수 간의 균형을 유지하는 데 사용하고 F1 점수 정밀도와 Recall 값 사이의 조화 평균을 나타내기 때문에 성능 지표로 사용된다.[그림 7]

L1 정규화를 사용하여 회귀 모델 정의를 마치면 c = 0.1을 사용하여 얻은 매개 변수가 차원을 줄이고 실행 시간을 최적화한다

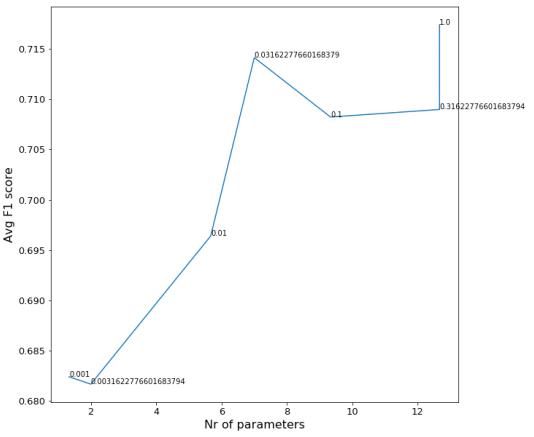

그림 7. F1-score 정밀도와 Recall 값 사이의 성능
Fig. 7. Performance between F1-score precision and recall value

[그림 8]은 훈련 및 테스트 데이터 세트에서 XGBoost 모델의 로그 손실이다.

훈련 및 테스트 데이터 세트에서 XGBoost 모델의 분류 오류이다. 이 모델이 epoch 43에서 학습을 중단했음을 [표 5]로서 알 수 있다. epoch를 평가하기 위해 개선이 없는 epoch 수를 10을 추가하였지만, 로그 손실이 개선되지 않았음을 확인할 수 있다. [그림 9]의 분류 오류 곡선에서 학습을 조기에 중단할 수 있는 시점을 시각화한 것이다.

[표 6]은 쇼핑몰 광고 데이터에 대한 3가지 알고리즘의 비교 분석한 결과이다. Logistic Regression 모델과 Decision Tree 모델로 Scikit Learn 함수를 사용하여 모델을 초기화, 훈련을 시키고 최종 예측 평가를 한다. Logistic Regression 모델의 정확도는 0.769(76.9%)이며, Confusion matrix에서 정확한 예측의 수는 18 + 236 = 254, 잘못된 예측의 수는 49 + 27 = 76이다. Decision Tree 모델은 Confusion matrix에서 정확한 예측한 수는 28 + 225 = 253, 잘

