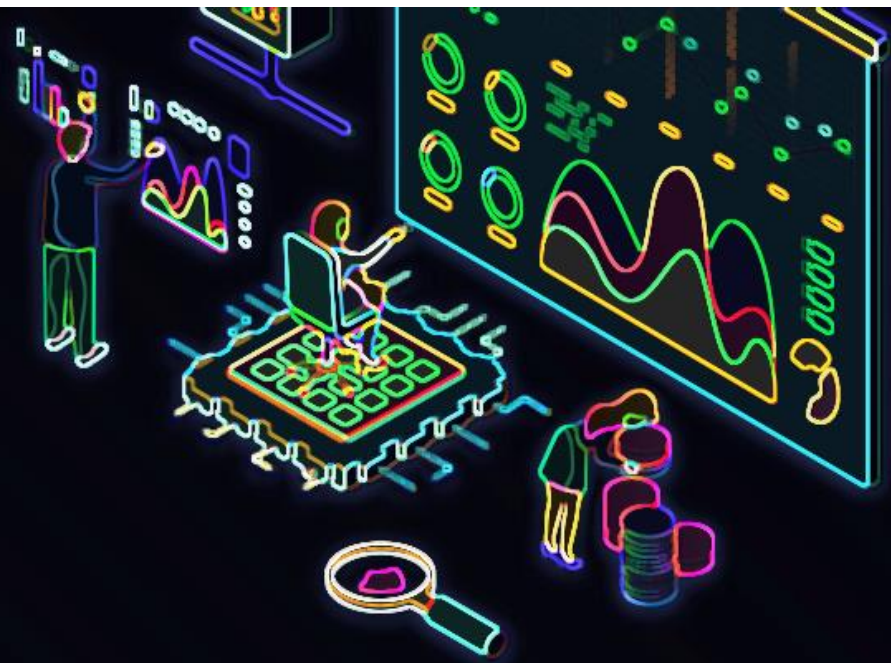
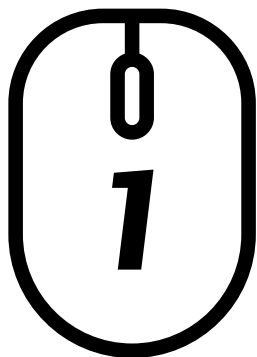


2019 IGAWorks BIG DATA COMPETITION

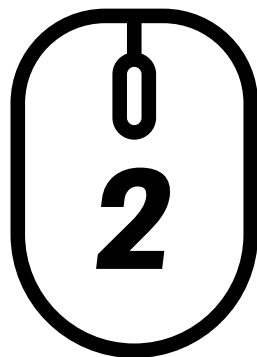


- 광고 클릭률(CTR) 예측 -

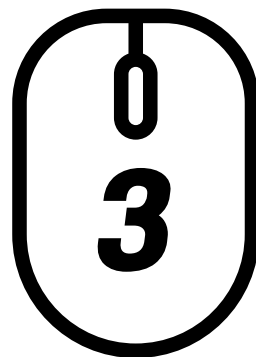
목차



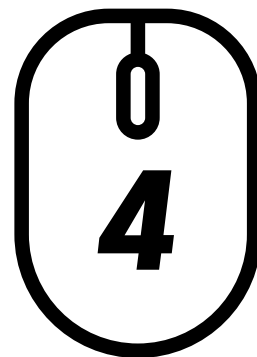
EDA



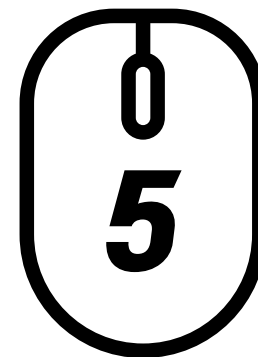
전처리



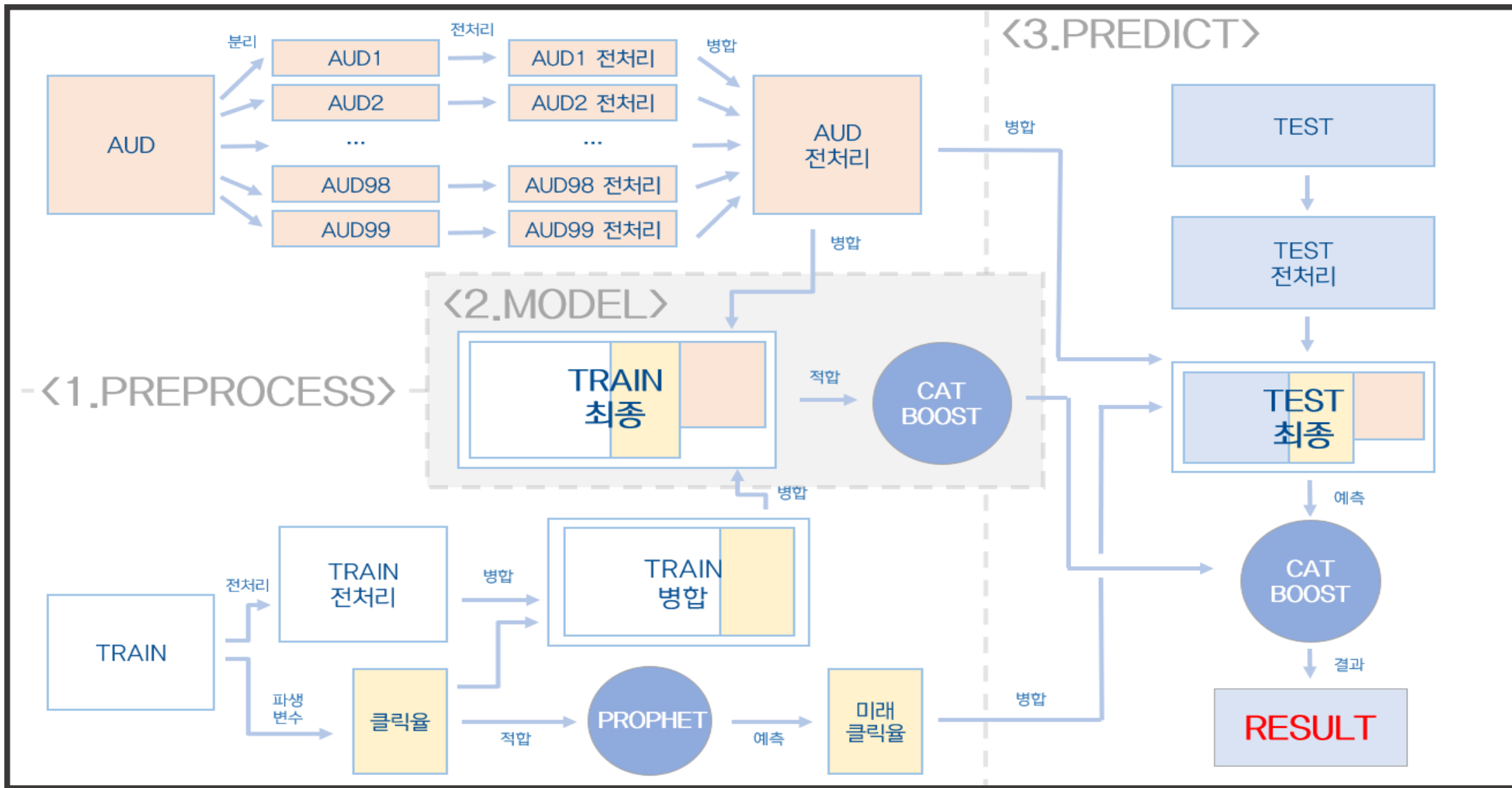
모델 비교

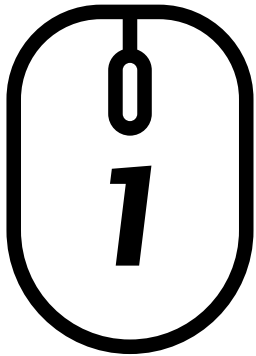


모델 선정





최종 결과





EDA



- TRAIN & TEST
 - 범주형 변수 분석
 - 시간별 데이터 시각화

- AUDIENCE_PROFILE
 - Key 값 확인
 - Transaction Data



1. TRAIN & TEST

TRAIN

550만 x 25 변수

CLICK

TEST

55만 x 24 변수

- TRAIN에만 있는 목표 값(Y)
→ Binary DATA : CLICK
- 공통으로 존재하는 24개의 변수
→ 범주형 DATA : 23개
→ 시간 DATA : 1개



1. TRAIN & TEST

범주형 변수 분석

대부분의 변수가 범주형 변수, **level의 개수**를 정리해보았다

	TRAIN	BOTH	TEST
ssp_id	17(0)	17	17(0)
campaign_id	186(47)	139	149(10)
adset_id	872(288)	584	636(52)
placement_type	4(0)	4	4(0)
media_id	5815(3127)	2688	2883(195)
media_bundle	6286(3944)	2342	2560(218)
media_name	6810(4156)	2654	2875(221)

괄호 안의 숫자는 각각 TRAIN, TEST에만 있는 level 개수



1. TRAIN & TEST

범주형 변수 분석

	TRAIN	BOTH	TEST
media_domain	162(60)	102	102(0)
publisher_name	1222(529)	693	720(27)
device_ifa	1,869,137 (1,614,485)	264,652	375,069 (120,417)
device_os	2(0)	2	2(0)
device_os_version	125(35)	90	90(0)
device_model	1667(786)	878	915(37)

괄호 안의 숫자는 각각 TRAIN, TEST에만 있는 level 개수



1. TRAIN & TEST

범주형 변수 분석

	TRAIN	BOTH	TEST
device_carrier	536(337)	199	209(10)
device_make	299(165)	134	142(8)
device_connection_type	8(0)	8	8(0)
device_country	1(0)	1	1(0)
device_region	148(63)	85	86(1)
device_city	1326(710)	616	638(22)
device_language	33(14)	19	20(1)
advertisement_id	30(5)	25	26(1)

괄호 안의 숫자는 각각 TRAIN, TEST에만 있는 level 개수



● 범주의 level이 많음

1. TRAIN & TEST

범주형 변수 분석

- test set에서 새로운 변수 level이 출현하면, 이 문제를 해결하지 못하는 모델에서는 사용 불가

	TRAIN	BOTH	TEST
device_carrier	536(337)	199	209(10)
device_make	299(165)	134	142(8)
device_connection_type	8(0)	8	8(0)
device_country	1(0)	1	1(0)
device_region	148(63)	85	86(1)
device_city	1326(710)	616	638(22)
device_language	33(14)	19	20(1)
advertisement_id	30(5)	25	26(1)

