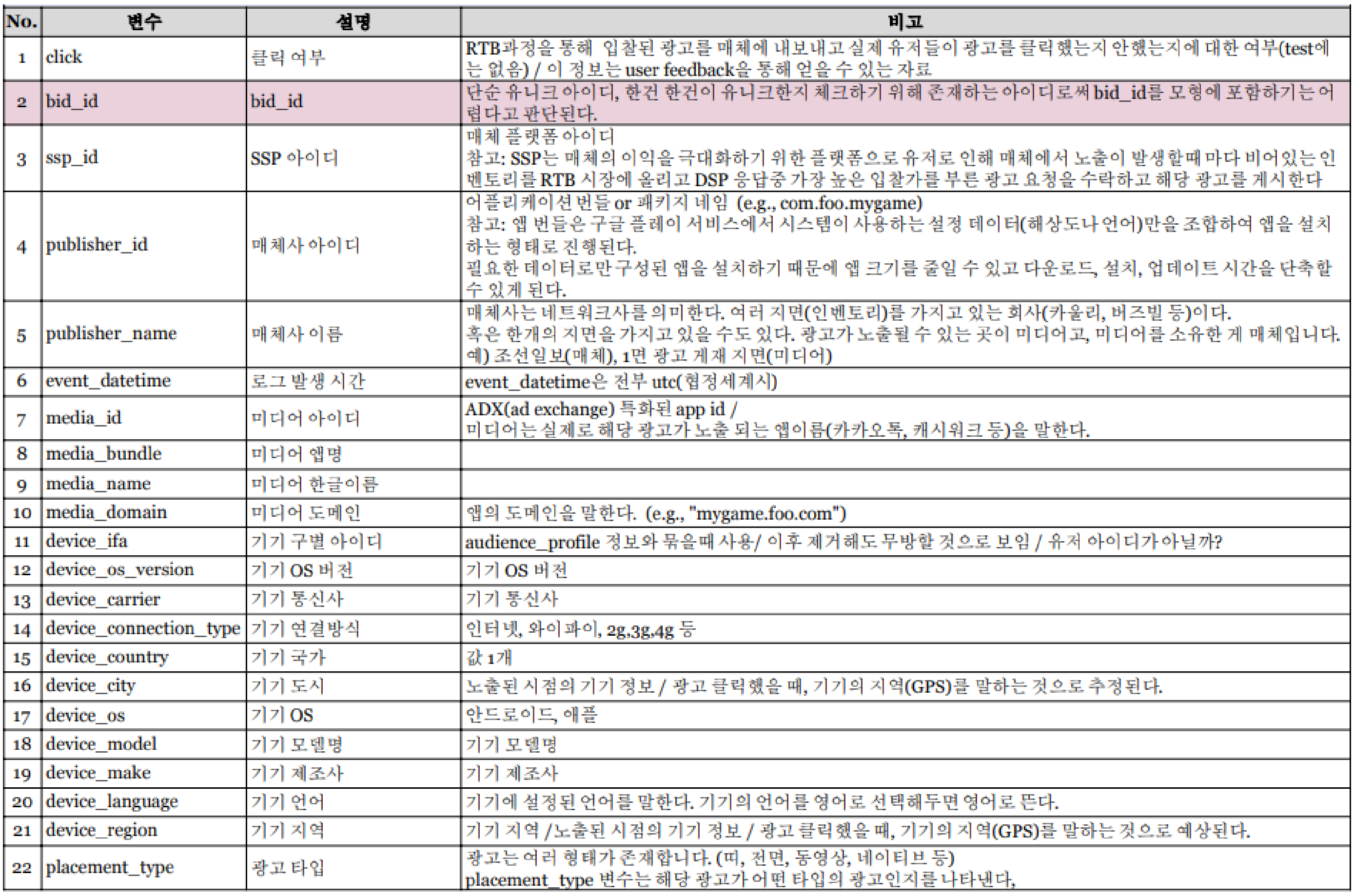
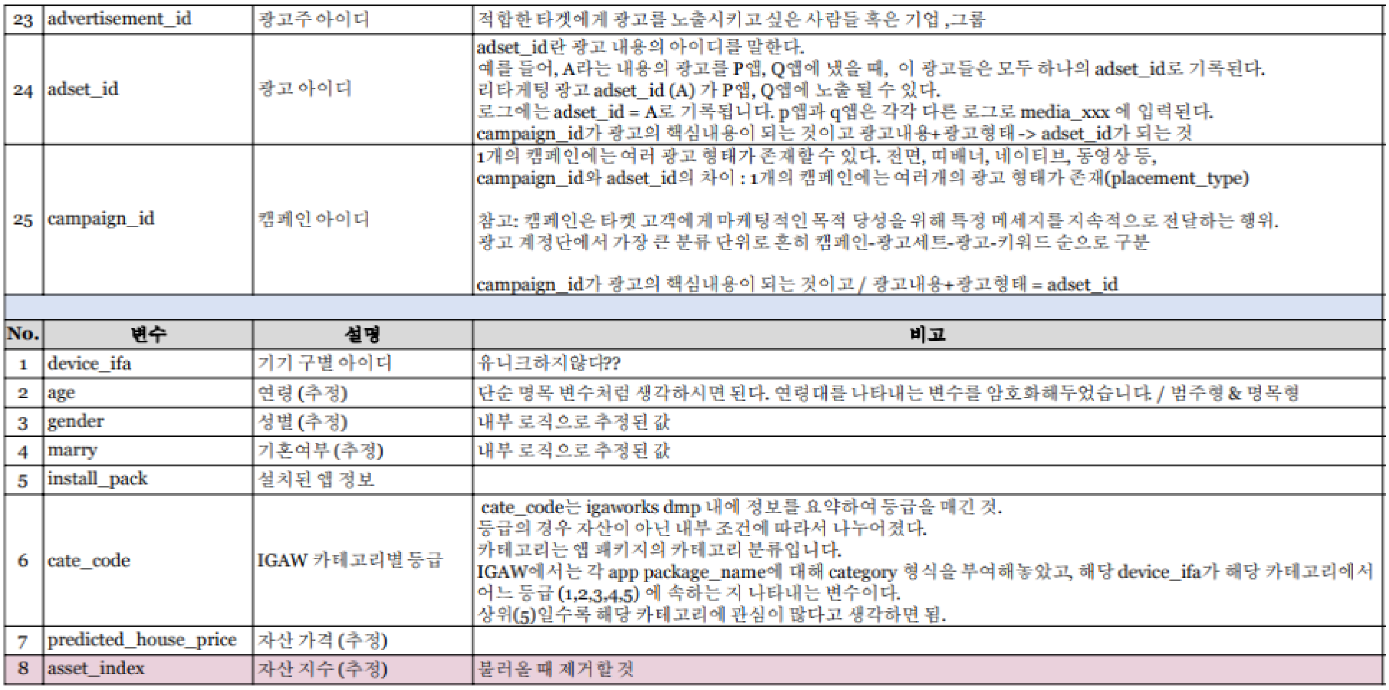
0.변수 파악





No.	변수	설명	비고
1	click	클릭 여부	RTB과정을 통해 입찰된 광고를 매체에 내보내고 실제 유저들이 광고를 클릭했는지 안했는지에 대한 여부(test에는 없음) / 이 정보는 user feedback을 통해 얻을 수 있는 자료
2	bid_id	bid_id	단순 유니크 아이디, 한건 한건이 유니크한지 체크하기 위해 존재하는 아이디로써 bid_id를 모형에 포함하기는 어렵다고 판단된다.
3	ssp_id	SSP 아이디	매체 플랫폼 아이디 참고: SSP는 매체의 이익을 극대화하기 위한 플랫폼으로 유저로 인해 매체에서 노출이 발생할때 마다 비어있는 인벤토리를 RTB 시장에 올리고 DSP 응답중 가장 높은 입찰가를 부른 광고 요청을 수락하고 해당 광고를 게시한다
4	publisher_id	매체사 아이디	어플리케이션 번들 or 패키지 네임 (e.g., com.foo.mygame) 참고: 앱 번들은 구글 플레이 서비스에서 시스템이 사용하는 설정 데이터(해상도나 언어)만을 조합하여 앱을 설치하는 형태로 진행된다. 필요한 데이터로만 구성된 앱을 설치하기 때문에 앱 크기를 줄일 수 있고 다운로드, 설치, 업데이트 시간을 단축할 수 있게 된다.
5	publisher_name	매체사 이름	매체사는 네트워크사를 의미한다. 여러 지면(인벤토리)를 가지고 있는 회사(카울리, 버즈빌 등)이다. 혹은 한개의 지면을 가지고 있을 수도 있다. 광고가 노출될 수 있는 곳이 미디어고, 미디어를 소유한 게 매체입니다. 예) 조선일보(매체), 1면 광고 게재 지면(미디어)
6	event_datetime	로그 발생 시간	event_datetime은 전부 utc(협정세계시)
7	media_id	미디어 아이디	ADX(ad exchange) 특화된 app id / 미디어는 실제로 해당 광고가 노출 되는 앱이름(카카오톡, 캐시워크 등)을 말한다.
8	media_bundle	미디어 앱명	
9	media_name	미디어 한글이름	
10	media_domain	미디어 도메인	앱의 도메인을 말한다. (e.g., "mygame.foo.com")
11	device_ifa	기기 구별 아이디	audience_profile 정보와 묶을때 사용/ 이후 제거해도 무방할 것으로 보임 / 유저 아이디가아닐까?
12	device_os_version	기기 OS 버전	기기 OS 버전
13	device_carrier	기기 통신사	기기 통신사
14	device_connection_type	기기 연결방식	인터넷, 와이파이, 2g,3g,4g 등
15	device_country	기기 국가	값 1개
16	device_city	기기 도시	노출된 시점의 기기 정보 / 광고 클릭했을 때, 기기의 지역(GPS)를 말하는 것으로 추정된다.
17	device_os	기기 OS	안드로이드, 애플
18	device_model	기기 모델명	기기 모델명
19	device_make	기기 제조사	기기 제조사
20	device_language	기기 언어	기기에 설정된 언어를 말한다. 기기의 언어를 영어로 선택해두면 영어로 뜬다.
21	device_region	기기 지역	기기 지역 /노출된 시점의 기기 정보 / 광고 클릭했을 때, 기기의 지역(GPS)를 말하는 것으로 예상된다.
22	placement_type	광고 타입	광고는 여러 형태가 존재합니다. (띠, 전면, 동영상, 네이티브 등) placement_type 변수는 해당 광고가 어떤 타입의 광고인지를 나타낸다,
23	advertisement_id	광고주 아이디	적합한 타겟에게 광고를 노출시키고 싶은 사람들 혹은 기업 ,그룹
24	adset_id	광고 아이디	adset_id란 광고 내용의 아이디를 말한다. 예를 들어, A라는 내용의 광고를 P앱, Q앱에 냈을 때, 이 광고들은 모두 하나의 adset_id로 기록된다. 리타게팅 광고 adset_id (A) 가 P앱, Q앱에 노출 될 수 있다. 로그에는 adset_id = A로 기록됩니다. p앱과 q앱은 각각 다른 로그로 media_xxx 에 입력된다. campaign_id가 광고의 핵심내용이 되는 것이고 광고내용+광고형태 -> adset_id가 되는 것
25	campaign_id	캠페인 아이디	1개의 캠페인에는 여러 광고 형태가 존재할 수 있다. 전면, 띠배너, 네이티브, 동영상 등, campaign_id와 adset_id의 차이 : 1개의 캠페인에는 여러개의 광고 형태가 존재(placement_type) 참고: 캠페인은 타켓 고객에게 마케팅적인 목적 당성을 위해 특정 메세지를 지속적으로 전달하는 행위. 광고 계정단에서 가장 큰 분류 단위로 흔히 캠페인-광고세트-광고-키워드 순으로 구분 campaign_id가 광고의 핵심내용이 되는 것이고 / 광고내용+광고형태 = adset_id

No.	변수	설명	비고
1	device_ifa	기기 구별 아이디	유니크하지않다??
2	age	연령 (추정)	단순 명목 변수처럼 생각하시면 된다. 연령대를 나타내는 변수를 암호화해두었습니다. / 범주형 & 명목형
3	gender	성별 (추정)	내부 로직으로 추정된 값
4	marry	기혼여부 (추정)	내부 로직으로 추정된 값
5	install_pack	설치된 앱 정보	
6	cate_code	IGAW 카테고리별 등급	cate_code는 igaworks dmp 내에 정보를 요약하여 등급을 매긴 것. 등급의 경우 자산이 아닌 내부 조건에 따라서 나누어졌다. 카테고리는 앱 패키지의 카테고리 분류입니다. IGAW에서는 각 app package_name에 대해 category 형식을 부여해놓았고, 해당 device_ifa가 해당 카테고리에서 어느 등급 (1,2,3,4,5) 에 속하는 지 나타내는 변수이다. 상위(5)일수록 해당 카테고리에 관심이 많다고 생각하면 됨.
7	predicted_house_price	자산 가격 (추정)	
8	asset_index	자산 지수 (추정)	불러올 때 제거할 것

1.Data Collection

1.1 Train data only

In [1]:

```
import pandas as pd
#train set 불러오기
train = pd.read_csv('train.csv')
train.head()
```

Out[1]:

	click	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_ty
0	0	2019-10-01 00:00:04.878	1b1Yz4S9wG	A6E0SZLhXP	taRA9jVfVL	GdBSlETcLy	1pcQ3RJg
1	0	2019-10-01 00:00:04.940	eCYeFjnExb	CD3hRiI3bN	jWRKzxzhyX	GlheP2trvZ	1pcQ3RJg
2	0	2019-10-01 00:00:04.963	QHcMnYqF3h	SrN77Arvqh	DW5C3As8ij	WGJnvetv2a	1pcQ3RJg
3	1	2019-10-01 00:00:05.186	p5v9KCdjS6	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kIeE1J0K
4	0	2019-10-01 00:00:05.289	aAEDD9AeIv	SrN77Arvqh	2T5sOm2MoW	UASfSkWw7S	1pcQ3RJg

5 rows × 25 columns

1.2 Data Merging(audience profile 정보가 있는 train 세트의 행들만 병합)

In [2]:

```
import gc
import pandas as pd
# audience_profile의 크기가 ram 용량에 비해 커서 작은 단위로 나누어 ram에 올리고 지우기
n=1
for chunk in pd.read_csv('audience_profile.csv',sep='delimiter', delimiter = "!@#", chunksize=5000(
    if n ==1:
        train_with_ap =pd.merge(train, chunk, how='inner', on='device_ifa')
    else:
        train_with_ap = pd.concat([train_with_ap,pd.merge(train, chunk, how='inner', on='device_ifa
    del chunk
    gc.collect()  #ram에서 삭제
    n+=1

train_with_ap.head(5)
```

```
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:5: ParserWa
rning: Falling back to the 'python' engine because the 'c' engine does not support r
egex separators (separators > 1 char and different from '₩s+' are interpreted as reg
ex); you can avoid this warning by specifying engine='python'.
  """
```

Out[2]:

	click	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_ty
0	0	2019-10-01 00:00:07.747	PLderpDXmr	Uox85xVMSC	MpoQeFR8wO	QqPbvcxynB	kIeE1J0K(
1	0	2019-10-02 06:39:49.757	yxBV351Fwe	VKAHCb2KFB	MpoQeFR8wO	pNK9aa5e24	WnrXFsYXI
2	0	2019-10-03 02:28:04.498	SodbXq6mas	VKAHCb2KFB	MpoQeFR8wO	pNK9aa5e24	WnrXFsYXI
3	1	2019-10-04 12:18:26.383	ejmRHmaCnv	Uox85xVMSC	MpoQeFR8wO	QqPbvcxynB	kIeE1J0K(
4	0	2019-10-01 00:00:11.831	MdVm8KPdsJ	VKAHCb2KFB	miiLZF3mqs	cdcVRpnJ2L	kIeE1J0K(

5 rows × 32 columns

2. 탐색적 자료분석

In [3]:

```
#데이터의 타입을 체크
train_with_ap.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2232520 entries, 0 to 112131
Data columns (total 32 columns):
click                     int64
event_datetime            object
bid_id                    object
ssp_id                    object
campaign_id               object
adset_id                  object
placement_type            object
media_id                  object
media_name                object
media_bundle              object
media_domain              object
publisher_id              object
publisher_name            object
device_ifa                object
device_os                 object
device_os_version         object
device_model              object
device_carrier            object
device_make               object
device_connection_type    object
device_language           object
device_country            object
device_region             object
device_city               object
advertisement_id          object
age                       int64
gender                    object
marry                     object
install_pack              object
cate_code                 object
predicted_house_price     float64
asset_index               float64
dtypes: float64(2), int64(2), object(28)
memory usage: 562.1+ MB
```

In [4]:

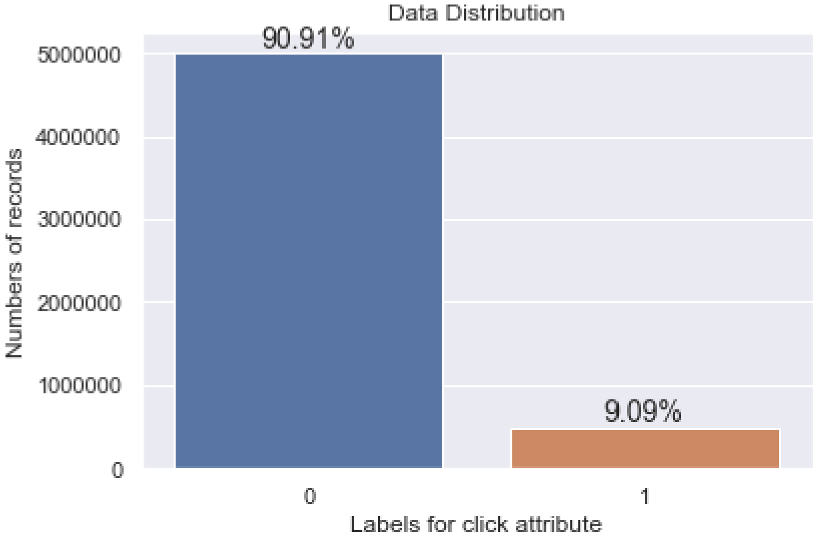
```
#train 데이터 Null 값을 체크
train.isnull().sum()
```

Out[4]:

```
click                   0
event_datetime          0
bid_id                  0
ssp_id                  0
campaign_id             0
adset_id                0
placement_type          0
media_id                0
media_name              0
media_bundle            0
media_domain            0
publisher_id            0
publisher_name          0
device_ifa              0
device_os               0
device_os_version       0
device_model            0
device_carrier          0
device_make             0
device_connection_type  0
device_language         0
device_country          0
device_region           0
device_city             0
advertisement_id        0
dtype: int64
```

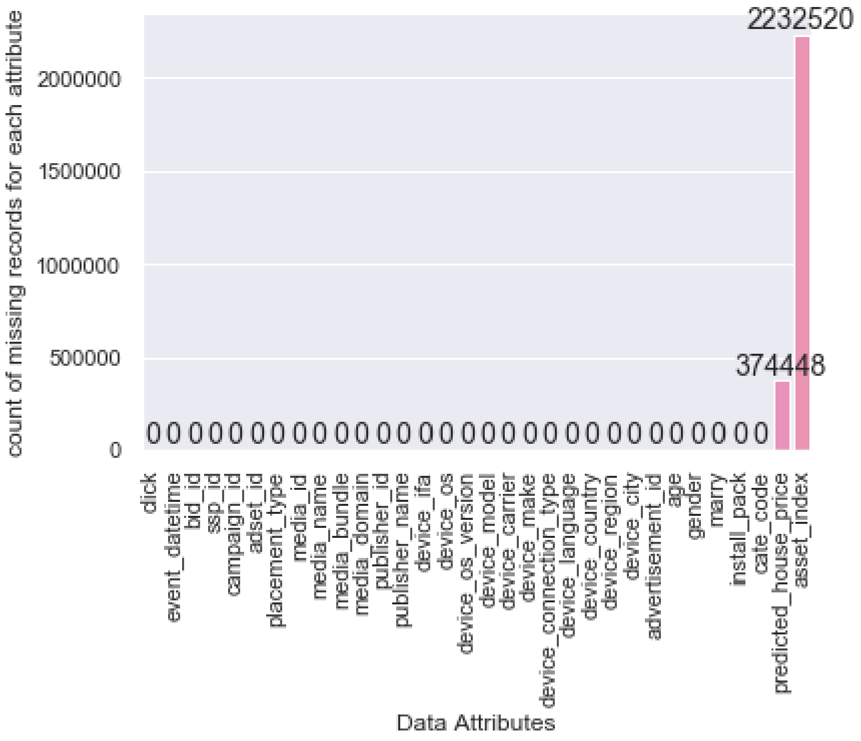
In [5]:

```
# train 데이터 종속변수 분포 체크
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set()
total_len = len(train['click'])
sns.countplot(train.click).set_title('Data Distribution')
ax = plt.gca()
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x() + p.get_width()/2.,
            height + 2,
            '{:.2f}%'.format(100*(height/total_len)),
            fontsize=14, ha='center', va='bottom')
sns.set(font_scale=1.5)
ax.set_xlabel("Labels for click attribute")
ax.set_ylabel("Numbers of records")
plt.show()
```



In [6]:

```
#train _with_ap 데이터 Null 값을 체크
x = train_with_ap.columns
y = train_with_ap.isnull().sum()
sns.set()
sns.barplot(x,y)
ax = plt.gca()
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x() + p.get_width()/2.,
            height + 2,
            int(height),
            fontsize=14, ha='center', va='bottom')
sns.set(font_scale=1.5)
ax.set_xlabel("Data Attributes")
ax.set_ylabel("count of missing records for each attribute")
plt.xticks(rotation=90) # 그래프를 보여줄때 위로 뻗을지 옆으로뻗을지 , rotaion을 0으로 하면 오른쪽
plt.show()
```



In [7]:

```
#feature 별 unique 개수
for column in train_with_ap:
    print(column,": ",len(train_with_ap[column].unique()))
```

```
click :  2
event_datetime :  2228869
bid_id :  2232520
ssp_id :  17
campaign_id :  178
adset_id :  844
placement_type :  4
media_id :  4041
media_name :  4520
media_bundle :  3881
media_domain :  135
publisher_id :  2808
publisher_name :  981
device_ifa :  767235
device_os :  1
device_os_version :  54
device_model :  778
device_carrier :  347
device_make :  181
device_connection_type :  8
device_language :  23
device_country :  1
device_region :  130
device_city :  983
advertisement_id :  28
age :  12
gender :  2
marry :  2
install_pack :  767235
cate_code :  767235
predicted_house_price :  3467
asset_index :  1
```

결론:

1.대부분의 변수가 범주형 데이터이다.(age feature포함) ->encoding 필요

2.install_pack, cate_code는 데이터를 가공해서 새로운 정보를 만들어내야한다.(가공 방안을 생각해내지 못해 해당열 제외 결정)

3.predicted_house_price에 존재하는 null 값을 채워주어야한다.

4.event_datetime은 unique값이 너무 크기 때문에 hour 정보만 추출하여야 한다.

5.unique값이 1인 device_os, device_country는 click에 영향을 미치지 않는다. 제외가능.

6.bid_id의 경우 unique값이 너무 크기 때문에 영향을 미치지 않을 것으로 보임

3.Preprocessing

predicted house price의 nan값 채우기

predicted_house_price의 결측값을 채우기 위해 같은 어디언스에 있는 정보인 age,gender,marry와의 상관관계를 파악하여 해당 정보를 바탕으로 결측값을 채우려한다.

In [8]:

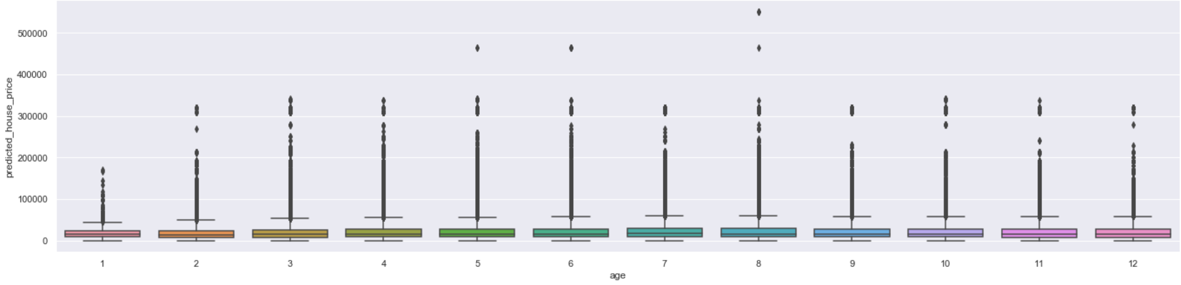
```
# age
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
sns.set()

sns.catplot(data=train_with_ap, x='age', y='predicted_house_price',kind='box',aspect=4)
```

Out[8]:

```
<seaborn.axisgrid.FacetGrid at 0x1d933ff1c08>
```

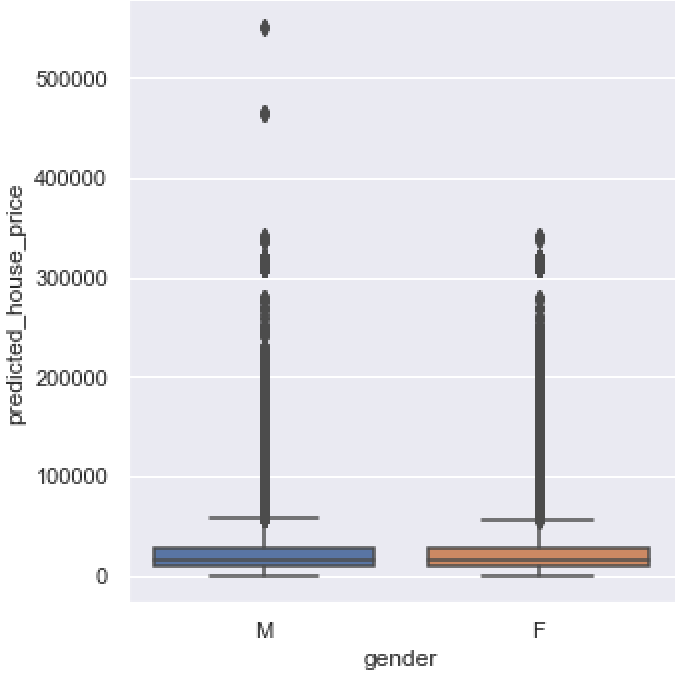


In [9]:

```
# gender
sns.catplot(data=train_with_ap, x='gender', y='predicted_house_price',kind='box')
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x1d9b77d1788>
```

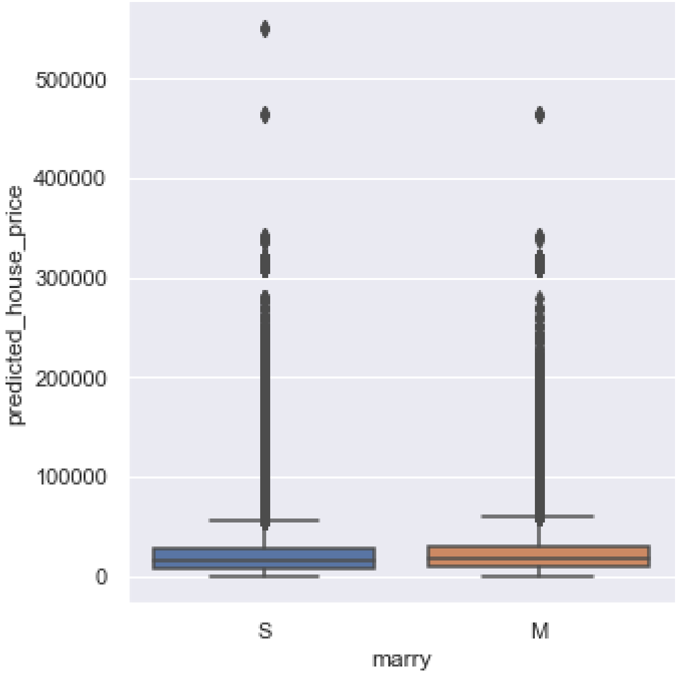


In [10]:

```
# marry
sns.catplot(data=train_with_ap, x='marry', y='predicted_house_price',kind='box')
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x1d99fd5ec48>
```



결론: age, gender, marry와 predicted_house_price 사이의 유의미한 상관관계가 거의 없어 predicted_house_priced의 평균 값으로 nan값 대체해야함

In [11]:

```
#predicted_house_price nan 값 채우기
train_with_ap['predicted_house_price']=train_with_ap['predicted_house_price'].fillna(value=train_wi
```

In [12]:

```
#시간 정보만 추출(train_with_ap)
train_with_ap['event_datetime'] = pd.to_datetime(train_with_ap['event_datetime']).dt.hour.astype(int
train_with_ap.head()
```