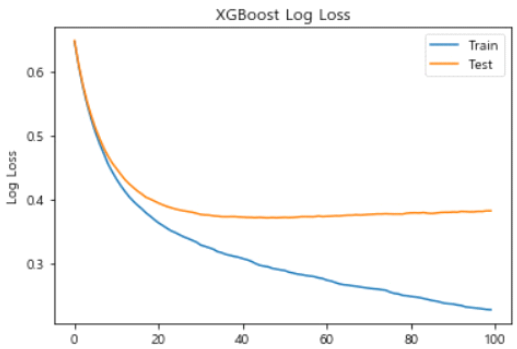

그림 8. XGBoost 학습 곡선 로그 손실
Fig. 8. XGBoost Learning curve log loss

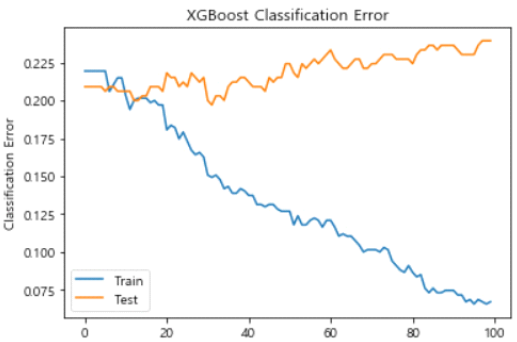

그림 9. XGBoost 학습 곡선 분류 오류
Fig. 9. XGBoost Learning curve classification error

www.dbpia.co.kr

표 6. 쇼핑몰 광고 클릭 알고리즘 비교
Table 6. Compare Shopping Mall Ad Click Algorithms

Algorithm	Corp	Data(Byte)	Accuracy	F1-Score	avg
XGBoost	A	107,885	0.7848	0.88	0.8221
	B	828,676	0.7564	0.75	
	C	930,477	0.9252	0.92	
Logistic Regression	A	107,885	0.7696	0.86	0.7989
	B	828,676	0.7617	0.75	
	C	930,477	0.8656	0.86	
Decision Tree	A	107,885	0.7666	0.85	0.7872
	B	828,676	0.6446	0.64	
	C	930,477	0.9505	0.95	

못된 예측의 수는 39 + 38 = 77개이며, 모델의 정확도는 0.766(76.6%)이다. XGBoost 모델은 광고 데이터의 총 클릭 대비 비 클릭 정확도를 0.784 (78.4%) 예측을 성능 평가한 결과이다.

이 연구의 XGBoost의 알고리즘은 조기 중지 위주 기반으로 정확도가 대부분 높게 나왔으며, Logistic Regression과 Decision Tree는 예측 분석을 위한 도구이나 XGBoost에 비해 분류 정확도가 낮음을 알 수 있다. 데이터 클래스가 많거나 데이터 불균형이 기형으로 높을수록 XGBoost가 정확도와 안정성 면에서 적합하다는 것을 알 수 있다. Logistic Regression 모델과 Decision Tree 모델의 결과는 분석을 수행하여 변수 간의 새로운 종속성을 찾아 시각화하였고 분류 보고서로 검증하였다. XGBoost 모델에선 동일한 표본으로 교육 성과 모니터링과 학습곡선으로 모델을 평가한다. XGBoost 모델은 훈련 중 모델에 대한 테스트 세트의 성능을 평가하고 보고할 수 있다. 모델을 학습하고 상세 출력을 지정할 때 model.fit () 호출 시 테스트 데이터 세트와 평가 메트릭을 모두 지정하여 이 기능을 지원한다. 쇼핑몰 데이터에 대한 광고 클릭 예측을 3가지 알고리즘으로 분석한 결과 본 연구의 알고리즘이 정확도와 f1-score 등에서 2~6% 정도의 성능 개선을 했다.

이 실험 결과는 XGBoost가 모델을 고정된 횟수의 교육 반복 후에 테스트 데이터 셋의 성능이 향상되지 않은 경우 교육 절차를 중지하는 알고리즘이 Logistic Regression과 Decision Tree 알고리즘보다 평균적으로 향상됨을 확인했다. [표 7]은 XGBoost에서 제공하는 기능이다. XGBoost의 처리는 단일 시스템에서 다중 스레드가 가능하고, 훈련 알고리즘 전에 데이터를 전처리하며, 멀티 스레딩은 스레드 수에 거의 선형 적

표 7. 트리 부스팅 시스템 비교
Table 7. Comparison of major tree boosting system

SYSTEM	exact greedy	approximate global	approximate local	out-of-core	sparisity aware	parallel
XGBoost	yes	yes	yes	yes	yes	yes
pGBRT	no	no	yes	no	no	yes
Spark MLLib	no	yes	no	no	partially	yes
H2O	no	yes	no	no	partially	yes
scikit-learn	yes	no	no	no	no	no
R GBM	yes	no	no	no	partially	no

이므로 효율성이 더욱 향상된다.

Ⅳ. 결 론

본 연구에서는 Logistic Regression 모델과 Decision Tree 모델, XGBoost 모델을 동일한 표본으로 수행하였다. 각 모델들은 분석을 수행하여 변수 간의 새로운 종속성을 찾아 시각화하였으며 분류 문제는 어느 정도 해결하는 것을 관찰했다. XGBoost는 데이터의 예측을 생성하고 테스트하고 튜닝하여 과적합을 피하며 조기 중지 기법으로서 최적의 성능을 발휘했으며, 데이터 품질의 높은 정확도와 처리 시간 단축에 기여할 수 있다.