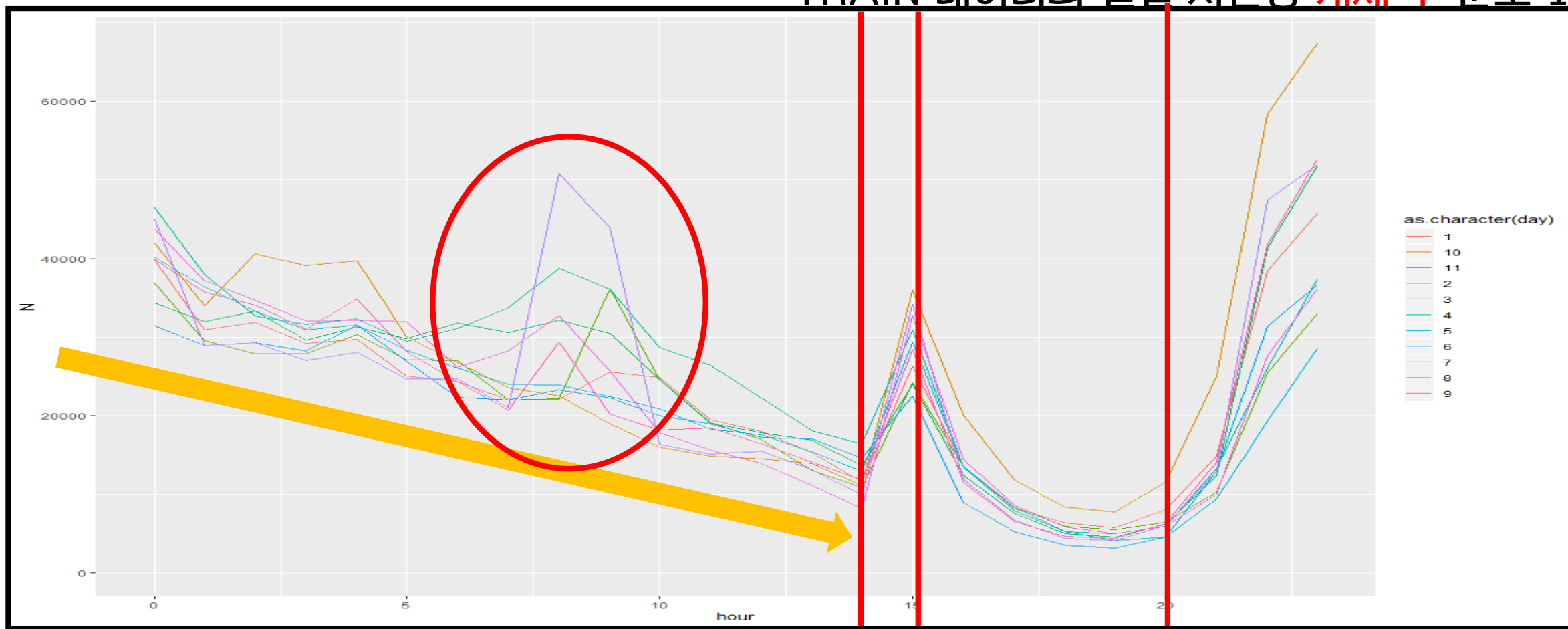
괄호 안의 숫자는 각각 TRAIN, TEST에만 있는 level 개수



1. TRAIN & TEST

시간별 데이터 시각화

TRAIN 데이터의 일별 시간당 개체 수 분포 1



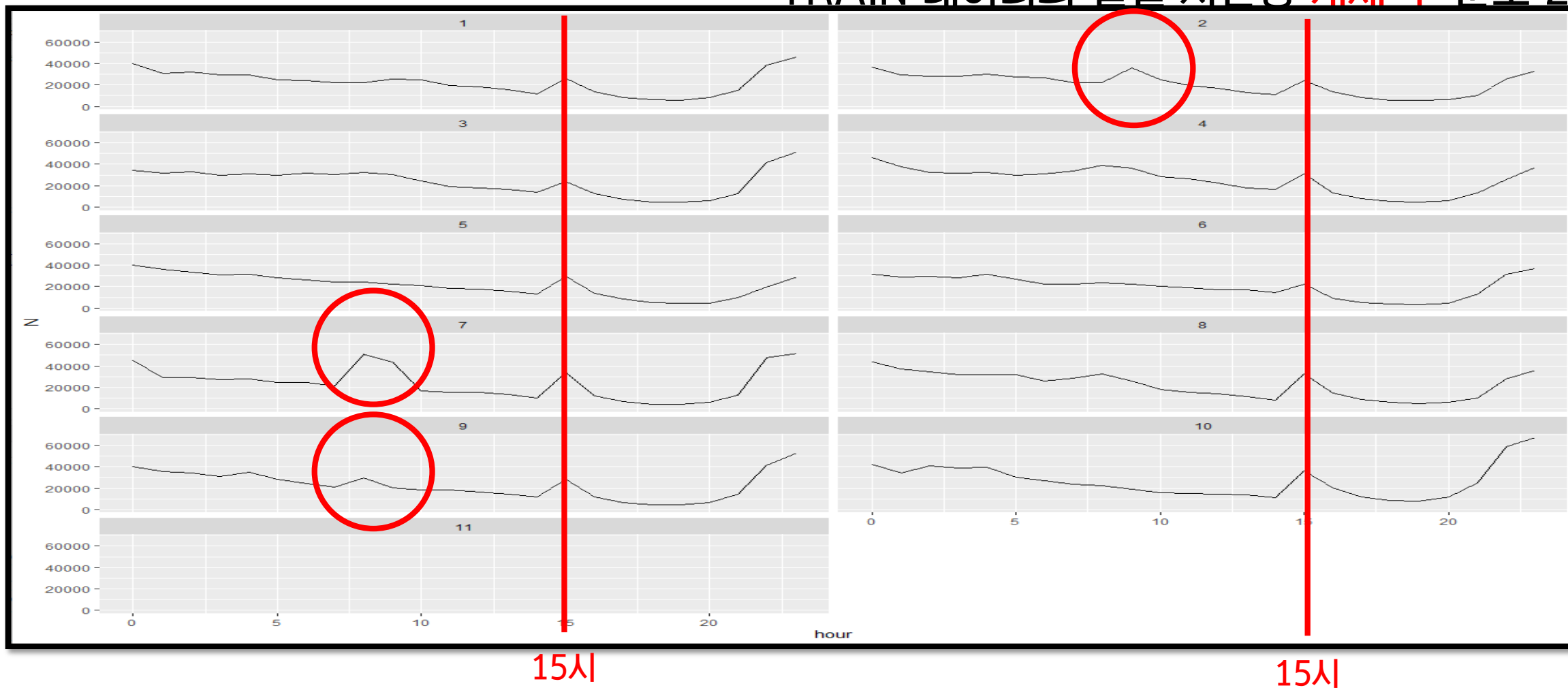
14시 15시

20시

1. TRAIN & TEST

시간별 데이터 시각화

TRAIN 데이터의 일별 시간당 개체 수 분포 2

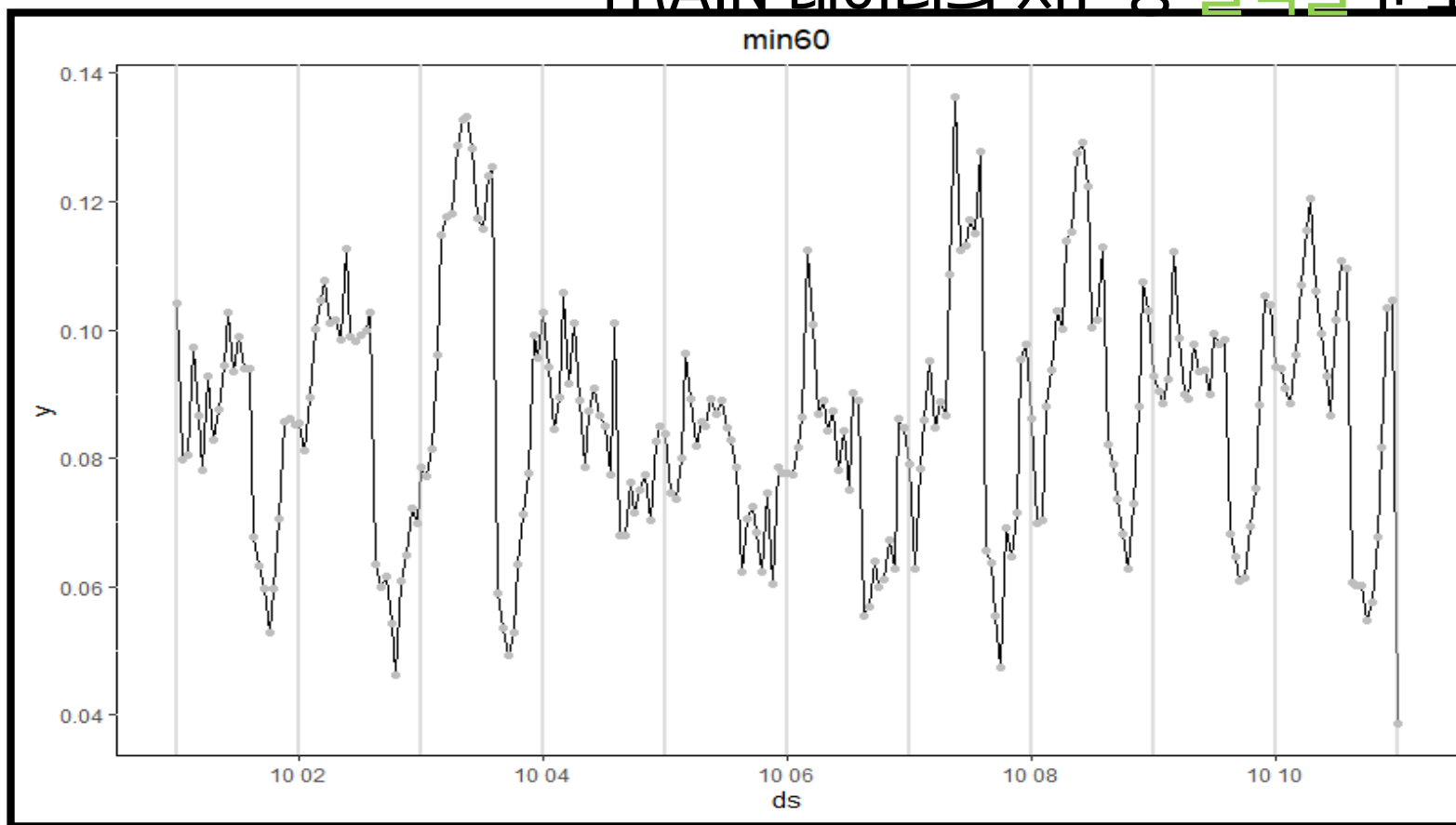




1. TRAIN & TEST

시간별 데이터 시각화

TRAIN 데이터의 시간당 클릭률 분포

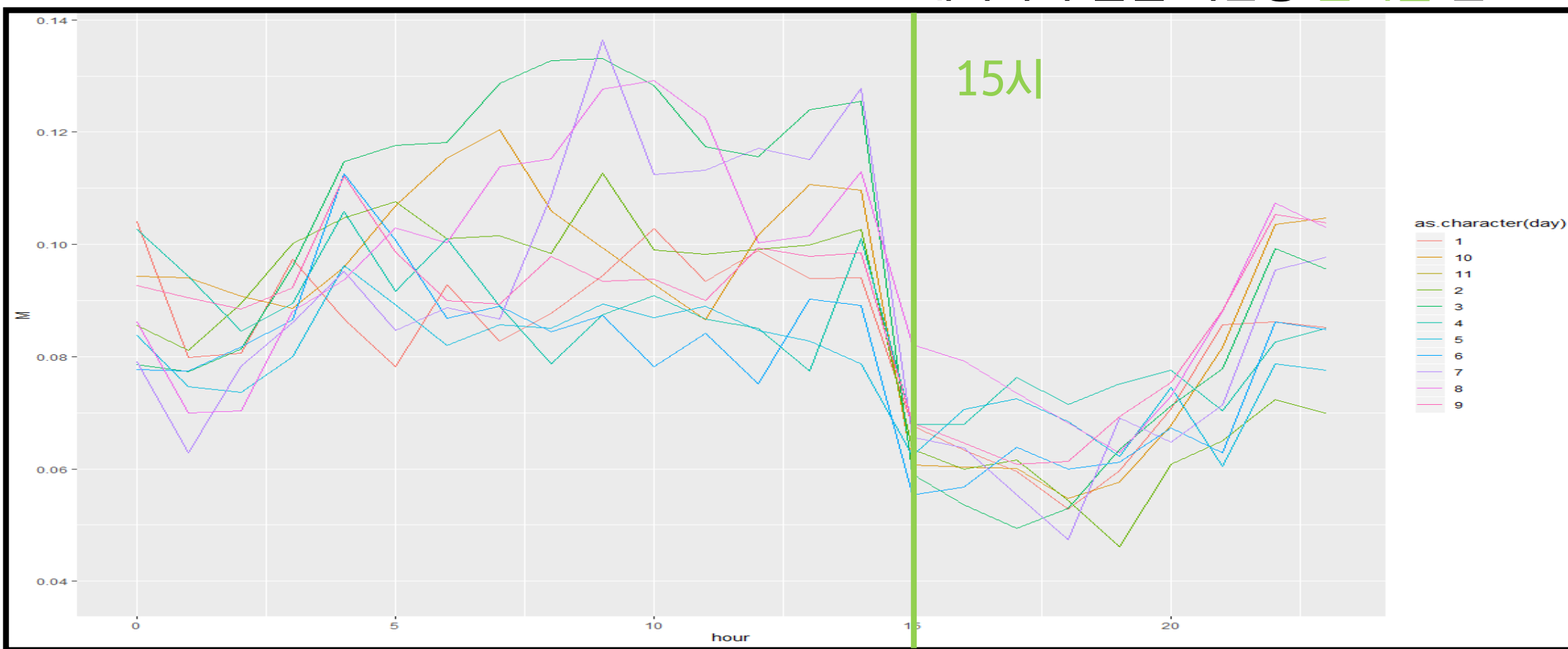




1. TRAIN & TEST

시간별 데이터 시각화

TRAIN 데이터의 일별 시간당 클릭률 분포 1

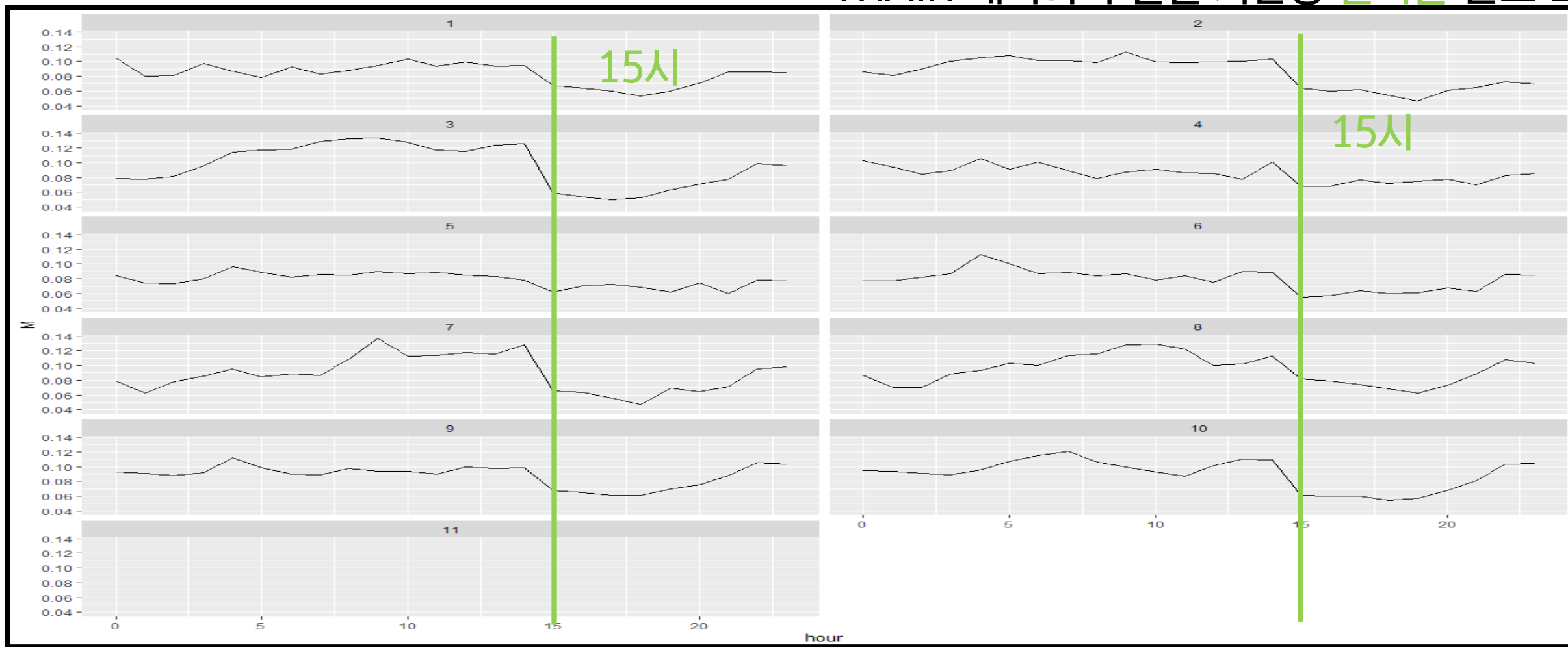




1. TRAIN & TEST

시간별 데이터 시각화

TRAIN 데이터의 일별 시간당 클릭률 분포 2





2. AUDIENCE_PROFILE

AUD

device_ifa	1000만 1 x 8 변수
------------	----------------

- 기존 TRAIN과 TEST에 병합할 수 있는
key 값 존재
→ device_ifa
- 범주형 DATA : 4개, 숫자형 DATA : 1개,
NA : 1개
- Transaction Data : 2개



2. AUDIENCE_PROFILE

Key 값 확인

[KEY] Device_ifa

666,026

TRAIN

101,209

45,545

TEST

AUD

10,000,001

4,732,765

TRAIN

406,246

TEST

- TEST나 TRAIN에만 존재하는 유저가 있다
- 기존 정보가 없는 새로운 유저도 있다



2. AUDIENCE_PROFILE

Transaction Data

AUD

-	Install_pack	Cate_code
---	--------------	-----------

p1006g, p1007g, ...

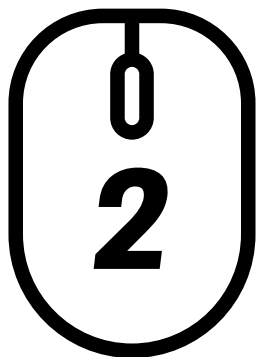
P00025:1, 000125:3, ...

41,968개의 app



242개의 cate_code



모델에 사용할 수 있도록 데이터 변환 필요!



전처리



- TRAIN & TEST 전처리
 - 인코딩(Encoding)
 - 클릭률 파생 변수 생성
 - 새로운 범주 level 처리
 - 기타 전처리
- AUDIENCE 전처리
 - Transaction 데이터
- 최종데이터
 - 샘플링(Sampling)

1. TRAIN & TEST 전처리

igaworks


EDA

전처리

모델 비교

모델 선정

최종 결과

 김홍도

TRAIN & TEST

TRAIN

550만 x 25 변수

CLICK

TEST

55만 x 24 변수

- TRAIN에만 있는 목표 값(Y)
→ Binary DATA : CLICK
- 공통으로 존재하는 24개의 변수
→ 범주형 DATA : 23개
→ 시간 DATA : 1개

igaworks


EDA

전처리

모델 비교

모델 선정

최종 결과

 김홍도

범주형 변수 분석

대부분의 변수가 범주형 변수, Factor level을 정리해보았다

	TRAIN	BOTH	TEST
ssp_id	17(0)	17	17(0)
campaign_id	186(47)	139	149(10)
adset_id	872(288)	584	636(52)
placement_type	4(0)	4	4(0)
media_id	5815(3127)	2688	2883(195)
media_bundle	6286(3944)	2342	2560(218)
media_name	6810(4156)	2654	2875(221)

범주형 변수와 그 level이 많음
→ 주로 사용되는 One-Hot 인코딩으로는 차원의 크기 때문에 데이터 핸들링 불가능한 문제

1. TRAIN & TEST 전처리

인코딩(Encoding)

Target Encoding (Mean Encoding)

- 범주형 변수의 level을 **반응변수의 평균으로 대체**
- 장점
 - 차원을 늘리지 않음
 - 빠른 학습
- 단점
 - **과적합 가능성**이 굉장히 큼
 - 교차 검증이나 다른 접근법을 사용한 정규화가 필요

[Y] 토익점수	[X] 학과	Target Encoding
780	경영	855
930	경영	855
850	통계	820
870	통계	820
810	통계	820
750	통계	820
660	경제	863.333
980	경제	863.333
950	경제	863.333

1. TRAIN & TEST 전처리

인코딩(Encoding)

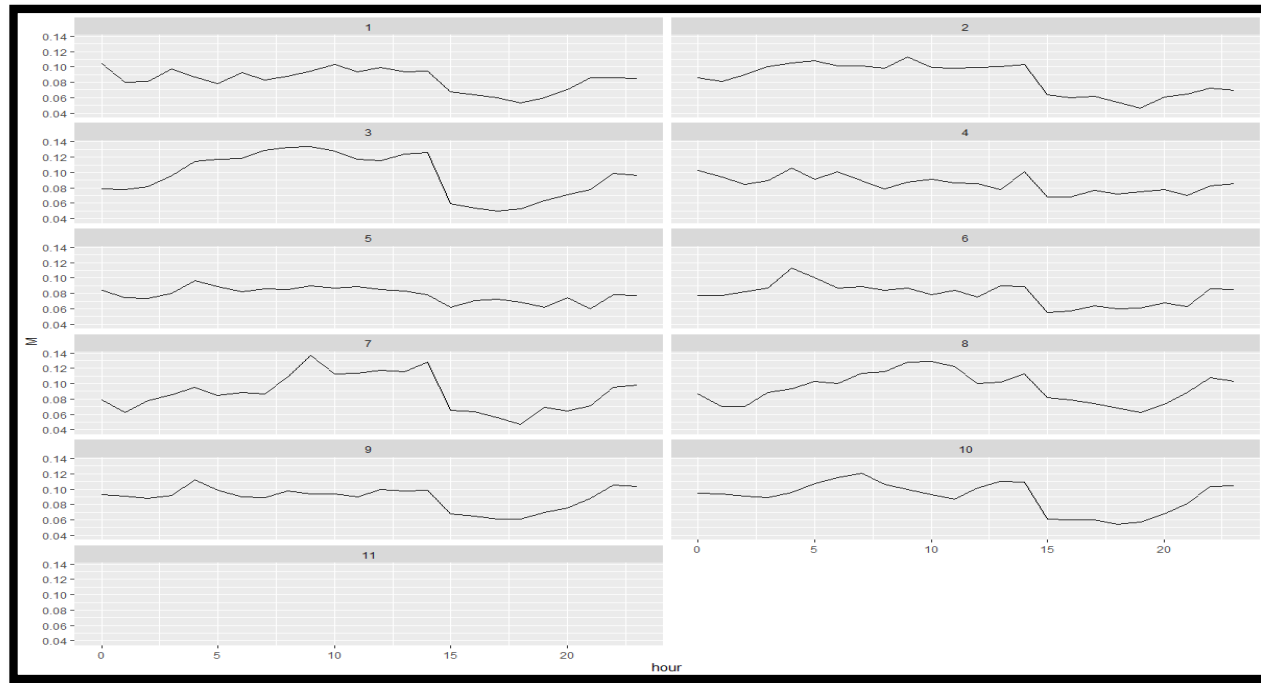
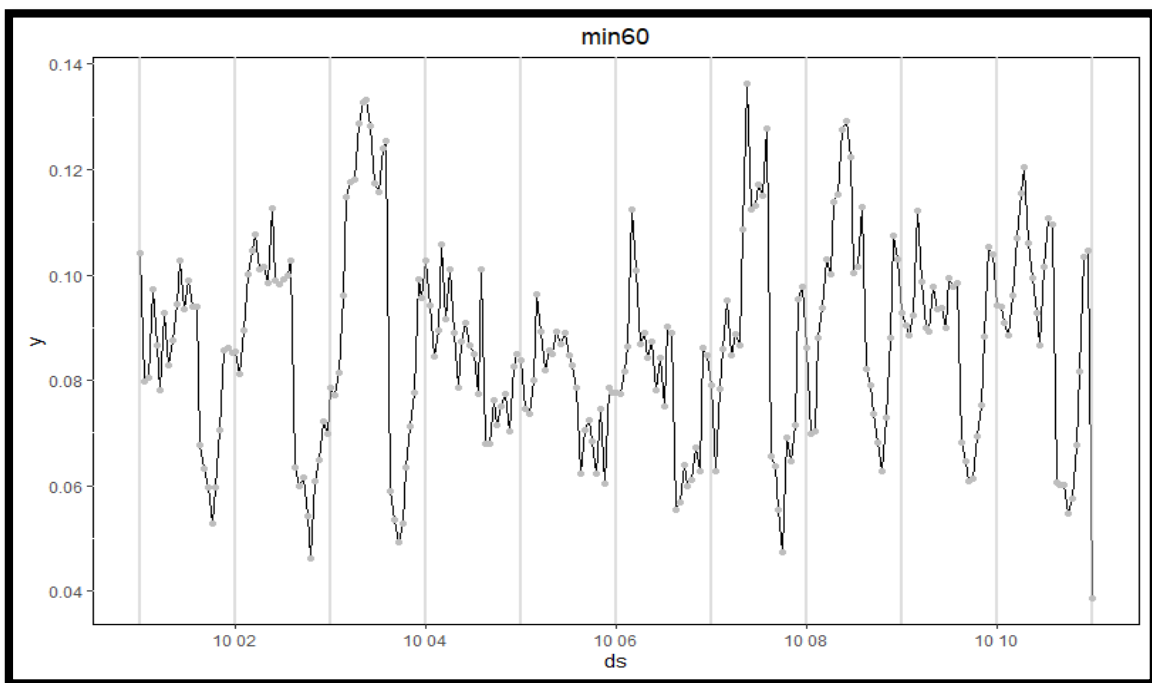
Leave One Out Encoding

- Data leakage를 방지하기 위해 **해당 행을 제외**한 Target Encoding
- 장점
 - Data leakage를 방지해 과적합 위험 감소
- 단점
 - Level이 하나라면 NA값이 생김

[Y] 합격	[X] 학과	LOO Encoding
1	경영	50% + ϵ
1	경영	50% + ϵ
0	경영	100% + ϵ
1	통계	33.33% + ϵ
1	통계	33.33% + ϵ
0	통계	66.67% + ϵ
0	통계	66.67% + ϵ
0	경제	50% + ϵ
0	경제	50% + ϵ
1	경제	0% + ϵ