Out[12]:

	click	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_
0	0	0	PLderpDXmr	Uox85xVMSC	MpoQeFR8wO	QqPbvcxynB	kIeE1J
1	0	6	yxBV351Fwe	VKAHCb2KFB	MpoQeFR8wO	pNK9aa5e24	WnrXFs
2	0	2	SodbXq6mas	VKAHCb2KFB	MpoQeFR8wO	pNK9aa5e24	WnrXFs
3	1	12	ejmRHmaCnv	Uox85xVMSC	MpoQeFR8wO	QqPbvcxynB	kIeE1J
4	0	0	MdVm8KPdsJ	VKAHCb2KFB	miiLZF3mqs	cdcVRpnJ2L	kIeE1J

5 rows × 32 columns

In [13]:

```
#시간 정보만 추출(train)
train['event_datetime'] = pd.to_datetime(train['event_datetime']).dt.hour.astype(int)
train.head()
```

Out[13]:

	click	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_ty
0	0	0	1b1Yz4S9wG	A6E0SZLhXP	taRA9jVfVL	GdBSlETcLy	1pcQ3RJg
1	0	0	eCYeFjnExb	CD3hRiI3bN	jWRKzxzhyX	GlheP2trvZ	1pcQ3RJg
2	0	0	QHcMnYqF3h	SrN77Arvqh	DW5C3As8ij	WGJnvetv2a	1pcQ3RJg
3	1	0	p5v9KCdjS6	nwf1A3O5cO	qR4Xa60DLI	FiSRHSfVaf	kIeE1J0K
4	0	0	aAEDD9AeIv	SrN77Arvqh	2T5sOm2MoW	UASfSkWw7S	1pcQ3RJg

5 rows × 25 columns

In [14]:

```
# converting gender,marry feature to numeical value
train_with_ap['gender']=train_with_ap['gender'].map({'M':0,
                                                    'F':1})
train_with_ap['marry']=train_with_ap['marry'].map({'M':0,
                                                  'S':1})
train_with_ap.head()
```

Out[14]:

	click	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_ty
0	0	0	PLderpDXmr	Uox85xVMSC	MpoQeFR8wO	QqPbvcxynB	kIeE1J0K
1	0	6	yxBV351Fwe	VKAHCb2KFB	MpoQeFR8wO	pNK9aa5e24	WnrXFsYX
2	0	2	SodbXq6mas	VKAHCb2KFB	MpoQeFR8wO	pNK9aa5e24	WnrXFsYX
3	1	12	ejmRHmaCnv	Uox85xVMSC	MpoQeFR8wO	QqPbvcxynB	kIeE1J0K
4	0	0	MdVm8KPdsJ	VKAHCb2KFB	miiLZF3mqs	cdcVRpnJ2L	kIeE1J0K

5 rows × 32 columns

In [15]:

```
#encoding을 위해 age feature str으로 변환
train_with_ap['age']=train_with_ap['age'].astype('str')
```

In [16]:

```
#데이터의 타입을 체크
train_with_ap.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2232520 entries, 0 to 112131
Data columns (total 32 columns):
click                     int64
event_datetime            int32
bid_id                    object
ssp_id                    object
campaign_id               object
adset_id                  object
placement_type            object
media_id                  object
media_name                object
media_bundle              object
media_domain              object
publisher_id              object
publisher_name            object
device_ifa                object
device_os                 object
device_os_version         object
device_model              object
device_carrier            object
device_make               object
device_connection_type    object
device_language           object
device_country            object
device_region             object
device_city               object
advertisement_id          object
age                       object
gender                    int64
marry                     int64
install_pack              object
cate_code                 object
predicted_house_price     float64
asset_index               float64
dtypes: float64(2), int32(1), int64(3), object(26)
memory usage: 593.6+ MB
```

4.Feature Engineering

Dimension Reduction

1.asset_index: 처음부터 잘못 들어간 특성

2.install_pack, cate_code: 데이터 가공방안 찾지 못함

3.bid_id: unique 값이 너무 큼

4.device_os, device_country:unique 값이 하나로 클릭여부에 영향 없음

In [17]:

```
#train_with_ap set
train_with_ap = train_with_ap.drop(['install_pack','bid_id','cate_code','asset_index','device_os','c
train_with_ap.head()
```

Out[17]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	med
0	0	0	Uox85xVMSC	MpoQeFR8wO	QqPbvcxynB	kleE1J0KCa	3rKjE
1	0	6	VKAHCb2KFB	MpoQeFR8wO	pNK9aa5e24	WnrXFsYXNs	8fKHvDl
2	0	2	VKAHCb2KFB	MpoQeFR8wO	pNK9aa5e24	WnrXFsYXNs	8fKHvDl
3	1	12	Uox85xVMSC	MpoQeFR8wO	QqPbvcxynB	kleE1J0KCa	ECY7Pl
4	0	0	VKAHCb2KFB	miiLZF3mqs	cdcVRpnJ2L	kleE1J0KCa	WOU4QEFI

5 rows × 26 columns

In [18]:

```
#train set
train = train.drop(['bid_id','device_os','device_country'],axis=1)
train.head()
```

Out[18]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_
0	0	0	A6E0SZLhXP	taRA9jVfVL	GdBSlETcLy	1pcQ3RJgQt	lyDyyhXBn
1	0	0	CD3hRiI3bN	jWRKzxzhyX	GlheP2trvZ	1pcQ3RJgQt	hkFCnTpDp
2	0	0	SrN77Arvqh	DW5C3As8ij	WGJnvetv2a	1pcQ3RJgQt	hkFCnTpDp
3	1	0	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kleE1J0KCa	j7H2fWft
4	0	0	SrN77Arvqh	2T5sOm2MoW	UASfSkWw7S	1pcQ3RJgQt	hkFCnTpDp

5 rows × 22 columns

Target Encoding

In [19]:

```
#train_with_ap에 대해
from category_encoders import TargetEncoder

target_ap = train_with_ap['click']
input_values_ap = train_with_ap.drop(['click'],axis=1)
enc_with_ap = TargetEncoder().fit(input_values_ap, target_ap)
enc_with_ap
```

Out[19]:

```
TargetEncoder(cols=['ssp_id', 'campaign_id', 'adset_id', 'placement_type',
                    'media_id', 'media_name', 'media_bundle', 'media_domain',
                    'publisher_id', 'publisher_name', 'device_ifa',
                    'device_os_version', 'device_model', 'device_carrier',
                    'device_make', 'device_connection_type', 'device_language',
                    'device_region', 'device_city', 'advertisement_id', 'age'],
              drop_invariant=False, handle_missing='value',
              handle_unknown='value', min_samples_leaf=1, return_df=True,
              smoothing=1.0, verbose=0)
```

In [20]:

```
#target encoding 모델 저장
import pickle
with open("enc_preprocess_1.sav", 'wb') as file:
    pickle.dump(enc_with_ap, file)
```

In [21]:

```
numeric_train_with_ap = enc_with_ap.transform(input_values_ap)
numeric_train_with_ap
```

Out[21]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_nar
0	0	0.039832	0.146178	0.118166	0.139161	0.141414	0.1390
1	6	0.048989	0.146178	0.011869	0.052756	0.000272	0.0238
2	2	0.048989	0.146178	0.011869	0.052756	0.000272	0.0238
3	12	0.039832	0.146178	0.118166	0.139161	0.130435	0.1304
4	0	0.048989	0.020312	0.021305	0.139161	0.019410	0.0238
...	...	...	...	...	...	...	
112127	23	0.189793	0.047038	0.010073	0.021026	0.086199	0.0861
112128	23	0.036740	0.110731	0.062044	0.052756	0.016541	0.0238
112129	23	0.039832	0.110171	0.085671	0.139161	0.059028	0.0590
112130	23	0.143941	0.056158	0.097210	0.139161	0.081451	0.0814
112131	0	0.036740	0.047038	0.089109	0.052756	0.016541	0.0238

2232520 rows × 25 columns

In [22]:

```
# numeric_train_with_ap에 click column 추가
numeric_train_with_ap['click']=train_with_ap['click']
#column 순서 조정
cols = list(numeric_train_with_ap)
cols = [cols[-1]] + cols[:-1]
numeric_train_with_ap = numeric_train_with_ap[cols]

numeric_train_with_ap
```

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_name	mec
0	0	0	0.039832	0.146178	0.118166	0.139161	0.141414	0.139073	
1	0	6	0.048989	0.146178	0.011869	0.052756	0.000272	0.023879	
2	0	2	0.048989	0.146178	0.011869	0.052756	0.000272	0.023879	
3	1	12	0.039832	0.146178	0.118166	0.139161	0.130435	0.130435	
4	0	0	0.048989	0.020312	0.021305	0.139161	0.019410	0.023879	
...	...	...	...	...	...	...	...	...	
112127	0	23	0.189793	0.047038	0.010073	0.021026	0.086199	0.086199	
112128	0	23	0.036740	0.110731	0.062044	0.052756	0.016541	0.023879	
112129	0	23	0.039832	0.110171	0.085671	0.139161	0.059028	0.059028	
112130	0	23	0.143941	0.056158	0.097210	0.139161	0.081451	0.081451	
112131	0	0	0.036740	0.047038	0.089109	0.052756	0.016541	0.023879	

In [23]:

```
#train에 대해
from category_encoders import TargetEncoder

target = train['click']
input_values = train.drop(['click'],axis=1)

enc = TargetEncoder().fit(input_values, target)
enc
```

Out[23]:

```
TargetEncoder(cols=['ssp_id', 'campaign_id', 'adset_id', 'placement_type',
                    'media_id', 'media_name', 'media_bundle', 'media_domain',
                    'publisher_id', 'publisher_name', 'device_ifa',
                    'device_os_version', 'device_model', 'device_carrier',
                    'device_make', 'device_connection_type', 'device_language',
                    'device_region', 'device_city', 'advertisement_id'],
              drop_invariant=False, handle_missing='value',
              handle_unknown='value', min_samples_leaf=1, return_df=True,
              smoothing=1.0, verbose=0)
```

In [24]:

```
#target encoding 모델 저장
import pickle

with open("enc_preprocess_2.sav", 'wb') as file:
    pickle.dump(enc, file)
```

In [25]:

```
numeric_train = enc.transform(input_values)
numeric_train
```

Out[25]:

```
         event_datetime    ssp_id  campaign_id  adset_id  placement_type  media_id  media_na
      0               0  0.087898     0.130311  0.028476        0.022644  0.002079     0.002
      1               0  0.036473     0.030543  0.016824        0.022644  0.018766     0.024
      2               0  0.018592     0.023756  0.023756        0.022644  0.018766     0.024
      3               0  0.193911     0.287327  0.287327        0.143059  0.283167     0.279
      4               0  0.018592     0.005527  0.005605        0.022644  0.018766     0.024
    ...             ...       ...          ...       ...             ...       ...
5499995               0  0.018592     0.056631  0.007687        0.022644  0.018766     0.024
5499996               0  0.144511     0.209155  0.209285        0.143059  0.273958     0.279
5499997               0  0.018592     0.056631  0.005855        0.022644  0.018766     0.024
5499998               0  0.018592     0.005527  0.005889        0.022644  0.018766     0.024
5499999               0  0.193911     0.056129  0.085316        0.143059  0.020265     0.020

5500000 rows × 21 columns
```

In [26]:

```
# numeric_train에 click column 추가
numeric_train['click']=train['click']
#column 순서 조정
cols = list(numeric_train)
cols = [cols[-1]] + cols[:-1]
numeric_train = numeric_train[cols]

numeric_train
```

Out[26]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_name	me
0	0	0	0.087898	0.130311	0.028476	0.022644	0.002079	0.002079	
1	0	0	0.036473	0.030543	0.016824	0.022644	0.018766	0.024986	
2	0	0	0.018592	0.023756	0.023756	0.022644	0.018766	0.024986	
3	1	0	0.193911	0.287327	0.287327	0.143059	0.283167	0.279698	
4	0	0	0.018592	0.005527	0.005605	0.022644	0.018766	0.024986	
...	...	...	...	...	...	...	...	...	
5499995	0	0	0.018592	0.056631	0.007687	0.022644	0.018766	0.024986	

outlier 제거

numerical variable인 event_datetime, predicted_house_price 에 대해 이상치 제거 작업을 수행

In [27]:

```
numeric_train_with_ap
```

Out[27]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	r
0	0	0	0.039832	0.146178	0.118166	0.139161	0.141414	
1	0	6	0.048989	0.146178	0.011869	0.052756	0.000272	
2	0	2	0.048989	0.146178	0.011869	0.052756	0.000272	
3	1	12	0.039832	0.146178	0.118166	0.139161	0.130435	
4	0	0	0.048989	0.020312	0.021305	0.139161	0.019410	
...	...	...	...	...	...	...	...	
112127	0	23	0.189793	0.047038	0.010073	0.021026	0.086199	
112128	0	23	0.036740	0.110731	0.062044	0.052756	0.016541	
112129	0	23	0.039832	0.110171	0.085671	0.139161	0.059028	
112130	0	23	0.143941	0.056158	0.097210	0.139161	0.081451	
112131	0	0	0.036740	0.047038	0.089109	0.052756	0.016541	

2232520 rows × 26 columns

In [28]:

```
def remove_outlier_test(d_cp, column):
    fraud_column_data = d_cp[d_cp['click']==0][column]
    quan_25 = np.percentile(fraud_column_data.values,25)
    quan_75 = np.percentile(fraud_column_data.values,75)

    iqr = quan_75 - quan_25
    iqr = iqr*1.5
    lowest = quan_75 - iqr
    highest = quan_75 + iqr
    outlier_index = fraud_column_data[(fraud_column_data<lowest)| (fraud_column_data> highest)].ind
    print(len(outlier_index))
    d_cp.drop(outlier_index, axis = 0, inplace = True)
    print(d_cp.shape)
    return d_cp
```

In [29]:

```
import numpy as np
numeric_train_with_ap = remove_outlier_test(numeric_train_with_ap, 'event_datetime')
numeric_train_with_ap = remove_outlier_test(numeric_train_with_ap, 'predicted_house_price')
```

```
0
(2232520, 26)
191604
(377682, 26)
```

In [30]:

```
numeric_train_with_ap
```

Out[30]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	mec
2	0	2	0.048989	0.146178	0.011869	0.052756	0.000272	
12	0	2	0.143941	0.093064	0.104093	0.139161	0.105418	
13	0	15	0.189793	0.096605	0.123112	0.139161	0.119479	
14	0	23	0.189793	0.096605	0.123112	0.139161	0.119479	
15	0	22	0.143941	0.096605	0.123112	0.139161	0.105418	
...	...	...	...	...	...	...	...	
112117	0	23	0.143941	0.096605	0.123112	0.139161	0.273404	
112118	0	23	0.039832	0.057104	0.115930	0.139161	0.032964	
112123	0	23	0.015880	0.005032	0.005487	0.021026	0.016541	
112124	0	23	0.143941	0.036550	0.064146	0.139161	0.045987	
112128	0	23	0.036740	0.110731	0.062044	0.052756	0.016541	

377682 rows × 26 columns

In [31]:

```
numeric_train
```

Out[31]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	me
0	0	0	0.087898	0.130311	0.028476	0.022644	0.002079	
1	0	0	0.036473	0.030543	0.016824	0.022644	0.018766	
2	0	0	0.018592	0.023756	0.023756	0.022644	0.018766	
3	1	0	0.193911	0.287327	0.287327	0.143059	0.283167	
4	0	0	0.018592	0.005527	0.005605	0.022644	0.018766	
...	...	...	...	...	...	...	...	
5499995	0	0	0.018592	0.056631	0.007687	0.022644	0.018766	
5499996	0	0	0.144511	0.209155	0.209285	0.143059	0.273958	
5499997	0	0	0.018592	0.056631	0.005855	0.022644	0.018766	
5499998	0	0	0.018592	0.005527	0.005889	0.022644	0.018766	
5499999	0	0	0.193911	0.056129	0.085316	0.143059	0.020265	

5500000 rows × 22 columns

In [32]:

```
def remove_outlier_test(d_cp2, column):
    fraud_column_data = d_cp2[d_cp2['click']==0][column]
    quan_25 = np.percentile(fraud_column_data.values,25)
    quan_75 = np.percentile(fraud_column_data.values,75)

    iqr = quan_75 - quan_25
    iqr = iqr*1.5
    lowest = quan_75 - iqr
    highest = quan_75 + iqr
    outlier_index = fraud_column_data[(fraud_column_data<lowest)| (fraud_column_data> highest)].ind
    print(len(outlier_index))
    d_cp2.drop(outlier_index, axis = 0, inplace = True)
    print(d_cp2.shape)
    return d_cp2
```

In [33]:

```
numeric_train = remove_outlier_test(numeric_train, 'event_datetime')
```

```
0
(5500000, 22)
```

In [34]:

```
numeric_train
```

Out[34]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	me
0	0	0	0.087898	0.130311	0.028476	0.022644	0.002079	
1	0	0	0.036473	0.030543	0.016824	0.022644	0.018766	
2	0	0	0.018592	0.023756	0.023756	0.022644	0.018766	
3	1	0	0.193911	0.287327	0.287327	0.143059	0.283167	
4	0	0	0.018592	0.005527	0.005605	0.022644	0.018766	
...	...	...	...	...	...	...	...	
5499995	0	0	0.018592	0.056631	0.007687	0.022644	0.018766	
5499996	0	0	0.144511	0.209155	0.209285	0.143059	0.273958	
5499997	0	0	0.018592	0.056631	0.005855	0.022644	0.018766	
5499998	0	0	0.018592	0.005527	0.005889	0.022644	0.018766	
5499999	0	0	0.193911	0.056129	0.085316	0.143059	0.020265	

5500000 rows × 22 columns

Scaling

In [35]:

```
#ap 포함하는 파일
from sklearn.preprocessing import MinMaxScaler
Scaler = MinMaxScaler()
numeric_train_with_ap[['event_datetime','predicted_house_price']] = Scaler.fit_transform(numeric_tra
numeric_train_with_ap
```

Out[35]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	mec
2	0	0.086957	0.048989	0.146178	0.011869	0.052756	0.000272	
12	0	0.086957	0.143941	0.093064	0.104093	0.139161	0.105418	
13	0	0.652174	0.189793	0.096605	0.123112	0.139161	0.119479	
14	0	1.000000	0.189793	0.096605	0.123112	0.139161	0.119479	
15	0	0.956522	0.143941	0.096605	0.123112	0.139161	0.105418	
...	...	...	...	...	...	...	...	
112117	0	1.000000	0.143941	0.096605	0.123112	0.139161	0.273404	
112118	0	1.000000	0.039832	0.057104	0.115930	0.139161	0.032964	
112123	0	1.000000	0.015880	0.005032	0.005487	0.021026	0.016541	
112124	0	1.000000	0.143941	0.036550	0.064146	0.139161	0.045987	
112128	0	1.000000	0.036740	0.110731	0.062044	0.052756	0.016541	

377682 rows × 26 columns

In [37]:

```
#ap 포함하는 파일
from sklearn.preprocessing import MinMaxScaler
Scaler = MinMaxScaler()
numeric_train[['event_datetime']] = Scaler.fit_transform(numeric_train[['event_datetime']])
numeric_train
```

Out[37]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	me
0	0	0.0	0.087898	0.130311	0.028476	0.022644	0.002079	
1	0	0.0	0.036473	0.030543	0.016824	0.022644	0.018766	
2	0	0.0	0.018592	0.023756	0.023756	0.022644	0.018766	
3	1	0.0	0.193911	0.287327	0.287327	0.143059	0.283167	
4	0	0.0	0.018592	0.005527	0.005605	0.022644	0.018766	
...	...	...	...	...	...	...	...	
5499995	0	0.0	0.018592	0.056631	0.007687	0.022644	0.018766	
5499996	0	0.0	0.144511	0.209155	0.209285	0.143059	0.273958	
5499997	0	0.0	0.018592	0.056631	0.005855	0.022644	0.018766	
5499998	0	0.0	0.018592	0.005527	0.005889	0.022644	0.018766	
5499999	0	0.0	0.193911	0.056129	0.085316	0.143059	0.020265	

5500000 rows × 22 columns

5.Preprocessed Data Saving

In [40]:

```
numeric_train_with_ap.to_csv('train_preprocess_1.csv',index=False)
```

In [41]:

```
numeric_train.to_csv('train_preprocess_2.csv',index=False)
```

모델 선정기준

주어진 문제는 클릭여부를 예측하는 분류문제이다. 분류(Classification)은 학습데이터로 주어진 데이터의 피처와 레이블값을 머신러닝 알고리즘으로 학습해 모델을 생성하고, 생성된 모델에 새로운 데이터 값이 주어졌을 때 미지의 레이블 값을 예측하는 것이다. 즉, 기존 데이터가 어떤 레이블에 속하는지 패턴을 알고리즘으로 인지한 뒤에 새롭게 관측된 데이터에 대한 레이블을 판별하는 것이다. 회귀모델은 연속적인 값을 예측하는 방법이므로 적합하지않다. 분류 모델의 종류는 다음과 같다.

1.k최근접 이웃(k-nearest neighbor,KNN)

2.로지스틱 회귀(logistic regression)

3.결정 트리(Decision Tree)

4.나이브 베이즈(Naive Bayes)

5.서포트 벡터 머신(Support Vector Machine)

6.앙상블(Ensemble)

```
    -6.1. Boosting    6.1.1. 그레이디언트부스트(Gradient Boosting),
                      6.1.2. 에타부스트(AdaBoost)
                      6.1.3. XgBoost, 6
                      6.1.4. LightGBM

    -6.2. Bagging     6.2.1. 랜덤포레스트(Random Forest)
```

1.k 최근접 이웃(k-nearst neighbor, KNN) 일반적으로 k 최근접 이웃 알고리즘에서 출력되는 예측은 이웃과 동일한 경향을 따른다. k는 고려할 이웃의 수이다. k=3이면, 예측 출력이 이루어지는 동안에 세 개의 가장 가까운 이웃점을 검사하고, 하나의 이웃이 X 범주에 속하고 두 이웃이 Y 범주에 속한다면, 가장 가까운 점의 대다수가 Y 축에 속하기 때문에 예측된 레이블은 Y가된다.

2.로지스틱 회귀(logistic regression) 가장 널리 사용되고 가장 오래된 알고리즘 중 하나이다. 이 알고리즘은 포적 레이블을 예측하기 위해 signod 및 기타 비선형 함수를 사용해 표적 변수에 대한 확률을 생성한다

3.결정트리(Decision Tree) 앙상블의 기본 알고리즘 으로 일반적으로 사용함. 매우 쉽고 유연하게 적용될 수 있는 알고리즘 이다. 또한 데이터 scaling이나 nomalization 등의 사전 가공의 영향이 매우 적다. 하지만 예측 성능을 향상시키기 위해 복잡한 규칙 구조를 가져야 하며, 이로인한 과적합이 발생해 반대로 예측 성능이 저하될 수도 있다는 단점이 있다. 하지만 이러한 단점이 앙상블 기법에서는 오히려 장점으로 작용한다. 왜냐하면 앙상블은 매우 많은 여러개의 약한 학습기(즉, 예측 성능이 상대적으로 떨어지는 학습 알고리즘)를 결합해 확률적 보완과 오류가 발생한 부분에 대한 가중치를 계속 업데이트하면서 예측 성능을 향상시키는데, 결정 트리가 좋은 약한 학습기가 되기 때문이다. ML 알고리즘 중 직관적으로 이해하기 쉬운 알고리즘이다. 데이터에 있는 규칙을 학습하여 자동으로 찾아낸 Tree 기반의 분류 규칙을 만드는 것이다. if/else를 자동으로 찾아내 예측을 위한 규칙을 만드는 알고리즘으로 이해하면 된다. 따라서 데이터의 어떤 기준을 바탕으로 규칙을 만들어야 가장 효율적인 분류가 될 것인가가 알고리즘의 성능을 크게 좌우한다.

4.나이브 베이즈(Naive Bayes) 나이브 베이즈는 스팸 메일 필터, 텍스트 분류, 감정 분석, 추천 시스템 등에 광범위하게 활용되는 분류 기법이다. 베이즈 분류모델을 기반으로 한 모델로 조건부확률을 통해 분류를 진행한다. 하나의 변수에 한정된 베이즈 모델과 다르게 나이브 베이즈 다수의 변수들이 서로 독립적임을 가정하고 계산을 수행한다. 간단하고, 빠르며, 정확하고 computation cost가 작다.큰 데이터셋에 적합합니다. 연속형보다 이산형 데이터에서 성능이 좋습니다. Multiple class 예측을 위해서도 사용할 수 있습니다.

5.서포트 벡터 머신(Support Vector Machine) 서포트 벡터 머신은 인공지능의 기계학습 분야 중 하나로, 패턴인식, 자료분석을 위한 지도학습 모델이다. 2개의 범주를 분류하는 이진 분류기이다. 주로 분류와 회귀 분석을 위해 사용되며, SVM 알고리즘은 주어진 데이터 집합을 바탕으로 하여 새로운 데이터가 어느 카테고리에 속할 것인지 판단하는 비확률적 이진 선형 분류 모델을 만들게 된다.

6.앙상블 분류에서 가장 각광 받는 방법중 하나. 이미지, 영상, 음성, NLP 영역에서는 tslrduakd에 기반한 딥러닝이 머신러닝계를 선도하고 있지만, 이를 제외한 정형 데이터의 예측 분석 영역에서는 앙상블이 매우 높은 예측 성능으로 인해 많이 사용된다. 앙상블은 서로 다른/또는 같은 알고리즘을 단순히 결합한 형태도 있으나 대부분은 동리한 알고리즘을 결합한다. 일반 적으로 배깅(Bagging)과 부스팅(Boosting)방식으로 나뉜다. 근래의 앙상블 방법은 Boosting방식으로 지속해서 발전하고 있다.

6.1.1.그레이디언트 부스팅(GradientBoosting)=경사도 증폭 앙상블 중 Boosting방식으로, 뛰어난 예측 성능을 가지고있지만, 수행 시간이 너무 오래걸리는 단점으로 인해 최적화 모델 튜닝이 어렵다. 이 알고리즘에서는 기본 회귀 알고리즘을 사용해 모델을 학습한다. 훈련을 마친 후에는 error rate를 계산할 뿐만 아니라 알고리즘이 제대로 수행되지 않는 데이터 점을 찾고 다음 반복에서는 error를 도입한 데이터 점을 취해 더 나은 예측을 위해 모델을 다시 테스트 하게 된다. 알고리즘은 새로 생성된 모델 뿐만 아니라 이미 생성된 모델을 사용해 데이터 점의 값을 예측한다

6.1.2에이다부스트(AdaBoost)=어댑티브 부스팅(adaptive boosting)=적응형 증폭 부스팅(boosting, 증폭)은 weak classifier 여러개를 사용해 strong classifier를 작성하는 앙상블 방식이다. 에이다 부스트는 알고리즘 강화에 이진 분류 문제에 좋은 결과를 제공한다.이 특정 알고리즘의 반복과정은 N회이다. 첫 번째 반복과정에서는 훈련 데이터셋에서 마구잡이로 데이터 점을 가져와 모델을 작성하는 것으로 시작한다.각 반복과정이 지난 후에 알고리즘은 분류기의 성능이 좋지 않은 데이터 점을 확인한다. error rate를 기반으로 알고리즘에 의해 데이터 점이 식별되면 가중치 분포가 갱신된다. 따라서 반복에서는 알고리즘이 이전에 잘못 분류된 데이터 점을 선택하고 이를 분류하는 방법을 배울 수 있는 기호가 더 많아진다. 이 과정이 주어진 반복 횟수 동안 계속 실행된다.

6.1.3.XgBoost(eXtra Gradient Boost) 그레이디언트 부스팅 방식의 예측 성능을 한단계 발전시키면서도 수행 시간을 단축시킨 알고리즘으로 정형 데이터의 분류 영역에서 가장 활용도가 높은 알고리즘

6.1.4.LightGBM 그레이디언트 부스팅 방식의 예측 성능을 한단계 발전시키면서도 수행 시간을 단축시킨 알고리즘으로 정형 데이터의 분류 영역에서 가장 활용도가 높은 알고리즘

6.2.랜덤포레스트(RandomForest) Bagging방식의 대표적인 랜덤 포레스트는 뛰어난 예측 성능, 상대적으로 빠른 수행시간, 유연성 등으로 많이 사용되는 알고리즘 이다. 결정트리(Decision Trees)의 수를 생성하고 투표 방식을 사용해 표적 레이블을 예측한다. 이 알고리즘에는 여러 가지 결정 트리가 생성되어 숲을 이루므로 랜덤 포레스트 라고한다.

시간관계상 모든 모델을 테스트 할 수 없어 보통 가장 좋은 성능을 내는 앙상블 모델 5개(Gradient Boosting,AdaBoost, XgBoost, LightGBM, Random Forest)와 선형 분류 모델 중 로지스틱 회귀 모델까지 총 6종의 모델을 생성하였다.