향후 연구로는 catboost의 ordered boosting 기법을 활용하여 prediction shift 문제를 해결해 보고자 한다. 일반적인 GBM(Gradient Boosting Machine)은 다음 스텝의 새로운 Tree를 만들 때 현재 모델에서 사용된 데이터를 Gradient estimate를 사용하는데 다시 사용되기 때문에 과적합에 취약한 문제가 있었다.

이러한 문제를 기존의 tree 구조를 선택 후(internal node의 cut-off value를 정하는 방식) leaf value는 역순으로 먼저 구하고 tree 구조를 선택하는 과정을 ordered principle 개념을 적용해 해결하고자 한다.

References

[1] "*Boosting algorithm: XGBoost*," Towards Data Science, 2017-05-14. Retrieved 2020-01-04.

[2] D. M. Hawkins, "The problem of overfitting," *J. Chem. Info. and Modeling*, vol. 44, no. 1, pp. 1-12, 2004, doi:10.1021/ci0342472. PMID 14741005.

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining,*

www.dbpia.co.kr

Inference, and Prediction, 2nd Ed., New York, Springer, 2009, ISBN 978-0-387-84858-7.

[4] C. Li, "*A gentle introduction to gradient boosting*," https://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf.

[5] D. Nielsen, "Tree boosting with XGBoost – why does XGBoost win "Every" machine learning competition?," M.S. Thesis, Dept. Sci. in Phys. and Math., Norwegian Univ. of Sci. and Technol., Dec. 2016.

[6] C. Sammut, et al., "Bias – Variance decomposition," *Encyclopedia Mach. Learn.,* Springer, pp. 100-101, 2011.

[7] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*, Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1984, ISBN 978-0-412-04841-8.

[8] G. James, *An Introduction to Statistical Learning: with Applications in R,* Springer. p. 175, 2013, ISBN 978-1461471370.

[9] S. Geisser, *Predictive Inference,* New York, NY: Chapman and Hall, 1993, ISBN 978-0-412-03471-8.

[10] M. Stone, "An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion," *J. Royal Statistical Soc.: Series B (Methodological),* vol. 39, no. 1, pp. 44-47, 1977, JSTOR 2984877.

[11] T. Chen and C. Guestrin, "Xgboost: Reliable large-scale tree boosting system," in *Proc. 22nd SIGKDD Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2015.

[12] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Mining*, vol. 10, no. 35, Dec. 2017, doi:10.1186/s13040-017-0155-3

[13] David W. Hosmer and S. Lemeshow, *Applied Logistic Regression*, 2nd Ed., John Wiley & Sons, Inc., 2000, ISBN 0-471-35632-8.

[14] L. Rokach and O. Maimon, "Top-down induction of decision trees classifiers-a survey," *IEEE Trans. Syst., Man, and Cybernetics, Part C*, vol. 35, no. 4, pp. 476-487, 2005.

한 영 진 (Young-Jin Han)

2020년 : 한양대학교 컴퓨터공학 석사

2020년~현재 : 한양대학교 컴퓨터소프트웨어학과 박사과정

1989년: GENERAL BBS(전자게시판) 운영

1991년~1993년 : 한국교육진흥원 전산개발실

1994년~1999년 : SOFTCOM CEO

1999년~현재 : 에스엔아이(주) CEO

<관심분야> 머신러닝 및 딥러닝, IoT, 정보보안, 이미지 프로세싱, 네트워크 기반 제어

[ORCID:0000-0003-1014-166X]

조 인 휘 (In-Whee Joe)

1983년 : 한양대학교 전자공학과 학사

1995년 : University of Arizona 컴퓨터공학 석사

1998년 : Georgia Tech 컴퓨터공학 박사

1985년~1992년 : ㈜데이콤 종합연구소 선임연구원

1998년~2000년 : Oak Ridge 국립연구소 연구원

2000년~2002년 : Bellcore Lab(Telcordia) Scientist

2002년~현재 : 한양대학교 컴퓨터소프트웨어학부 교수

<관심분야> 이동통신, IoT, 딥러닝, XAI, 임베디드 시스템, EV 배터리 및 시뮬레이션

[ORCID:0000-0002-8435-0395]

www.dbpia.co.kr