1. TRAIN & TEST 전처리

클릭률 파생 변수 생성



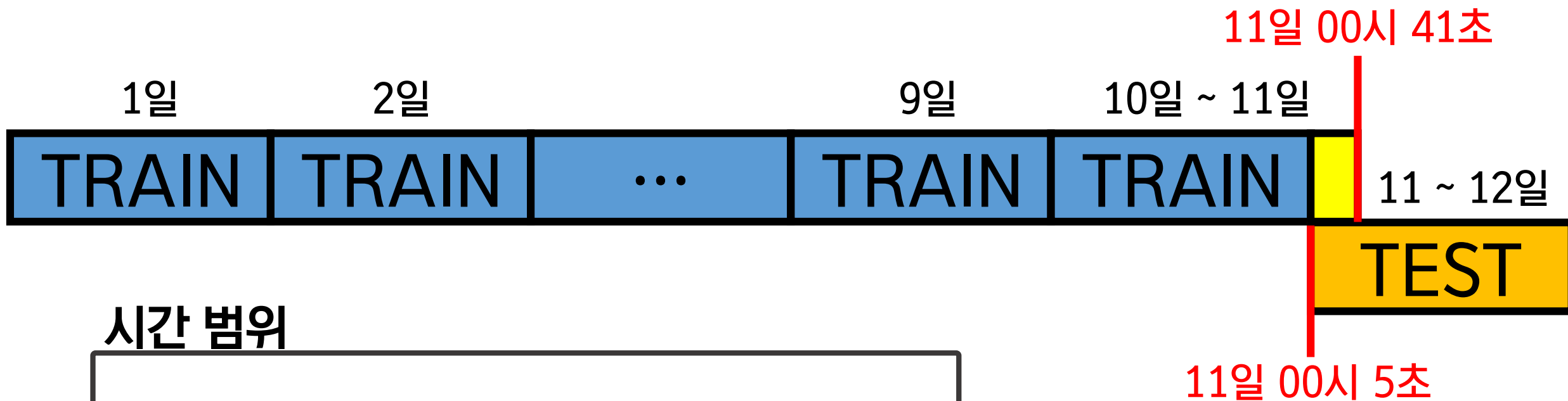
앞선 EDA에서 **클릭률**이
시계열 분포를 보인다고 판단하였다



시계열 모델을 활용하여
TEST 클릭률 파생변수를 생성



1. TRAIN & TEST 전처리

클릭률 파생 변수 생성

시간 범위

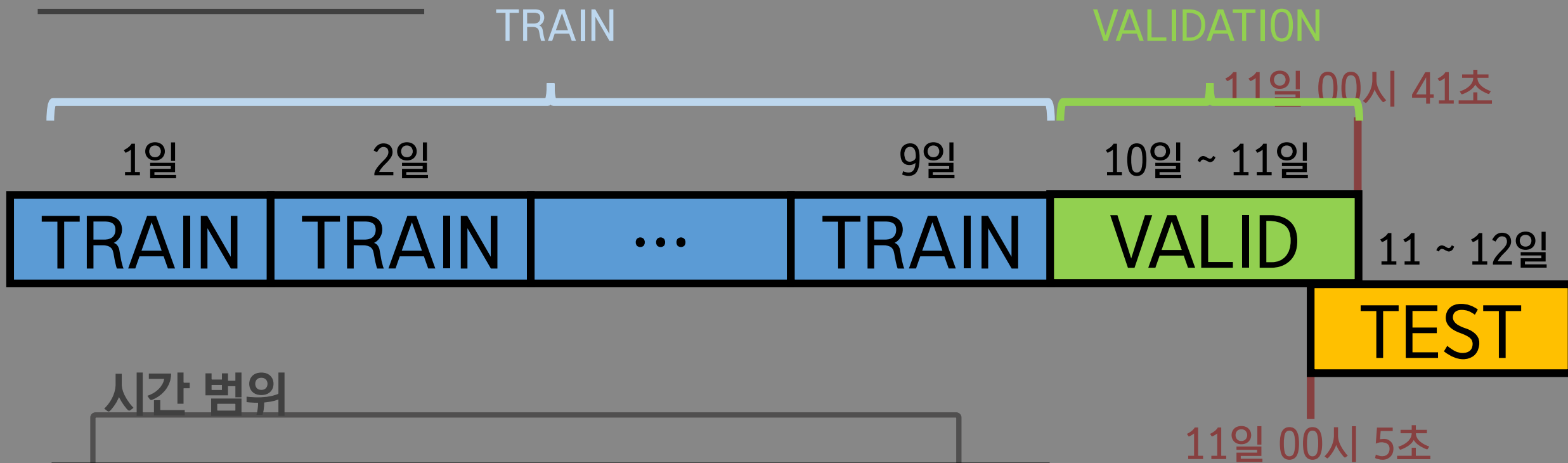
TRAIN : 1일 00시 00분 04초 ~ 11일 00시 00분 41초

TEST : 11일 00시 00분 05초 ~ 12일 00시 01분 22초



1. TRAIN & TEST 전처리

클릭률 파생 변수 생성



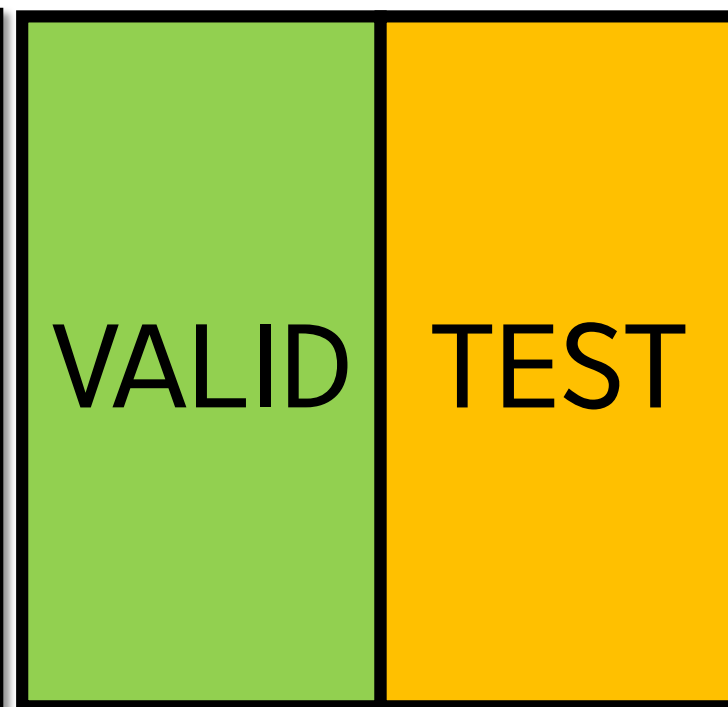
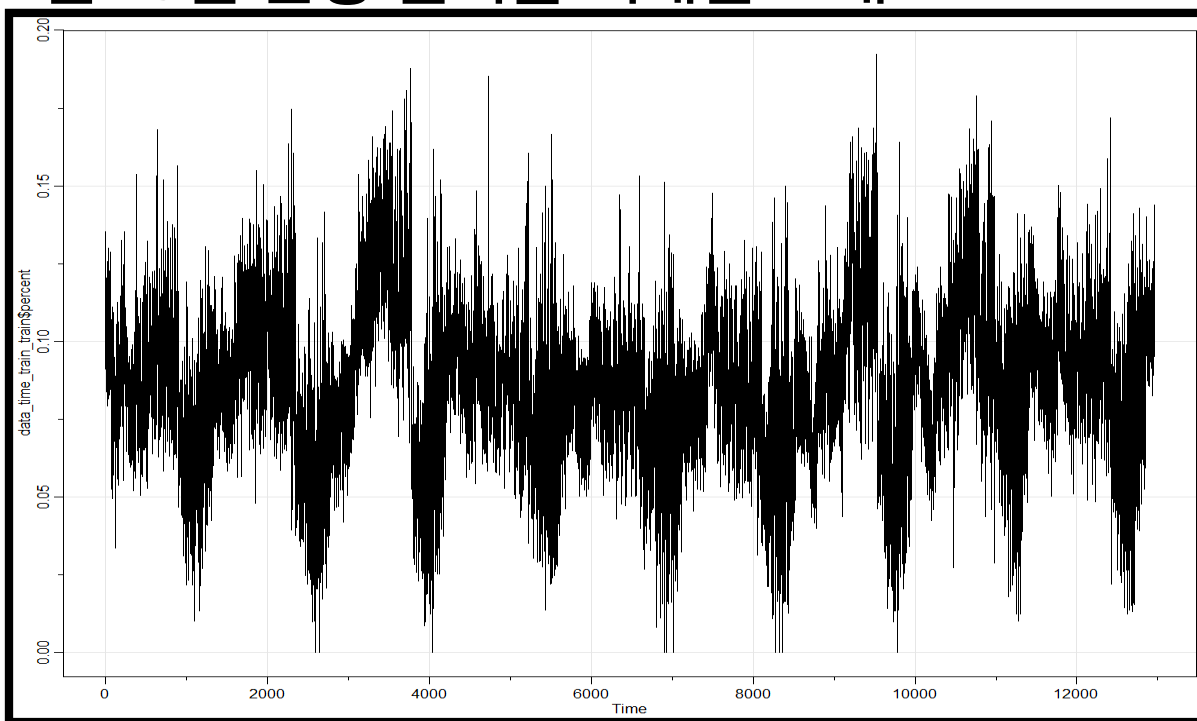
여러 시계열 모델 중 validation set의 분당 클릭률을
가장 잘 예측한 모델(가장 낮은 RMSE)을 찾아보자!



1. TRAIN & TEST 전처리

클릭률 파생 변수 생성

1일~ 9일 분당 클릭률 시계열 그래프



1일 ~ 9일

10일

11일

12일

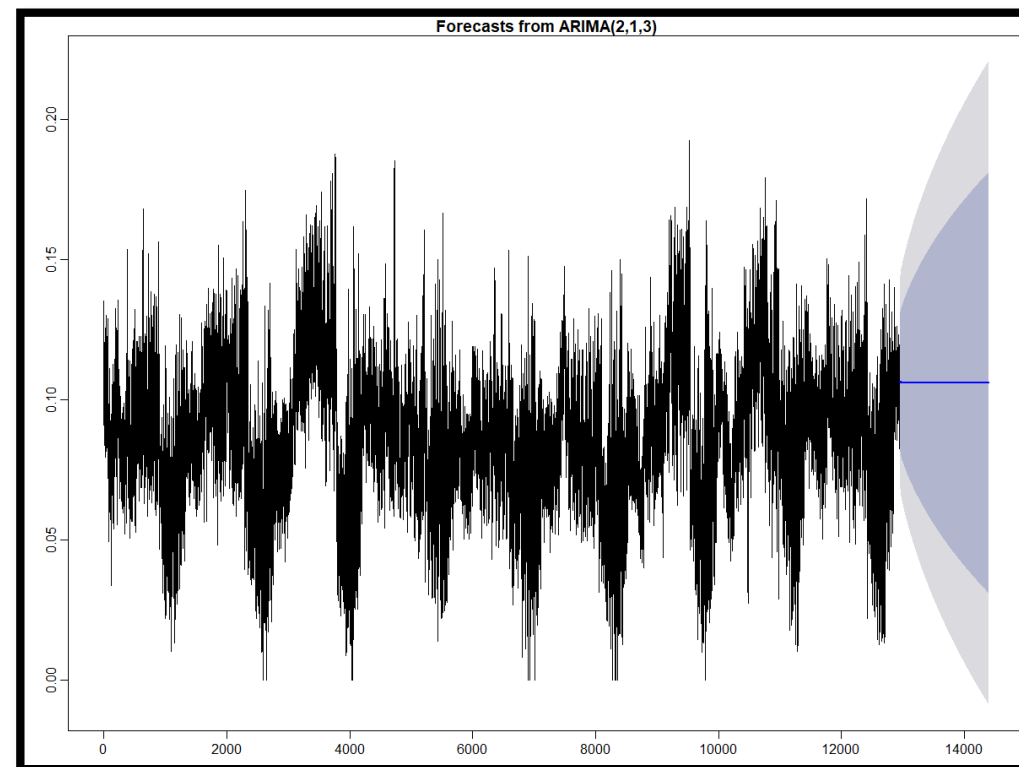
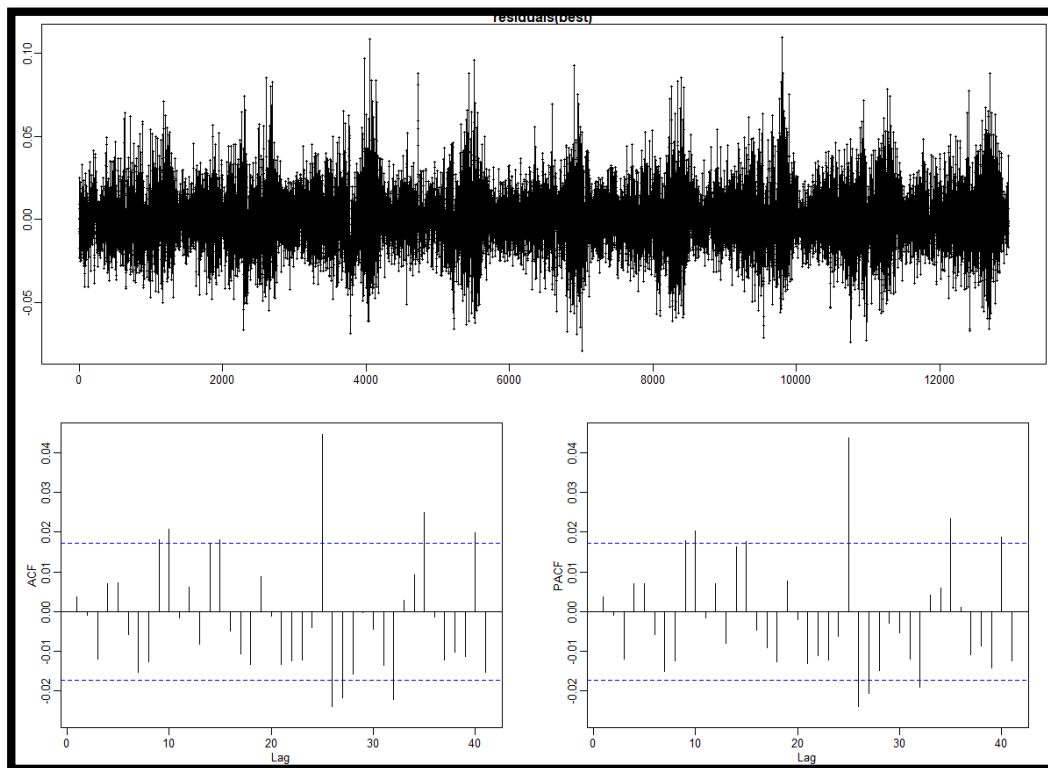


1. TRAIN & TEST 전처리

클릭률 파생 변수 생성

1

ARIMA 모델 - ARIMA(2, 1, 3) : [RMSE] 0.03149301



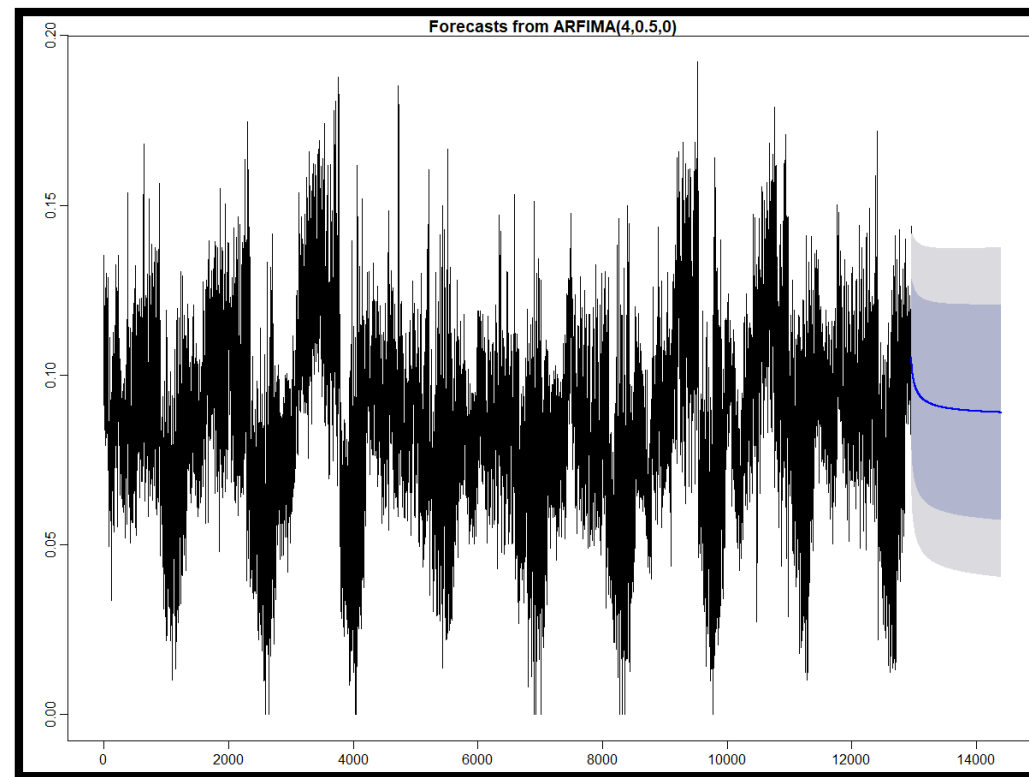
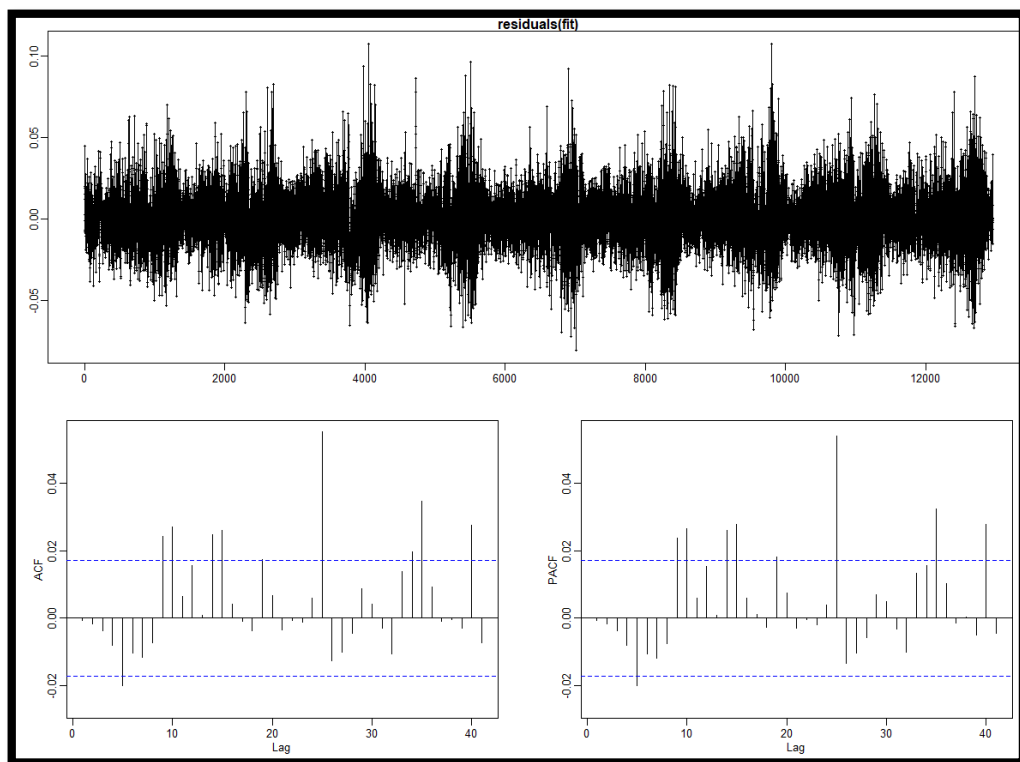