1.Data Reading

In [1]:

```
import pandas as pd

train_with_ap = pd.read_csv('train_preprocess_1.csv')
train_with_ap.head()
```

Out[1]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_na
0	0	0.086957	0.048989	0.146178	0.011869	0.052756	0.000272	0.023
1	0	0.086957	0.143941	0.093064	0.104093	0.139161	0.105418	0.105
2	0	0.652174	0.189793	0.096605	0.123112	0.139161	0.119479	0.119
3	0	1.000000	0.189793	0.096605	0.123112	0.139161	0.119479	0.119
4	0	0.956522	0.143941	0.096605	0.123112	0.139161	0.105418	0.105

5 rows × 26 columns

In [2]:

```
train = pd.read_csv('train_preprocess_2.csv')
train.head()
```

Out[2]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_na
0	0	0.0	0.087898	0.130311	0.028476	0.022644	0.002079	0.002
1	0	0.0	0.036473	0.030543	0.016824	0.022644	0.018766	0.024
2	0	0.0	0.018592	0.023756	0.023756	0.022644	0.018766	0.024
3	1	0.0	0.193911	0.287327	0.287327	0.143059	0.283167	0.279
4	0	0.0	0.018592	0.005527	0.005605	0.022644	0.018766	0.024

5 rows × 22 columns

2.Modeling

In [3]:

```
from sklearn.model_selection import train_test_split

train_data_with_ap = train_with_ap.drop('click',axis=1)
target_data_with_ap = train_with_ap['click']

x_train_with_ap, x_valid_with_ap, y_train_with_ap, y_valid_with_ap = train_test_split(train_data_wit

train_data = train.drop('click',axis=1)
target_data = train['click']

x_train, x_valid, y_train, y_valid = train_test_split(train_data, target_data)
```

1.RandomFroestClassifier

1.1audience_profile 없는 데이터세트

In [4]:

```
#RandomForestClassifier_without_ap 모델 생성
from sklearn.ensemble import RandomForestClassifier
import sklearn

RFC =  RandomForestClassifier(n_estimators=100,max_depth=5, n_jobs=-1)
RFC_without_ap = RFC.fit(x_train, y_train)

print('training set_with_ap accuracy :', RFC_without_ap.score(x_train, y_train))
print('validation set_with_ap accuracy :', RFC_without_ap.score(x_valid, y_valid))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid, RFC_without_ap.predict_
```

```
training set_with_ap accuracy : 0.9201093333333333
validation set_with_ap accuracy : 0.9202647272727272
validation set_with_ap log loss:  0.19489632292660777
```

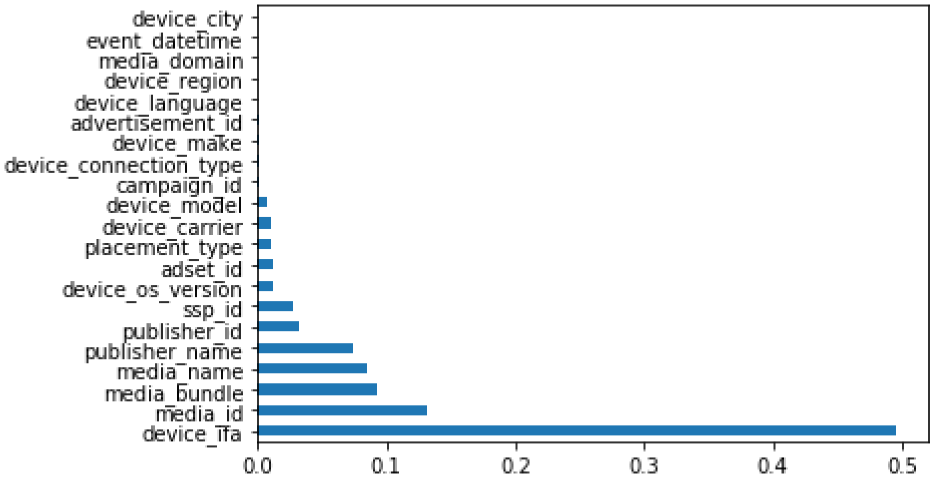
In [5]:

```
#plot feature importance
import matplotlib.pylab as plt
%matplotlib inline

feat_importances = pd.Series(RFC_without_ap.feature_importances_, index=x_train.columns)
feat_importances.nlargest(len(x_train.columns)).plot(kind='barh')
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2723ddb0048>
```



Feature selecion

In [6]:

```
#Feature selection
from sklearn.feature_selection import SelectFromModel

sel = SelectFromModel(RandomForestClassifier(n_estimators=100,max_depth=5, n_jobs=-1),threshold="0.
sel.fit(train_data, target_data)

selected_feat= train_data.columns[(sel.get_support())]
print(selected_feat)
```

```
Index(['ssp_id', 'campaign_id', 'adset_id', 'placement_type', 'media_id',
       'media_name', 'media_bundle', 'publisher_id', 'publisher_name',
       'device_ifa', 'device_os_version', 'device_model', 'device_carrier',
       'device_connection_type'],
      dtype='object')
```

In [7]:

```
x_train_f=x_train[selected_feat]
x_train_f.head()
```

Out[7]:

	ssp_id	campaign_id	adset_id	placement_type	media_id	media_name	media_bunc
716979	0.193911	0.136531	0.144954	0.143059	0.022128	0.022128	0.0215
1756644	0.193911	0.098465	0.110714	0.143059	0.006384	0.006384	0.0305
1218791	0.046613	0.056622	0.096883	0.143059	0.060584	0.060584	0.0601
2980572	0.193911	0.292706	0.292706	0.143059	0.283167	0.279698	0.2740
4854807	0.193911	0.294792	0.294792	0.143059	0.283167	0.279698	0.2740

In [8]:

```
x_valid_f = x_valid[selected_feat]
x_valid_f.head()
```

Out[8]:

	ssp_id	campaign_id	adset_id	placement_type	media_id	media_name	media_bunc
5178242	0.193911	0.075303	0.174953	0.143059	0.283167	0.279698	0.2740
387091	0.193911	0.287327	0.287327	0.143059	0.283167	0.279698	0.2740
5384832	0.046613	0.046747	0.015466	0.022644	0.030230	0.030230	0.0289
3360945	0.046613	0.075303	0.045820	0.143059	0.033853	0.033853	0.0306
1980357	0.193911	0.136572	0.137860	0.143059	0.283167	0.279698	0.2740

In [9]:

```
#RandomForestClassifier_without_ap
from sklearn.ensemble import RandomForestClassifier

RFC =  RandomForestClassifier(n_estimators=100,max_depth=5, n_jobs=-1)
RFC_without_ap = RFC.fit(x_train_f, y_train)

print('training set_with_ap accuracy :', RFC_without_ap.score(x_train_f, y_train))
print('validation set_with_ap accuracy :', RFC_without_ap.score(x_valid_f, y_valid))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid, RFC_without_ap.predict_
```

```
training set_with_ap accuracy : 0.9205267878787878
validation set_with_ap accuracy : 0.9206814545454546
validation set_with_ap log loss:  0.1903733970648135
```

1.2audience_profile 포함하는 데이터세트

In [10]:

```
#RandomForestClassifier_with_ap 모델 생성
from sklearn.ensemble import RandomForestClassifier

RFC =  RandomForestClassifier(n_estimators=100,max_depth=5, n_jobs=-1)
RFC_with_ap = RFC.fit(x_train_with_ap, y_train_with_ap)

print('training set_with_ap accuracy :', RFC_with_ap.score(x_train_with_ap, y_train_with_ap))
print('validation set_with_ap accuracy :', RFC_with_ap.score(x_valid_with_ap, y_valid_with_ap))
#validation set log loss
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid_with_ap, RFC_with_ap.pre
```

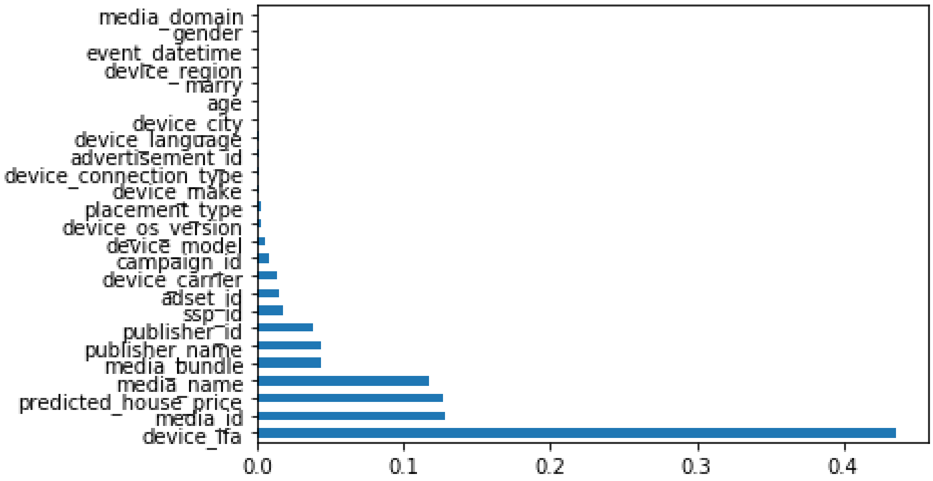
```
training set_with_ap accuracy : 0.9244724829750653
validation set_with_ap accuracy : 0.9228985077472173
validation set_with_ap log loss:  0.18900328463203622
```

In [11]:

```
#plot feature importance
feat_importances = pd.Series(RFC_with_ap.feature_importances_, index=x_train_with_ap.columns)
feat_importances.nlargest(len(x_train_with_ap.columns)).plot(kind='barh')
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2723df56a08>
```



Feature selection

In [12]:

```
#Feature selection
from sklearn.feature_selection import SelectFromModel

sel = SelectFromModel(RandomForestClassifier(n_estimators=100,max_depth=5, n_jobs=-1),threshold="0.
sel.fit(train_data_with_ap, target_data_with_ap)

selected_feat_ap= train_data_with_ap.columns[(sel.get_support())]
print(selected_feat_ap)
```

```
Index(['ssp_id', 'campaign_id', 'adset_id', 'placement_type', 'media_id',
       'media_name', 'media_bundle', 'publisher_id', 'publisher_name',
       'device_ifa', 'device_os_version', 'device_model', 'device_carrier',
       'predicted_house_price'],
      dtype='object')
```

In [13]:

```
x_train_with_ap_f=x_train_with_ap[selected_feat_ap]
x_train_with_ap_f.head()
```

Out[13]:

	ssp_id	campaign_id	adset_id	placement_type	media_id	media_name	media_bundl
297961	0.015880	0.019819	0.019819	0.021026	0.016541	0.023879	0.02174
132005	0.015880	0.047038	0.012108	0.021026	0.016541	0.023879	0.02406
67061	0.039832	0.161582	0.170724	0.173604	0.098299	0.098299	0.09786
196628	0.143941	0.164003	0.217998	0.139161	0.273404	0.278601	0.27263
340650	0.189793	0.297220	0.297220	0.139161	0.282208	0.278601	0.27263

In [14]:

```
x_valid_with_ap_f = x_valid_with_ap[selected_feat_ap]
x_valid_with_ap_f.head()
```

Out[14]:

	ssp_id	campaign_id	adset_id	placement_type	media_id	media_name	media_bundl
374661	0.143941	0.096605	0.123112	0.139161	0.273404	0.278601	0.27263
201129	0.015880	0.003569	0.003569	0.021026	0.016541	0.023879	0.00442
253992	0.034519	0.097937	0.014756	0.021026	0.014414	0.014414	0.05843
143413	0.048989	0.129205	0.130435	0.139161	0.129187	0.023879	0.10968
318988	0.039832	0.087930	0.136877	0.139161	0.032964	0.032964	0.02987

In [15]:

```
#RandomForestClassifier_with_ap
from sklearn.ensemble import RandomForestClassifier

RFC =  RandomForestClassifier(n_estimators=100,max_depth=5, n_jobs=-1)
RFC_with_ap = RFC.fit(x_train_with_ap_f, y_train_with_ap)

print('training set_with_ap accuracy :', RFC_with_ap.score(x_train_with_ap_f, y_train_with_ap))
print('validation set_with_ap accuracy :', RFC_with_ap.score(x_valid_with_ap_f, y_valid_with_ap))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid_with_ap, RFC_with_ap.pre
```

```
training set_with_ap accuracy : 0.9249137721041725
validation set_with_ap accuracy : 0.9234068692345982
validation set_with_ap log loss:  0.18761931542342092
```

3.하이퍼 파라미터 조정

RandomForestClassifier는 기본 하이퍼파라미터 설정값으로도 충분히 좋은 성능을 내기 때문에 하이퍼 파라미터 조정을 진행하지 않는다.

4.Final Validation

In [16]:

```
#final feature selection
train_data_with_ap_f=train_data_with_ap[selected_feat_ap]
train_data_without_ap_f=train_data[selected_feat]
```

In [17]:

```
RFC =  RandomForestClassifier(n_estimators=100,max_depth=5, n_jobs=-1)
RFC_with_ap = RFC.fit(train_data_with_ap_f, target_data_with_ap)
RFC =  RandomForestClassifier(n_estimators=100,max_depth=5, n_jobs=-1)
RFC_without_ap = RFC.fit(train_data_without_ap_f, target_data)
```

In [18]:

```
#Cross validation(최종 검증)
from sklearn.model_selection import StratifiedKFold,cross_val_score
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
score_ap= cross_val_score(RFC_with_ap,train_data_with_ap_f,target_data_with_ap,scoring="neg_log_los
score= cross_val_score(RFC_without_ap,train_data_without_ap_f,target_data,scoring="neg_log_loss",cv=
```

In [19]:

```
print("cross validation score of model_with_ap: ",score_ap.mean())
print("cross validation score of model_without_ap: ",score.mean())
```

```
cross validation score of model_with_ap:  -0.18812836569154887
cross validation score of model_without_ap:  -0.19490072748757653
```

4.Model saving

In [20]:

```
import pickle
with open("RFC_without_ap.sav", 'wb') as file:
    pickle.dump(RFC_without_ap, file)
with open("RFC_with_ap.sav", 'wb') as file:
    pickle.dump(RFC_with_ap, file)
```

In [ ]:

1. Test set reading

In [1]:

```
#test data reading
#bid_id 만 read vs 데이터 세트 preprocess에서 구성하여 읽기
import pandas as pd
raw_test= pd.read_csv('test.csv')
raw_test.head()
```

Out[1]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	2019-10-11 00:00:05.593	jlLrN0gGpx	nwf1A3O5cO	7Noz5InNj5	yH0QQDoPNl	kleE1J0KCa	
1	2019-10-11 00:00:06.024	zA3WyymOcJ	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kleE1J0KCa	9
2	2019-10-11 00:00:06.126	Pwlj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kleE1J0KCa	9
3	2019-10-11 00:00:06.598	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kleE1J0KCa	Zı
4	2019-10-11 00:00:06.639	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9

5 rows × 24 columns

In [2]:

```
#test data 와 audience profile merging
#필요열만 읽어들이기
import gc
import pandas as pd
# audience_profile의 크기가 ram 용량에 비해 커서 작은 단위로 나누어 ram에 올리고 지우기
n=1
for chunk in pd.read_csv('audience_profile.csv',sep='delimiter', delimiter = "!@#", chunksize=5000(
    if n ==1:
        test_with_ap =pd.merge(raw_test, chunk, how='inner', on='device_ifa')
    else:
        test_with_ap = pd.concat([test_with_ap,pd.merge(raw_test, chunk, how='inner', on='device_if
    del chunk
    gc.collect()  #ram에서 삭제
    n+=1

test_with_ap.head(5)
```

```
C:\Users\Administrator\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: ParserWa
rning: Falling back to the 'python' engine because the 'c' engine does not support r
egex separators (separators > 1 char and different from '\s+' are interpreted as reg
ex); you can avoid this warning by specifying engine='python'.
  import sys
```

Out[2]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	2019-10-11 00:00:23.202	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kIeE1J0KCa	E
1	2019-10-11 00:00:27.861	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9l
2	2019-10-11 03:52:07.245	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kIeE1J0KCa	9l
3	2019-10-11 00:00:35.423	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kIeE1J0KCa	hk
4	2019-10-11 00:00:52.950	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kIeE1J0KCa	

5 rows × 31 columns

In [3]:

```
bid_id_with_ap=test_with_ap['bid_id']
bid_id_with_ap
```

Out[3]:

```
0        EHTpBC8c9L
1        0JdfjVSNi8
2        d7g9YcPv7u
3        KJryVcuWQc
4        0bZPTVVYcA
            ...
10765    faWjGdHRmw
10766    mAl8SUv657
10767    kuCl8qgDp9
10768    SL3vs75mac
10769    dv5HX2ehaK
Name: bid_id, Length: 214642, dtype: object
```

In [4]:

```
#test set without ap set 생성
test=pd.merge(raw_test,test_with_ap[['device_ifa','gender']], how='left',on='device_ifa')
test=test[(test['gender'].isnull()==1)].drop(['gender'],axis=1)
test
```

4	2019-10-11 00:00:06.639	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9bC9qJ8
6	2019-10-11 00:00:07.166	PAG5INDY6L	y7QKxSwhwV	dS6rpIpBHY	G0n6acDmBk	tg9mzu7kFm	8eTC2j
...	...	...	...	...	...	...	
782300	2019-10-12 00:01:21.559	cLU5mO3z89	SrN77Arvqh	2NIOV3Vhjb	RBkPIE7zXi	1pcQ3RJgQt	hkFCnTpl
782301	2019-10-12 00:01:21.570	2YxtmVzvpB	Uox85xVMSC	SHpt2lzYOT	AlVulu17z5	kIeE1J0KCa	JzMEh8RE
782304	2019-10-12 00:01:21.886	bSxh3i0gN3	nwf1A3O5cO	w6ERRwu6pk	tr7cYrEXuJ	kIeE1J0KCa	WG9YHz
782305	2019-10-12 00:01:22.026	LAyxamwNxm	M6QaRvdZ8h	wvAODZefbN	GdGZ3dDmhQ	kIeE1J0KCa	EWk3Gk
782307	2019-10-12 00:01:22.270	W8XuFXZw4v	nwf1A3O5cO	qR4Xa60DLI	FiSRHSfVaf	kIeE1J0KCa	j7H2fW

In [5]:

```
bid_id_without_ap=test['bid_id']
```

2.Test set preprocessing

In [6]:

```
test_with_ap['predicted_house_price']=test_with_ap['predicted_house_price'].fillna(value=test_with_a
```

In [7]:

```
#시간 정보만 추출(test_with_ap)
test_with_ap['event_datetime'] = pd.to_datetime(test_with_ap['event_datetime']).dt.hour.astype(int)
test_with_ap.head()
```

Out[7]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kleE1J0KCa	E
1	0	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9b
2	3	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kleE1J0KCa	9b
3	0	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kleE1J0KCa	hk
4	0	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLI	FiSRHSfVaf	kleE1J0KCa	

5 rows × 31 columns

In [8]:

```
#시간 정보만 추출(test)
test['event_datetime'] = pd.to_datetime(test['event_datetime']).dt.hour.astype(int)
test.head()
```

Out[8]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	jILrN0gGpx	nwf1A3O5cO	7Noz5lnNj5	yH0QQDoPNl	kleE1J0KCa	
2	0	Pwlj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kleE1J0KCa	9
3	0	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kleE1J0KCa	Z
4	0	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9
6	0	PAG5INDY6L	y7QKxSwhwV	dS6rpIpBHY	G0n6acDmBk	tg9mzu7kFm	

5 rows × 24 columns

In [9]:

```
# converting gender,marry feature to numeical value
test_with_ap['gender']=test_with_ap['gender'].map({'M':0,
                                                    'F':1})
test_with_ap['marry']=test_with_ap['marry'].map({'M':0,
                                                  'S':1})
test_with_ap.head()
```

Out[9]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kleE1J0KCa	E
1	0	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9b
2	3	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kleE1J0KCa	9b
3	0	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kleE1J0KCa	hk
4	0	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kleE1J0KCa	

5 rows × 31 columns

In [10]:

```
#encoding을 위해 age feature str으로 변환
test_with_ap['age']=test_with_ap['age'].astype('str')
```

In [11]:

```
test_with_ap=test_with_ap.drop(['install_pack','bid_id','cate_code','asset_index','device_os','devic
test = test.drop(['bid_id','device_os','device_country'],axis=1)
```

3.Feature engineering

Target encoding

In [12]:

```
#target encoding
from category_encoders import TargetEncoder
import pickle

enc_with_ap = pickle.load(open('enc_preprocess_1.sav', 'rb'))

numeric_test_with_ap=enc_with_ap.transform(test_with_ap)
numeric_test_with_ap
```

Out[12]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_nam
0	0	0.143941	0.042074	0.141686	0.139161	0.273404	0.27860
1	0	0.039832	0.057104	0.115930	0.139161	0.032964	0.03296
2	3	0.039832	0.047038	0.109618	0.139161	0.032964	0.03296
3	0	0.009178	0.008072	0.008072	0.139161	0.016541	0.00917
4	0	0.189793	0.285726	0.285726	0.139161	0.282208	0.27860
...	...	...	...	...	...	...	.
10765	0	0.011040	0.012040	0.006623	0.021026	0.016541	0.01724
10766	0	0.039832	0.105717	0.130554	0.139161	0.090909	0.09090
10767	0	0.189793	0.135373	0.136251	0.139161	0.023806	0.02380
10768	0	0.189793	0.106897	0.142785	0.139161	0.034730	0.03473
10769	0	0.143941	0.087930	0.136877	0.139161	0.273404	0.27860

214642 rows × 25 columns

In [13]:

```
enc = pickle.load(open('enc_preprocess_2.sav', 'rb'))

numeric_test=enc.transform(test)
numeric_test
```

Out[13]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_nar
0	0	0.193911	0.254406	0.244391	0.143059	0.283167	0.2796
2	0	0.046613	0.046747	0.104764	0.143059	0.033853	0.0338
3	0	0.144511	0.064228	0.092414	0.143059	0.025916	0.0259
4	0	0.046613	0.056631	0.119014	0.143059	0.033853	0.0338
6	0	0.184407	0.190157	0.201794	0.184516	0.174362	0.1743
...	...	...	...	...	...	...	
782300	0	0.018592	0.126839	0.003182	0.022644	0.018766	0.0249
782301	0	0.046613	0.108579	0.133637	0.143059	0.020502	0.0205
782304	0	0.193911	0.088609	0.136681	0.143059	0.260223	0.2602
782305	0	0.144511	0.136572	0.137860	0.143059	0.273958	0.2796
782307	0	0.193911	0.287327	0.287327	0.143059	0.283167	0.2796

335358 rows × 21 columns

In [14]:

```
#test_with_ap set
numeric_test_with_ap = numeric_test_with_ap[['ssp_id', 'campaign_id', 'adset_id', 'placement_type',
       'media_name', 'media_bundle', 'publisher_id', 'publisher_name',
       'device_ifa', 'device_os_version', 'device_model', 'device_carrier',
       'predicted_house_price']]
numeric_test_with_ap.head()
```

Out[14]:

	ssp_id	campaign_id	adset_id	placement_type	media_id	media_name	media_bundle	pu
0	0.143941	0.042074	0.141686	0.139161	0.273404	0.278601	0.272632	
1	0.039832	0.057104	0.115930	0.139161	0.032964	0.032964	0.029874	
2	0.039832	0.047038	0.109618	0.139161	0.032964	0.032964	0.029874	
3	0.009178	0.008072	0.008072	0.139161	0.016541	0.009178	0.009178	
4	0.189793	0.285726	0.285726	0.139161	0.282208	0.278601	0.272632	

In [15]:

```
#test set
numeric_test = numeric_test[['ssp_id', 'campaign_id', 'adset_id', 'placement_type', 'media_id',
       'media_name', 'media_bundle', 'publisher_id', 'publisher_name',
       'device_ifa', 'device_os_version', 'device_model', 'device_carrier',
       'device_connection_type']]
numeric_test.head()
```

Out[15]:

	ssp_id	campaign_id	adset_id	placement_type	media_id	media_name	media_bundle	pu
0	0.193911	0.254406	0.244391	0.143059	0.283167	0.279698	0.274011	
2	0.046613	0.046747	0.104764	0.143059	0.033853	0.033853	0.030615	
3	0.144511	0.064228	0.092414	0.143059	0.025916	0.025916	0.030615	
4	0.046613	0.056631	0.119014	0.143059	0.033853	0.033853	0.030615	
6	0.184407	0.190157	0.201794	0.184516	0.174362	0.174362	0.173042	

scaling

In [16]:

```
#ap 포함하는 파일
from sklearn.preprocessing import MinMaxScaler
Scaler = MinMaxScaler()
numeric_test_with_ap[['predicted_house_price']] = Scaler.fit_transform(numeric_test_with_ap[['predic
numeric_test_with_ap
```

```
C:\Users\Administrator\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  after removing the cwd from sys.path.
C:\Users\Administrator\Anaconda3\lib\site-packages\pandas\core\frame.py:3498: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self.loc._setitem_with_indexer((slice(None), indexer), value)
C:\Users\Administrator\Anaconda3\lib\site-packages\pandas\core\frame.py:3469: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self._setitem_array(key, value)
```

Out[16]:

	ssp_id	campaign_id	adset_id	placement_type	media_id	media_name	media_bundle
0	0.143941	0.042074	0.141686	0.139161	0.273404	0.278601	0.272632
1	0.039832	0.057104	0.115930	0.139161	0.032964	0.032964	0.029874
2	0.039832	0.047038	0.109618	0.139161	0.032964	0.032964	0.029874
3	0.009178	0.008072	0.008072	0.139161	0.016541	0.009178	0.009178
4	0.189793	0.285726	0.285726	0.139161	0.282208	0.278601	0.272632
...	...	...	...	...	...	...	...
10765	0.011040	0.012040	0.006623	0.021026	0.016541	0.017241	0.017241
10766	0.039832	0.105717	0.130554	0.139161	0.090909	0.090909	0.090909
10767	0.189793	0.135373	0.136251	0.139161	0.023806	0.023806	0.026056
10768	0.189793	0.106897	0.142785	0.139161	0.034730	0.034730	0.036685
10769	0.143941	0.087930	0.136877	0.139161	0.273404	0.278601	0.272632

214642 rows × 14 columns

3.Model loading

In [17]:

```
#Model reading
import pickle

RFC = pickle.load(open('RFC_without_ap.sav', 'rb'))
RFC_with_ap = pickle.load(open('RFC_with_ap.sav', 'rb'))
```

3.Prediction

1.RandomForestClassifier

In [18]:

```
#test_with_ap 세트 확률과 bid id 연결
probs_with_ap = RFC_with_ap.predict_proba(numeric_test_with_ap)
probs_with_ap = probs_with_ap[:, 1]
bid_ap = pd.DataFrame(bid_id_with_ap)
bid_ap['probs']=probs_with_ap
bid_ap
```

Out[18]:

	bid_id	probs
0	EHTpBC8c9L	0.196267
1	0JdfjVSNi8	0.016246
2	d7g9YcPv7u	0.016318
3	KJryVcuWQc	0.012006
4	ObZPTVVYcA	0.650776
...	...	...
10765	faWjGdHRmw	0.013217
10766	mAI8SUv657	0.073003
10767	kuCI8qgDp9	0.015639
10768	SL3vs75mac	0.024731
10769	dv5HX2ehaK	0.133572

214642 rows × 2 columns

In [19]:

```
#test 세트 확률과 bid id 연결

probs = RFC.predict_proba(numeric_test)
probs = probs[:, 1]
bid = pd.DataFrame(bid_id_without_ap)
bid['probs']=probs
bid
```

Out[19]:

	bid_id	probs
0	jlLrN0gGpx	0.215444
2	PwIj11RYvM	0.027068
3	W0o0KwmTSQ	0.025356
4	UpL3kLWqZy	0.026818
6	PAG5INDY6L	0.156978
...	...	...
782300	cLU5mO3z89	0.006349
782301	2YxtmVzvpB	0.032682
782304	bSxh3i0gN3	0.197404
782305	LAyxamwNxm	0.134171
782307	W8XuFXZw4v	0.129640

335358 rows × 2 columns

In [20]:

```
#모든 확률값 병합

bid_all= pd.concat([bid,bid_ap])
submit = pd.merge(raw_test['bid_id'],bid_all,how="left",on='bid_id')
submit
```

Out[20]:

```
              bid_id     probs
     0    jlLrN0gGpx  0.215444
     1    zA3WyymOcJ  0.022740
     2    PwIj11RYvM  0.027068
     3    W0o0KwmTSQ  0.025356
     4    UpL3kLWqZy  0.026818
   ...           ...       ...
549995    bSxh3i0gN3  0.197404
549996    LAyxamwNxm  0.134171
549997    3sF8PXgPom  0.661520
549998    W8XuFXZw4v  0.129640
549999    WNke5qEQC1  0.197315

550000 rows × 2 columns
```

In [21]:

```
submit.to_csv('submit_RFC_ap.csv', index=False,header=False)
```

1.Data Reading

In [1]:

```
import pandas as pd

train_with_ap = pd.read_csv('train_preprocess_1.csv')
train_with_ap.head()
```