1. TRAIN & TEST 전처리

클릭률 파생 변수 생성

2

ARFIMA 모델 - ARFIMA(4, 0.5, 0) : [RMSE] 0.02682807



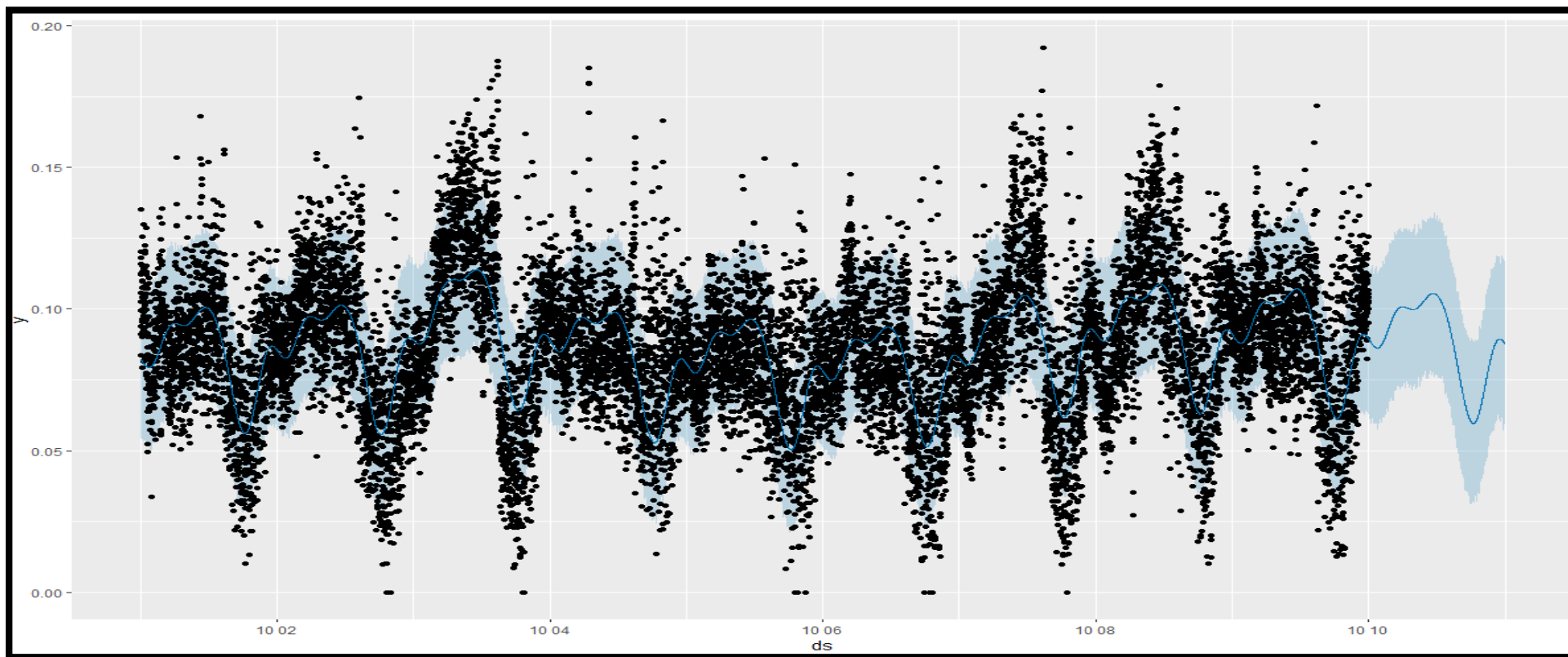


1. TRAIN & TEST 전처리

클릭률 파생 변수 생성

3

Prophet 모델 : [RMSE] 0.02213476





1. TRAIN & TEST 전처리

클릭률 파생 변수 생성

3

Prophet 모델

PROPHET

• 페이스북이 만든 시계열 예측 라이브러리

• 통계적 지식이 없어도 직관적 파라미터를 통해
모형을 조정할 수 있음

• 일반적인 경우 기본값만 사용해도 높은 성능을
보여줌

1. TRAIN & TEST 전처리

클릭률 파생 변수 생성

3

Prophet 모델

PROPHET

<장점>

- 기존 시계열 모델이 가정이 매우 엄격한 데 비해 엄격한 가정 없이도 모델링을 할 수 있다.
- 기준 시점 직전에 변동이 있어도 큰 흐름을 놓치지 않는다.
- 계절성 처리에 좋다
- NA를 핸들링해준다

<단점>

- 내부가 어떻게 동작하는지는 알지 못한다.
- 완전자동화 되는 시계열은 튜닝하기가 어렵다

1. TRAIN & TEST 전처리

클릭룰 파생 변수 생성

3

1일 Prophet 모델 2일

9일 <장점>

TRAIN

TRAIN

...

TRAIN

VALID

가정



1분

5분

10분

20분

30분

60분

RMSE



파라미터 튜닝:

- Linear vs logistic
- Logistic이라면 floor, cap
- Changepoints
- Holiday(10/5, 10/6)

<단점>

는지는 알지 못한다.

별은 튜닝하기가 어렵다

1. TRAIN & TEST 전처리

클릭률 파생 변수 생성

3

Prophet 모델

	Logistic 여부	Floor	Cap	Changepoints	Holiday 여부	RMSE
60분당	0	0	0.09	30	X	0.0144363
30분당	0	0.02	0.09	15	X	0.01471585
20분당	0	0.02	0.2	20	X	0.01472416
10분당	0	0.03	0.11	10	X	0.01515923
5분당	0	0.01	0.2	15	0	0.01607231
1분당	0	0	0.09	20	X	0.02212643



1. TRAIN & TEST 전처리

새로운 범주 level 처리

train

	device_language	device_ifa
1	A	나
2	A	나
3	B	가
4	B	다
5	B	다
6	C	가

새로운 범주 level



test

	device_language	device_ifa
1	A	가
2	B	나
3	C	가
4	D	나
5	E	다
6	F	라



1. TRAIN & TEST 전처리

새로운 범주 level 처리

train

	device_language	device_ifa
1	A	나
2	A	나
3	B	가
4	B	다
5	B	다
6	C	가

Catboost에서는
새로운 범주 level을
모두 **기타 범주**로 묶
어 처리한다



test

	device_language	device_ifa
1	A	가
2	B	나
3	C	가
4	기타	나
5	기타	다
6	기타	기타

결론: 새로운 범주 레벨을 NA로 처리하여 NA를 채우자!



1. TRAIN & TEST 전처리

새로운 범주 level 처리

- 그런데 데이터의 대부분의 변수들이 범주형
 - 따라서 범주형 데이터의 NA를 채우기 위해서는 범주형 변수 사이의 연관관계를 알아야 한다
 - 일반적인 상관관계의 지표인 Correlation은 범주형에서 사용 불가능

→ Goodman-Kruskal tau 이용



1. TRAIN & TEST 전처리

새로운 범주 level 처리

Goodman-Kruskal $\tau(\text{tau})$

- 명목형 변수의 교차표에 대한 연관성을 측정하는 척도
- 특정 변수의 값이 주어졌을 때, 다른 변수 예측에서 개선 비율을 측정.

분할표에서,
$$\tau(x, y) = \frac{\sum_{i=1}^K \sum_{j=1}^L \left(\frac{\pi_{ij}^2 - \pi_{i+}^2 \pi_{+j}^2}{\pi_{+j}} \right)}{1 - \sum_{j=1}^L \pi_{+j}^2}$$

분할표		X2			
		a	b	c	합계
X1	가	24	30	22	76
	나	2	39	31	72
	다	5	11	10	26
	라	3	3	3	9
	합계	34	83	66	183

ex) $\tau(x_1, x_2) = 0.0435$, $\tau(x_2, x_1) = 0.0699$

1. TRAIN & TEST 전처리

새로운 범주 level 처리

train

	변수1	변수2	변수3
1	A	나	ㄱ
2	A	나	ㄱ
3	B	가	ㄱ
4	B	다	ㄱ
5	B	다	ㄴ
6	C	가	ㄴ
...

각각의 범주형 변수끼리
분할표 생성

분할표		변수2			
		가	나	다	...
1차원	A	0	2	0	...
	B	1	0	2	...
	C	1	0	0	...

...

분할표		변수3			
		ㄱ	ㄴ
1차원	A	2	0
	B	2	1
	C	0	1

Tau (1, 2)
Tau (2, 1)

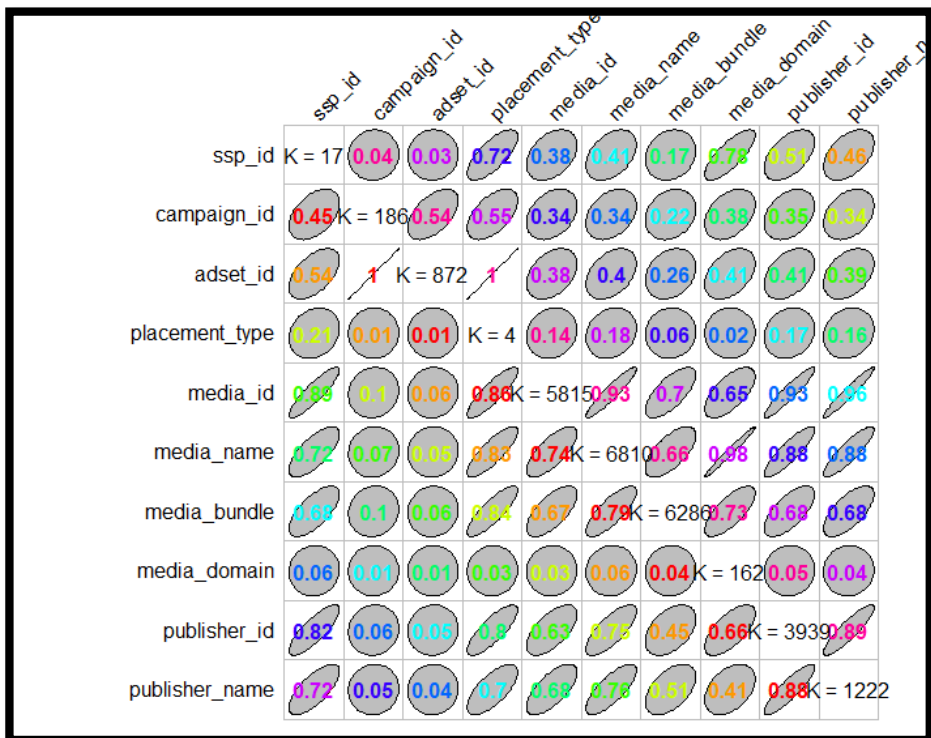
그리고 이를 통해서
연관관계를 나타내는
tau를 도출

Tau (1, 3)
Tau (3, 1)



1. TRAIN & TEST 전처리

새로운 범주 level 처리



	변수1	변수2	변수3	...
변수1	-	0.9	0.8	...
변수2	0.95	-	0.6	...
변수3	0.4	0.5	-	...
...

- TRAIN에 있는 모든 범주형 변수 끼리 Gktau 계산, 높은 연관성을 가진 변수들 관계 정리

1. TRAIN & TEST 전처리

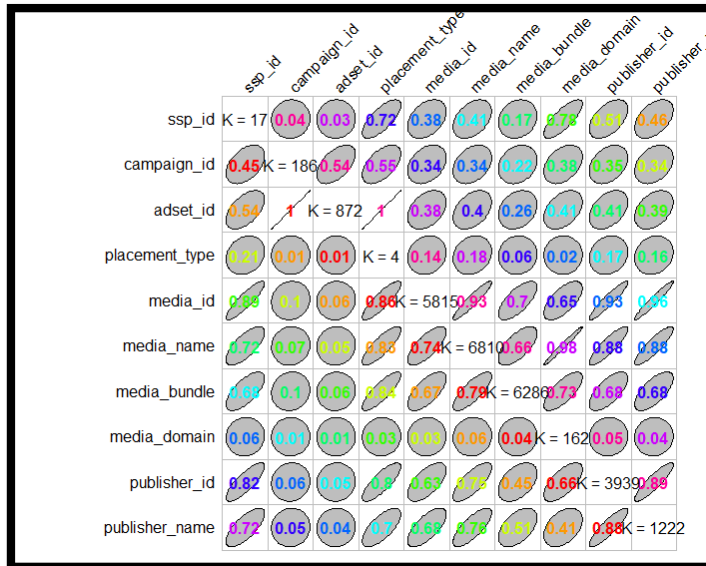
새로운 범주 level 처리

상황1

1. NA를 채우고 싶은 목표 변수를 설정!

예) device_language의 NA를 채우자!

2. 목표변수와 다른 변수 간의 Gktau를 보고 높은 값을 갖는 변수를 찾는다!



	device_language
ssp_id	0.38
campaign_id	0.34
adset_id	0.38
placement_type	0.14
device_ifa	0.74
...	...

1. TRAIN & TEST 전처리

새로운 범주 level 처리

상황1

train

	device_ language	device_ ifa
1	A	나
2	A	나
3	B	가
4	B	다
5	B	다
6	C	가
...

변수 1

목표변수

device_ ifa	device_ language	빈도수
가	A	102
가	B	51
나	B	95
나	C	62
다	D	85
다	A	15
...

device_ifa의 각 범주 level 마다
가장 빈도수가 많은
Device_language의
범주 level을 찾는다!

3. Gktau가 높은 범주형 변수 간의 빈도수를 보고
가장 빈도수가 높은 범주 level 조합을 찾는다!



1. TRAIN & TEST 전처리

새로운 범주 level 처리

상황1

변수 1

목표변수

device_ifa	device_language	빈도수
가	A	102
가	B	51
나	B	95
나	C	62
다	D	85
다	A	15
...

test

	device_language	device_ifa
1	A	가
2	B	나
3	C	가
4	NA -> B	나
5	NA -> D	다
6	NA -> A	가

4. 찾은 범주 level의 조합으로 test set의 NA를 대체한다!

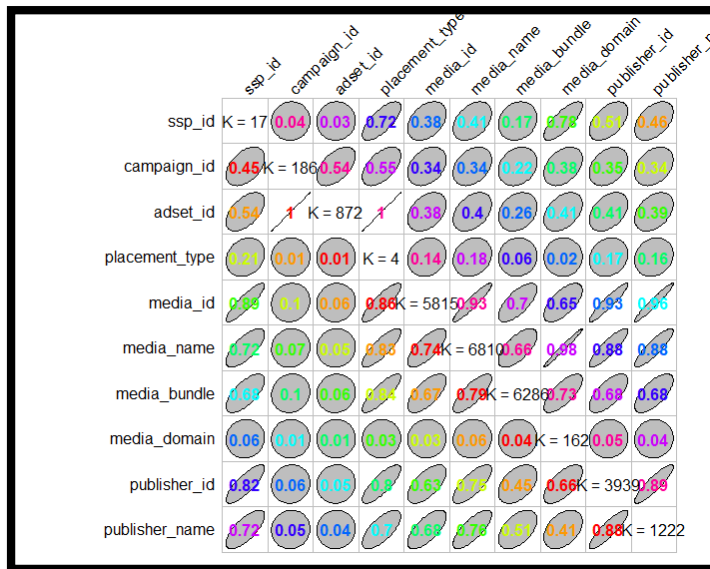
1. TRAIN & TEST 전처리

새로운 범주 level 처리

상황2

1. NA를 채우고 싶은 목표 변수를 설정!

예) media_carrier의 NA를 채우자!

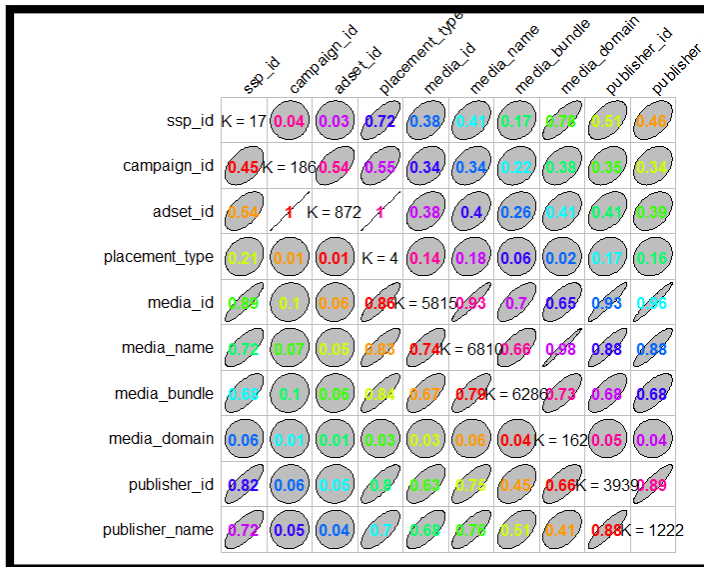
2. 목표변수와 다른 변수 간의 **Gktau**가 모두 0.7 보다 높지 않다면?

	media_carrier
ssp_id	0.18
campaign_id	0.24
adset_id	0.56
placement_type	0.44
media_name	0.64
...	...

1. TRAIN & TEST 전처리

새로운 범주 level 처리

상황2



	media_carrier
media_name + ssp_id	0.18
media_name + campaign_id	0.24
media_name + adset_id	0.56
media_name + placement_type	0.44
media_name + device_ifa	0.85
...	...

3. 두 개 이상의 변수를 조합했을 때 Gktau가 높은 조합을 찾는다!

1. TRAIN & TEST 전처리

새로운 범주 level 처리

상황2

train

	device_carrier	media_name	device_ifa
1	A	가	ㄱ
2	A	가	ㄱ
3	B	나	ㄴ
4	B	나	ㄷ
5	B	다	ㅅ
6	C	라	ㅇ
7	C	다	ㄹ
8	C	라	ㅁ
...

변수1 + 변수2

목표변수

media_name + device_ifa	device_carrier	빈도수
가ㄱ	A	215
가ㄱ	B	17
나ㄴ	B	156
나ㄴ	D	84
나ㄴ	E	26
라ㅇ	C	67
라ㅇ	A	3
...

media_name + device_ifa의

각 범주 level 마다
가장 빈도수가 많은Device_carrier의
범주 level을 찾는다!

4. Gktau가 높은 범주형 조합 변수 간의 빈도수를 보고
가장 빈도수가 높은 범주 level 조합을 찾는다!



1. TRAIN & TEST 전처리

새로운 범주 level 처리

상황2

test

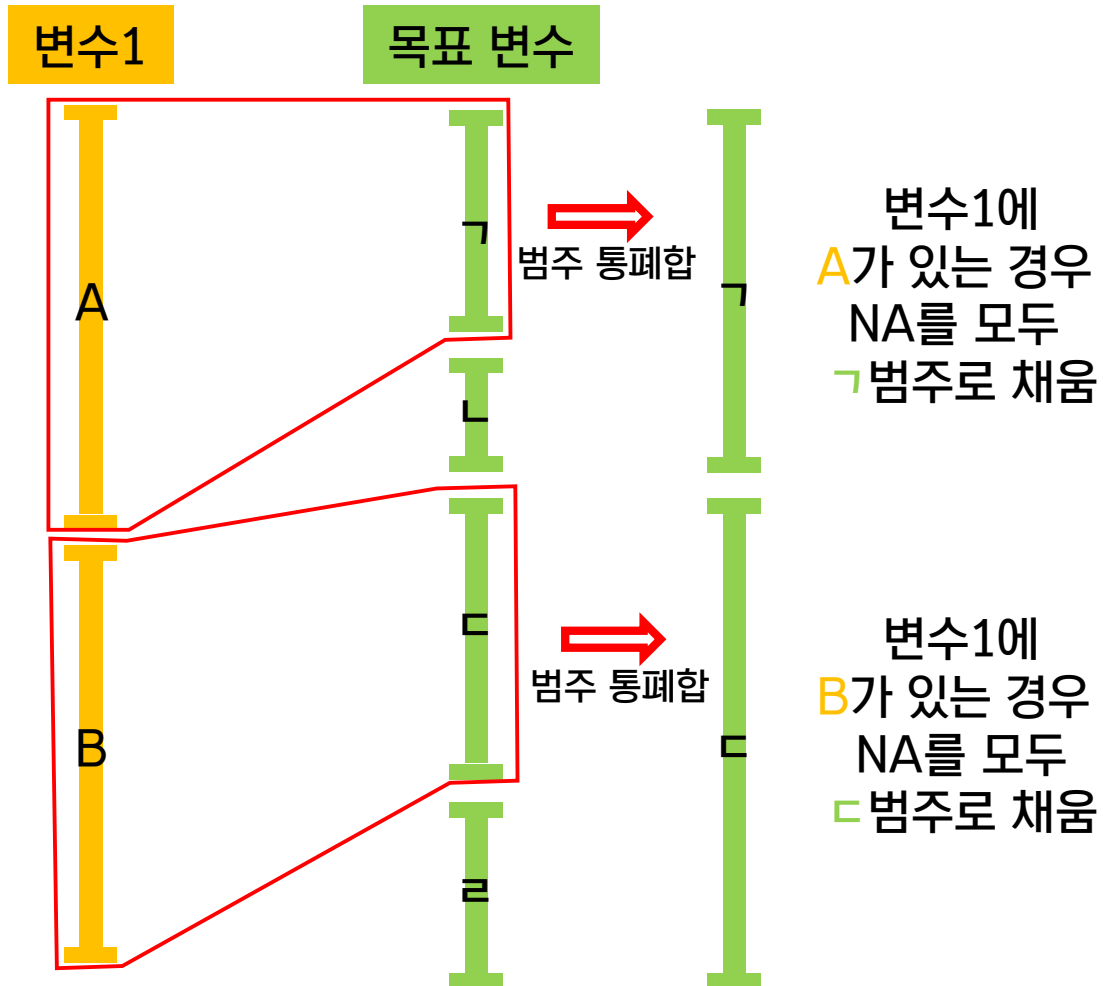
변수1 + 변수2	목표변수	
media_name + device_ifa	device_carrier	빈도수
가ㄱ	A	215
가ㄱ	B	17
나ㄴ	B	156
나ㄴ	D	84
나ㄴ	E	26
라ㅇ	C	67
라ㅇ	A	3
...

	device_carrier	media_name	device_ifa
1	NA -> A	가	ㄱ
2	NA -> B	나	ㄴ
3	C	가	ㄴ
4	A	나	ㄷ
5	A	다	ㄴ
6	NA -> C	라	ㅇ
...

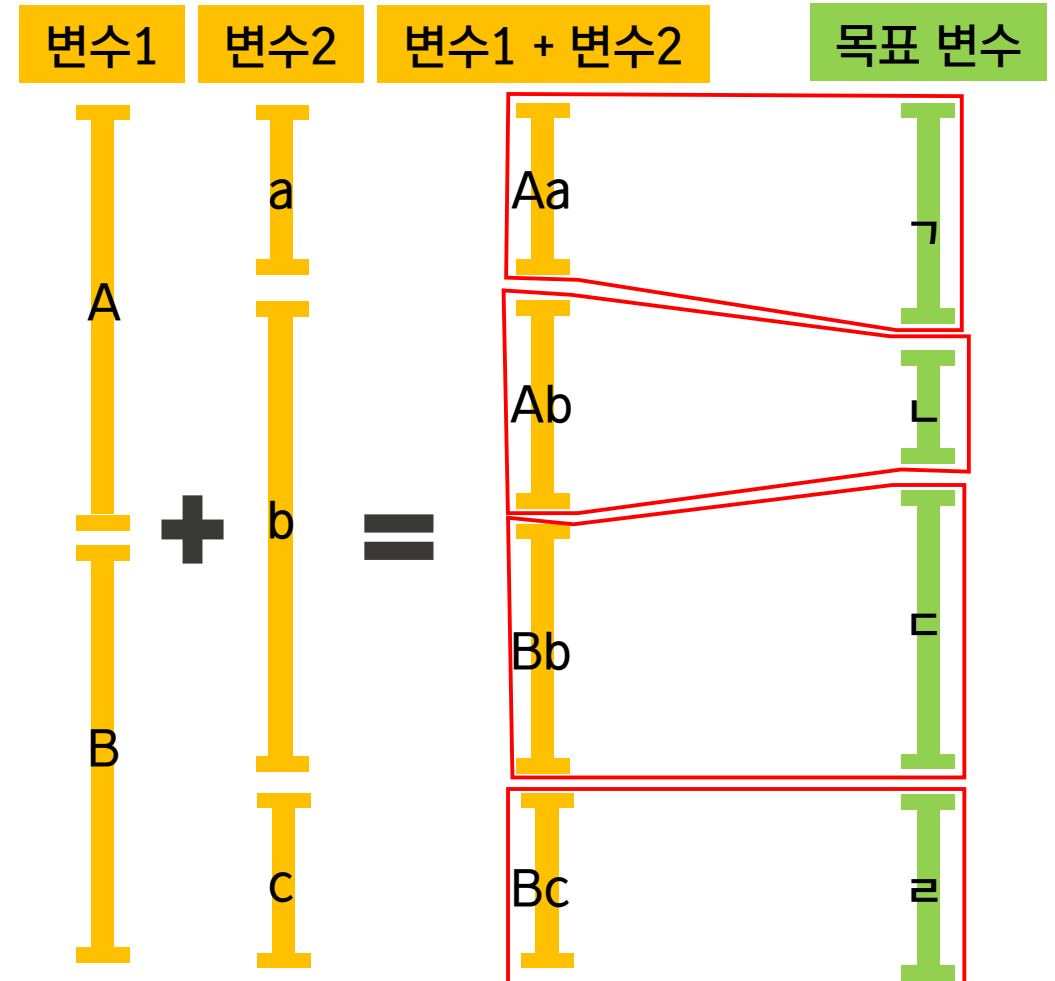
5. 찾은 범주 level 조합으로 test set의 NA를 대체한다!

* 왜 조합할까?

범주 개수가 적으면 목표 변수의 범주가
통폐합되어 잘 설명하지 못함



변수1과 변수2를 조합하여 세분화해 **설명력**을 높임



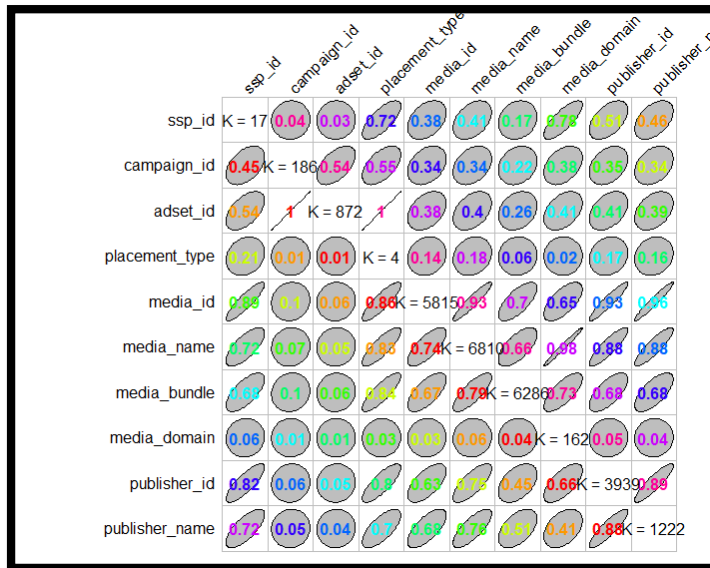
1. TRAIN & TEST 전처리

새로운 범주 level 처리

상황3

1. NA를 채우고 싶은 **목표 변수**를 설정!

예) publisher_name의 NA를 채우자!

2. 목표변수와 다른 변수 간의 **Gktau**를 보고 **높은 값**을 갖는 변수 혹은 변수 조합을 찾는다!

	publisher_name
ssp_id	0.43
campaign_id	0.32
adset_id	0.45
placement_type	0.75
media_name	0.89
...	...

1. TRAIN & TEST 전처리

새로운 범주 level 처리

상황3

train

	publisher_name	media_name	media_bundle	device_make
1	A	가	ㄱ	a
2	A	가	ㄱ	b
3	B	나	ㄴ	d
4	B	나	ㄷ	c
5	B	다	ㄹ	a
6	C	라	ㅇ	a
7	C	다	ㄹ	b
8	C	라	ㅁ	c
...

변수1

목표변수

media_name	publisher_name	빈도수
가	A	196
가	D	18
나	B	159
나	E	16
라	C	148
라	A	25
...

media_name 의
각 범주 level 마다
가장 빈도수가 많은
Publisher_name 의
범주 level을 찾는다!

3. Gktau가 높은 범주형 변수 간의 빈도수를 보고
가장 빈도수가 높은 범주 level 조합을 찾는다!

1. TRAIN & TEST 전처리

새로운 범주 level 처리

상황3

변수1 media_name	목표변수 publisher_name	빈도수
가	A	196
가	D	18
나	B	159
나	E	16
라	C	148
라	A	25
...

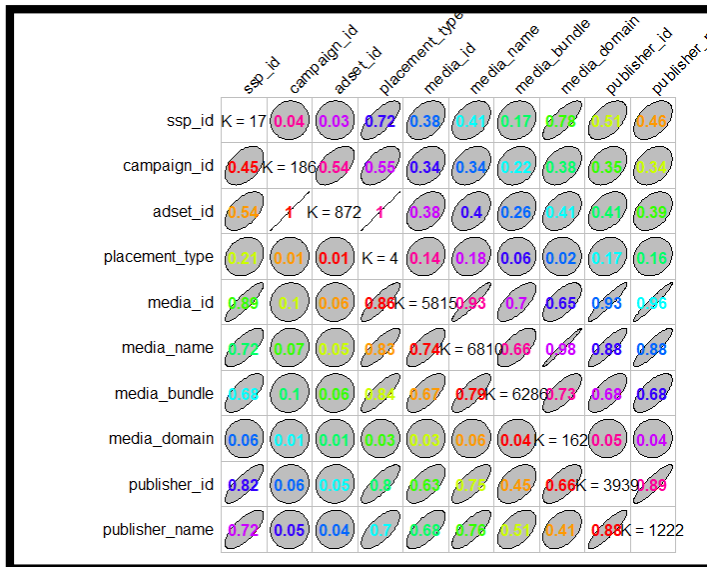
	publisher_name	media_name
1	NA -> A	가
2	NA -> B	나
3	NA -> NA	다
4	NA -> NA	다
5	NA -> NA	다
6	NA -> C	라
...

4. 찾은 범주 level 조합으로 test set의 NA를 대체하였으나
해당 변수에 새로운 범주 level이 생겨 NA가 많이 채워지지 않았다면?

1. TRAIN & TEST 전처리

새로운 범주 level 처리

상황3



	Publisher_name
media_bundle + ssp_id	0.36
media_bundle + campaign_id	0.54
media_bundle + adset_id	0.65
media_bundle + placement_type	0.45
media_bundle + device_make	0.73
...	...

5. Gktau가 높은 또 다른 변수 혹은 조합을 찾는다!



1. TRAIN & TEST 전처리

새로운 범주 level 처리

상황3

train

	publisher_name	media_name	media_bundle	device_make
1	A	가	ㄱ	a
2	A	가	ㄱ	b
3	B	나	ㄴ	d
4	B	나	ㄷ	c
5	B	다	ㄴ	a
6	C	라	ㅇ	a
7	C	다	ㄹ	b
8	C	라	ㅁ	c
...

변수2 + 변수3

목표변수

media_bundle + device_make	publisher_name	빈도수
ㄱa	A	156
ㄱa	B	18
ㄴd	B	124
ㄴd	C	121
ㄴd	A	11
ㄹb	C	89
ㄹb	D	17
...

publisher_name의
각 범주 level 마다
가장 빈도수가 많은
Media_bundle
+ device_make의
범주 level을 찾는다!

6. Gktau가 높은 범주형 변수 혹은 변수 조합 간의 빈도수를 보고
가장 빈도수가 높은 범주 level 조합을 찾는다!

1. TRAIN & TEST 전처리

새로운 범주 level 처리

상황3

변수2 + 변수3

목표변수

media_bundle + device_make	publisher_ name	빈도수
ㄱa	A	156
ㄱa	B	18
ㄴd	B	124
ㄴd	C	121
ㄴd	A	11
ㄹb	C	89
ㄹb	D	17
...

test

	publisher_n ame	media_ name	media_ bundle	device _make
1	NA -> A	가	ㄱ	d
2	NA -> B	나	ㄴ	a
3	NA -> C	다	ㄹ	b
4	NA -> NA	다	ㅁ	a
5	NA -> B	다	ㄴ	d
6	NA -> C	라	ㄹ	c
...

7. NA를 못 채워 넣은 부분에만 채워넣는다!



1. TRAIN & TEST 전처리

새로운 범주 level 처리

위 상황들과 같이 **NA를 잘 채우면서도**

Gktau가 높아 **정확도**가 높은 변수 조합들을 찾아주었다. 결과는 다음 표와 같다.

목표 변수	범주 level 개수	NA 개수	1단계	Tau	2단계	tau
advertisement_id	30	5771	device_ifa	0.849		
device_language	33	1	media_bundle & media_id	0.989		
device_region	148	1	device_city & media_bundle	0.964		
campaign_id	186	8384	device_ifa	0.633		
device_make	299	8	media_id & device_model	0.991	media_id & device_os_version	0.8281



목표 변수	범주 level 개수	NA 개수	1단계	Tau	2단계	tau
device_carrier	536	12	media_name & device_ifa	0.9829		
adset_id	872	9869	device_ifa & media_bundle	0.6701	device_ifa	0.551
publisher_name	1222	260	media_name	0.8803	media_bundle & device_make	0.8792
device_city	1326	30	device_ifa & device_region	0.9544	device_region & publisher_id	0.6814
device_model	1664	48	device_ifa	0.9450		



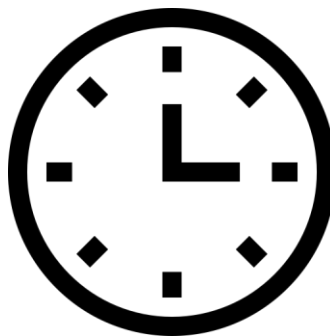
목표 변수	범주 level 개수	NA 개수	1단계	Tau	2단계	tau
publisher_id	3939	422	media_name	0.8747		
media_id	5815	893	media_bundle & ssp_id	0.9514	media_bundle & device_make	0.8568
media_bundle	6286	298	device_ifa	0.766	publisher_name	0.507
media_name	6810	686	ssp_id & media_bundle	0.9918	ssp_id & publisher_name	0.8378
device_ifa	186만 9137	152636	전체			

1. TRAIN & TEST 전처리

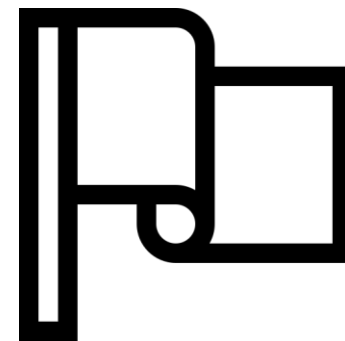
기타 전처리



주말, 평일
변수 생성

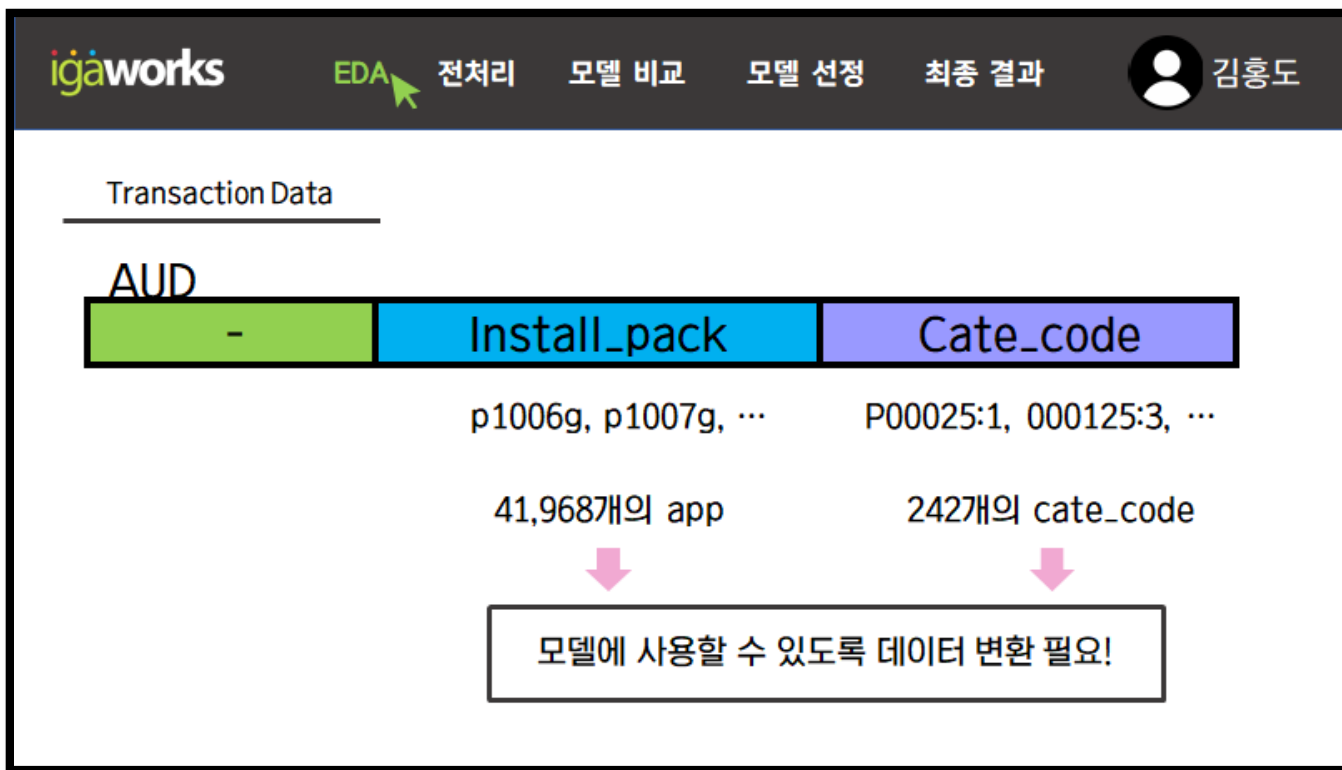


시간 변수 생성
(범주형)



device_country
제거

2. AUDIENCE 전처리



Transaction Data

AUD

	Install_pack	Cate_code
	p1006g, p1007g, ...	P00025:1, 000125:3, ...
	41,968개의 app	242개의 cate_code

모델에 사용할 수 있도록 데이터 변환 필요!