Out[1]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_na
0	0	0.086957	0.048989	0.146178	0.011869	0.052756	0.000272	0.023
1	0	0.086957	0.143941	0.093064	0.104093	0.139161	0.105418	0.105
2	0	0.652174	0.189793	0.096605	0.123112	0.139161	0.119479	0.119
3	0	1.000000	0.189793	0.096605	0.123112	0.139161	0.119479	0.119
4	0	0.956522	0.143941	0.096605	0.123112	0.139161	0.105418	0.105

5 rows × 26 columns

In [2]:

```
train = pd.read_csv('train_preprocess_2.csv')
train.head()
```

Out[2]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_na
0	0	0.0	0.087898	0.130311	0.028476	0.022644	0.002079	0.002
1	0	0.0	0.036473	0.030543	0.016824	0.022644	0.018766	0.024
2	0	0.0	0.018592	0.023756	0.023756	0.022644	0.018766	0.024
3	1	0.0	0.193911	0.287327	0.287327	0.143059	0.283167	0.279
4	0	0.0	0.018592	0.005527	0.005605	0.022644	0.018766	0.024

5 rows × 22 columns

2.Modeling

In [3]:

```
from sklearn.model_selection import train_test_split

train_data_with_ap = train_with_ap.drop('click',axis=1)
target_data_with_ap = train_with_ap['click']

x_train_with_ap, x_valid_with_ap, y_train_with_ap, y_valid_with_ap = train_test_split(train_data_wit

train_data = train.drop('click',axis=1)
target_data = train['click']

x_train, x_valid, y_train, y_valid = train_test_split(train_data, target_data)
```

2. Logistic Regression

2.1audience_profile 없는 데이터세트

In [4]:

```
#logistic Regression_without_ap
from sklearn.linear_model import LogisticRegression
import sklearn

lr = LogisticRegression(C=1000.0, random_state=0)
lr_without_ap=lr.fit(x_train, y_train)

print('training set_with_ap accuracy :', lr_without_ap.score(x_train, y_train))
print('validation set_with_ap accuracy :', lr_without_ap.score(x_valid, y_valid))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid, lr_without_ap.predict_p
```

```
C:\Users\Administrator\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a sol
ver to silence this warning.
  FutureWarning)

training set_with_ap accuracy : 0.9223963636363637
validation set_with_ap accuracy : 0.9221883636363636
validation set_with_ap log loss:  0.1912311867864279
```

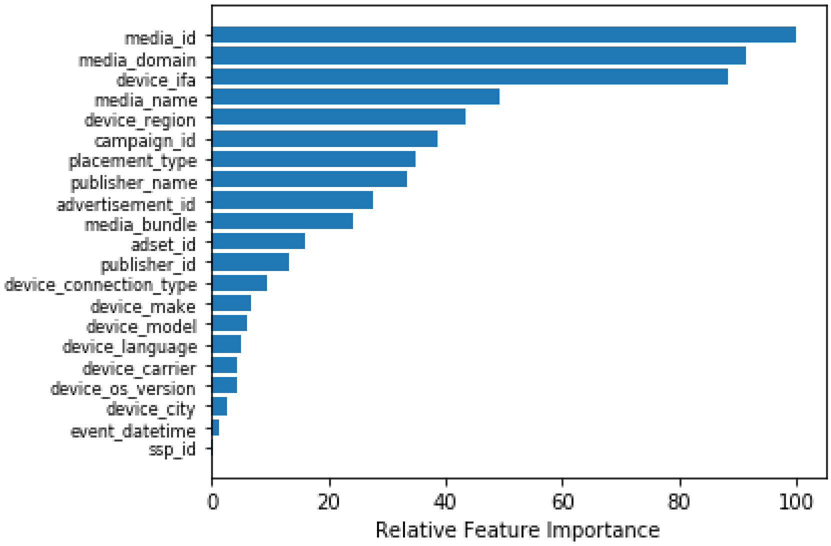
In [5]:

```
import matplotlib.pylab as plt
import numpy as np
%matplotlib inline

feature_importance = abs(lr_without_ap.coef_[0])
feature_importance = 100.0 * (feature_importance / feature_importance.max())
sorted_idx = np.argsort(feature_importance)
pos = np.arange(sorted_idx.shape[0]) + .5

featfig = plt.figure()
featax = featfig.add_subplot(1, 1, 1)
featax.barh(pos, feature_importance[sorted_idx], align='center')
featax.set_yticks(pos)
featax.set_yticklabels(np.array(x_train.columns)[sorted_idx], fontsize=8)
featax.set_xlabel('Relative Feature Importance')

plt.tight_layout()
plt.show()
```



In [6]:

```
#Feature selection
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression

sel = SelectFromModel(LogisticRegression(C=1000.0, random_state=0),threshold = "0.25*median")
sel.fit(train_data, target_data)

sel.get_support()
```

```
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a sol
ver to silence this warning.
  FutureWarning)
```

Out[6]:

```
array([False, False,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True, False,  True])
```

In [7]:

```
selected_feat= train_data.columns[(sel.get_support())]
print(selected_feat)
```

```
Index(['campaign_id', 'adset_id', 'placement_type', 'media_id', 'media_name',
       'media_bundle', 'media_domain', 'publisher_id', 'publisher_name',
       'device_ifa', 'device_os_version', 'device_model', 'device_carrier',
       'device_make', 'device_connection_type', 'device_language',
       'device_region', 'advertisement_id'],
      dtype='object')
```

In [8]:

```
x_train_f=x_train[selected_feat]
x_train_f.head()
```

Out[8]:

	campaign_id	adset_id	placement_type	media_id	media_name	media_bundle	media_
1435597	0.052936	0.047512	0.143059	0.031161	0.031161	0.030615	(
3313250	0.048909	0.052600	0.022644	0.018766	0.024986	0.024414	(
380710	0.098465	0.110714	0.143059	0.023646	0.024449	0.025723	(
2361171	0.285384	0.285384	0.143059	0.283167	0.279698	0.274011	(
3313857	0.209155	0.209285	0.143059	0.166355	0.166355	0.164328	(

In [9]:

```
x_valid_f = x_valid[selected_feat]
x_valid_f.head()
```

Out[9]:

	campaign_id	adset_id	placement_type	media_id	media_name	media_bundle	media_
3279390	0.136572	0.137860	0.143059	0.026774	0.018900	0.021018	(
5340276	0.023756	0.023756	0.022644	0.018766	0.024986	0.024414	(
3825907	0.128094	0.171690	0.143059	0.033853	0.033853	0.030615	(
3871623	0.129479	0.167415	0.143059	0.273958	0.279698	0.274011	(
1898388	0.209155	0.209285	0.143059	0.283167	0.279698	0.274011	(

In [10]:

```
#logistic Regression_without_ap after feature selection
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=1000.0, random_state=0)
lr_without_ap=lr.fit(x_train_f, y_train)

print('training set_with_ap accuracy :', lr_without_ap.score(x_train_f, y_train))
print('validation set_with_ap accuracy :', lr_without_ap.score(x_valid_f, y_valid))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid, lr_without_ap.predict_p
```

```
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a sol
ver to silence this warning.
  FutureWarning)

training set_with_ap accuracy : 0.9223757575757576
validation set_with_ap accuracy : 0.9221585454545455
validation set_with_ap log loss:  0.19125442885571037
```

2.2audience_profile 포함하는 데이터세트

In [11]:

```
#logistic Regression_with_ap
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=1000.0, random_state=0)
lr_with_ap=lr.fit(x_train_with_ap, y_train_with_ap)

print('training set_with_ap accuracy :', lr_with_ap.score(x_train_with_ap, y_train_with_ap))
print('validation set_with_ap accuracy :', lr_with_ap.score(x_valid_with_ap, y_valid_with_ap))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid_with_ap, lr_with_ap.prec
```

```
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a sol
ver to silence this warning.
  FutureWarning)

training set_with_ap accuracy : 0.9221283551212486
validation set_with_ap accuracy : 0.9239258215862997
validation set_with_ap log loss:  0.18846499958069646
```

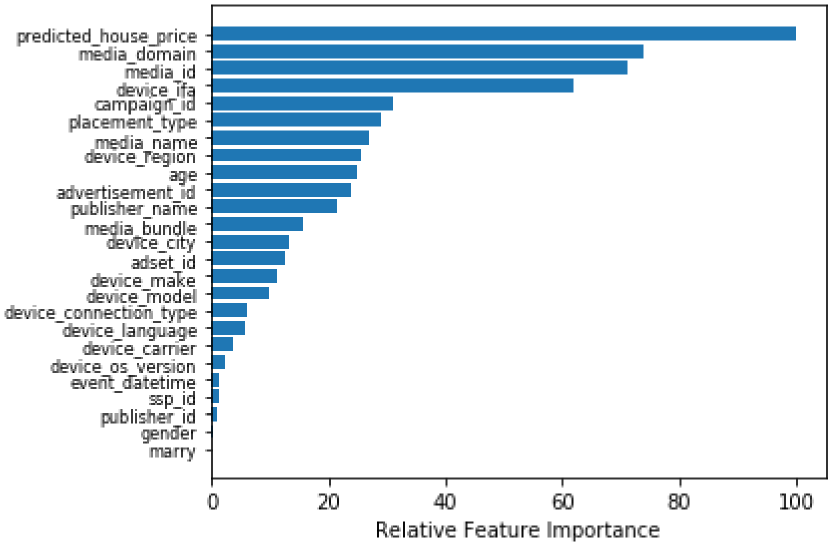
In [12]:

```
#feature importance plot
import matplotlib.pylab as plt
import numpy as np
%matplotlib inline

feature_importance = abs(lr_with_ap.coef_[0])
feature_importance = 100.0 * (feature_importance / feature_importance.max())
sorted_idx = np.argsort(feature_importance)
pos = np.arange(sorted_idx.shape[0]) + .5

featfig = plt.figure()
featax = featfig.add_subplot(1, 1, 1)
featax.barh(pos, feature_importance[sorted_idx], align='center')
featax.set_yticks(pos)
featax.set_yticklabels(np.array(x_train_with_ap.columns)[sorted_idx], fontsize=8)
featax.set_xlabel('Relative Feature Importance')

plt.tight_layout()
plt.show()
```



In [13]:

```
#Feature selection
from sklearn.feature_selection import SelectFromModel

sel = SelectFromModel(LogisticRegression(C=1000.0, random_state=0),threshold = "0.25*median")
sel.fit(train_data_with_ap, target_data_with_ap)

sel.get_support()
```

```
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a sol
ver to silence this warning.
  FutureWarning)
```

Out[13]:

```
array([False, False,  True,  True,  True,  True,  True,  True,  True,
       False,  True,  True, False,  True,  True,  True,  True,  True,
        True,  True,  True,  True, False, False,  True])
```

In [14]:

```
selected_feat_ap= train_data_with_ap.columns[(sel.get_support())]
print(selected_feat_ap)
```

```
Index(['campaign_id', 'adset_id', 'placement_type', 'media_id', 'media_name',
       'media_bundle', 'media_domain', 'publisher_name', 'device_ifa',
       'device_model', 'device_carrier', 'device_make',
       'device_connection_type', 'device_language', 'device_region',
       'device_city', 'advertisement_id', 'age', 'predicted_house_price'],
      dtype='object')
```

In [15]:

```
x_train_with_ap_f=x_train_with_ap[selected_feat_ap]
x_train_with_ap_f.head()
```

Out[15]:

	campaign_id	adset_id	placement_type	media_id	media_name	media_bundle	media_(
304529	0.070628	0.144559	0.139161	0.273404	0.278601	0.272632	0.
110317	0.065546	0.093710	0.139161	0.025807	0.025807	0.029874	0.
60135	0.027056	0.027056	0.021026	0.016541	0.023879	0.023664	0.
126027	0.075370	0.057576	0.052756	0.032345	0.032345	0.048611	0.
337713	0.042074	0.051673	0.139161	0.019410	0.023879	0.029874	0.

In [16]:

```
x_valid_with_ap_f = x_valid_with_ap[selected_feat_ap]
x_valid_with_ap_f.head()
```

Out[16]:

	campaign_id	adset_id	placement_type	media_id	media_name	media_bundle	media_(
216097	0.206682	0.206209	0.139161	0.282208	0.278601	0.272632	0.
56004	0.057104	0.009009	0.021026	0.017450	0.018437	0.018962	0.
175719	0.061998	0.119803	0.139161	0.273404	0.278601	0.272632	0.
323311	0.256653	0.255025	0.139161	0.282208	0.278601	0.272632	0.
304412	0.106897	0.142785	0.139161	0.282208	0.278601	0.272632	0.

In [17]:

```
#logistic Regression_with_ap after feature selection
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=1000.0, random_state=0)
lr_with_ap=lr.fit(x_train_with_ap_f, y_train_with_ap)

print('training set_with_ap accuracy :', lr_with_ap.score(x_train_with_ap_f, y_train_with_ap))
print('validation set_with_ap accuracy :', lr_with_ap.score(x_valid_with_ap_f, y_valid_with_ap))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid_with_ap, lr_with_ap.pred
```

```
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a sol
ver to silence this warning.
  FutureWarning)
```

```
training set_with_ap accuracy : 0.9220083244781315
validation set_with_ap accuracy : 0.9239787759079019
validation set_with_ap log loss:  0.18850900982155783
```

3.Final Validation

In [18]:

```
#final feature selection
train_data_with_ap_f=train_data_with_ap[selected_feat_ap]
train_data_without_ap_f=train_data[selected_feat]
```

In [20]:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=1000.0, random_state=0)
lr_without_ap=lr.fit(train_data_without_ap_f, target_data)
```

```
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a sol
ver to silence this warning.
  FutureWarning)
```

In [21]:

```
lr = LogisticRegression(C=1000.0, random_state=0)
lr_with_ap=lr.fit(train_data_with_ap_f, target_data_with_ap)
```

```
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a sol
ver to silence this warning.
  FutureWarning)
```

In [22]:

```
#Cross validation(최종 검증)
from sklearn.model_selection import StratifiedKFold,cross_val_score
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
score_ap= cross_val_score(lr_with_ap,train_data_with_ap_f,target_data_with_ap,scoring="neg_log_loss'
score= cross_val_score(lr_without_ap,train_data_without_ap_f,target_data,scoring="neg_log_loss",cv=
```

```
solver to silence this warning.
  FutureWarning)
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.p
y:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.p
y:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.p
y:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.p
y:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.p
y:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
```

In [23]:

```
print("cross validation score of model_with_ap: ",score_ap.mean())
print("cross validation score of model_without_ap: ",score.mean())
```

```
cross validation score of model_with_ap:  -0.19132497695156153
cross validation score of model_without_ap:  -0.19109382526159813
```

5.Model saving

In [24]:

```
import pickle

with open("lr_without_ap.sav", 'wb') as file:
    pickle.dump(lr_without_ap, file)
with open("lr_with_ap.sav", 'wb') as file:
    pickle.dump(lr_with_ap, file)
```

In [ ]:

```
```

1. Test set reading

In [1]:

```
#test data reading
#bid_id 만 read vs 데이터 세트 preprocess에서 구성하여 읽기
import pandas as pd
raw_test= pd.read_csv('test.csv')
raw_test.head()
```

Out[1]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type
0	2019-10-11 00:00:05.593	jILrN0gGpx	nwf1A3O5cO	7Noz5InNj5	yH0QQDoPNl	kIeE1J0KCa
1	2019-10-11 00:00:06.024	zA3WyymOcJ	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kIeE1J0KCa
2	2019-10-11 00:00:06.126	PwIj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kIeE1J0KCa
3	2019-10-11 00:00:06.598	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kIeE1J0KCa
4	2019-10-11 00:00:06.639	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa

5 rows × 24 columns

In [2]:

```
#test data 와 audience profile merging
#필요열만 읽어들이기
import gc
import pandas as pd
# audience_profile의 크기가 ram 용량에 비해 커서 작은 단위로 나누어 ram에 올리고 지우기
n=1
for chunk in pd.read_csv('audience_profile.csv',sep='delimiter', delimiter = "!@#", chunksize=50000
    if n ==1:
        test_with_ap =pd.merge(raw_test, chunk, how='inner', on='device_ifa')
    else:
        test_with_ap = pd.concat([test_with_ap,pd.merge(raw_test, chunk, how='inner', on='device_if
    del chunk
    gc.collect()  #ram에서 삭제
    n+=1

test_with_ap.head(5)
```

```
C:\Users\Administrator\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: ParserWa
rning: Falling back to the 'python' engine because the 'c' engine does not support r
egex separators (separators > 1 char and different from '\s+' are interpreted as reg
ex); you can avoid this warning by specifying engine='python'.
  import sys
```

Out[2]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	2019-10-11 00:00:23.202	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kIeE1J0KCa	E
1	2019-10-11 00:00:27.861	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9b
2	2019-10-11 03:52:07.245	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kIeE1J0KCa	9b
3	2019-10-11 00:00:35.423	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kIeE1J0KCa	hk
4	2019-10-11 00:00:52.950	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLI	FiSRHSfVaf	kIeE1J0KCa	

5 rows × 31 columns

In [3]:

```
bid_id_with_ap=test_with_ap['bid_id']
bid_id_with_ap
```

Out[3]:

```
0        EHTpBC8c9L
1        OJdfjVSNi8
2        d7g9YcPv7u
3        KJryVcuWQc
4        ObZPTVVYcA
            ...
10765    faWjGdHRmw
10766    mAl8SUv657
10767    kuCl8qgDp9
10768    SL3vs75mac
10769    dv5HX2ehaK
Name: bid_id, Length: 214642, dtype: object
```

In [4]:

```
#test set without ap set 생성
test=pd.merge(raw_test,test_with_ap[['device_ifa','gender']], how='left',on='device_ifa')
test=test[(test['gender'].isnull()==1)].drop(['gender'],axis=1)
test
```

4	2019-10-11 00:00:06.639	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9bC9qJ8
6	2019-10-11 00:00:07.166	PAG5INDY6L	y7QKxSwhwV	dS6rpIpBHY	G0n6acDmBk	tg9mzu7kFm	8eTC2j
...	...	...	...	...	...	...	
782300	2019-10-12 00:01:21.559	cLU5mO3z89	SrN77Arvqh	2NIOV3Vhjb	RBkPIE7zXi	1pcQ3RJgQt	hkFCnTpl
782301	2019-10-12 00:01:21.570	2YxtmVzvpB	Uox85xVMSC	SHpt2lzYOT	AlVulu17z5	kleE1J0KCa	JzMEh8RE
782304	2019-10-12 00:01:21.886	bSxh3i0gN3	nwf1A3O5cO	w6ERRwu6pk	tr7cYrEXuJ	kleE1J0KCa	WG9YHz
782305	2019-10-12 00:01:22.026	LAyxamwNxm	M6QaRvdZ8h	wvAODZefbN	GdGZ3dDmhQ	kleE1J0KCa	EWk3Gk
782307	2019-10-12 00:01:22.270	W8XuFXZw4v	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kleE1J0KCa	j7H2fW

In [5]:

```
bid_id_without_ap=test['bid_id']
```

2.Test set preprocessing

In [6]:

```
test_with_ap['predicted_house_price']=test_with_ap['predicted_house_price'].fillna(value=test_with_a
```

In [7]:

```
#시간 정보만 추출(test_with_ap)
test_with_ap['event_datetime'] = pd.to_datetime(test_with_ap['event_datetime']).dt.hour.astype(int)
test_with_ap.head()
```

Out[7]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kIeE1J0KCa	E
1	0	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9
2	3	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kIeE1J0KCa	9
3	0	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kIeE1J0KCa	hk
4	0	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kIeE1J0KCa	

5 rows × 31 columns

In [8]:

```
#시간 정보만 추출(test)
test['event_datetime'] = pd.to_datetime(test['event_datetime']).dt.hour.astype(int)
test.head()
```

Out[8]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	jILrN0gGpx	nwf1A3O5cO	7Noz5InNj5	yH0QQDoPNl	kIeE1J0KCa	
2	0	Pwlj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kIeE1J0KCa	9
3	0	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kIeE1J0KCa	Z
4	0	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9
6	0	PAG5INDY6L	y7QKxSwhwV	dS6rpIpBHY	G0n6acDmBk	tg9mzu7kFm	

5 rows × 24 columns

In [9]:

```
# converting gender,marry feature to numeical value
test_with_ap['gender']=test_with_ap['gender'].map({'M':0,
                                                    'F':1})
test_with_ap['marry']=test_with_ap['marry'].map({'M':0,
                                                  'S':1})
test_with_ap.head()
```

Out[9]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kleE1J0KCa	E
1	0	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9b
2	3	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kleE1J0KCa	9b
3	0	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kleE1J0KCa	hk
4	0	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kleE1J0KCa	

5 rows × 31 columns

In [10]:

```
#encoding을 위해 age feature str으로 변환
test_with_ap['age']=test_with_ap['age'].astype('str')
```

In [11]:

```
test_with_ap=test_with_ap.drop(['install_pack','bid_id','cate_code','asset_index','device_os','devic
test = test.drop(['bid_id','device_os','device_country'],axis=1)
```

3.Feature engineering

Target encoding

In [12]:

```
#target encoding
from category_encoders import TargetEncoder
import pickle

enc_with_ap = pickle.load(open('enc_preprocess_1.sav', 'rb'))

numeric_test_with_ap=enc_with_ap.transform(test_with_ap)
numeric_test_with_ap
```

Out[12]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_nam
0	0	0.143941	0.042074	0.141686	0.139161	0.273404	0.27860
1	0	0.039832	0.057104	0.115930	0.139161	0.032964	0.03296
2	3	0.039832	0.047038	0.109618	0.139161	0.032964	0.03296
3	0	0.009178	0.008072	0.008072	0.139161	0.016541	0.00917
4	0	0.189793	0.285726	0.285726	0.139161	0.282208	0.27860
...	...	...	...	...	...	...	.
10765	0	0.011040	0.012040	0.006623	0.021026	0.016541	0.01724
10766	0	0.039832	0.105717	0.130554	0.139161	0.090909	0.09090
10767	0	0.189793	0.135373	0.136251	0.139161	0.023806	0.02380
10768	0	0.189793	0.106897	0.142785	0.139161	0.034730	0.03473
10769	0	0.143941	0.087930	0.136877	0.139161	0.273404	0.27860

214642 rows × 25 columns

In [13]:

```
enc = pickle.load(open('enc_preprocess_2.sav', 'rb'))

numeric_test=enc.transform(test)
numeric_test
```

```
     4          0  0.046613    0.056631  0.119014       0.143059  0.033853    0.033853     0.030
     6          0  0.184407    0.190157  0.201794       0.184516  0.174362    0.174362     0.173
   ...        ...       ...         ...       ...            ...       ...         ...
782300          0  0.018592    0.126839  0.003182       0.022644  0.018766    0.024986     0.004
782301          0  0.046613    0.108579  0.133637       0.143059  0.020502    0.020502     0.086
782304          0  0.193911    0.088609  0.136681       0.143059  0.260223    0.260223     0.258
782305          0  0.144511    0.136572  0.137860       0.143059  0.273958    0.279698     0.274
782307          0  0.193911    0.287327  0.287327       0.143059  0.283167    0.279698     0.274
```

In [14]:

```
#test_with_ap set
numeric_test_with_ap = numeric_test_with_ap[['campaign_id', 'adset_id', 'placement_type', 'media_id'
        'media_bundle', 'media_domain', 'publisher_name', 'device_ifa',
        'device_model', 'device_carrier', 'device_make',
        'device_connection_type', 'device_language', 'device_region',
        'device_city', 'advertisement_id', 'age', 'predicted_house_price']]
numeric_test_with_ap.head()
```

Out[14]:

```
   campaign_id  adset_id  placement_type  media_id  media_name  media_bundle  media_domai
0     0.042074  0.141686        0.139161  0.273404    0.278601      0.272632      0.09193
1     0.057104  0.115930        0.139161  0.032964    0.032964      0.029874      0.09193
2     0.047038  0.109618        0.139161  0.032964    0.032964      0.029874      0.09193
3     0.008072  0.008072        0.139161  0.016541    0.009178      0.009178      0.00917
4     0.285726  0.285726        0.139161  0.282208    0.278601      0.272632      0.09193
```

In [15]:

```
#test set
numeric_test = numeric_test[['campaign_id', 'adset_id', 'placement_type', 'media_id', 'media_name',
       'media_bundle', 'media_domain', 'publisher_id', 'publisher_name',
       'device_ifa', 'device_os_version', 'device_model', 'device_carrier',
       'device_make', 'device_connection_type', 'device_language',
       'device_region', 'advertisement_id']]
numeric_test.head()
```

Out[15]:

	campaign_id	adset_id	placement_type	media_id	media_name	media_bundle	media_domai
0	0.254406	0.244391	0.143059	0.283167	0.279698	0.274011	0.09384
2	0.046747	0.104764	0.143059	0.033853	0.033853	0.030615	0.09384
3	0.064228	0.092414	0.143059	0.025916	0.025916	0.030615	0.09384
4	0.056631	0.119014	0.143059	0.033853	0.033853	0.030615	0.09384
6	0.190157	0.201794	0.184516	0.174362	0.174362	0.173042	0.09384

scaling

In [16]:

```
#ap 포함하는 파일
from sklearn.preprocessing import MinMaxScaler
Scaler = MinMaxScaler()
numeric_test_with_ap[['predicted_house_price']] = Scaler.fit_transform(numeric_test_with_ap[['predic
numeric_test_with_ap
```

```
C:\Users\Administrator\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  after removing the cwd from sys.path.
C:\Users\Administrator\Anaconda3\lib\site-packages\pandas\core\frame.py:3498: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self.loc._setitem_with_indexer((slice(None), indexer), value)
C:\Users\Administrator\Anaconda3\lib\site-packages\pandas\core\frame.py:3469: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self._setitem_array(key, value)
```

Out[16]:

	campaign_id	adset_id	placement_type	media_id	media_name	media_bundle	media_d
0	0.042074	0.141686	0.139161	0.273404	0.278601	0.272632	0.0
1	0.057104	0.115930	0.139161	0.032964	0.032964	0.029874	0.0
2	0.047038	0.109618	0.139161	0.032964	0.032964	0.029874	0.0
3	0.008072	0.008072	0.139161	0.016541	0.009178	0.009178	0.0
4	0.285726	0.285726	0.139161	0.282208	0.278601	0.272632	0.0
...	...	...	...	...	...	...	
10765	0.012040	0.006623	0.021026	0.016541	0.017241	0.017241	0.0
10766	0.105717	0.130554	0.139161	0.090909	0.090909	0.090909	0.0
10767	0.135373	0.136251	0.139161	0.023806	0.023806	0.026056	0.0
10768	0.106897	0.142785	0.139161	0.034730	0.034730	0.036685	0.0
10769	0.087930	0.136877	0.139161	0.273404	0.278601	0.272632	0.0

214642 rows × 19 columns

3.Model loading

In [17]:

```
#Model reading
import pickle

lr = pickle.load(open('lr_without_ap.sav', 'rb'))
lr_with_ap = pickle.load(open('lr_with_ap.sav', 'rb'))
```

3.Prediction

1.RandomForestClassifier

In [18]:

```
#test_with_ap 세트 확률과 bid id 연결
probs_with_ap = lr_with_ap.predict_proba(numeric_test_with_ap)
probs_with_ap = probs_with_ap[:, 1]
bid_ap = pd.DataFrame(bid_id_with_ap)
bid_ap['probs']=probs_with_ap
bid_ap
```

Out[18]:

```
           bid_id     probs
    0  EHTpBC8c9L  0.097043
    1  0JdfjVSNi8  0.019430
    2  d7g9YcPv7u  0.019992
    3  KJryVcuWQc  0.017692
    4  ObZPTVVYcA  0.278338
  ...         ...       ...
10765  faWjGdHRmw  0.015253
10766  mAI8SUv657  0.036691
10767  kuCI8qgDp9  0.022921
10768  SL3vs75mac  0.021370
10769  dv5HX2ehaK  0.047436

214642 rows × 2 columns
```

In [19]:

```
#test 세트 확률과 bid id 연결

probs = lr.predict_proba(numeric_test)
probs = probs[:, 1]
bid = pd.DataFrame(bid_id_without_ap)
bid['probs']=probs
bid
```

Out[19]:

	bid_id	probs
0	jlLrN0gGpx	0.088512
2	Pwlj11RYvM	0.041699
3	W0o0KwmTSQ	0.044277
4	UpL3kLWqZy	0.041806
6	PAG5INDY6L	0.080264
...	...	...
782300	cLU5mO3z89	0.009073
782301	2YxtmVzvpB	0.043081
782304	bSxh3i0gN3	0.100475
782305	LAyxamwNxm	0.044772
782307	W8XuFXZw4v	0.043497

335358 rows × 2 columns

In [20]:

```
#모든 확률값 병합

bid_all= pd.concat([bid,bid_ap])
submit = pd.merge(raw_test['bid_id'],bid_all,how="left",on='bid_id')
submit
```

Out[20]:

	bid_id	probs
0	jlLrN0gGpx	0.088512
1	zA3WyymOcJ	0.067790
2	Pwlj11RYvM	0.041699
3	W0o0KwmTSQ	0.044277
4	UpL3kLWqZy	0.041806
...	...	...
549995	bSxh3i0gN3	0.100475
549996	LAyxamwNxm	0.044772
549997	3sF8PXgPom	0.164399
549998	W8XuFXZw4v	0.043497
549999	WNke5qEQC1	0.092771

550000 rows × 2 columns

In [21]:

```
submit.to_csv('submit_lr_ap.csv', index=False,header=False)
```

1.Data Reading

In [2]:

```
import pandas as pd

train_with_ap = pd.read_csv('train_preprocess_1.csv')
train_with_ap.head()
```

Out[2]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_na
0	0	0.086957	0.048989	0.146178	0.011869	0.052756	0.000272	0.023
1	0	0.086957	0.143941	0.093064	0.104093	0.139161	0.105418	0.105
2	0	0.652174	0.189793	0.096605	0.123112	0.139161	0.119479	0.119
3	0	1.000000	0.189793	0.096605	0.123112	0.139161	0.119479	0.119
4	0	0.956522	0.143941	0.096605	0.123112	0.139161	0.105418	0.105

5 rows × 26 columns

In [3]:

```
train = pd.read_csv('train_preprocess_2.csv')
train.head()
```

Out[3]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_na
0	0	0.0	0.087898	0.130311	0.028476	0.022644	0.002079	0.002
1	0	0.0	0.036473	0.030543	0.016824	0.022644	0.018766	0.024
2	0	0.0	0.018592	0.023756	0.023756	0.022644	0.018766	0.024
3	1	0.0	0.193911	0.287327	0.287327	0.143059	0.283167	0.279
4	0	0.0	0.018592	0.005527	0.005605	0.022644	0.018766	0.024

5 rows × 22 columns

2.Modeling

In [4]:

```
from sklearn.model_selection import train_test_split

train_data_with_ap = train_with_ap.drop('click',axis=1)
target_data_with_ap = train_with_ap['click']

x_train_with_ap, x_valid_with_ap, y_train_with_ap, y_valid_with_ap = train_test_split(train_data_wit

train_data = train.drop('click',axis=1)
target_data = train['click']

x_train, x_valid, y_train, y_valid = train_test_split(train_data, target_data)
```

3.XGBClassifier

3.1audience_profile 없는 데이터세트

In [5]:

```
#XGBClassifier_without_ap
evals = [(x_valid, y_valid)]
from xgboost import XGBClassifier
import sklearn

xgbC = XGBClassifier(n_estimators=100, learning_rate=0.1)
xgbC_without_ap=xgbC.fit(x_train, y_train, early_stopping_rounds=100, eval_metric="logloss", eval_se

print('training set_with_ap accuracy :', xgbC_without_ap.score(x_train, y_train))
print('validation set_with_ap accuracy :', xgbC_without_ap.score(x_valid, y_valid))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid, xgbC_without_ap.predict
```

```
training set_with_ap accuracy : 0.9230693333333333
validation set_with_ap accuracy : 0.9232087272727273
validation set_with_ap log loss:  0.16775986935544202
```

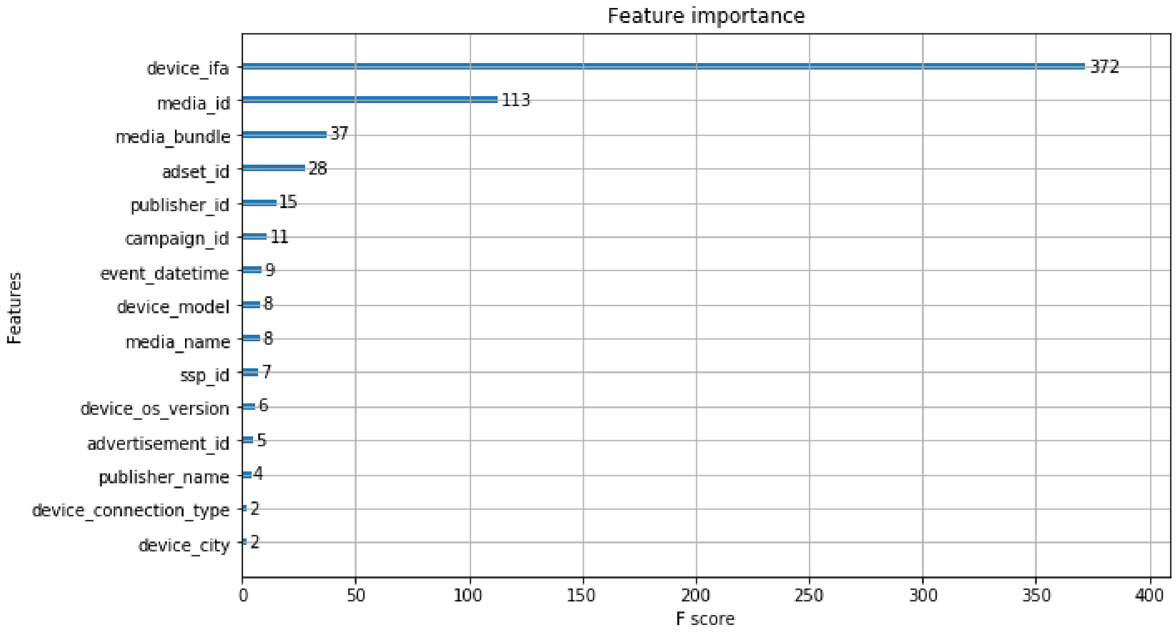
In [6]:

```
#feature importance plot
from xgboost import plot_importance
import matplotlib as mpl
import matplotlib.pylab as plt
%matplotlib inline

fig, ax = plt.subplots(figsize = (10,6))
plot_importance(xgbC_without_ap,ax = ax)
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22645a29f48>
```



In [7]:

```
from sklearn.feature_selection import SelectFromModel
# select features using threshold
sel = SelectFromModel(xgbC_without_ap, threshold = "0.25*median",prefit=True)

selected_feat= train_data.columns[(sel.get_support())]
print(selected_feat)
```

```
Index(['event_datetime', 'ssp_id', 'campaign_id', 'adset_id', 'media_id',
       'media_name', 'media_bundle', 'publisher_id', 'publisher_name',
       'device_ifa', 'device_os_version', 'device_connection_type',
       'advertisement_id'],
      dtype='object')
```

In [8]:

```
x_train_f=x_train[selected_feat]
x_train_f.head()
```

Out[8]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bund
426916	0.826087	0.193911	0.285384	0.285384	0.283167	0.279698	0.2740
5271995	0.652174	0.193911	0.047194	0.079032	0.283167	0.279698	0.2740
4758174	0.913043	0.144511	0.292706	0.292706	0.273958	0.279698	0.2740
3617543	0.652174	0.184407	0.198130	0.198867	0.174362	0.174362	0.1730
169335	0.217391	0.018592	0.035098	0.035098	0.018766	0.024986	0.0629

In [9]:

```
x_valid_f = x_valid[selected_feat]
x_valid_f.head()
```

Out[9]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bund
2089465	0.652174	0.026049	0.047194	0.015895	0.007864	0.007864	0.0078
2438145	0.260870	0.144511	0.158065	0.219374	0.273958	0.279698	0.2740
2413201	0.217391	0.144511	0.209155	0.209285	0.273958	0.279698	0.2740
903509	0.608696	0.046613	0.158065	0.213067	0.058105	0.058105	0.0581
5422919	0.956522	0.046613	0.046747	0.148515	0.033853	0.033853	0.0306

In [10]:

```
#XGBClassifier_without_ap after feature selection
evals = [(x_valid_f, y_valid)]
from xgboost import XGBClassifier

xgbC = XGBClassifier(n_estimators=100, learning_rate=0.1)
xgbC_without_ap=xgbC.fit(x_train_f, y_train, early_stopping_rounds=100, eval_metric="logloss", eval_

print('training set_with_ap accuracy :', xgbC_without_ap.score(x_train_f, y_train))
print('validation set_with_ap accuracy :', xgbC_without_ap.score(x_valid_f, y_valid))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid, xgbC_without_ap.predict
```

```
training set_with_ap accuracy : 0.923321696969697
validation set_with_ap accuracy : 0.9234450909090909
validation set_with_ap log loss:  0.16777806825858244
```

3.2audience_profile 포함하는 데이터세트

In [11]:

```
#XGBClassifier_with_ap
evals = [(x_valid_with_ap, y_valid_with_ap)]
from xgboost import XGBClassifier

xgbC = XGBClassifier(n_estimators=100, learning_rate=0.1)
xgbC_with_ap=xgbC.fit(x_train_with_ap, y_train_with_ap, early_stopping_rounds=100, eval_metric="log

print('training set_with_ap accuracy :', xgbC_with_ap.score(x_train_with_ap, y_train_with_ap))
print('validation set_with_ap accuracy :', xgbC_with_ap.score(x_valid_with_ap, y_valid_with_ap))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid_with_ap, xgbC_with_ap.pr
```

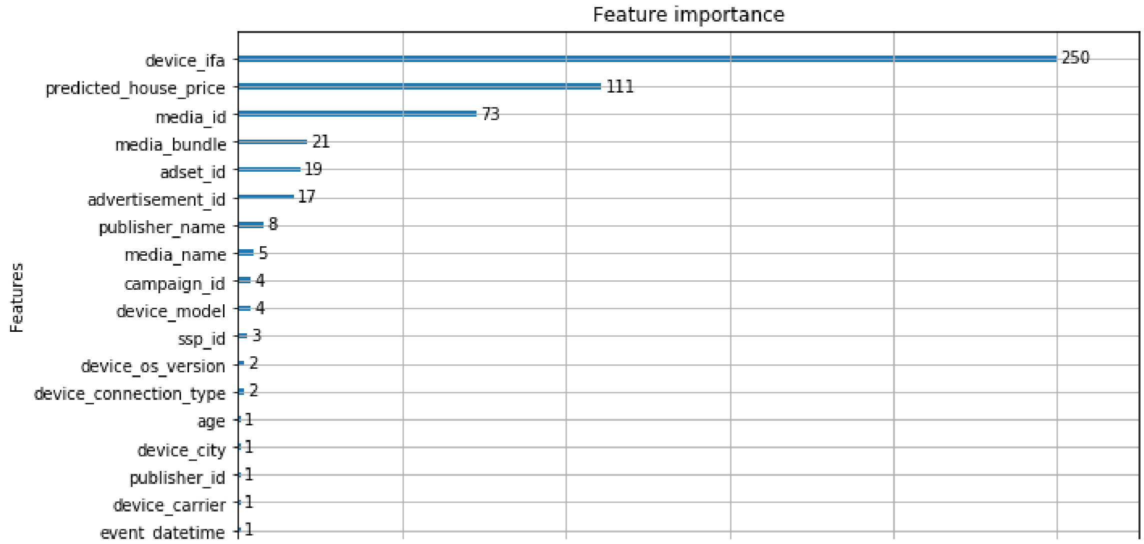
```
training set_with_ap accuracy : 0.9279674928775935
validation set_with_ap accuracy : 0.9276749875557344
validation set_with_ap log loss:  0.16004003990695378
```

In [12]:

```
fig, ax = plt.subplots(figsize = (10,6))
plot_importance(xgbC_with_ap,ax = ax)
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22646949f48>
```



In [13]:

```
from sklearn.feature_selection import SelectFromModel
# select features using threshold
sel = SelectFromModel(xgbC_with_ap, threshold = "0.25*median",prefit=True)

selected_feat_ap= train_data_with_ap.columns[(sel.get_support())]
print(selected_feat)
```

```
Index(['event_datetime', 'ssp_id', 'campaign_id', 'adset_id', 'media_id',
       'media_name', 'media_bundle', 'publisher_id', 'publisher_name',
       'device_ifa', 'device_os_version', 'device_connection_type',
       'advertisement_id'],
      dtype='object')
```

In [15]:

```
x_train_with_ap_f=x_train_with_ap[selected_feat_ap]
x_train_with_ap_f.head()
```

Out[15]:

	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundle	publisher_id
25229	0.039832	0.117587	0.033625	0.016005	0.016005	0.017584	0.016004
329550	0.088627	0.065056	0.014604	0.002239	0.002239	0.002312	0.002239
27360	0.025348	0.065056	0.064371	0.048965	0.048965	0.048951	0.051276
225334	0.189793	0.256156	0.257817	0.282208	0.278601	0.272632	0.282208
271895	0.143941	0.081519	0.129543	0.273404	0.278601	0.272632	0.168428

In [17]:

```
x_valid_with_ap_f = x_valid_with_ap[selected_feat_ap]
x_valid_with_ap_f.head()
```

Out[17]:

	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundle	publisher_id
70566	0.039832	0.093064	0.104093	0.032964	0.032964	0.029874	0.032964
224133	0.143941	0.132918	0.174315	0.016091	0.015839	0.017266	0.168428
307427	0.015880	0.048118	0.042266	0.016541	0.023879	0.021740	0.023796
141011	0.015880	0.003773	0.003773	0.016541	0.023879	0.004529	0.023796
169090	0.015880	0.024206	0.024206	0.016541	0.023879	0.011771	0.023796

In [18]:

```
#XGBClassifier_without_ap after feature selection
evals = [(x_valid_with_ap_f, y_valid_with_ap)]
from xgboost import XGBClassifier

xgbC = XGBClassifier(n_estimators=100, learning_rate=0.1)
xgbC_with_ap=xgbC.fit(x_train_with_ap_f, y_train_with_ap, early_stopping_rounds=100, eval_metric="log

print('training set_with_ap accuracy :', xgbC_with_ap.score(x_train_with_ap_f, y_train_with_ap))
print('validation set_with_ap accuracy :', xgbC_with_ap.score(x_valid_with_ap_f, y_valid_with_ap))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid_with_ap, xgbC_with_ap.pre
```

```
training set_with_ap accuracy : 0.9279498413124292
validation set_with_ap accuracy : 0.9274313976763644
validation set_with_ap log loss:  0.16010173304833145
```

3.Final Validation

In [19]:

```
#final feature selection
train_data_with_ap_f=train_data_with_ap[selected_feat_ap]
train_data_without_ap_f=train_data[selected_feat]
```

```
xgbC = XGBClassifier(n_estimators=100, learning_rate=0.1)
xgbC_without_ap=xgbC.fit(train_data_without_ap_f, target_data, early_stopping_rounds=100,
eval_metric="logloss", eval_set=evals, verbose=False)
xgbC = XGBClassifier(n_estimators=100, learning_rate=0.1)
xgbC_with_ap=xgbC.fit(train_data_with_ap_f, target_data_with_ap, early_stopping_rounds=100,
eval_metric="logloss", eval_set=evals, verbose=False)
```

In [23]:

```
#Cross validation(최종 검증)
from sklearn.model_selection import StratifiedKFold,cross_val_score
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
score_ap= cross_val_score(xgbC_with_ap,train_data_with_ap_f,target_data_with_ap,scoring="neg_log_los
score= cross_val_score(xgbC_without_ap,train_data_without_ap_f,target_data,scoring="neg_log_loss",cv
```

```
[04:22:40] WARNING: C:₩Jenkins₩workspace₩xgboost-win64_release_0.90₩src₩learner.cc:6
86: Tree method is automatically selected to be 'approx' for faster speed. To use ol
d behavior (exact greedy algorithm on single machine), set tree_method to 'exact'.
[04:36:41] WARNING: C:₩Jenkins₩workspace₩xgboost-win64_release_0.90₩src₩learner.cc:6
86: Tree method is automatically selected to be 'approx' for faster speed. To use ol
d behavior (exact greedy algorithm on single machine), set tree_method to 'exact'.
[04:50:57] WARNING: C:₩Jenkins₩workspace₩xgboost-win64_release_0.90₩src₩learner.cc:6
86: Tree method is automatically selected to be 'approx' for faster speed. To use ol
d behavior (exact greedy algorithm on single machine), set tree_method to 'exact'.
[05:05:13] WARNING: C:₩Jenkins₩workspace₩xgboost-win64_release_0.90₩src₩learner.cc:6
86: Tree method is automatically selected to be 'approx' for faster speed. To use ol
d behavior (exact greedy algorithm on single machine), set tree_method to 'exact'.
[05:18:27] WARNING: C:₩Jenkins₩workspace₩xgboost-win64_release_0.90₩src₩learner.cc:6
86: Tree method is automatically selected to be 'approx' for faster speed. To use ol
d behavior (exact greedy algorithm on single machine), set tree_method to 'exact'.
```

In [24]:

```
print("cross validation logloss of model_with_ap: ",score_ap.mean())
print("cross validation logloss of model_without_ap: ",score.mean())
```

```
cross validation logloss of model_with_ap:  -0.1596142260036695
cross validation logloss of model_without_ap:  -0.16875295251515496
```

4.Model saving

In [26]:

```
import pickle

with open("xgbC_without_ap.sav", 'wb') as file:
    pickle.dump(xgbC_without_ap, file)
with open("xgbC_with_ap.sav", 'wb') as file:
    pickle.dump(xgbC_with_ap, file)
```

In [ ]:

1. Test set reading

In [1]:

```
#test data reading
#bid_id 만 read vs 데이터 세트 preprocess에서 구성하여 읽기
import pandas as pd
raw_test= pd.read_csv('test.csv')
raw_test.head()
```

Out[1]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	2019-10-11 00:00:05.593	jILrN0gGpx	nwf1A3O5cO	7Noz5InNj5	yH0QQDoPNI	kIeE1J0KCa	
1	2019-10-11 00:00:06.024	zA3WyymOcJ	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kIeE1J0KCa	9
2	2019-10-11 00:00:06.126	PwIj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kIeE1J0KCa	9
3	2019-10-11 00:00:06.598	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kIeE1J0KCa	Z
4	2019-10-11 00:00:06.639	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9

5 rows × 24 columns

In [2]:

```
#test data 와 audience profile merging
#필요열만 읽어들이기
import gc
import pandas as pd
# audience_profile의 크기가 ram 용량에 비해 커서 작은 단위로 나누어 ram에 올리고 지우기
n=1
for chunk in pd.read_csv('audience_profile.csv',sep='delimiter', delimiter = "!@#", chunksize=5000(
    if n ==1:
        test_with_ap =pd.merge(raw_test, chunk, how='inner', on='device_ifa')
    else:
        test_with_ap = pd.concat([test_with_ap,pd.merge(raw_test, chunk, how='inner', on='device_if
    del chunk
    gc.collect()  #ram에서 삭제
    n+=1

test_with_ap.head(5)
```

```
C:\Users\Administrator\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: ParserWa
rning: Falling back to the 'python' engine because the 'c' engine does not support r
egex separators (separators > 1 char and different from '\s+' are interpreted as reg
ex); you can avoid this warning by specifying engine='python'.
  import sys
```

Out[2]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	2019-10-11 00:00:23.202	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kIeE1J0KCa	E
1	2019-10-11 00:00:27.861	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9b
2	2019-10-11 03:52:07.245	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kIeE1J0KCa	9b
3	2019-10-11 00:00:35.423	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kIeE1J0KCa	hk
4	2019-10-11 00:00:52.950	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLI	FiSRHSfVaf	kIeE1J0KCa	

5 rows × 31 columns

In [3]:

```
bid_id_with_ap=test_with_ap['bid_id']
bid_id_with_ap
```

Out[3]:

```
0        EHTpBC8c9L
1        0JdfjVSNi8
2        d7g9YcPv7u
3        KJryVcuWQc
4        0bZPTVVYcA
            ...
10765    faWjGdHRmw
10766    mAI8SUv657
10767    kuCI8qgDp9
10768    SL3vs75mac
10769    dv5HX2ehaK
Name: bid_id, Length: 214642, dtype: object
```

In [4]:

```
#test set without ap set 생성
test=pd.merge(raw_test,test_with_ap[['device_ifa','gender']], how='left',on='device_ifa')
test=test[(test['gender'].isnull()==1)].drop(['gender'],axis=1)
test
```

Out[4]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	media
0	2019-10-11 00:00:05.593	jILrN0gGpx	nwf1A3O5cO	7Noz5InNj5	yH0QQDoPNl	kleE1J0KCa	j7H2fW
2	2019-10-11 00:00:06.126	PwIj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kleE1J0KCa	9bC9qJ8
3	2019-10-11 00:00:06.598	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kleE1J0KCa	ZrCGAwg
4	2019-10-11 00:00:06.639	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9bC9qJ8
6	2019-10-11 00:00:07.166	PAG5INDY6L	y7QKxSwhwV	dS6rpIpBHY	G0n6acDmBk	tg9mzu7kFm	8eTC2j
...	...	...	...	...	...	...	
782300	2019-10-12	cLU5mO3z89	SrNZZArvgh	2NIQV3Vhib	RRkPIFZzXi	1pcO3RJgOt	hkFCnTpl

In [5]:

```
bid_id_without_ap=test['bid_id']
```

2.Test set preprocessing

In [6]:

```
test_with_ap['predicted_house_price']=test_with_ap['predicted_house_price'].fillna(value=test_with_a
```

In [7]:

```
#시간 정보만 추출(test_with_ap)
test_with_ap['event_datetime'] = pd.to_datetime(test_with_ap['event_datetime']).dt.hour.astype(int)
test_with_ap.head()
```

Out[7]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kIeE1J0KCa	E
1	0	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9b
2	3	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kIeE1J0KCa	9b
3	0	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kIeE1J0KCa	hk
4	0	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLI	FiSRHSfVaf	kIeE1J0KCa	

5 rows × 31 columns

In [8]:

```
#시간 정보만 추출(test)
test['event_datetime'] = pd.to_datetime(test['event_datetime']).dt.hour.astype(int)
test.head()
```

Out[8]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	jILrN0gGpx	nwf1A3O5cO	7Noz5InNj5	yH0QQDoPNI	kIeE1J0KCa	
2	0	PwIj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kIeE1J0KCa	9
3	0	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kIeE1J0KCa	Z
4	0	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9
6	0	PAG5INDY6L	y7QKxSwhwV	dS6rpIpBHY	G0n6acDmBk	tg9mzu7kFm	

5 rows × 24 columns

In [9]:

```
# converting gender,marry feature to numeical value
test_with_ap['gender']=test_with_ap['gender'].map({'M':0,
                                                   'F':1})
test_with_ap['marry']=test_with_ap['marry'].map({'M':0,
                                                 'S':1})
test_with_ap.head()
```

Out[9]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kleE1J0KCa	E
1	0	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9b
2	3	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kleE1J0KCa	9b
3	0	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kleE1J0KCa	hk
4	0	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kleE1J0KCa	

5 rows × 31 columns

In [10]:

```
#encoding을 위해 age feature str으로 변환
test_with_ap['age']=test_with_ap['age'].astype('str')
```

In [11]:

```
test_with_ap=test_with_ap.drop(['install_pack','bid_id','cate_code','asset_index','device_os','devic
test = test.drop(['bid_id','device_os','device_country'],axis=1)
```

3.Feature engineering

Target encoding

In [29]:

```
#target encoding
from category_encoders import TargetEncoder
import pickle

enc_with_ap = pickle.load(open('enc_preprocess_1.sav', 'rb'))

numeric_test_with_ap=enc_with_ap.transform(test_with_ap)
numeric_test_with_ap
```

Out[29]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_nam
0	0	0.143941	0.042074	0.141686	0.139161	0.273404	0.27860
1	0	0.039832	0.057104	0.115930	0.139161	0.032964	0.03296
2	3	0.039832	0.047038	0.109618	0.139161	0.032964	0.03296
3	0	0.009178	0.008072	0.008072	0.139161	0.016541	0.00917
4	0	0.189793	0.285726	0.285726	0.139161	0.282208	0.27860
...	...	...	...	...	...	...	.
10765	0	0.011040	0.012040	0.006623	0.021026	0.016541	0.01724
10766	0	0.039832	0.105717	0.130554	0.139161	0.090909	0.09090
10767	0	0.189793	0.135373	0.136251	0.139161	0.023806	0.02380
10768	0	0.189793	0.106897	0.142785	0.139161	0.034730	0.03473
10769	0	0.143941	0.087930	0.136877	0.139161	0.273404	0.27860

214642 rows × 25 columns

In [30]:

```
enc = pickle.load(open('enc_preprocess_2.sav', 'rb'))

numeric_test=enc.transform(test)
numeric_test
```

Out[30]:

```
        event_datetime    ssp_id  campaign_id  adset_id  placement_type  media_id  media_nar
0                    0  0.193911     0.254406  0.244391        0.143059  0.283167     0.2796
2                    0  0.046613     0.046747  0.104764        0.143059  0.033853     0.0338
3                    0  0.144511     0.064228  0.092414        0.143059  0.025916     0.0259
4                    0  0.046613     0.056631  0.119014        0.143059  0.033853     0.0338
6                    0  0.184407     0.190157  0.201794        0.184516  0.174362     0.1743
...                ...       ...          ...       ...             ...       ...
782300               0  0.018592     0.126839  0.003182        0.022644  0.018766     0.0249
782301               0  0.046613     0.108579  0.133637        0.143059  0.020502     0.0205
782304               0  0.193911     0.088609  0.136681        0.143059  0.260223     0.2602
782305               0  0.144511     0.136572  0.137860        0.143059  0.273958     0.2796
782307               0  0.193911     0.287327  0.287327        0.143059  0.283167     0.2796

335358 rows × 21 columns
```

In [31]:

```
#test_with_ap set
numeric_test_with_ap = numeric_test_with_ap[['ssp_id', 'campaign_id', 'adset_id', 'media_id', 'media
       'media_bundle', 'publisher_id', 'publisher_name', 'device_ifa',
       'device_os_version', 'device_model', 'device_city', 'age',
       'predicted_house_price']]
numeric_test_with_ap.head()
```

Out[31]:

	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundle	publisher_id	publis
0	0.143941	0.042074	0.141686	0.273404	0.278601	0.272632	0.168428	
1	0.039832	0.057104	0.115930	0.032964	0.032964	0.029874	0.032964	
2	0.039832	0.047038	0.109618	0.032964	0.032964	0.029874	0.032964	
3	0.009178	0.008072	0.008072	0.016541	0.009178	0.009178	0.009178	
4	0.189793	0.285726	0.285726	0.282208	0.278601	0.272632	0.282208	

In [32]:

```
#test set
numeric_test = numeric_test[['event_datetime', 'ssp_id', 'campaign_id', 'adset_id', 'media_id',
       'media_name', 'media_bundle', 'publisher_id', 'publisher_name',
       'device_ifa', 'device_os_version', 'device_connection_type',
       'advertisement_id']]
numeric_test.head()
```

Out[32]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundle	pul
0	0	0.193911	0.254406	0.244391	0.283167	0.279698	0.274011	
2	0	0.046613	0.046747	0.104764	0.033853	0.033853	0.030615	
3	0	0.144511	0.064228	0.092414	0.025916	0.025916	0.030615	
4	0	0.046613	0.056631	0.119014	0.033853	0.033853	0.030615	
6	0	0.184407	0.190157	0.201794	0.174362	0.174362	0.173042	

scaling

In [38]:

```
#ap 비포함 파일
from sklearn.preprocessing import MinMaxScaler
Scaler = MinMaxScaler()
numeric_test[['event_datetime']] = Scaler.fit_transform(numeric_test[['event_datetime']])
numeric_test
```

```
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:4: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  after removing the cwd from sys.path.
C:₩Users₩Administrator₩Anaconda3₩lib₩site-packages₩pandas₩core₩indexing.py:494: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self.obj[item] = s
```

Out[38]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundl
0	0.0	0.193911	0.254406	0.244391	0.283167	0.279698	0.27401
2	0.0	0.046613	0.046747	0.104764	0.033853	0.033853	0.03061
3	0.0	0.144511	0.064228	0.092414	0.025916	0.025916	0.03061
4	0.0	0.046613	0.056631	0.119014	0.033853	0.033853	0.03061
6	0.0	0.184407	0.190157	0.201794	0.174362	0.174362	0.17304
...	...	...	...	...	...	...	..
782300	0.0	0.018592	0.126839	0.003182	0.018766	0.024986	0.00456
782301	0.0	0.046613	0.108579	0.133637	0.020502	0.020502	0.08664
782304	0.0	0.193911	0.088609	0.136681	0.260223	0.260223	0.25874
782305	0.0	0.144511	0.136572	0.137860	0.273958	0.279698	0.27401
782307	0.0	0.193911	0.287327	0.287327	0.283167	0.279698	0.27401

335358 rows × 13 columns

3.Model loading

In [39]:

```
#Model reading
import pickle

xgbC = pickle.load(open('xgbC_without_ap.sav', 'rb'))
xgbC_with_ap = pickle.load(open('xgbC_with_ap.sav', 'rb'))
```

3.Prediction

6.XgBooostClassifier

In [40]:

```
numeric_test_with_ap
```

Out[40]:

	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundle	publisher_id	p
0	0.143941	0.042074	0.141686	0.273404	0.278601	0.272632	0.168428	
1	0.039832	0.057104	0.115930	0.032964	0.032964	0.029874	0.032964	
2	0.039832	0.047038	0.109618	0.032964	0.032964	0.029874	0.032964	
3	0.009178	0.008072	0.008072	0.016541	0.009178	0.009178	0.009178	
4	0.189793	0.285726	0.285726	0.282208	0.278601	0.272632	0.282208	
...	...	...	...	...	...	...	...	
10765	0.011040	0.012040	0.006623	0.016541	0.017241	0.017241	0.009709	
10766	0.039832	0.105717	0.130554	0.090909	0.090909	0.090909	0.100000	
10767	0.189793	0.135373	0.136251	0.023806	0.023806	0.026056	0.060699	
10768	0.189793	0.106897	0.142785	0.034730	0.034730	0.036685	0.041657	
10769	0.143941	0.087930	0.136877	0.273404	0.278601	0.272632	0.168428	

214642 rows × 14 columns

In [41]:

```
#test_with_ap 세트 확률과 bid id 연결
probs_with_ap = xgbC_with_ap.predict_proba(numeric_test_with_ap)
probs_with_ap = probs_with_ap[:, 1]
bid_ap = pd.DataFrame(bid_id_with_ap)
bid_ap['probs']=probs_with_ap
bid_ap
```

Out[41]:

```
            bid_id     probs
    0   EHTpBC8c9L  0.989204
    1   0JdfjVSNi8  0.001708
    2   d7g9YcPv7u  0.001708
    3   KJryVcuWQc  0.819127
    4   ObZPTVVYcA  0.988905
  ...          ...       ...
10765   faWjGdHRmw  0.827769
10766   mAI8SUv657  0.981077
10767   kuCI8qgDp9  0.001708
10768   SL3vs75mac  0.940694
10769   dv5HX2ehaK  0.001333

214642 rows × 2 columns
```

In [42]:

```
#test 세트 확률과 bid id 연결

probs = xgbC.predict_proba(numeric_test)
probs = probs[:, 1]
bid = pd.DataFrame(bid_id_without_ap)
bid['probs']=probs
bid
```

Out[42]:

	bid_id	probs
0	jlLrN0gGpx	0.212692
2	PwIj11RYvM	0.045867
3	W0o0KwmTSQ	0.041069
4	UpL3kLWqZy	0.043078
6	PAG5INDY6L	0.177034
...	...	...
782300	cLU5mO3z89	0.000068
782301	2YxtmVzvpB	0.036899
782304	bSxh3i0gN3	0.220303
782305	LAyxamwNxm	0.000073
782307	W8XuFXZw4v	0.000071

335358 rows × 2 columns

In [43]:

```
#모든 확률값 병합

bid_all= pd.concat([bid,bid_ap])
submit = pd.merge(raw_test['bid_id'],bid_all,how="left",on='bid_id')
submit
```

Out[43]:

```
              bid_id     probs
     0    jlLrN0gGpx  0.212692
     1    zA3WyymOcJ  0.940694
     2    Pwlj11RYvM  0.045867
     3    W0o0KwmTSQ  0.041069
     4    UpL3kLWqZy  0.043078
   ...           ...       ...
549995    bSxh3i0gN3  0.220303
549996    LAyxamwNxm  0.000073
549997    3sF8PXgPom  0.990385
549998    W8XuFXZw4v  0.000071
549999    WNke5qEQC1  0.989342

550000 rows × 2 columns
```

In [44]:

```
submit.to_csv('submit_xgbC_ap.csv', index=False,header=False)
```

In [ ]:

```

```

1.Data Reading

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.metrics import accuracy_score
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split

from sklearn.metrics import log_loss
```

In [2]:

```
import pandas as pd

train_with_ap = pd.read_csv('train_preprocess_1.csv')
train_with_ap.head()
```

Out[2]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_na
0	0	0.086957	0.048989	0.146178	0.011869	0.052756	0.000272	0.023
1	0	0.086957	0.143941	0.093064	0.104093	0.139161	0.105418	0.105
2	0	0.652174	0.189793	0.096605	0.123112	0.139161	0.119479	0.119
3	0	1.000000	0.189793	0.096605	0.123112	0.139161	0.119479	0.119
4	0	0.956522	0.143941	0.096605	0.123112	0.139161	0.105418	0.105

5 rows × 26 columns

In [3]:

```
train = pd.read_csv('train_preprocess_2.csv')
train.head()
```

Out[3]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_na
0	0	0.0	0.087898	0.130311	0.028476	0.022644	0.002079	0.002
1	0	0.0	0.036473	0.030543	0.016824	0.022644	0.018766	0.024
2	0	0.0	0.018592	0.023756	0.023756	0.022644	0.018766	0.024
3	1	0.0	0.193911	0.287327	0.287327	0.143059	0.283167	0.279
4	0	0.0	0.018592	0.005527	0.005605	0.022644	0.018766	0.024

5 rows × 22 columns

2.Modeling

In [4]:

```
from sklearn.model_selection import train_test_split

train_data_with_ap = train_with_ap.drop('click',axis=1)
target_data_with_ap = train_with_ap['click']

x_train_with_ap, x_valid_with_ap, y_train_with_ap, y_valid_with_ap = train_test_split(train_data_wit
################################################
train_data = train.drop('click',axis=1)
target_data = train['click']

x_train, x_valid, y_train, y_valid = train_test_split(train_data, target_data)
```

5.GradientBoostingClassifier

5.1audience_profile 없는 데이터세트

In [5]:

```
#GradientBoostingClassifier_without_ap
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier(random_state=0) # 기본값: max_depth=3, learning_rate=0.1
gbc_without_ap=gbc.fit(x_train, y_train)

print('training set_with_ap accuracy :', gbc_without_ap.score(x_train, y_train))
print('validation set_with_ap accuracy :', gbc_without_ap.score(x_valid, y_valid))
```

```
training set_with_ap accuracy : 0.9237515151515151
validation set_with_ap accuracy : 0.9237738181818181
```

5.2audience_profile 포함하는 데이터세트

In [6]:

```
#GradientBoostingClassifier_with_ap
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier(random_state=0) # 기본값: max_depth=3, learning_rate=0.1
gbc_with_ap=gbc.fit(x_train_with_ap, y_train_with_ap)

print('training set_with_ap accuracy :', gbc_with_ap.score(x_train_with_ap, y_train_with_ap))
print('validation set_with_ap accuracy :', gbc_with_ap.score(x_valid_with_ap, y_valid_with_ap))
```

```
training set_with_ap accuracy : 0.9277274315913592
validation set_with_ap accuracy : 0.9304603848720094
```

In [ ]:

```

```

In [7]:

```
#Feature selection with
from sklearn.feature_selection import SelectFromModel

sel = SelectFromModel(GradientBoostingClassifier(max_depth=3, learning_rate=0.1,random_state=0),thr
sel.fit(train_data_with_ap, target_data_with_ap)

selected_feat_ap= train_data_with_ap.columns[(sel.get_support())]
print(selected_feat_ap)
```

```
Index(['event_datetime', 'ssp_id', 'campaign_id', 'adset_id', 'media_id',
       'media_bundle', 'publisher_id', 'device_ifa', 'device_os_version',
       'device_model', 'device_connection_type', 'device_city',
       'advertisement_id', 'predicted_house_price'],
      dtype='object')
```

In [8]:

```
x_train_with_ap_f=x_train_with_ap[selected_feat_ap]
x_train_with_ap_f.head()
```

Out[8]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_bundle	publisher_ic
352008	1.000000	0.015880	0.019819	0.019819	0.016541	0.011536	0.023796
110515	0.391304	0.143941	0.036550	0.060212	0.024390	0.033114	0.168428
106864	0.130435	0.015880	0.104747	0.014516	0.016541	0.011771	0.023796
371257	0.043478	0.044999	0.057087	0.085628	0.038745	0.033832	0.038343
39897	0.304348	0.143941	0.093064	0.104093	0.030814	0.109687	0.168428

In [9]:

```
x_valid_with_ap_f = x_valid_with_ap[selected_feat_ap]
x_valid_with_ap_f.head()
```

Out[9]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_bundle	publisher_ic
186953	1.000000	0.015880	0.004163	0.004175	0.016541	0.021379	0.023796
26653	1.000000	0.015880	0.004163	0.004175	0.016541	0.024063	0.023796
229990	0.260870	0.015880	0.074334	0.015041	0.016541	0.004529	0.023796
28361	0.608696	0.143941	0.135373	0.136251	0.039506	0.041016	0.029469
371280	0.130435	0.143941	0.072638	0.126557	0.061602	0.062685	0.038167

In [10]:

```
#Feature selection without
from sklearn.feature_selection import SelectFromModel

sel = SelectFromModel(GradientBoostingClassifier(max_depth=3, learning_rate=0.1,random_state=0),thr
sel.fit(train_data, target_data)

selected_feat= train_data.columns[(sel.get_support())]
print(selected_feat)
```

```
Index(['event_datetime', 'ssp_id', 'campaign_id', 'adset_id', 'media_id',
       'media_name', 'media_bundle', 'publisher_id', 'publisher_name',
       'device_ifa', 'device_os_version', 'device_model', 'device_carrier',
       'device_connection_type', 'advertisement_id'],
      dtype='object')
```

In [11]:

```
#final feature selection
x_train_f=x_train[selected_feat]
x_train_f.head()
```

Out[11]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bund
5265085	0.652174	0.046613	0.056631	0.051476	0.028990	0.028990	0.03694
4829786	1.000000	0.193911	0.258716	0.257943	0.283167	0.279698	0.27401
3329220	0.130435	0.144511	0.219865	0.223457	0.273958	0.279698	0.27401
3973659	0.217391	0.018592	0.037721	0.037721	0.018766	0.024986	0.06290
3204202	1.000000	0.144511	0.056622	0.096883	0.273958	0.279698	0.27401

In [12]:

```
x_valid_f = x_valid[selected_feat]
x_valid_f.head()
```

Out[12]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bund
2761683	0.043478	0.018592	0.003295	0.003295	0.018766	0.024986	0.00382
5284085	0.652174	0.018592	0.046747	0.008207	0.018766	0.024986	0.00456
3850214	0.043478	0.193911	0.128094	0.081582	0.087301	0.087301	0.08460
5330641	0.826087	0.046613	0.056631	0.119014	0.025582	0.029090	0.02389
3611379	0.652174	0.144511	0.128094	0.081582	0.273958	0.279698	0.27401

비교 feature selection 이후

In [13]:

```
#GradientBoostingClassifier_with_ap
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier(max_depth=3, learning_rate=0.1,random_state=0)
gbc_with_ap=gbc.fit(x_train_with_ap_f, y_train_with_ap)

print('training set_with_ap accuracy :', gbc_with_ap.score(x_train_with_ap_f, y_train_with_ap))
print('validation set_with_ap accuracy :', gbc_with_ap.score(x_valid_with_ap_f, y_valid_with_ap))
```

```
training set_with_ap accuracy : 0.9277662650347206
validation set_with_ap accuracy : 0.9304074305504072
```

In [14]:

```
#GradientBoostingClassifier_without_ap
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier(max_depth=3, learning_rate=0.1,random_state=0)
gbc_without_ap=gbc.fit(x_train_f, y_train)

print('training set_with_ap accuracy :', gbc_without_ap.score(x_train_f, y_train))
print('validation set_with_ap accuracy :', gbc_without_ap.score(x_valid_f, y_valid))
```

```
training set_with_ap accuracy : 0.9237566060606061
validation set_with_ap accuracy : 0.9237447272727273
```

3.Final Validation

In [15]:

```
#final feature selection
train_data_with_ap_f=train_data_with_ap[selected_feat_ap]
train_data_without_ap_f=train_data[selected_feat]
```

In [16]:

```
gbc = GradientBoostingClassifier(max_depth=3, learning_rate=0.1,random_state=0)
gbc_without_ap=gbc.fit(train_data_without_ap_f, target_data)
gbc = GradientBoostingClassifier(max_depth=3, learning_rate=0.1,random_state=0)
gbc_with_ap=gbc.fit(train_data_with_ap_f, target_data_with_ap)
```

In [17]:

```
#Cross validation(최종 검증)
from sklearn.model_selection import StratifiedKFold,cross_val_score
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
score_ap= cross_val_score(gbc_with_ap,train_data_with_ap_f,target_data_with_ap,scoring="neg_log_los
score= cross_val_score(gbc_without_ap,train_data_without_ap_f,target_data,scoring="neg_log_loss",cv
```

In [18]:

```
print("cross validation score of model_with_ap: ",score_ap.mean())
print("cross validation score of model_without_ap: ",score.mean())
```

```
cross validation score of model_with_ap:  -0.1599139101655365
cross validation score of model_without_ap:  -0.1680542761042562
```

4.최종모델 생성

4.Model saving

In [19]:

```
import pickle
with open("gbc_without_ap.sav", 'wb') as file:
    pickle.dump(gbc_without_ap, file)
with open("gbc_with_ap.sav", 'wb') as file:
    pickle.dump(gbc_with_ap, file)
```

In [ ]:

1. Test set reading

In [1]:

```
#test data reading
#bid_id 만 read vs 데이터 세트 preprocess에서 구성하여 읽기
import pandas as pd
raw_test= pd.read_csv('test.csv')
raw_test.head()
```

Out[1]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	2019-10-11 00:00:05.593	jlLrN0gGpx	nwf1A3O5cO	7Noz5lnNj5	yH0QQDoPNl	kIeE1J0KCa	
1	2019-10-11 00:00:06.024	zA3WyymOcJ	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kIeE1J0KCa	9
2	2019-10-11 00:00:06.126	PwIj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kIeE1J0KCa	9
3	2019-10-11 00:00:06.598	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kIeE1J0KCa	Z
4	2019-10-11 00:00:06.639	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9

5 rows × 24 columns

In [2]:

```
#test data 와 audience profile merging
#필요열만 읽어들이기
import gc
import pandas as pd
# audience_profile의 크기가 ram 용량에 비해 커서 작은 단위로 나누어 ram에 올리고 지우기
n=1
for chunk in pd.read_csv('audience_profile.csv',sep='delimiter', delimiter = "!@#", chunksize=50000
    if n ==1:
        test_with_ap =pd.merge(raw_test, chunk, how='inner', on='device_ifa')
    else:
        test_with_ap = pd.concat([test_with_ap,pd.merge(raw_test, chunk, how='inner', on='device_ifa
    del chunk
    gc.collect()  #ram에서 삭제
    n+=1

test_with_ap.head(5)
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:7: ParserWarning: F
alling back to the 'python' engine because the 'c' engine does not support regex sep
arators (separators > 1 char and different from '₩s+' are interpreted as regex); you
can avoid this warning by specifying engine='python'.
  import sys
```

Out[2]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	2019-10-11 00:00:23.202	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kIeE1J0KCa	E
1	2019-10-11 00:00:27.861	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9t
2	2019-10-11 03:52:07.245	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kIeE1J0KCa	9t
3	2019-10-11 00:00:35.423	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kIeE1J0KCa	hk
4	2019-10-11 00:00:52.950	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kIeE1J0KCa	

5 rows × 31 columns

In [3]:

```
bid_id_with_ap=test_with_ap['bid_id']
bid_id_with_ap
```

Out[3]:

```
0        EHTpBC8c9L
1        0JdfjVSNi8
2        d7g9YcPv7u
3        KJryVcuWQc
4        0bZPTVVYcA
            ...
10765    faWjGdHRmw
10766    mAl8SUv657
10767    kuCl8qgDp9
10768    SL3vs75mac
10769    dv5HX2ehaK
Name: bid_id, Length: 214642, dtype: object
```

In [4]:

```
#test set without ap set 생성
test=pd.merge(raw_test,test_with_ap[['device_ifa','gender']], how='left',on='device_ifa')
test=test[(test['gender'].isnull()==1)].drop(['gender'],axis=1)
test
```

4	2019-10-11 00:00:06.639	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9bC9qJ8
6	2019-10-11 00:00:07.166	PAG5INDY6L	y7QKxSwhwV	dS6rpIpBHY	G0n6acDmBk	tg9mzu7kFm	8eTC2j
...	...	...	...	...	...	...	
782300	2019-10-12 00:01:21.559	cLU5mO3z89	SrN77Arvqh	2NIOV3Vhjb	RBkPIE7zXi	1pcQ3RJgQt	hkFCnTpl
782301	2019-10-12 00:01:21.570	2YxtmVzvpB	Uox85xVMSC	SHpt2lzYOT	AIVulu17z5	kIeE1J0KCa	JzMEh8RI
782304	2019-10-12 00:01:21.886	bSxh3i0gN3	nwf1A3O5cO	w6ERRwu6pk	tr7cYrEXuJ	kIeE1J0KCa	WG9YHz
782305	2019-10-12 00:01:22.026	LAyxamwNxm	M6QaRvdZ8h	wvAODZefbN	GdGZ3dDmhQ	kIeE1J0KCa	EWk3Gk
782307	2019-10-12 00:01:22.270	W8XuFXZw4v	nwf1A3O5cO	qR4Xa60DLI	FiSRHSfVaf	kIeE1J0KCa	j7H2fW

In [5]:

```
bid_id_without_ap=test['bid_id']
```

2.Test set preprocessing

In [6]:

```
test_with_ap['predicted_house_price']=test_with_ap['predicted_house_price'].fillna(value=test_with_a
```

In [7]:

```
#시간 정보만 추출(test_with_ap)
test_with_ap['event_datetime'] = pd.to_datetime(test_with_ap['event_datetime']).dt.hour.astype(int)
test_with_ap.head()
```

Out[7]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kIeE1J0KCa	E
1	0	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9b
2	3	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kIeE1J0KCa	9b
3	0	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kIeE1J0KCa	hk
4	0	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kIeE1J0KCa	

5 rows × 31 columns

In [8]:

```
#시간 정보만 추출(test)
test['event_datetime'] = pd.to_datetime(test['event_datetime']).dt.hour.astype(int)
test.head()
```

Out[8]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	jILrN0gGpx	nwf1A3O5cO	7Noz5InNj5	yH0QQDoPNl	kIeE1J0KCa	
2	0	PwIj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kIeE1J0KCa	9
3	0	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kIeE1J0KCa	Z
4	0	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9
6	0	PAG5INDY6L	y7QKxSwhwV	dS6rpIpBHY	G0n6acDmBk	tg9mzu7kFm	

5 rows × 24 columns

In [9]:

```
# converting gender,marry feature to numeical value
test_with_ap['gender']=test_with_ap['gender'].map({'M':0,
                                                    'F':1})
test_with_ap['marry']=test_with_ap['marry'].map({'M':0,
                                                  'S':1})
test_with_ap.head()
```

Out[9]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kleE1J0KCa	E
1	0	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9t
2	3	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kleE1J0KCa	9t
3	0	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kleE1J0KCa	hk
4	0	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kleE1J0KCa	

5 rows × 31 columns

In [10]:

```
#encoding을 위해 age feature str으로 변환
test_with_ap['age']=test_with_ap['age'].astype('str')
```

In [11]:

```
test_with_ap=test_with_ap.drop(['install_pack','bid_id','cate_code','asset_index','device_os','devic
test = test.drop(['bid_id','device_os','device_country'],axis=1)
```

3.Feature engineering

Target encoding

In [12]:

```
#target encoding
from category_encoders import TargetEncoder
import pickle

enc_with_ap = pickle.load(open('enc_preprocess_1.sav', 'rb'))

numeric_test_with_ap=enc_with_ap.transform(test_with_ap)
numeric_test_with_ap
```

Out[12]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_nam
0	0	0.143941	0.042074	0.141686	0.139161	0.273404	0.27860
1	0	0.039832	0.057104	0.115930	0.139161	0.032964	0.03296
2	3	0.039832	0.047038	0.109618	0.139161	0.032964	0.03296
3	0	0.009178	0.008072	0.008072	0.139161	0.016541	0.00917
4	0	0.189793	0.285726	0.285726	0.139161	0.282208	0.27860
...	...	...	...	...	...	...	.
10765	0	0.011040	0.012040	0.006623	0.021026	0.016541	0.01724
10766	0	0.039832	0.105717	0.130554	0.139161	0.090909	0.09090
10767	0	0.189793	0.135373	0.136251	0.139161	0.023806	0.02380
10768	0	0.189793	0.106897	0.142785	0.139161	0.034730	0.03473
10769	0	0.143941	0.087930	0.136877	0.139161	0.273404	0.27860

214642 rows × 25 columns

In [13]:

```
enc = pickle.load(open('enc_preprocess_2.sav', 'rb'))

numeric_test=enc.transform(test)
numeric_test
```

_ifa	device_os_version	device_model	device_carrier	device_make	device_connection_type	device_language
)9e-02	0.135901	0.151469	0.172943	0.125915	0.161345	0.124895
)9e-02	0.036010	0.099170	0.172943	0.125915	0.112029	0.124895
)9e-02	0.135901	0.147155	0.041873	0.061362	0.161345	0.124895
)9e-02	0.036010	0.125626	0.164716	0.125915	0.047138	0.124895
)9e-02	0.063991	0.109436	0.054759	0.125915	0.047138	0.124895
...	...	...	...	...	...	...
29e-	0.021274	0.021369	0.029291	0.061362	0.024992	0.028811

In [14]:

```
#test_with_ap set
numeric_test_with_ap = numeric_test_with_ap[['event_datetime', 'ssp_id', 'campaign_id', 'adset_id',
        'media_bundle', 'publisher_id', 'device_ifa', 'device_os_version',
        'device_model', 'device_connection_type', 'device_city',
        'advertisement_id', 'predicted_house_price']]
numeric_test_with_ap.head()
```

Out[14]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_bundle	publisher_id	dev
0	0	0.143941	0.042074	0.141686	0.273404	0.272632	0.168428	0.
1	0	0.039832	0.057104	0.115930	0.032964	0.029874	0.032964	0.
2	3	0.039832	0.047038	0.109618	0.032964	0.029874	0.032964	0.
3	0	0.009178	0.008072	0.008072	0.016541	0.009178	0.009178	0.
4	0	0.189793	0.285726	0.285726	0.282208	0.272632	0.282208	0.

In [15]:

```
#test set
numeric_test = numeric_test[['event_datetime', 'ssp_id', 'campaign_id', 'adset_id', 'media_id',
        'media_name', 'media_bundle', 'publisher_id', 'publisher_name',
        'device_ifa', 'device_os_version', 'device_model', 'device_carrier',
        'device_connection_type', 'advertisement_id']]
numeric_test.head()
```

Out[15]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundle	pul
0	0	0.193911	0.254406	0.244391	0.283167	0.279698	0.274011	
2	0	0.046613	0.046747	0.104764	0.033853	0.033853	0.030615	
3	0	0.144511	0.064228	0.092414	0.025916	0.025916	0.030615	
4	0	0.046613	0.056631	0.119014	0.033853	0.033853	0.030615	
6	0	0.184407	0.190157	0.201794	0.174362	0.174362	0.173042	

scaling

In [16]:

```
#ap 포함하는 파일
from sklearn.preprocessing import MinMaxScaler
Scaler = MinMaxScaler()
numeric_test_with_ap[['predicted_house_price']] = Scaler.fit_transform(numeric_test_with_ap[['predic
numeric_test_with_ap
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:4: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  after removing the cwd from sys.path.
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩pandas₩core₩indexing.py:494: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self.obj[item] = s
```

Out[16]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_bundle	publisher_id
0	0	0.143941	0.042074	0.141686	0.273404	0.272632	0.168428
1	0	0.039832	0.057104	0.115930	0.032964	0.029874	0.032964
2	3	0.039832	0.047038	0.109618	0.032964	0.029874	0.032964
3	0	0.009178	0.008072	0.008072	0.016541	0.009178	0.009178
4	0	0.189793	0.285726	0.285726	0.282208	0.272632	0.282208
...	...	...	...	...	...	...	...
10765	0	0.011040	0.012040	0.006623	0.016541	0.017241	0.009709
10766	0	0.039832	0.105717	0.130554	0.090909	0.090909	0.100000
10767	0	0.189793	0.135373	0.136251	0.023806	0.026056	0.060699
10768	0	0.189793	0.106897	0.142785	0.034730	0.036685	0.041657
10769	0	0.143941	0.087930	0.136877	0.273404	0.272632	0.168428

214642 rows × 14 columns

In [17]:

```
#ap 비포함 파일
from sklearn.preprocessing import MinMaxScaler
Scaler = MinMaxScaler()
numeric_test[['event_datetime']] = Scaler.fit_transform(numeric_test[['event_datetime']])
numeric_test
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  after removing the cwd from sys.path.
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:494: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self.obj[item] = s
```

Out[17]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundl
0	0.0	0.193911	0.254406	0.244391	0.283167	0.279698	0.27401
2	0.0	0.046613	0.046747	0.104764	0.033853	0.033853	0.03061
3	0.0	0.144511	0.064228	0.092414	0.025916	0.025916	0.03061
4	0.0	0.046613	0.056631	0.119014	0.033853	0.033853	0.03061
6	0.0	0.184407	0.190157	0.201794	0.174362	0.174362	0.17304
...	...	...	...	...	...	...	..
782300	0.0	0.018592	0.126839	0.003182	0.018766	0.024986	0.00456
782301	0.0	0.046613	0.108579	0.133637	0.020502	0.020502	0.08664
782304	0.0	0.193911	0.088609	0.136681	0.260223	0.260223	0.25874
782305	0.0	0.144511	0.136572	0.137860	0.273958	0.279698	0.27401
782307	0.0	0.193911	0.287327	0.287327	0.283167	0.279698	0.27401

335358 rows × 15 columns

3.Model loading

In [18]:

```
#Model reading
import pickle

gbc_with_ap = pickle.load(open('gbc_with_ap.sav', 'rb'))
gbc = pickle.load(open('gbc_without_ap.sav', 'rb'))
```

3.Prediction

5.GradientBoostingClassifier

In [19]:

```
#test_with_ap 세트 확률과 bid id 연결
probs_with_ap = gbc_with_ap.predict_proba(numeric_test_with_ap)
probs_with_ap = probs_with_ap[:, 1]
bid_ap = pd.DataFrame(bid_id_with_ap)
bid_ap['probs']=probs_with_ap
bid_ap
```

Out[19]:

	bid_id	probs
0	EHTpBC8c9L	0.207128
1	0JdfjVSNi8	0.001044
2	d7g9YcPv7u	0.000990
3	KJryVcuWQc	0.017181
4	ObZPTVVYcA	0.988369
...	...	...
10765	faWjGdHRmw	0.019426
10766	mAI8SUv657	0.077443
10767	kuCI8qgDp9	0.001035
10768	SL3vs75mac	0.038667
10769	dv5HX2ehaK	0.001139

214642 rows × 2 columns

In [20]:

```
#test 세트 확률과 bid id 연결

probs = gbc.predict_proba(numeric_test)
probs = probs[:, 1]
bid = pd.DataFrame(bid_id_without_ap)
bid['probs']=probs
bid
```

Out[20]:

	bid_id	probs
0	jlLrN0gGpx	0.224327
2	Pwlj11RYvM	0.040792
3	W0o0KwmTSQ	0.041134
4	UpL3kLWqZy	0.038955
6	PAG5INDY6L	0.169731
...	...	...
782300	cLU5mO3z89	0.000611
782301	2YxtmVzvpB	0.043459
782304	bSxh3i0gN3	0.230427
782305	LAyxamwNxm	0.001438
782307	W8XuFXZw4v	0.001344

335358 rows × 2 columns

In [21]:

```
#모든 확률값 병합

bid_all= pd.concat([bid,bid_ap])
submit = pd.merge(raw_test['bid_id'],bid_all,how="left",on='bid_id')
submit
```

Out[21]:

```
            bid_id      probs
     0  jlLrN0gGpx   0.224327
     1  zA3WyymOcJ   0.038667
     2  Pwlj11RYvM   0.040792
     3  W0o0KwmTSQ   0.041134
     4  UpL3kLWqZy   0.038955
   ...         ...        ...
549995  bSxh3i0gN3   0.230427
549996  LAyxamwNxm   0.001438
549997  3sF8PXgPom   0.992003
549998  W8XuFXZw4v   0.001344
549999  WNke5qEQC1   0.209815

550000 rows × 2 columns
```

In [23]:

```
submit.to_csv('submit_gbc.csv', index=False,header=False)
```

In [ ]:

```

```

1.Data Reading

In [1]:

```
import pandas as pd

train_with_ap = pd.read_csv('train_preprocess_1.csv')
train_with_ap.head()
```

Out[1]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_na
0	0	0.086957	0.048989	0.146178	0.011869	0.052756	0.000272	0.023
1	0	0.086957	0.143941	0.093064	0.104093	0.139161	0.105418	0.105
2	0	0.652174	0.189793	0.096605	0.123112	0.139161	0.119479	0.119
3	0	1.000000	0.189793	0.096605	0.123112	0.139161	0.119479	0.119
4	0	0.956522	0.143941	0.096605	0.123112	0.139161	0.105418	0.105