- AUD 데이터에서는 **Transaction Data** 형태의 자료들 존재
→ 모델링을 위해서 **일반적인 데이터 형태로 변환**
- AUD 데이터의 크기 때문에 **여러 파일로 분리**하여 진행



2. AUDIENCE 전처리

Transaction 데이터**INSTALL_PACK 변수**

device_ifa	install_pack_num	p1006g	p1007g	p1008g
j00Wr9NpsL	114	1	1	0
y9tMCHCnx0	100	0	1	0
kJ0NENXpPx	228	0	0	0
D0D3n4ZkVc	217	0	0	1

- p1006g, p1007g, ... 총 41,968개의 app
- device_ifa를 기준으로 app 개수를 나타내는 column
→ **install_pack_num**
- 각 app을 feature로 하는 **One-hot encoding** 형식의 데이터로 변환



2. AUDIENCE 전처리

Transaction 데이터

INSTALL_PACK 변수

Mean (click)	device_ifa	p1006g	p1007g	p1008g
1	j00Wr9NpsL	1	1	0
0.77	y9tMCHCnx0	0	1	0
0.5	kJ0NENXpPx	0	0	0
0.8	D0D3n4ZkVc	0	0	1

- app 개수가 너무 많음
- Train의 **click**과 app의 binary encoding과 **상관관계**를 계산해 상관관계가 **높은 84개**의 app만을 추출해 feature 생성
- 클릭률에 의미 있는 app만을 선택



2. AUDIENCE 전처리

Transaction 데이터

CATE_CODE 변수

device_ifa	cat1	cat2	cat3	cat4	cat5	01001	01002	01003
j00Wr9NpSL	8	8	12	15	11	0	4	4
y9tMCHCnx0	7	7	13	12	7	4	1	4
kJ0NENXpPx	32	13	14	9	12	2	1	3
D0D3n4ZkVc	24	15	19	12	11	3	1	1

- P00025:1, 000125:3, ... 총 242개의 cate_code

- device_ifa를 기준으로 1부터 5까지의 평점 개수를 feature로 생성
- 각 cate_code에 대한 평점을 feature로 생성
- 평점이 NA라면 0으로 대체
→ 관심이 적은 것으로 판단



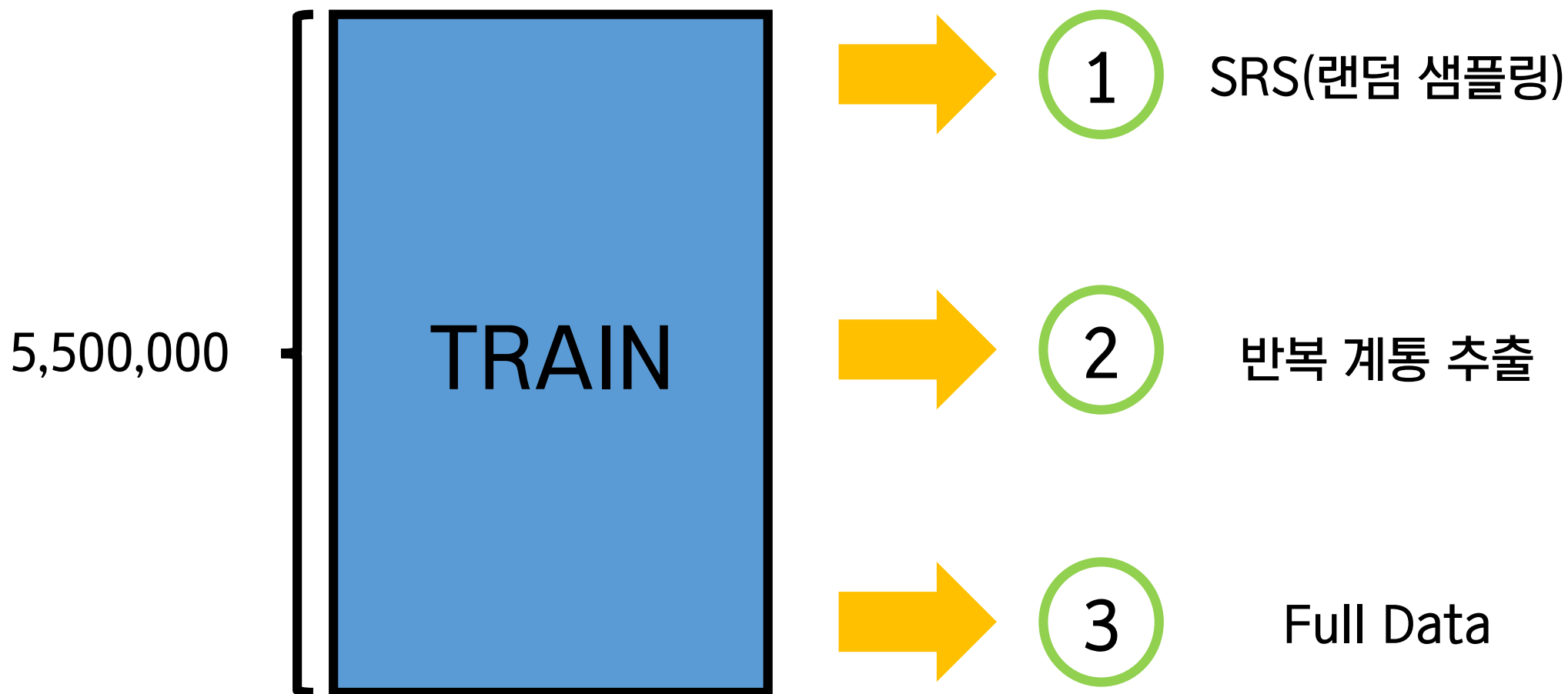
3. 최종데이터

TRAIN 전처리		TRAIN 클릭률		AUD
click	550만 x 23 변수	weekend	hour	550만 x 6 변수
				223만 2520 x 335 변수
				NA

TEST 전처리		TEST 클릭률 예측		AUD
	55만 x 23 변수	weekend	hour	55만 x 6 변수
				21만 4642 x 335 변수
				NA



3. 최종데이터

샘플링(Sampling)



3. 최종데이터

샘플링(Sampling)

1

SRS(랜덤 샘플링)

5,500,000

TRAIN

10% 랜덤 샘플링



550,000



3. 최종데이터

샘플링(Sampling) 주어진 train 데이터는 1일부터 10일까지의 데이터!

- ① SRS(랜덤 샘플링) 만약, 어느 한 시점에 샘플링 데이터가 모여있다면?
5,500,000

TRAIN

10% 랜덤 샘플링



계통 추출을 이용하자!

3. 최종데이터

샘플링(Sampling)

2

반복 계통 추출

계통 추출: 첫 번째 요소는 무작위로 선택한 후
목록의 매번 **k번째 요소**를 표본으로 선정하는 표집방법

예시

K = 10

TRAIN

4

14

24

34

44

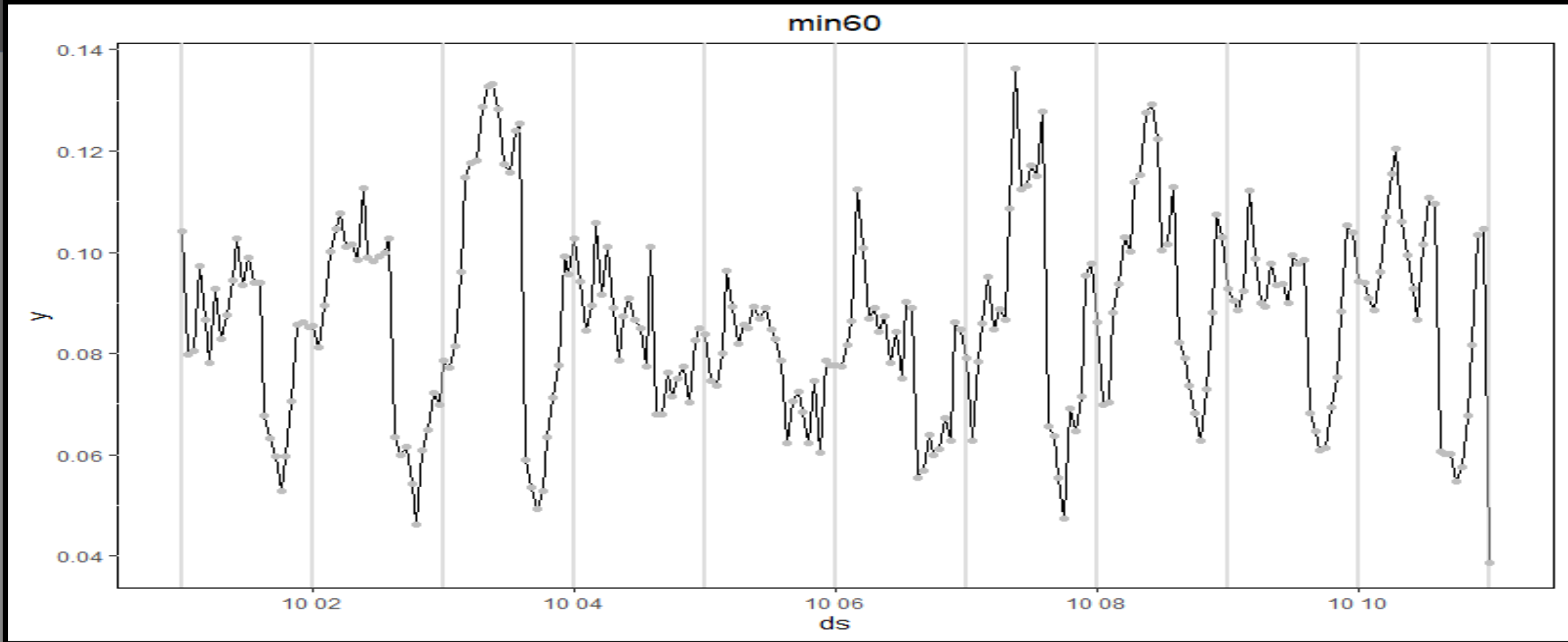
54

64

74

84

94



클릭률이 일별로 **주기성**을 갖기 때문에
계통추출을 했을 경우 문제가 될 수 있다!



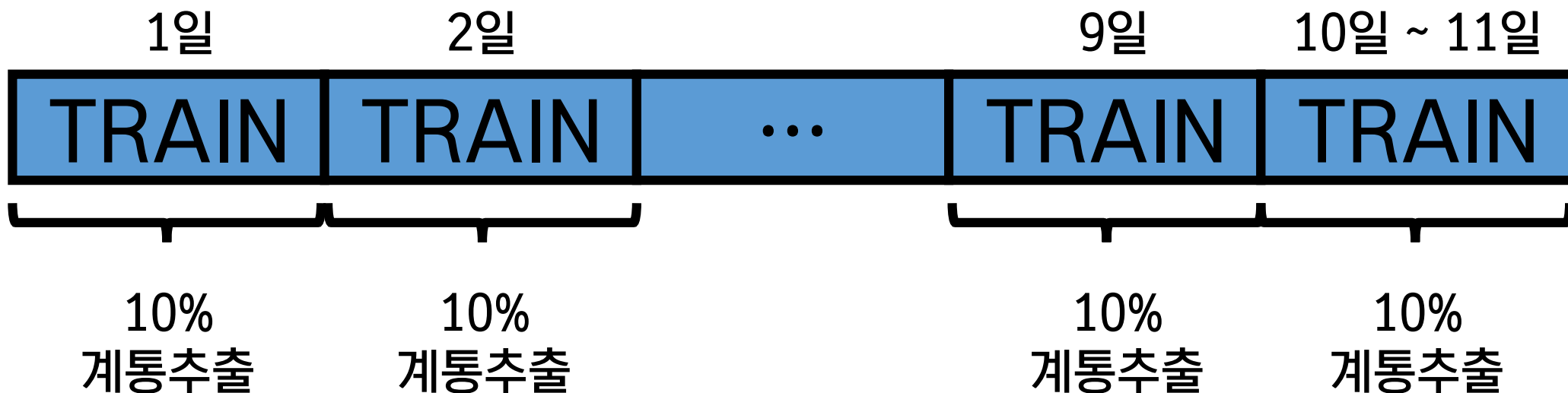
반복 계통 추출!



3. 최종데이터

샘플링(Sampling)

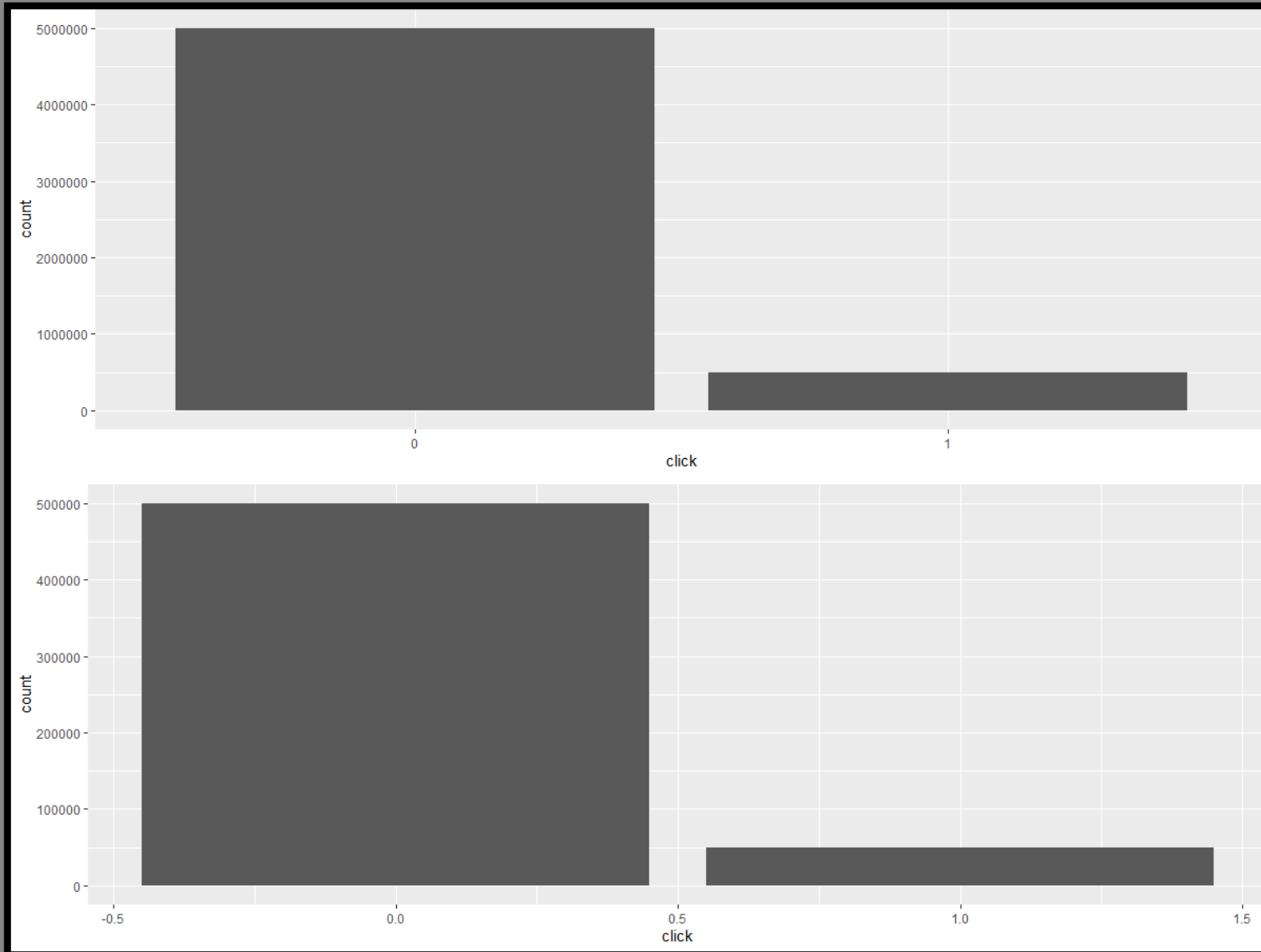
2 반복 계통 추출



그룹 별로 각각 계통 추출을 해준다!



<원 데이터>



	0	1
개수	5,000,000	500,000

9일

10일 ~ 10%

<샘플링 데이터>

	0	1
개수	499,963	50,041

9.99%



3. 최종데이터

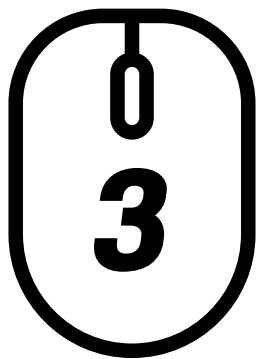
샘플링(Sampling)

3

Full Data

TRAIN

몇몇 모델들은 샘플링 없이도 돌릴 수 있었다!



모델 비교

◀ ▶ 🔍 모델 비교	
• MODEL	
GLM	XGBoost
SVM	LightGBM
NBC	CatBoost
Random Forest	DNN



GLM

- 다양한 연결함수를 사용할 수 있음
 - cloglog가 가장 낮은 Logloss를 기록 (click이 0에 많이 분포하기 때문이라고 추정)

장점

- 해석이 쉬운 모델
- 적합 속도, 예측 속도 빠름

단점

- 자동적으로 더미변수를 만들기 때문에 큰 수준의 범주형 변수의 경우 모델 사이즈가 커짐
- 새로운 범주 level 처리 불가능
- NA 처리 불가능



SVM

선형방식과 비선형 방식(커널 트릭)이 있음

장점

- 새로운 범주 level 처리 가능
- Linear와 Kernel 모두 파라미터(각 1, 2개)를 수정하여 더 나은 값을 얻을 수 있음

단점

- 차원이 클수록 모델 사이즈의 문제가 커짐
- 적합 속도, 예측 속도 느림
- NA 처리 불가능



NBC

자체적으로 변수의 독립성을 가정

장점

- 단순한 모형
- 새로운 범주 level 처리 가능
- 적합 속도 빠름

단점

- 독립이라는 가정이 위배되면 성능의 저하가 일어날 수 있음
- 결과값으로 나오는 확률은 정확하지 않음
- 예측 속도 느림
- NA 처리 불가능



Random Forest

트리의 배경 모델

장점

- 모델을 조절하는 중요 파라미터로 mtry값을 조정하면 좋은 결과를 얻을 수 있음
- 적합 속도 빠름
- 새로운 범주 level 처리 가능

단점

- 차원이 클수록 모델 사이즈의 문제가 커짐
- 범주 level이 커지면(53개 이상) 처리할 수 없음
- NA 처리 불가능



XGBoost

트리의 부스팅 모델, 수치형 데이터만 받음

장점

- 일반적으로 트리 기반 부스팅 모델 중 성능 가장 우수
- 새로운 범주 level 처리 가능
- NA 처리 가능

단점

- 인코딩 필요 (python)
- 대부분이 범주형 변수인 현 데이터의 특성 상 성능 저하



LightGBM

트리의 부스팅 모델, 수치형 데이터만 받음(R에서는 자체 인코딩)

장점

- XGBoost보다 **적합 및 예측 속도 빠름**
- **새로운 범주** level 처리 가능
- **NA** 처리 가능

단점

- 인코딩 필요 (python)
- 대부분이 **범주형 변수**인 현 데이터의 특성
상 **성능 저하**

CatBoost

트리의 부스팅 모델

장점

- Ordered Boosting, Random Permutation, Ordered Target Encoding → 과적합 방지
- Default 파라미터 값의 성능이 좋음
- 병렬화와 GPU 사용이 가능하여 효과적인 훈련이 가능
- 예측 속도 빠름
- 새로운 범주 level 처리 및 NA 처리 가능

단점

- NA 핸들링이 정교하지 못함 (default = min)
- LGBM보다 속도가 상대적으로 느림 (특히 데이터가 수치형일 때)
- train과 test의 column 순서가 일치해야 함 (그렇지 않으면 error 증가)



DNN with entity embedding

장점

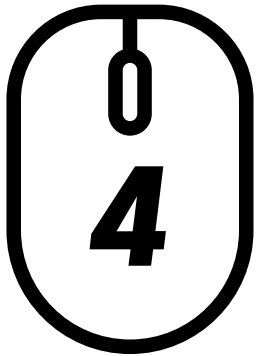
- Data 양이 많아지면 성능이 계속 좋아짐
- Feature extraction이 자동
- Entity embedding으로 범주 정보 손실 최소화
- 새로운 범주 level 처리 가능

단점

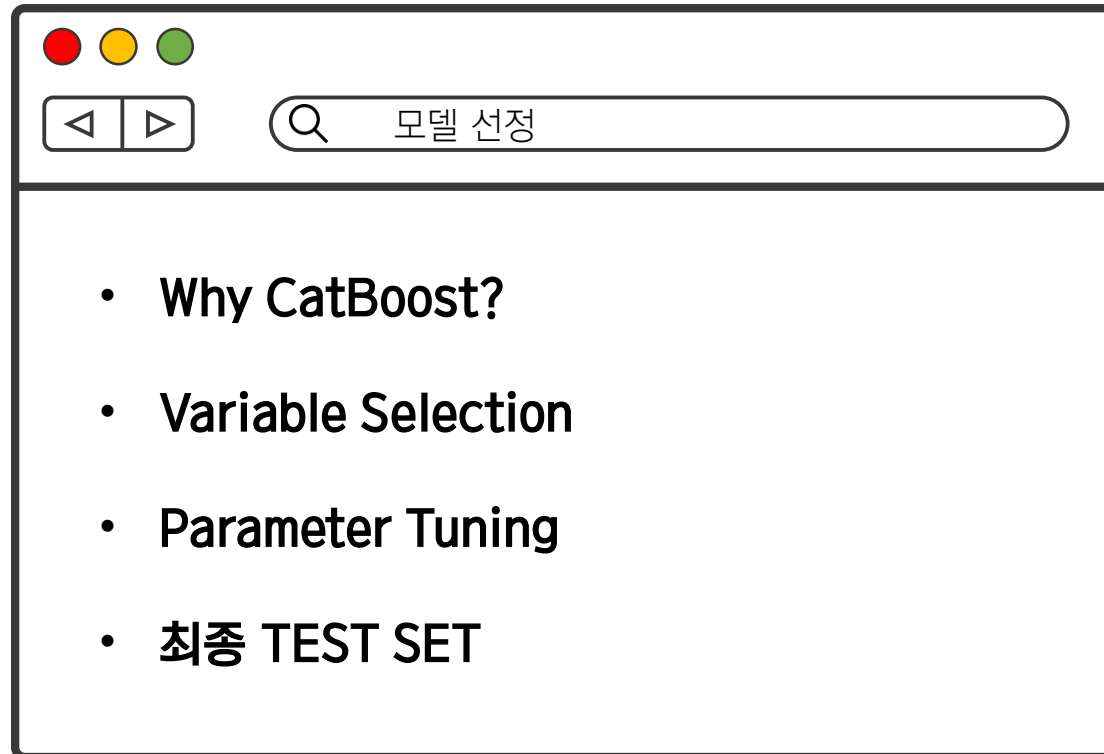
- 해석이 어려운 모델
- NA 처리 불가능



모델	Test Logloss	예측 속도	Full data 가능	인코딩 여부	New level 가능	NA 핸들링
CatBoost	0.24222	약 2.8초	0	Ordered target	0	가능
DNN	0.24672	약 12초	0	Entity embedding	0	불가능
GLM	0.27022	약 0.39초	X	더미변수 + target	X	불가능
LGBM	0.30655	0.03초 (sample)	0	EFB	0	가능
RF	0.31149	약 9.9초	X	부분적으로 target	0	불가능
XGBoost	0.43930	약 4.3초	0	더미변수 + target	0	가능



모델 선정

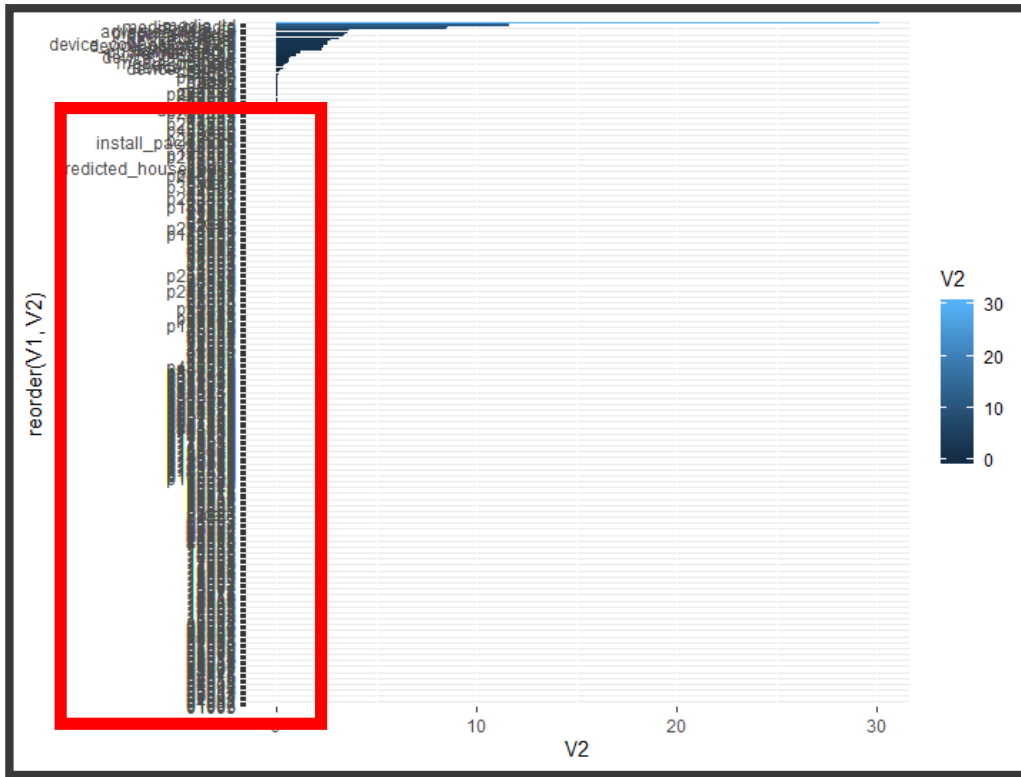


Why CatBoost?

- 1) 총 31개의 설명변수 중 1개(추정 자산 가격)를 제외한 **모든 변수가 범주형 변수**
→ CatBoost가 최적 모델!
- 2) 각 **범주형 변수의 level이 매우 큼** (최대 187만여 개, 평균 45만여 개, 중간값 174)
→ One-hot encoding은 너무 sparse
→ CatBoost의 자체적인 Encoding이 효과적임

가장 낮은 Logloss & 적당한 예측 속도

Variable selection



- Variable Importance를 보고 0인 변수는 제외하여 변수 선택을 하였다.
- 그러나 변수 선택후에 logloss가 낮아져 **기존의 모든 변수를 사용**하였다.

Parameter tuning

Best tuning

Default tuning

Max_ctr_complexity

...

1

2

3

4

5

6

...

Boosting_type

Ordered

Plain

Bootstrap_type

Bayesian

Bernoulli

MVS

Poisson

No

Sampling_frequency

PerTree

PerTreeLevel

One_hot_max_size

0

2

10

100

255

...



Parameter tuning

Leaf_estimation_method

Newton

Gradient

Exact

Leaf_estimation_backtracking

No

AnyImprovement

Armijo

Score_function

Cosine

L2

L00L2

Newton
Cosine

NewtonL2

SatL2

SolarL2

Model_shrink_rate

0

0.25

0.5

1



Parameter tuning

Feature_border_type

Median

UniformAndQuantiles

Uniform

GreedyLogSum

MaxLogSum

MinEntropy

Model_size_reg

0

0.25

0.5

Sparse_features_
conflict_fraction

0

0.25

0.5

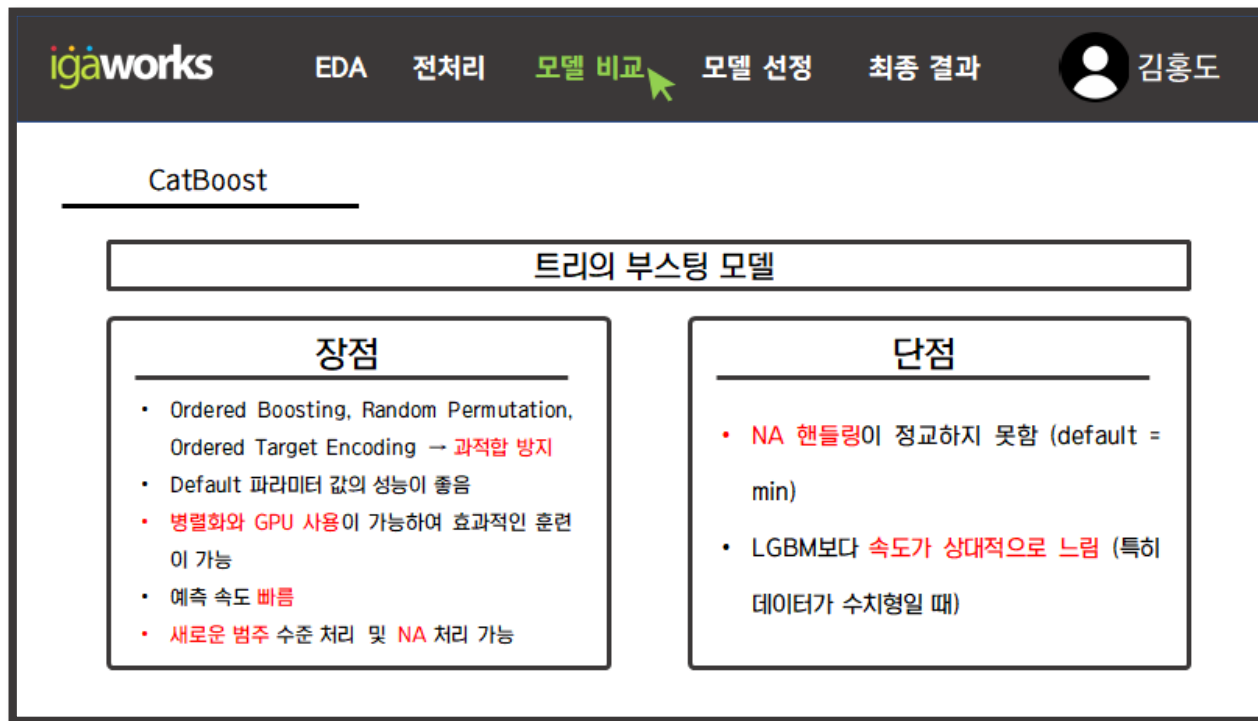
1

Counter_calc_method

SkipTest

Full

Parameter tuning



igaworks EDA 전처리 모델 비교 모델 선정 최종 결과 김홍도

CatBoost

트리의 부스팅 모델

장점	단점
<ul style="list-style-type: none">Ordered Boosting, Random Permutation, Ordered Target Encoding → 과적합 방지Default 파라미터 값의 성능이 좋음병렬화와 GPU 사용이 가능하여 효과적인 훈련이 가능예측 속도 빠름새로운 범주 수준 처리 및 NA 처리 가능	<ul style="list-style-type: none">NA 핸들링이 정교하지 못함 (default = min)LGBM보다 속도가 상대적으로 느림 (특히 데이터가 수치형일 때)

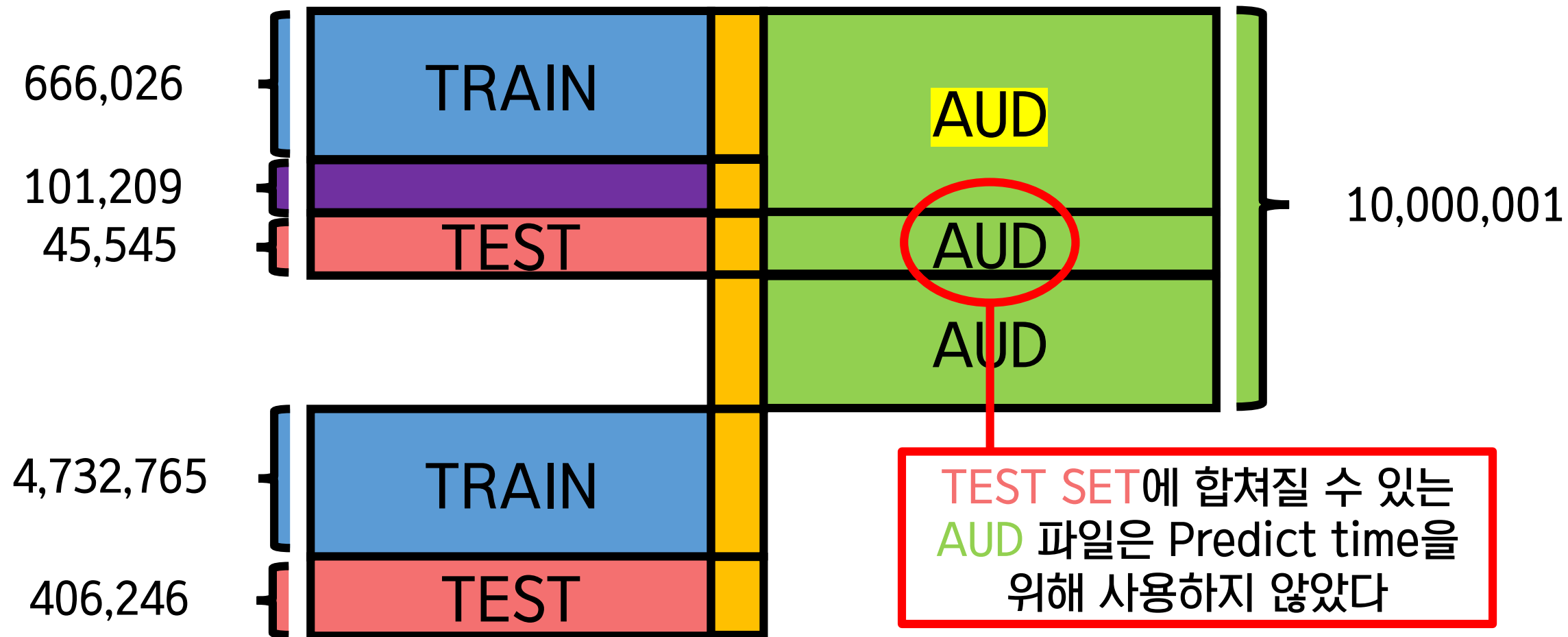
결과

- 앞서 본 것과 같이 CAT BOOST의 Default 파라미터는 좋은 성능
- 모델 최종 사이즈와 Logloss를 기준으로 파라미터 튜닝을 해도 Default와 같았다.



최종 TEST SET

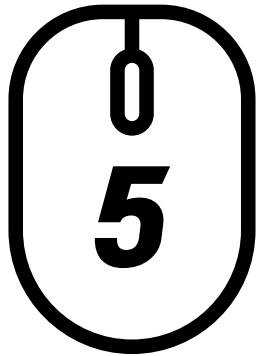
[KEY] Device_ifa



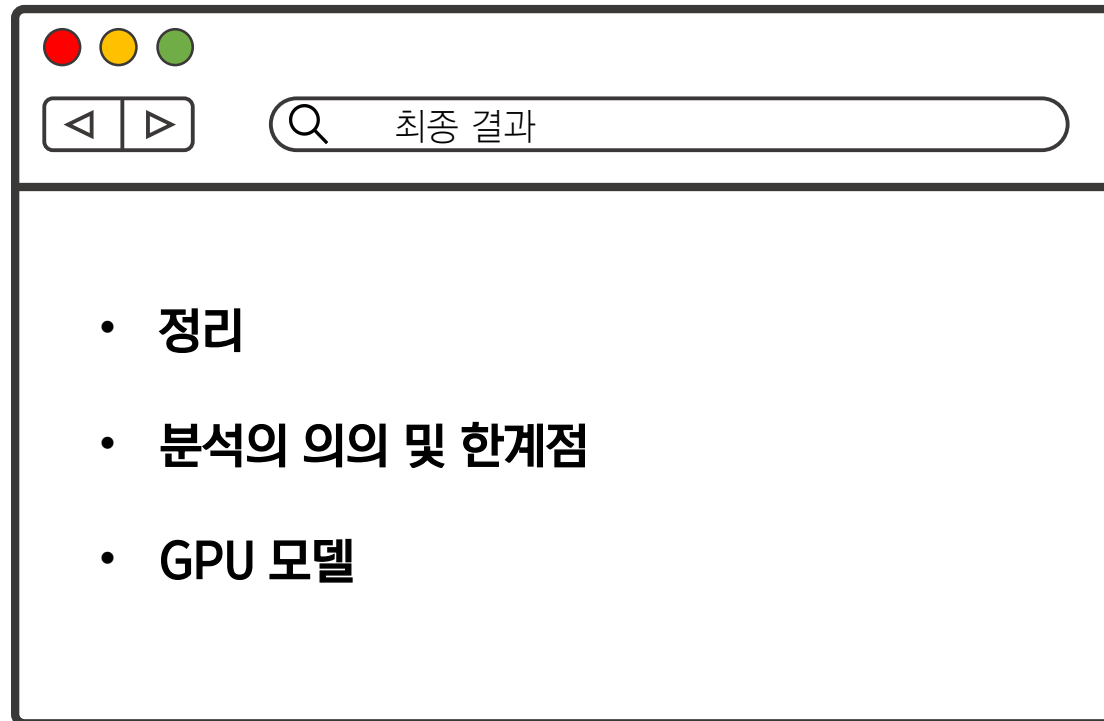
최종 TEST SET

	Logloss	Predict time
AUD TRAIN + TEST	0.24197	약 100 ~ 110초
AUD TRAIN	0.24204	약 70초 ~ 80초

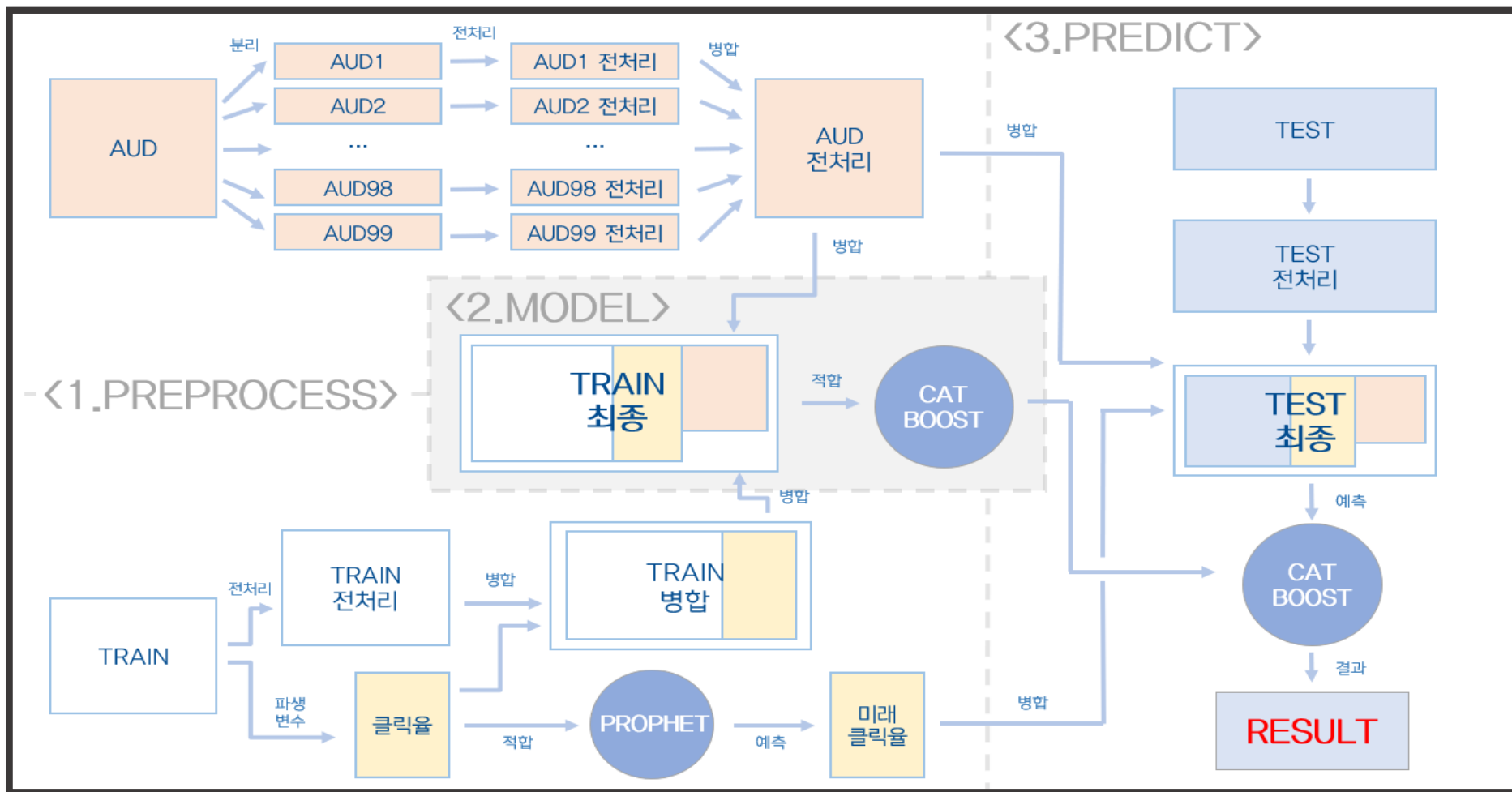
Logloss와 predict time 간의 trade off 관계 고려시,
Train data에 나타난 AUD 파일만 이용하는 것이 효율적이라 판단



최종 결과




정리





분석의 의의 및 한계점

의의

- 여러 모델을 통해 가장 나은 모델을 택할 수 있었음
- EDA에서 발견한 시간에 따른 클릭률을 반영할 수 있었음
- 가장 많은 제출횟수(119회) 

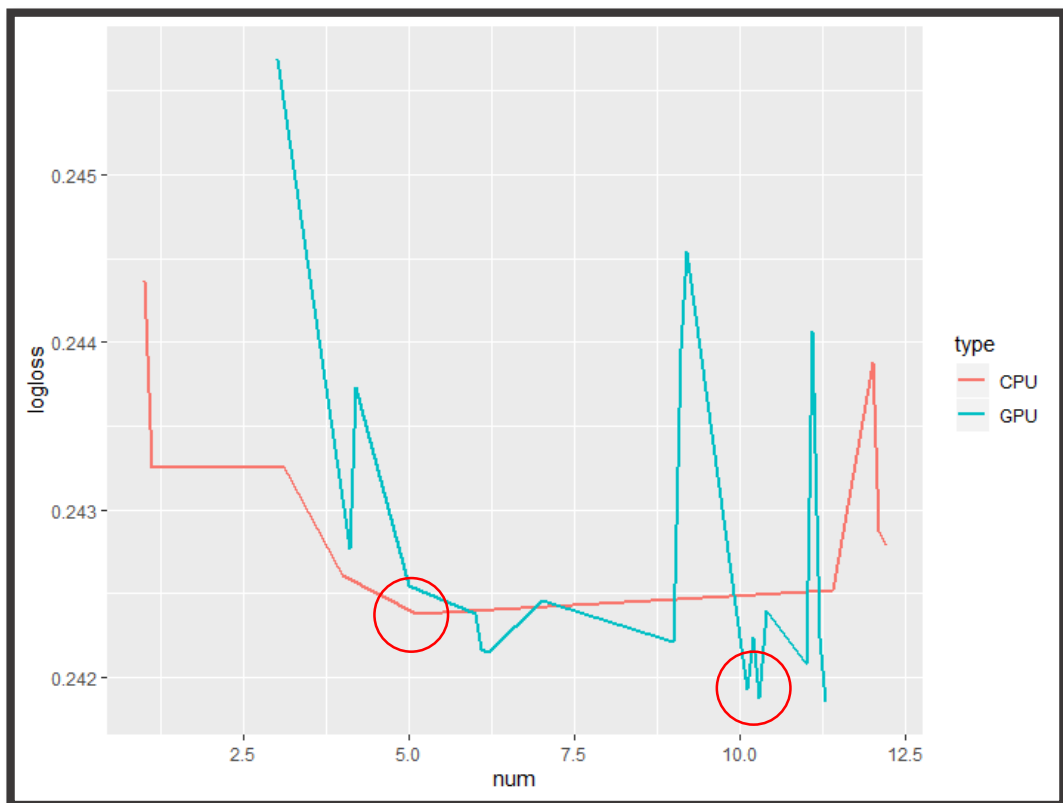
최종 모델 : CatBoost with Prophet

한계점

- 수치형 NA 처리를 못하였다
- Default 값 보다 좋은 파라미터를 찾지 못했다
- GPU보다 좋은 성능을 내지 못했다



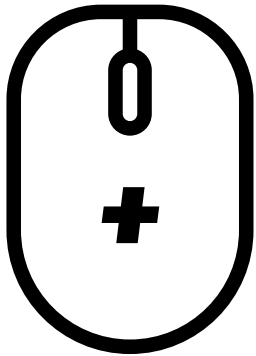
GPU 모델



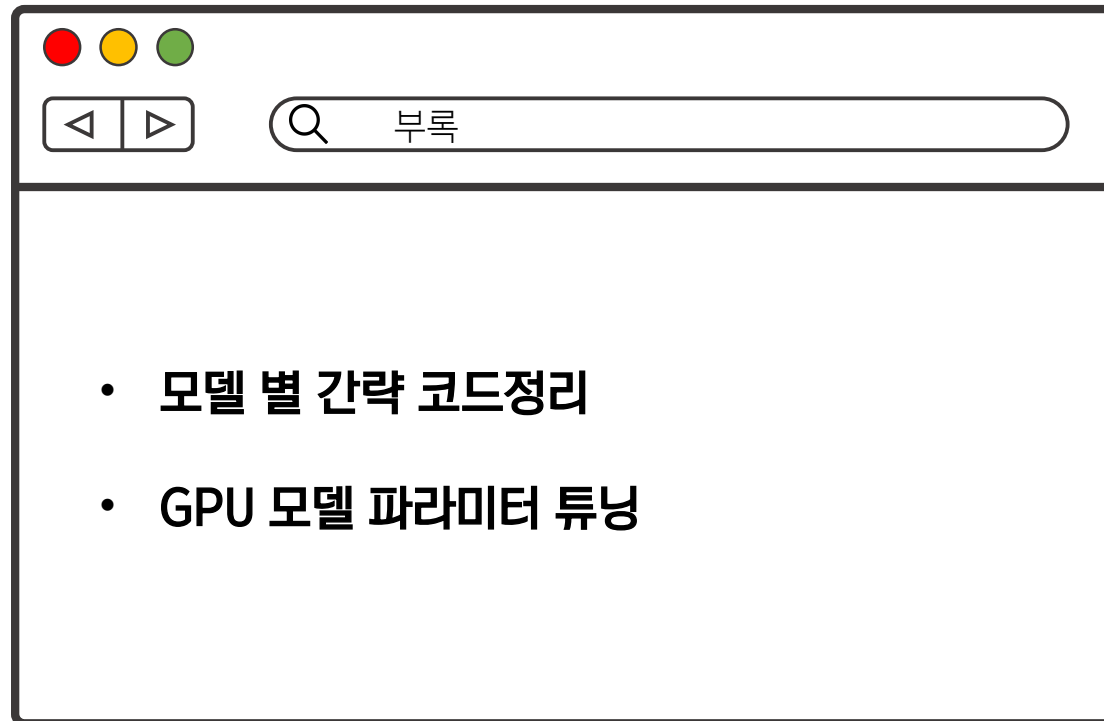
개선점

- GPU 모델도 같이 병행하였음
 - 5~8배 정도 차이나는 적합속도
 - 파라미터 튜닝을 통해 더 개선된 결과
- GPU 모델을 통하여 결과를 더 개선할 수 있음

감사합니다



부록



1. 모델별 간단 코드 정리

GLM

Input : Audience X Data_Sample

Model :

```
parglm(click ~ ., data = data_train_model, family = binomial(link = "cloglog"),  
        control = parglm.control(nthreads = (detectCores(logical = T) - 1)))
```

Logloss : 0.27002

1. 모델별 간단 코드 정리

Support Vector Machine

Input : Audience X Data_sample (Target 인코딩)

Model :

```
parallelSVM(click ~ ., data = data_train_svm_learn, kernel = "linear",  
            probability = T, cost = i, numberCores = (detectCores(logical = T) - 1))
```

Logloss : -

→ 차원이 커지면 처리할 수 없는 size

1. 모델별 간단 코드 정리

Naive Bayesian Classification

Input : Audience X Data_sample

Model :

```
model <- naiveBayes(click ~ .,  
                    data = data_train_sampled, laplace = 1)
```

Logloss : -

→ 너무 긴 predict 시간!

1. 모델별 간단 코드 정리

Random Forest

Input : Audience X Data_sample (Target 인코딩)

Model :

```
cores <- detectCores() - 1; cl <- makeCluster(cores); registerDoParallel(cl)
```

```
model <- foreach(ntree = rep(floor(500/cores), cores),  
  .combine = randomForest::combine, .multicombine = TRUE,  
  .packages = "randomForest") %dopar% { randomForest(l(as.factor(click)) ~ .,  
    data = data_train_model, do.trace = T, ntree = ntree) }  
stopCluster(cl); rm(cl, cores)
```

Logloss : 0.31149

1. 모델별 간단 코드 정리

XGBoost

Input : Audience X Data (Target 인코딩)

Model :

```
XGBClassifier(max_depth=3, min_child_weight=3,  
               colsample_bytree = 0.5, subsample = 0.5,  
               gamma = 0, reg_lambda=0.05,  
               eta = 0.01, num_round = 1000,  
               objective="binary:logistic", eval_metric='logloss', n_jobs=6)
```

Logloss : 0.43930

1. 모델별 간단 코드 정리

LightGBM

Input : Audience X Data

Model :

```
params1 <- list(learning_rate = 0.01, objective = "binary", metric = "binary_logloss",  
               bagging_fraction = 0.9, bagging_freq = 1, feature_fraction = 0.9,  
               num_leaves = 60, max_depth = -1, lambda_l1 = 0.25, lambda_l2 = 0.25)
```

```
lgb1 <- lightgbm(params = params1, data = lgb.train1, nrounds = 3000, label = train1.y,  
                early_stopping_rounds = 50L, boosting = "gbdt", num_threads = 4, tree_learner = "voting")
```

Logloss : 0.30655

1. 모델별 간단 코드 정리

CatBoost

Input : Audience 0 Data

Model :

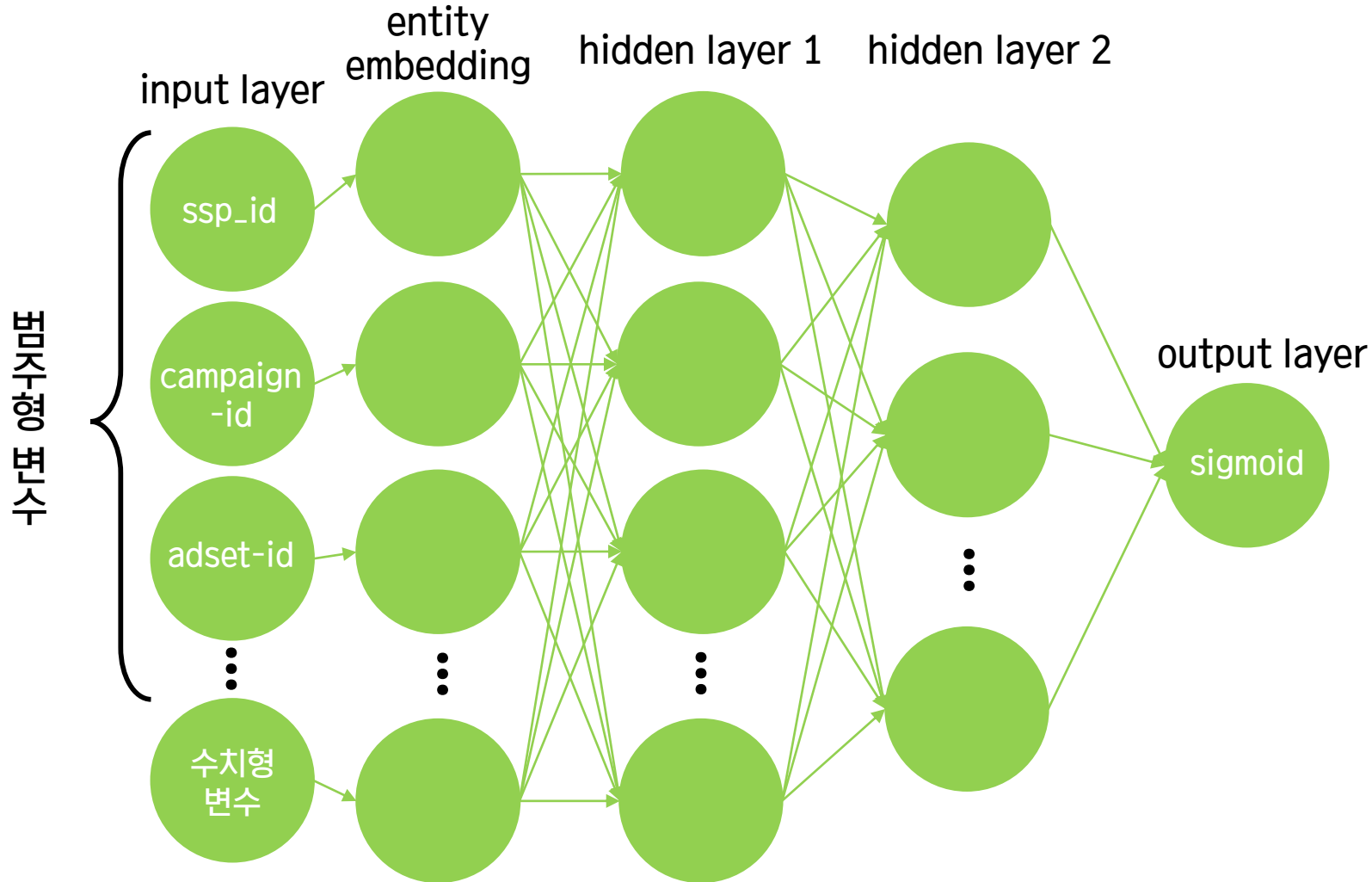
```
params <- list(random_seed = 1, loss_function = "Logloss", rsm = 1,  
  logging_level = "Verbose", depth = 6, l2_leaf_reg = 3, border_count = 254,  
  iterations = 500, learning_rate = 0.1, has_time = T, task_type = "CPU")
```

```
model <- catboost.train(learn_pool = data_train, params = params)
```

Logloss : 0.24222

1. 모델별 간단 코드 정리

DNN with entity embedding



Input : Audience X Data

Model :
DNN using Entity Embedding

- batch_size = 1024
- epochs = 30
- activation = relu
- optimizer = adam
- kerner_init = he_uniform

Output : 0.24672

2. GPU 모델 파라미터 튜닝

Parameter tuning (GPU)

Best tuning

Default
tuning

Max_ctr_complexity

...

1

2

3

4

5

6

...

Boosting_type

Ordered

Plain

Bootstrap_type

Bayesian

Bernoulli

Poisson

No

grow_policy

Depthwise

SymmetricTree

Lossguide

One_hot_max_size

0

2

10

100

255

...

2. GPU 모델 파라미터 튜닝

Parameter tuning (GPU)

Leaf_estimation_method

Newton

Gradient

Exact

Leaf_estimation_backtracking

No

AnyImprovement

Armijo

Score_function

Cosine

L2

L00L2

Newton
Cosine

NewtonL2

SatL2

SolarL2

2. GPU 모델 파라미터 튜닝

Parameter tuning (GPU)

Feature_border_type

Median	UniformAndQuantiles	Uniform
GreedyLogSum	MaxLogSum	MinEntropy

Counter_calc_method

SkipTest	Full
----------	------