5 rows × 26 columns

In [2]:

```
train = pd.read_csv('train_preprocess_2.csv')
train.head()
```

Out[2]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_na
0	0	0.0	0.087898	0.130311	0.028476	0.022644	0.002079	0.002
1	0	0.0	0.036473	0.030543	0.016824	0.022644	0.018766	0.024
2	0	0.0	0.018592	0.023756	0.023756	0.022644	0.018766	0.024
3	1	0.0	0.193911	0.287327	0.287327	0.143059	0.283167	0.279
4	0	0.0	0.018592	0.005527	0.005605	0.022644	0.018766	0.024

5 rows × 22 columns

2.Modeling

In [3]:

```
from sklearn.model_selection import train_test_split

train_data_with_ap = train_with_ap.drop('click',axis=1)
target_data_with_ap = train_with_ap['click']

x_train_with_ap, x_valid_with_ap, y_train_with_ap, y_valid_with_ap = train_test_split(train_data_wit

train_data = train.drop('click',axis=1)
target_data = train['click']

x_train, x_valid, y_train, y_valid = train_test_split(train_data, target_data)
```

4.AdaBoostClassifier

4.1audience_profile 없는 데이터세트

In [4]:

```
#AdaBoostClassifier_without_ap
from sklearn.ensemble import AdaBoostClassifier
import sklearn
adaMod = AdaBoostClassifier(base_estimator=None, n_estimators=100, learning_rate=1.0)
adaMod_without_ap=adaMod.fit(x_train, y_train)

print('training set_with_ap accuracy :', adaMod_without_ap.score(x_train, y_train))
print('validation set_with_ap accuracy :', adaMod_without_ap.score(x_valid, y_valid))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid, adaMod_without_ap.predi
```

```
training set_with_ap accuracy : 0.9231430303030304
validation set_with_ap accuracy : 0.9233425454545454
validation set_with_ap log loss:  0.5963244648371769
```

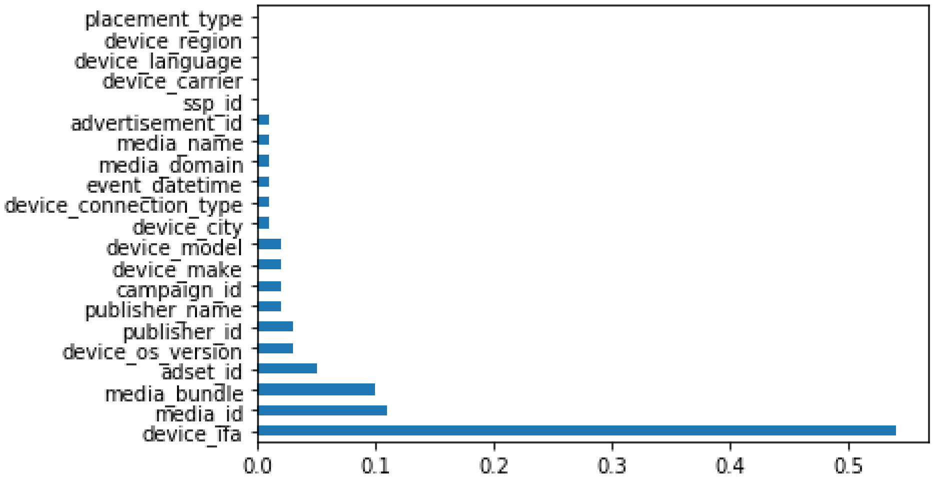
In [5]:

```
#plot feature importance
import matplotlib.pylab as plt
%matplotlib inline

feat_importances = pd.Series(adaMod_without_ap.feature_importances_, index=x_train.columns)
feat_importances.nlargest(len(x_train.columns)).plot(kind='barh')
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cf70563d88>
```



In [6]:

```
#Feature selection
from sklearn.feature_selection import SelectFromModel

sel = SelectFromModel(AdaBoostClassifier(base_estimator=None, n_estimators=100, learning_rate=1.0),
sel.fit(train_data, target_data)

selected_feat= train_data.columns[(sel.get_support())]
print(selected_feat)
```

```
Index(['event_datetime', 'ssp_id', 'campaign_id', 'adset_id', 'media_id',
       'media_name', 'media_bundle', 'media_domain', 'publisher_id',
       'publisher_name', 'device_ifa', 'device_os_version', 'device_model',
       'device_make', 'device_connection_type', 'device_city',
       'advertisement_id'],
      dtype='object')
```

In [7]:

```
x_train_f=x_train[selected_feat]
x_train_f.head()
```

Out[7]:

me	media_bundle	media_domain	publisher_id	publisher_name	device_ifa	device_os_version	c
709	0.202709	0.093845	0.202727	0.031964	0.198038	0.063991	
414	0.034252	0.093845	0.046414	0.046414	0.090909	0.130898	
598	0.274011	0.093845	0.283167	0.283167	0.090909	0.097264	
598	0.274011	0.093845	0.169584	0.273958	0.444326	0.156081	
474	0.052419	0.093845	0.036538	0.046549	0.024449	0.042296	

In [8]:

```
x_valid_f = x_valid[selected_feat]
x_valid_f.head()
```

Out[8]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundle
171615	0.608696	0.193911	0.101862	0.101862	0.087301	0.087301	0.084604
5111013	0.260870	0.144511	0.075303	0.174953	0.079612	0.079612	0.084604
626964	0.652174	0.018592	0.005527	0.005889	0.018766	0.024986	0.013298
730303	0.000000	0.046613	0.091650	0.102840	0.033853	0.033853	0.030615
562068	0.478261	0.193911	0.097297	0.097297	0.087301	0.087301	0.084604

In [10]:

```
#AdaBoostClassifier_without_ap after feature selection
from sklearn.ensemble import AdaBoostClassifier

adaMod = AdaBoostClassifier(base_estimator=None, n_estimators=100, learning_rate=1.0)
adaMod_without_ap=adaMod.fit(x_train_f, y_train)

print('training set_with_ap accuracy :', adaMod_without_ap.score(x_train_f, y_train))
print('validation set_with_ap accuracy :', adaMod_without_ap.score(x_valid_f, y_valid))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid, adaMod_without_ap.predi
```

```
training set_with_ap accuracy : 0.9231430303030304
validation set_with_ap accuracy : 0.9233425454545454
validation set_with_ap log loss:  0.5963244648371613
```

4.2audience_profile 포함하는 데이터세트

In [11]:

```
#AdaBoostClassifier_with_ap
from sklearn.ensemble import AdaBoostClassifier

adaMod = AdaBoostClassifier(base_estimator=None, n_estimators=100, learning_rate=1.0)
adaMod_with_ap=adaMod.fit(x_train_with_ap, y_train_with_ap)

print('training set_with_ap accuracy :', adaMod_with_ap.score(x_train_with_ap, y_train_with_ap))
print('validation set_with_ap accuracy :', adaMod_with_ap.score(x_valid_with_ap, y_valid_with_ap))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid_with_ap, adaMod_with_ap.
```

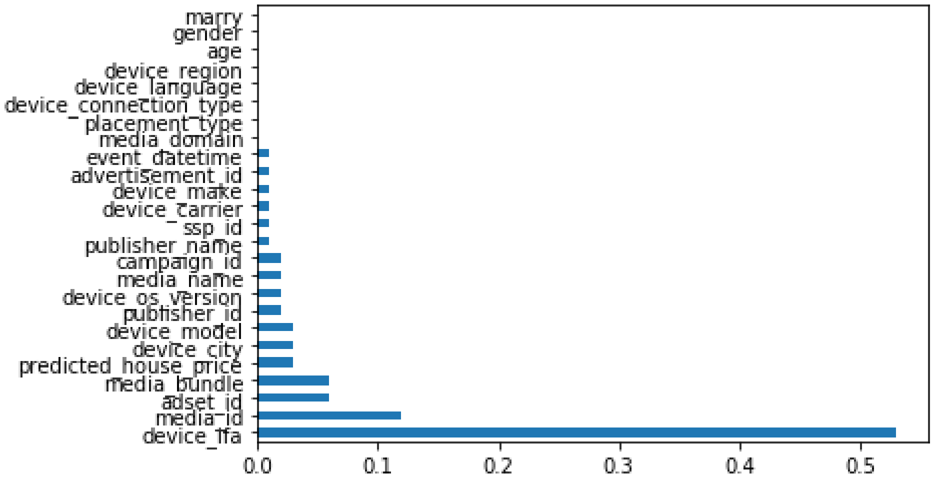
```
training set_with_ap accuracy : 0.9279251291211992
validation set_with_ap accuracy : 0.9274631702693257
validation set_with_ap log loss:  0.5941373194999544
```

In [12]:

```
#plot feature importance
feat_importances = pd.Series(adaMod_with_ap.feature_importances_, index=x_train_with_ap.columns)
feat_importances.nlargest(len(x_train_with_ap.columns)).plot(kind='barh')
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cf70970588>
```



In [13]:

```
#Feature selection
from sklearn.feature_selection import SelectFromModel

sel = SelectFromModel(AdaBoostClassifier(base_estimator=None, n_estimators=100, learning_rate=1.0),
sel.fit(train_data_with_ap, target_data_with_ap)

selected_feat_ap= train_data_with_ap.columns[(sel.get_support())]
print(selected_feat_ap)
```

```
Index(['event_datetime', 'ssp_id', 'campaign_id', 'adset_id', 'media_id',
       'media_name', 'media_bundle', 'publisher_id', 'publisher_name',
       'device_ifa', 'device_os_version', 'device_model', 'device_carrier',
       'device_make', 'device_city', 'advertisement_id',
       'predicted_house_price'],
      dtype='object')
```

In [14]:

```
x_train_with_ap_f=x_train_with_ap[selected_feat_ap]
x_train_with_ap_f.head()
```

Out[14]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundl
289497	0.826087	0.189793	0.134734	0.154823	0.282208	0.278601	0.27263
156311	0.782609	0.015880	0.048118	0.052307	0.016541	0.023879	0.02174
288981	0.956522	0.015880	0.024206	0.024206	0.016541	0.023879	0.02174
36503	0.260870	0.189793	0.280284	0.280284	0.282208	0.278601	0.27263
369562	0.260870	0.015880	0.026130	0.026130	0.016541	0.023879	0.02174

In [15]:

```
x_valid_with_ap_f = x_valid_with_ap[selected_feat_ap]
x_valid_with_ap_f.head()
```

Out[15]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundl
327879	1.000000	0.015880	0.025626	0.028735	0.016541	0.023879	0.00452
58542	0.304348	0.015880	0.048118	0.042266	0.016541	0.023879	0.02137
315633	0.608696	0.189793	0.097937	0.110151	0.282208	0.278601	0.27263
100549	0.086957	0.048989	0.102987	0.103697	0.080721	0.023879	0.08936
238529	0.347826	0.189793	0.111476	0.143055	0.034730	0.034730	0.03668

In [16]:

```
#AdaBoostClassifier_with_ap after feature selection
from sklearn.ensemble import AdaBoostClassifier

adaMod = AdaBoostClassifier(base_estimator=None, n_estimators=100, learning_rate=1.0)
adaMod_with_ap=adaMod.fit(x_train_with_ap_f, y_train_with_ap)

print('training set_with_ap accuracy :', adaMod_with_ap.score(x_train_with_ap_f, y_train_with_ap))
print('validation set_with_ap accuracy :', adaMod_with_ap.score(x_valid_with_ap_f, y_valid_with_ap))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid_with_ap, adaMod_with_ap.
```

```
training set_with_ap accuracy : 0.9279251291211992
validation set_with_ap accuracy : 0.9274631702693257
validation set_with_ap log loss:  0.5941373194999541
```

3.Final Validation

In [ ]:

```
#final feature selection
train_data_with_ap_f=train_data_with_ap[selected_feat_ap]
train_data_without_ap_f=train_data[selected_feat]
```

In [ ]:

```
adaMod = AdaBoostClassifier(base_estimator=None, n_estimators=100, learning_rate=1.0)
adaMod_without_ap=adaMod.fit(train_data_without_ap_f, target_data)
adaMod = AdaBoostClassifier(base_estimator=None, n_estimators=100, learning_rate=1.0)
adaMod_with_ap=adaMod.fit(train_data_with_ap_f, target_data_with_ap)
```

In [ ]:

```
#Cross validation(최종 검증)
from sklearn.model_selection import StratifiedKFold,cross_val_score
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
score_ap= cross_val_score(adaMod_with_ap,train_data_with_ap_f,target_data_with_ap,scoring="neg_log_
score= cross_val_score(adaMod_without_ap,train_data_without_ap_f,target_data,scoring="neg_log_loss"
```

In [ ]:

```
print("cross validation score of model_with_ap: ",score_ap.mean())
print("cross validation score of model_without_ap: ",score.mean())
```

4.Model saving

In [18]:

```
import pickle
with open("adaMod_without_ap.sav", 'wb') as file:
    pickle.dump(adaMod_without_ap, file)
with open("adaMod_with_ap.sav", 'wb') as file:
    pickle.dump(adaMod_with_ap, file)
```

In [ ]:

```
```

1. Test set reading

In [1]:

```
#test data reading
#bid_id 만 read vs 데이터 세트 preprocess에서 구성하여 읽기
import pandas as pd
raw_test= pd.read_csv('test.csv')
raw_test.head()
```

Out[1]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	2019-10-11 00:00:05.593	jlLrN0gGpx	nwf1A3O5cO	7Noz5lnNj5	yH0QQDoPNl	kleE1J0KCa	
1	2019-10-11 00:00:06.024	zA3WyymOcJ	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kleE1J0KCa	9
2	2019-10-11 00:00:06.126	PwIj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kleE1J0KCa	9
3	2019-10-11 00:00:06.598	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kleE1J0KCa	Z
4	2019-10-11 00:00:06.639	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9

5 rows × 24 columns

In [2]:

```
#test data 와 audience profile merging
#필요열만 읽어들이기
import gc
import pandas as pd
# audience_profile의 크기가 ram 용량에 비해 커서 작은 단위로 나누어 ram에 올리고 지우기
n=1
for chunk in pd.read_csv('audience_profile.csv',sep='delimiter', delimiter = "!@#", chunksize=5000
    if n ==1:
        test_with_ap =pd.merge(raw_test, chunk, how='inner', on='device_ifa')
    else:
        test_with_ap = pd.concat([test_with_ap,pd.merge(raw_test, chunk, how='inner', on='device_if
    del chunk
    gc.collect()  #ram에서 삭제
    n+=1

test_with_ap.head(5)
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:7: ParserWarning: F
alling back to the 'python' engine because the 'c' engine does not support regex sep
arators (separators > 1 char and different from '₩s+' are interpreted as regex); you
can avoid this warning by specifying engine='python'.
  import sys
```

Out[2]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	2019-10-11 00:00:23.202	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kleE1J0KCa	E
1	2019-10-11 00:00:27.861	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9b
2	2019-10-11 03:52:07.245	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kleE1J0KCa	9b
3	2019-10-11 00:00:35.423	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kleE1J0KCa	hk
4	2019-10-11 00:00:52.950	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kleE1J0KCa	

5 rows × 31 columns

In [3]:

```
bid_id_with_ap=test_with_ap['bid_id']
bid_id_with_ap
```

Out[3]:

```
0        EHTpBC8c9L
1        OJdfjVSNi8
2        d7g9YcPv7u
3        KJryVcuWQc
4        ObZPTVVYcA
            ...
10765    faWjGdHRmw
10766    mAI8SUv657
10767    kuCI8qgDp9
10768    SL3vs75mac
10769    dv5HX2ehaK
Name: bid_id, Length: 214642, dtype: object
```

In [4]:

```
#test set without ap set 생성
test=pd.merge(raw_test,test_with_ap[['device_ifa','gender']], how='left',on='device_ifa')
test=test[(test['gender'].isnull()==1)].drop(['gender'],axis=1)
test
```

Out[4]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	medi
0	2019-10-11 00:00:05.593	jILrN0gGpx	nwf1A3O5cO	7Noz5InNj5	yH0QQDoPNI	kIeE1J0KCa	j7H2fV
2	2019-10-11 00:00:06.126	PwIj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kIeE1J0KCa	9bC9qJ8
3	2019-10-11 00:00:06.598	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kIeE1J0KCa	ZrCGAwg
4	2019-10-11 00:00:06.639	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9bC9qJ8
6	2019-10-11 00:00:07.166	PAG5INDY6L	y7QKxSwhwV	dS6rpIpBHY	G0n6acDmBk	tg9mzu7kFm	8eTC2
...	...	...	...	...	...	...	
	2019-10-12						

In [5]:

```
bid_id_without_ap=test['bid_id']
```

2.Test set preprocessing

In [6]:

```
test_with_ap['predicted_house_price']=test_with_ap['predicted_house_price'].fillna(value=test_with_a
```

In [7]:

```
#시간 정보만 추출(test_with_ap)
test_with_ap['event_datetime'] = pd.to_datetime(test_with_ap['event_datetime']).dt.hour.astype(int)
test_with_ap.head()
```

Out[7]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kleE1J0KCa	E
1	0	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9b
2	3	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kleE1J0KCa	9b
3	0	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kleE1J0KCa	hk
4	0	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLI	FiSRHSfVaf	kleE1J0KCa	

5 rows × 31 columns

In [8]:

```
#시간 정보만 추출(test)
test['event_datetime'] = pd.to_datetime(test['event_datetime']).dt.hour.astype(int)
test.head()
```

Out[8]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	jlLrN0gGpx	nwf1A3O5cO	7Noz5lnNj5	yH0QQDoPNl	kleE1J0KCa	
2	0	Pwlj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kleE1J0KCa	9
3	0	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kleE1J0KCa	Z
4	0	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9
6	0	PAG5INDY6L	y7QKxSwhwV	dS6rpIpBHY	G0n6acDmBk	tg9mzu7kFm	

5 rows × 24 columns

In [9]:

```
# converting gender,marry feature to numeical value
test_with_ap['gender']=test_with_ap['gender'].map({'M':0,
                                                    'F':1})
test_with_ap['marry']=test_with_ap['marry'].map({'M':0,
                                                  'S':1})
test_with_ap.head()
```

Out[9]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kleE1J0KCa	E
1	0	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9t
2	3	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kleE1J0KCa	9t
3	0	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kleE1J0KCa	hk
4	0	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kleE1J0KCa	

5 rows × 31 columns

In [10]:

```
#encoding을 위해 age feature str으로 변환
test_with_ap['age']=test_with_ap['age'].astype('str')
```

In [11]:

```
test_with_ap=test_with_ap.drop(['install_pack','bid_id','cate_code','asset_index','device_os','devic
test = test.drop(['bid_id','device_os','device_country'],axis=1)
```

3.Feature engineering

Target encoding

In [18]:

```
#target encoding
from category_encoders import TargetEncoder
import pickle

enc_with_ap = pickle.load(open('enc_preprocess_1.sav', 'rb'))

numeric_test_with_ap=enc_with_ap.transform(test_with_ap)
numeric_test_with_ap
```

Out[18]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_nam
0	0	0.143941	0.042074	0.141686	0.139161	0.273404	0.27860
1	0	0.039832	0.057104	0.115930	0.139161	0.032964	0.03296
2	3	0.039832	0.047038	0.109618	0.139161	0.032964	0.03296
3	0	0.009178	0.008072	0.008072	0.139161	0.016541	0.00917
4	0	0.189793	0.285726	0.285726	0.139161	0.282208	0.27860
...	...	...	...	...	...	...	.
10765	0	0.011040	0.012040	0.006623	0.021026	0.016541	0.01724
10766	0	0.039832	0.105717	0.130554	0.139161	0.090909	0.09090
10767	0	0.189793	0.135373	0.136251	0.139161	0.023806	0.02380
10768	0	0.189793	0.106897	0.142785	0.139161	0.034730	0.03473
10769	0	0.143941	0.087930	0.136877	0.139161	0.273404	0.27860

214642 rows × 25 columns

In [19]:

```
enc = pickle.load(open('enc_preprocess_2.sav', 'rb'))

numeric_test=enc.transform(test)
numeric_test
```

Out[19]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_nar
0	0	0.193911	0.254406	0.244391	0.143059	0.283167	0.2796
2	0	0.046613	0.046747	0.104764	0.143059	0.033853	0.0338
3	0	0.144511	0.064228	0.092414	0.143059	0.025916	0.0259
4	0	0.046613	0.056631	0.119014	0.143059	0.033853	0.0338
6	0	0.184407	0.190157	0.201794	0.184516	0.174362	0.1743
...	...	...	...	...	...	...	
782300	0	0.018592	0.126839	0.003182	0.022644	0.018766	0.0249
782301	0	0.046613	0.108579	0.133637	0.143059	0.020502	0.0205
782304	0	0.193911	0.088609	0.136681	0.143059	0.260223	0.2602
782305	0	0.144511	0.136572	0.137860	0.143059	0.273958	0.2796
782307	0	0.193911	0.287327	0.287327	0.143059	0.283167	0.2796

335358 rows × 21 columns

In [20]:

```
#test_with_ap set
numeric_test_with_ap = numeric_test_with_ap[['event_datetime', 'ssp_id', 'campaign_id', 'adset_id',
       'media_name', 'media_bundle', 'publisher_id', 'publisher_name',
       'device_ifa', 'device_os_version', 'device_model', 'device_carrier',
       'device_make', 'device_city', 'advertisement_id',
       'predicted_house_price']]
numeric_test_with_ap.head()
```

Out[20]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundle	pul
0	0	0.143941	0.042074	0.141686	0.273404	0.278601	0.272632	
1	0	0.039832	0.057104	0.115930	0.032964	0.032964	0.029874	
2	3	0.039832	0.047038	0.109618	0.032964	0.032964	0.029874	
3	0	0.009178	0.008072	0.008072	0.016541	0.009178	0.009178	
4	0	0.189793	0.285726	0.285726	0.282208	0.278601	0.272632	

In [21]:

```
#test set
numeric_test = numeric_test[['event_datetime', 'ssp_id', 'campaign_id', 'adset_id', 'media_id',
       'media_name', 'media_bundle', 'media_domain', 'publisher_id',
       'publisher_name', 'device_ifa', 'device_os_version', 'device_model',
       'device_make', 'device_connection_type', 'device_city',
       'advertisement_id']]
numeric_test.head()
```

Out[21]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundle	me
0	0	0.193911	0.254406	0.244391	0.283167	0.279698	0.274011	
2	0	0.046613	0.046747	0.104764	0.033853	0.033853	0.030615	
3	0	0.144511	0.064228	0.092414	0.025916	0.025916	0.030615	
4	0	0.046613	0.056631	0.119014	0.033853	0.033853	0.030615	
6	0	0.184407	0.190157	0.201794	0.174362	0.174362	0.173042	

scaling

In [22]:

```
ㅓ일
reprocessing import MinMaxScaler
xScaler()
 th_ap[['predicted_house_price','event_datetime']] = Scaler.fit_transform(numeric_test_with_ap[['pred
 th_ap
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  after removing the cwd from sys.path.
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:494: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self.obj[item] = s
```

Out[22]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundle
0	0.000000	0.143941	0.042074	0.141686	0.273404	0.278601	0.272632
1	0.000000	0.039832	0.057104	0.115930	0.032964	0.032964	0.029874
2	0.130435	0.039832	0.047038	0.109618	0.032964	0.032964	0.029874
3	0.000000	0.009178	0.008072	0.008072	0.016541	0.009178	0.009178
4	0.000000	0.189793	0.285726	0.285726	0.282208	0.278601	0.272632
...	...	...	...	...	...	...	...
10765	0.000000	0.011040	0.012040	0.006623	0.016541	0.017241	0.017241
10766	0.000000	0.039832	0.105717	0.130554	0.090909	0.090909	0.090909
10767	0.000000	0.189793	0.135373	0.136251	0.023806	0.023806	0.026056
10768	0.000000	0.189793	0.106897	0.142785	0.034730	0.034730	0.036685
10769	0.000000	0.143941	0.087930	0.136877	0.273404	0.278601	0.272632

214642 rows × 17 columns

In [23]:

```
#ap 비포함 파일
from sklearn.preprocessing import MinMaxScaler
Scaler = MinMaxScaler()
numeric_test[['event_datetime']] = Scaler.fit_transform(numeric_test[['event_datetime']])
numeric_test
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  after removing the cwd from sys.path.
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:494: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self.obj[item] = s
```

Out[23]:

	event_datetime	ssp_id	campaign_id	adset_id	media_id	media_name	media_bundl
0	0.0	0.193911	0.254406	0.244391	0.283167	0.279698	0.27401
2	0.0	0.046613	0.046747	0.104764	0.033853	0.033853	0.03061
3	0.0	0.144511	0.064228	0.092414	0.025916	0.025916	0.03061
4	0.0	0.046613	0.056631	0.119014	0.033853	0.033853	0.03061
6	0.0	0.184407	0.190157	0.201794	0.174362	0.174362	0.17304
...	...	...	...	...	...	...	..
782300	0.0	0.018592	0.126839	0.003182	0.018766	0.024986	0.00456
782301	0.0	0.046613	0.108579	0.133637	0.020502	0.020502	0.08664
782304	0.0	0.193911	0.088609	0.136681	0.260223	0.260223	0.25874
782305	0.0	0.144511	0.136572	0.137860	0.273958	0.279698	0.27401
782307	0.0	0.193911	0.287327	0.287327	0.283167	0.279698	0.27401

335358 rows × 17 columns

3.Model loading

In [24]:

```
#Model reading
import pickle

adaMod = pickle.load(open('adaMod_without_ap.sav', 'rb'))
adaMod_with_ap = pickle.load(open('adaMod_with_ap.sav', 'rb'))
```

3.Prediction

3.AdaBoostClassifier

In [25]:

```
#test_with_ap 세트 확률과 bid id 연결
probs_with_ap = adaMod_with_ap.predict_proba(numeric_test_with_ap)
probs_with_ap = probs_with_ap[:, 1]
bid_ap = pd.DataFrame(bid_id_with_ap)
bid_ap['probs']=probs_with_ap
bid_ap
```

Out[25]:

	bid_id	probs
0	EHTpBC8c9L	0.495820
1	0JdfjVSNi8	0.461617
2	d7g9YcPv7u	0.462050
3	KJryVcuWQc	0.488016
4	ObZPTVVYcA	0.584312
...	...	...
10765	faWjGdHRmw	0.489561
10766	mAl8SUv657	0.494305
10767	kuCI8qgDp9	0.401552
10768	SL3vs75mac	0.492840
10769	dv5HX2ehaK	0.405182

214642 rows × 2 columns

In [26]:

```
#test 세트 확률과 bid id 연결

probs = adaMod.predict_proba(numeric_test)
probs = probs[:, 1]
bid = pd.DataFrame(bid_id_without_ap)
bid['probs']=probs
bid
```

Out[26]:

	bid_id	probs
0	jILrN0gGpx	0.495454
2	PwIj11RYvM	0.493058
3	W0o0KwmTSQ	0.492427
4	UpL3kLWqZy	0.492359
6	PAG5INDY6L	0.495890
...	...	...
782300	cLU5mO3z89	0.394232
782301	2YxtmVzvpB	0.492332
782304	bSxh3i0gN3	0.495543
782305	LAyxamwNxm	0.401920
782307	W8XuFXZw4v	0.401775

335358 rows × 2 columns

In [27]:

```
#모든 확률값 병합

bid_all= pd.concat([bid,bid_ap])
submit = pd.merge(raw_test['bid_id'],bid_all,how="left",on='bid_id')
submit
```

Out[27]:

```
            bid_id     probs
     0  jlLrN0gGpx  0.495454
     1  zA3WyymOcJ  0.492857
     2  PwIj11RYvM  0.493058
     3  W0o0KwmTSQ  0.492427
     4  UpL3kLWqZy  0.492359
   ...         ...       ...
549995  bSxh3i0gN3  0.495543
549996  LAyxamwNxm  0.401920
549997  3sF8PXgPom  0.584624
549998  W8XuFXZw4v  0.401775
549999  WNke5qEQC1  0.495546

550000 rows × 2 columns
```

In [28]:

```
submit.to_csv('submit_adaMod_ap.csv', index=False,header=False)
```

In [ ]:

```
```

1.Data Reading ¶

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.metrics import accuracy_score
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split

from sklearn.metrics import log_loss
```

In [2]:

```
import pandas as pd

train_with_ap = pd.read_csv('train_preprocess_1.csv')
train_with_ap.head()
```

Out[2]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_na
0	0	0.086957	0.048989	0.146178	0.011869	0.052756	0.000272	0.023
1	0	0.086957	0.143941	0.093064	0.104093	0.139161	0.105418	0.105
2	0	0.652174	0.189793	0.096605	0.123112	0.139161	0.119479	0.119
3	0	1.000000	0.189793	0.096605	0.123112	0.139161	0.119479	0.119
4	0	0.956522	0.143941	0.096605	0.123112	0.139161	0.105418	0.105

5 rows × 26 columns

In [3]:

```
train = pd.read_csv('train_preprocess_2.csv')
train.head()
```

Out[3]:

	click	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_na
0	0	0.0	0.087898	0.130311	0.028476	0.022644	0.002079	0.002
1	0	0.0	0.036473	0.030543	0.016824	0.022644	0.018766	0.024
2	0	0.0	0.018592	0.023756	0.023756	0.022644	0.018766	0.024
3	1	0.0	0.193911	0.287327	0.287327	0.143059	0.283167	0.279
4	0	0.0	0.018592	0.005527	0.005605	0.022644	0.018766	0.024

5 rows × 22 columns

2.Modeling

In [4]:

```
# 둘다 나눠줌
from sklearn.model_selection import train_test_split

train_data_with_ap = train_with_ap.drop('click',axis=1)
target_data_with_ap = train_with_ap['click']

x_train_with_ap, x_valid_with_ap, y_train_with_ap, y_valid_with_ap = train_test_split(train_data_wit
##################################################
train_data = train.drop('click',axis=1)
target_data = train['click']

x_train, x_valid, y_train, y_valid = train_test_split(train_data, target_data)
```

6.LightgbmClassifier

In [5]:

```
! pip install lightgbm
```

```
Requirement already satisfied: lightgbm in c:₩users₩user₩anaconda3₩lib₩site-packages
(2.3.1)
Requirement already satisfied: numpy in c:₩users₩user₩anaconda3₩lib₩site-packages (f
rom lightgbm) (1.16.5)
Requirement already satisfied: scikit-learn in c:₩users₩user₩anaconda3₩lib₩site-pack
ages (from lightgbm) (0.21.3)
Requirement already satisfied: scipy in c:₩users₩user₩anaconda3₩lib₩site-packages (f
rom lightgbm) (1.3.1)
Requirement already satisfied: joblib>=0.11 in c:₩users₩user₩anaconda3₩lib₩site-pack
ages (from scikit-learn->lightgbm) (0.13.2)
```

6.1audience_profile 없는 데이터세트

In [6]:

```
import sklearn
#LightgbmClassifier_without_ap
from lightgbm import LGBMClassifier

LGBMC = LGBMClassifier(n_estimators=100,max_depth=5, n_jobs=-1)
LGBMC_without_ap=LGBMC.fit(x_train, y_train)

print('training set_without_ap accuracy :', LGBMC_without_ap.score(x_train, y_train))
print('validation set_without_ap accuracy :', LGBMC_without_ap.score(x_valid, y_valid))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid, LGBMC_without_ap.predic
```

```
training set_without_ap accuracy : 0.9239187878787879
validation set_without_ap accuracy : 0.9235134545454545
validation set_with_ap log loss:  0.16666479572149098
```

In [7]:

```
#Feature selection without ap
from sklearn.feature_selection import SelectFromModel

sel = SelectFromModel(LGBMClassifier(n_estimators=100,max_depth=5, n_jobs=-1),threshold = "median")
sel.fit(train_data, target_data)

selected_feat= train_data.columns[(sel.get_support())]
print(selected_feat)
```

```
Index(['event_datetime', 'campaign_id', 'adset_id', 'media_id', 'media_name',
       'media_bundle', 'publisher_id', 'publisher_name', 'device_ifa',
       'device_model', 'advertisement_id'],
      dtype='object')
```

In [8]:

```
#final feature selection
x_train_f=x_train[selected_feat]
x_train_f.head()
```

Out[8]:

	event_datetime	campaign_id	adset_id	media_id	media_name	media_bundle	pub
3095208	0.652174	0.041295	0.017454	0.007194	0.024986	0.012811	
2537191	0.434783	0.056942	0.015316	0.018766	0.024986	0.034252	
919794	0.652174	0.034143	0.045397	0.000266	0.024986	0.000319	
1387861	0.478261	0.129479	0.031134	0.018766	0.294118	0.294118	
349252	0.565217	0.061817	0.015354	0.018766	0.024986	0.003820	

In [9]:

```
x_valid_f = x_valid[selected_feat]
x_valid_f.head()
```

Out[9]:

	event_datetime	campaign_id	adset_id	media_id	media_name	media_bundle	publish
2595429	0.608696	0.009610	0.009610	0.018766	0.008994	0.008994	0.0
5284137	0.652174	0.047194	0.015895	0.018766	0.024986	0.034252	0.0
5129651	0.304348	0.033107	0.011905	0.011011	0.011011	0.016914	0.0
2639224	0.695652	0.091650	0.102840	0.283167	0.279698	0.274011	0.2
3144017	0.956522	0.107643	0.038007	0.017412	0.017412	0.021556	0.0

In [10]:

```
#LightgbmClassifier_without_ap
from lightgbm import LGBMClassifier

LGBMC = LGBMClassifier(n_estimators=100,max_depth=5, n_jobs=-1)
LGBMC_without_ap=LGBMC.fit(x_train_f, y_train)

print('training set_without_ap accuracy :', LGBMC_without_ap.score(x_train_f, y_train))
print('validation set_without_ap accuracy :', LGBMC_without_ap.score(x_valid_f, y_valid))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid, LGBMC_without_ap.predic
```

```
training set_without_ap accuracy : 0.9238943030303031
validation set_without_ap accuracy : 0.9235338181818182
validation set_with_ap log loss:  0.16664625109867365
```

6.2audience_profile 포함하는 데이터세트

In [11]:

```
#LightgbmClassifier_with_ap
from lightgbm import LGBMClassifier

LGBMC = LGBMClassifier(n_estimators=100,max_depth=5, n_jobs=-1)
LGBMC_with_ap=LGBMC.fit(x_train_with_ap, y_train_with_ap)

print('training set_with_ap accuracy :', LGBMC_with_ap.score(x_train_with_ap, y_train_with_ap))
print('validation set_with_ap accuracy :', LGBMC_with_ap.score(x_valid_with_ap, y_valid_with_ap))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid_with_ap, LGBMC_with_ap.p
```

```
training set_with_ap accuracy : 0.9289948139701547
validation set_with_ap accuracy : 0.9271772169326739
validation set_with_ap log loss:  0.1595694247723334
```

In [12]:

```
#Feature selection with
from sklearn.feature_selection import SelectFromModel

sel = SelectFromModel(LGBMClassifier(n_estimators=100,max_depth=5, n_jobs=-1),threshold = "0.25*mec
sel.fit(train_data_with_ap, target_data_with_ap)

selected_feat_ap= train_data_with_ap.columns[(sel.get_support())]
print(selected_feat_ap)
```

```
Index(['event_datetime', 'ssp_id', 'campaign_id', 'adset_id', 'placement_type',
       'media_id', 'media_name', 'media_bundle', 'publisher_id',
       'publisher_name', 'device_ifa', 'device_os_version', 'device_model',
       'device_carrier', 'device_region', 'device_city', 'advertisement_id',
       'age', 'marry', 'predicted_house_price'],
      dtype='object')
```

In [13]:

```
x_train_with_ap_f=x_train_with_ap[selected_feat_ap]
x_train_with_ap_f.head()
```

Out[13]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_nar
133237	0.130435	0.143941	0.075370	0.092102	0.139161	0.023305	0.0240
234684	0.130435	0.088627	0.057458	0.014884	0.021026	0.002239	0.0022
207638	0.347826	0.048989	0.021239	0.014625	0.021026	0.021441	0.0238
172083	0.130435	0.189793	0.281599	0.281599	0.139161	0.282208	0.2786
122362	0.478261	0.015880	0.048118	0.042266	0.021026	0.016541	0.0238

In [14]:

```
x_valid_with_ap_f = x_valid_with_ap[selected_feat_ap]
x_valid_with_ap_f.head()
```

Out[14]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_nar
280129	0.391304	0.189793	0.131316	0.099466	0.139161	0.282208	0.2786
345420	0.217391	0.189793	0.042074	0.051673	0.139161	0.031396	0.0313
221721	0.391304	0.015880	0.048118	0.052307	0.021026	0.016541	0.0238
207250	0.956522	0.036740	0.047038	0.022556	0.021026	0.016541	0.0238
171652	0.130435	0.048989	0.102987	0.103697	0.139161	0.035627	0.0238

In [15]:

```
#LightgbmClassifier_with_ap
from lightgbm import LGBMClassifier

LGBMC = LGBMClassifier(n_estimators=100,max_depth=5, n_jobs=-1)
LGBMC_with_ap=LGBMC.fit(x_train_with_ap_f, y_train_with_ap)

print('training set_with_ap accuracy :', LGBMC_with_ap.score(x_train_with_ap_f, y_train_with_ap))
print('validation set_with_ap accuracy :', LGBMC_with_ap.score(x_valid_with_ap_f, y_valid_with_ap))
print('validation set_with_ap log loss: ', sklearn.metrics.log_loss(y_valid_with_ap, LGBMC_with_ap.p
```

```
training set_with_ap accuracy : 0.92893126833556333
validation set_with_ap accuracy : 0.9272725347115578
validation set_with_ap log loss:  0.15952251551632274
```

final validation

In [16]:

```
#final feature selection
train_data_with_ap_f=train_data_with_ap[selected_feat_ap]
train_data_without_ap_f=train_data[selected_feat]
```

In [17]:

```
from lightgbm import LGBMClassifier
LGBMC = LGBMClassifier(n_estimators=100,max_depth=5, n_jobs=-1)
LGBMC_without_ap=LGBMC.fit(train_data_without_ap_f, target_data)
```

In [18]:

```
LGBMC = LGBMClassifier(n_estimators=100,max_depth=5, n_jobs=-1)
LGBMC_with_ap=LGBMC.fit(train_data_with_ap_f, target_data_with_ap)
```

In [19]:

```
#Cross validation(최종 검증)
from sklearn.model_selection import StratifiedKFold,cross_val_score
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
score_ap= cross_val_score(LGBMC_with_ap,train_data_with_ap_f,target_data_with_ap,scoring="neg_log_l
score= cross_val_score(LGBMC_without_ap,train_data_without_ap_f,target_data,scoring="neg_log_loss",
```

In [20]:

```
print("cross validation score of model_with_ap: ",score_ap.mean())
print("cross validation score of model_without_ap: ",score.mean())
```

```
cross validation score of model_with_ap:  -0.15824082443423423
cross validation score of model_without_ap:  -0.16636872993329427
```

4.Model saving

In [21]:

```
import pickle
with open("lgbm_without_ap.sav", 'wb') as file:
    pickle.dump(LGBMC_without_ap, file)
with open("lgbm_with_ap.sav", 'wb') as file:
    pickle.dump(LGBMC_with_ap, file)
```

In [ ]:

1. Test set reading

In [1]:

```
#test data reading
#bid_id 만 read vs 데이터 세트 preprocess에서 구성하여 읽기
import pandas as pd
raw_test= pd.read_csv('test.csv')
raw_test.head()
```

Out[1]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	2019-10-11 00:00:05.593	jlLrN0gGpx	nwf1A3O5cO	7Noz5lnNj5	yH0QQDoPNl	kleE1J0KCa	
1	2019-10-11 00:00:06.024	zA3WyymOcJ	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kleE1J0KCa	9
2	2019-10-11 00:00:06.126	Pwlj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kleE1J0KCa	9
3	2019-10-11 00:00:06.598	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kleE1J0KCa	Z
4	2019-10-11 00:00:06.639	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9

5 rows × 24 columns

In [2]:

```
#test data 와 audience profile merging
#필요열만 읽어들이기
import gc
import pandas as pd
# audience_profile의 크기가 ram 용량에 비해 커서 작은 단위로 나누어 ram에 올리고 지우기
n=1
for chunk in pd.read_csv('audience_profile.csv',sep='delimiter', delimiter = "!@#", chunksize=50000
    if n ==1:
        test_with_ap =pd.merge(raw_test, chunk, how='inner', on='device_ifa')
    else:
        test_with_ap = pd.concat([test_with_ap,pd.merge(raw_test, chunk, how='inner', on='device_ifa
    del chunk
    gc.collect()  #ram에서 삭제
    n+=1

test_with_ap.head(5)
```

```
C:₩Users₩user₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:7: ParserWarning: Fa
lling back to the 'python' engine because the 'c' engine does not support regex sepa
rators (separators > 1 char and different from '₩s+' are interpreted as regex); you
can avoid this warning by specifying engine='python'.
  import sys
```

Out[2]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	2019-10-11 00:00:23.202	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kIeE1J0KCa	E
1	2019-10-11 00:00:27.861	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9
2	2019-10-11 03:52:07.245	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kIeE1J0KCa	9
3	2019-10-11 00:00:35.423	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kIeE1J0KCa	hk
4	2019-10-11 00:00:52.950	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kIeE1J0KCa	

5 rows × 31 columns

In [3]:

```
bid_id_with_ap=test_with_ap['bid_id']
bid_id_with_ap
```

Out[3]:

```
0        EHTpBC8c9L
1        OJdfjVSNi8
2        d7g9YcPv7u
3        KJryVcuWQc
4        ObZPTVVYcA
            ...
10765    faWjGdHRmw
10766    mAI8SUv657
10767    kuCI8qgDp9
10768    SL3vs75mac
10769    dv5HX2ehaK
Name: bid_id, Length: 214642, dtype: object
```

In [4]:

```
#test set without ap set 생성
test=pd.merge(raw_test,test_with_ap[['device_ifa','gender']], how='left',on='device_ifa')
test=test[(test['gender'].isnull()==1)].drop(['gender'],axis=1)
test
```

3	2019-10-11 00:00:06.598	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kIeE1J0KCa	ZrCGAw
4	2019-10-11 00:00:06.639	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9bC9qJ
6	2019-10-11 00:00:07.166	PAG5INDY6L	y7QKxSwhwV	dS6rpIpBHY	G0n6acDmBk	tg9mzu7kFm	8eTC2
...	...	...	...	...	...	...	
782300	2019-10-12 00:01:21.559	cLU5mO3z89	SrN77Arvqh	2NIOV3Vhjb	RBkPIE7zXi	1pcQ3RJgQt	hkFCnTp
782301	2019-10-12 00:01:21.570	2YxtmVzvpB	Uox85xVMSC	SHpt2lzYOT	AIVulu17z5	kIeE1J0KCa	JzMEh8R
782304	2019-10-12 00:01:21.886	bSxh3i0gN3	nwf1A3O5cO	w6ERRwu6pk	tr7cYrEXuJ	kIeE1J0KCa	WG9YH
782305	2019-10-12 00:01:22.026	LAyxamwNxm	M6QaRvdZ8h	wvAODZefbN	GdGZ3dDmhQ	kIeE1J0KCa	EWk3G
782307	2019-10-12	W8XuFXZw4v	nwf1A3O5cO	gR4Xa60DLI	FiSRHSfVaf	kIeE1J0KCa	i7H2fV

In [5]:

```
bid_id_without_ap=test['bid_id']
```

2.Test set preprocessing

In [6]:

```
test_with_ap['predicted_house_price']=test_with_ap['predicted_house_price'].fillna(value=test_with_
```

In [7]:

```
#시간 정보만 추출(test_with_ap)
test_with_ap['event_datetime'] = pd.to_datetime(test_with_ap['event_datetime']).dt.hour.astype(int)
test_with_ap.head()
```

Out[7]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kleE1J0KCa	E
1	0	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9b
2	3	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kleE1J0KCa	9b
3	0	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kleE1J0KCa	hk
4	0	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kleE1J0KCa	

5 rows × 31 columns

In [8]:

```
#시간 정보만 추출(test)
test['event_datetime'] = pd.to_datetime(test['event_datetime']).dt.hour.astype(int)
test.head()
```

Out[8]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	jlLrN0gGpx	nwf1A3O5cO	7Noz5InNj5	yH0QQDoPNl	kleE1J0KCa	
2	0	Pwlj11RYvM	Uox85xVMSC	NxzS8oTLt4	ytPy92XPEV	kleE1J0KCa	9
3	0	W0o0KwmTSQ	M6QaRvdZ8h	ctd4ThNAdz	2TWeHHdrJ8	kleE1J0KCa	Z
4	0	UpL3kLWqZy	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kleE1J0KCa	9
6	0	PAG5INDY6L	y7QKxSwhwV	dS6rpIpBHY	G0n6acDmBk	tg9mzu7kFm	

5 rows × 24 columns

In [9]:

```
# converting gender,marry feature to numeical value
test_with_ap['gender']=test_with_ap['gender'].map({'M':0,
                                                    'F':1})
test_with_ap['marry']=test_with_ap['marry'].map({'M':0,
                                                  'S':1})
test_with_ap.head()
```

Out[9]:

	event_datetime	bid_id	ssp_id	campaign_id	adset_id	placement_type	
0	0	EHTpBC8c9L	M6QaRvdZ8h	HXFpqSuEoP	xUg7NKz4Kb	kIeE1J0KCa	E
1	0	0JdfjVSNi8	Uox85xVMSC	A2FWaDfu78	AX8mFBH96H	kIeE1J0KCa	9l
2	3	d7g9YcPv7u	Uox85xVMSC	NxzS8oTLt4	HddNRHYvkt	kIeE1J0KCa	9l
3	0	KJryVcuWQc	tDmR2RkEPK	C1f0mepfnU	bPy6lzSOWP	kIeE1J0KCa	hk
4	0	ObZPTVVYcA	nwf1A3O5cO	qR4Xa60DLl	FiSRHSfVaf	kIeE1J0KCa	

5 rows × 31 columns

In [10]:

```
#encoding을 위해 age feature str으로 변환
test_with_ap['age']=test_with_ap['age'].astype('str')
```

In [11]:

```
test_with_ap=test_with_ap.drop(['install_pack','bid_id','cate_code','asset_index','device_os','devic
test = test.drop(['bid_id','device_os','device_country'],axis=1)
```

3.Feature engineering

Target encoding

In [12]:

```
#target encoding
from category_encoders import TargetEncoder
import pickle

enc_with_ap = pickle.load(open('enc_preprocess_1.sav', 'rb'))

numeric_test_with_ap=enc_with_ap.transform(test_with_ap)
numeric_test_with_ap
```

Out[12]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_nam
0	0	0.143941	0.042074	0.141686	0.139161	0.273404	0.27860
1	0	0.039832	0.057104	0.115930	0.139161	0.032964	0.03296
2	3	0.039832	0.047038	0.109618	0.139161	0.032964	0.03296
3	0	0.009178	0.008072	0.008072	0.139161	0.016541	0.00917
4	0	0.189793	0.285726	0.285726	0.139161	0.282208	0.27860
...	...	...	...	...	...	...	.
10765	0	0.011040	0.012040	0.006623	0.021026	0.016541	0.01724
10766	0	0.039832	0.105717	0.130554	0.139161	0.090909	0.09090
10767	0	0.189793	0.135373	0.136251	0.139161	0.023806	0.02380
10768	0	0.189793	0.106897	0.142785	0.139161	0.034730	0.03473
10769	0	0.143941	0.087930	0.136877	0.139161	0.273404	0.27860

214642 rows × 25 columns

In [13]:

```
enc = pickle.load(open('enc_preprocess_2.sav', 'rb'))

numeric_test=enc.transform(test)
numeric_test
```

Out[13]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_nar
0	0	0.193911	0.254406	0.244391	0.143059	0.283167	0.2796
2	0	0.046613	0.046747	0.104764	0.143059	0.033853	0.0338
3	0	0.144511	0.064228	0.092414	0.143059	0.025916	0.0259
4	0	0.046613	0.056631	0.119014	0.143059	0.033853	0.0338
6	0	0.184407	0.190157	0.201794	0.184516	0.174362	0.1743
...	...	...	...	...	...	...	
782300	0	0.018592	0.126839	0.003182	0.022644	0.018766	0.0249
782301	0	0.046613	0.108579	0.133637	0.143059	0.020502	0.0205
782304	0	0.193911	0.088609	0.136681	0.143059	0.260223	0.2602
782305	0	0.144511	0.136572	0.137860	0.143059	0.273958	0.2796
782307	0	0.193911	0.287327	0.287327	0.143059	0.283167	0.2796

335358 rows × 21 columns

In [14]:

```
#test_with_ap set
numeric_test_with_ap = numeric_test_with_ap[['event_datetime', 'ssp_id', 'campaign_id', 'adset_id',
         'media_id', 'media_name', 'media_bundle', 'publisher_id',
         'publisher_name', 'device_ifa', 'device_os_version', 'device_model',
         'device_carrier', 'device_region', 'device_city', 'advertisement_id',
         'age', 'marry', 'predicted_house_price']]
numeric_test_with_ap.head()
```

Out[14]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_name	m
0	0	0.143941	0.042074	0.141686	0.139161	0.273404	0.278601	
1	0	0.039832	0.057104	0.115930	0.139161	0.032964	0.032964	
2	3	0.039832	0.047038	0.109618	0.139161	0.032964	0.032964	
3	0	0.009178	0.008072	0.008072	0.139161	0.016541	0.009178	
4	0	0.189793	0.285726	0.285726	0.139161	0.282208	0.278601	

In [15]:

```
#test set
numeric_test = numeric_test[['event_datetime', 'campaign_id', 'adset_id', 'media_id', 'media_bundle'
         'publisher_id', 'publisher_name', 'device_ifa', 'device_os_version',
         'device_model', 'advertisement_id']]
numeric_test.head()
```

Out[15]:

	event_datetime	campaign_id	adset_id	media_id	media_bundle	publisher_id	publisher_nan
0	0	0.254406	0.244391	0.283167	0.274011	0.283167	0.2831
2	0	0.046747	0.104764	0.033853	0.030615	0.033853	0.0338
3	0	0.064228	0.092414	0.025916	0.030615	0.169584	0.0259
4	0	0.056631	0.119014	0.033853	0.030615	0.033853	0.0338
6	0	0.190157	0.201794	0.174362	0.173042	0.174222	0.0319

scaling

In [16]:

```
#ap 포함하는 파일
from sklearn.preprocessing import MinMaxScaler
Scaler = MinMaxScaler()
numeric_test_with_ap[['predicted_house_price']] = Scaler.fit_transform(numeric_test_with_ap[['predic
numeric_test_with_ap[['event_datetime']] = Scaler.fit_transform(numeric_test_with_ap[['event_datetim
numeric_test_with_ap
```

```
C:₩Users₩user₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:4: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  after removing the cwd from sys.path.
C:₩Users₩user₩Anaconda3₩lib₩site-packages₩pandas₩core₩indexing.py:494: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self.obj[item] = s
C:₩Users₩user₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:5: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  """
C:₩Users₩user₩Anaconda3₩lib₩site-packages₩pandas₩core₩indexing.py:494: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self.obj[item] = s
```

Out[16]:

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_
0	0.000000	0.143941	0.042074	0.141686	0.139161	0.273404	0.27
1	0.000000	0.039832	0.057104	0.115930	0.139161	0.032964	0.03
2	0.130435	0.039832	0.047038	0.109618	0.139161	0.032964	0.03
3	0.000000	0.009178	0.008072	0.008072	0.139161	0.016541	0.00
4	0.000000	0.189793	0.285726	0.285726	0.139161	0.282208	0.27
...	...	...	...	...	...	...	

	event_datetime	ssp_id	campaign_id	adset_id	placement_type	media_id	media_r
10765	0.000000	0.011040	0.012040	0.006623	0.021026	0.016541	0.01
10766	0.000000	0.039832	0.105717	0.130554	0.139161	0.090909	0.09
10767	0.000000	0.189793	0.135373	0.136251	0.139161	0.023806	0.02
10768	0.000000	0.189793	0.106897	0.142785	0.139161	0.034730	0.03
10769	0.000000	0.143941	0.087930	0.136877	0.139161	0.273404	0.27

214642 rows × 20 columns

In [17]:

```
#ap 비포함 파일
from sklearn.preprocessing import MinMaxScaler
Scaler = MinMaxScaler()
numeric_test[['event_datetime']] = Scaler.fit_transform(numeric_test[['event_datetime']])
numeric_test
```

```
C:₩Users₩user₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:4: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  after removing the cwd from sys.path.
C:₩Users₩user₩Anaconda3₩lib₩site-packages₩pandas₩core₩indexing.py:494: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self.obj[item] = s
```

Out[17]:

	event_datetime	campaign_id	adset_id	media_id	media_bundle	publisher_id	publishe
0	0.0	0.254406	0.244391	0.283167	0.274011	0.283167	0
2	0.0	0.046747	0.104764	0.033853	0.030615	0.033853	0
3	0.0	0.064228	0.092414	0.025916	0.030615	0.169584	0
4	0.0	0.056631	0.119014	0.033853	0.030615	0.033853	0
6	0.0	0.190157	0.201794	0.174362	0.173042	0.174222	0
...	...	...	...	...	...	...	
782300	0.0	0.126839	0.003182	0.018766	0.004567	0.025054	0
782301	0.0	0.108579	0.133637	0.020502	0.086647	0.020636	0
782304	0.0	0.088609	0.136681	0.260223	0.258748	0.260223	0
782305	0.0	0.136572	0.137860	0.273958	0.274011	0.169584	0
782307	0.0	0.287327	0.287327	0.283167	0.274011	0.283167	0

335358 rows × 11 columns

3.Model loading

In [19]:

```
#Model reading
import pickle

lgbm = pickle.load(open('lgbm_without_ap.sav', 'rb'))
lgbm_with_ap = pickle.load(open('lgbm_with_ap.sav', 'rb'))
```

3.Prediction

5.LightgbmClassifier

In [20]:

```
#test_with_ap 세트 확률과 bid id 연결
probs_with_ap = lgbm_with_ap.predict_proba(numeric_test_with_ap)
probs_with_ap = probs_with_ap[:, 1]
bid_ap = pd.DataFrame(bid_id_with_ap)
bid_ap['probs']=probs_with_ap
bid_ap
```

Out[20]:

	bid_id	probs
0	EHTpBC8c9L	0.229249
1	0JdfjVSNi8	0.000020
2	d7g9YcPv7u	0.000020
3	KJryVcuWQc	0.007987
4	ObZPTVVYcA	0.998562
...	...	...
10765	faWjGdHRmw	0.007318
10766	mAI8SUv657	0.091439
10767	kuCI8qgDp9	0.000020
10768	SL3vs75mac	0.036380
10769	dv5HX2ehaK	0.000020

214642 rows × 2 columns

In [21]:

```
#test 세트 확률과 bid id 연결

probs = lgbm.predict_proba(numeric_test)
probs = probs[:, 1]
bid = pd.DataFrame(bid_id_without_ap)
bid['probs']=probs
bid
```

Out[21]:

	bid_id	probs
0	jILrN0gGpx	0.164827
2	PwIj11RYvM	0.033326
3	W0o0KwmTSQ	0.089141
4	UpL3kLWqZy	0.030910
6	PAG5INDY6L	0.090196
...	...	...
782300	cLU5mO3z89	0.000967
782301	2YxtmVzvpB	0.025222
782304	bSxh3i0gN3	0.172497
782305	LAyxamwNxm	0.157774
782307	W8XuFXZw4v	0.138133

335358 rows × 2 columns

In [22]:

```
#모든 확률값 병합

bid_all= pd.concat([bid,bid_ap])
submit = pd.merge(raw_test['bid_id'],bid_all,how="left",on='bid_id')
submit
```

Out[22]:

	bid_id	probs
0	jlLrN0gGpx	0.164827
1	zA3WyymOcJ	0.035192
2	PwIj11RYvM	0.033326
3	W0o0KwmTSQ	0.089141
4	UpL3kLWqZy	0.030910
...	...	...
549995	bSxh3i0gN3	0.172497
549996	LAyxamwNxm	0.157774
549997	3sF8PXgPom	0.908994
549998	W8XuFXZw4v	0.138133
549999	WNke5qEQC1	0.246732

550000 rows × 2 columns

In [23]:

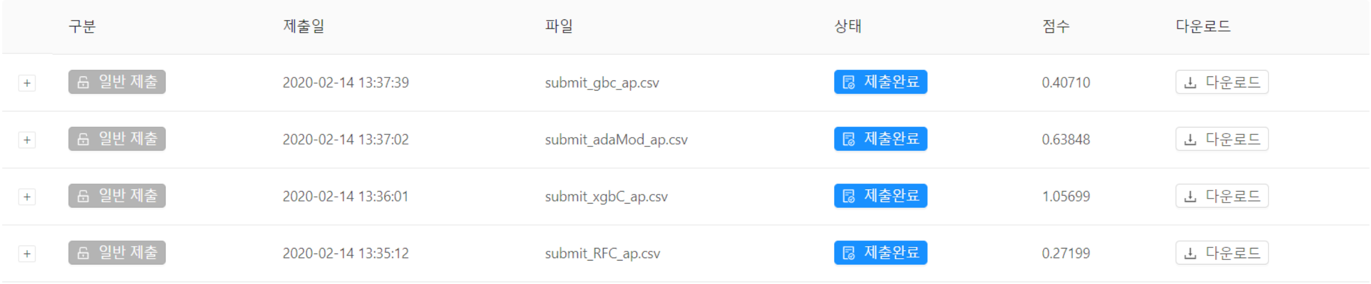
```
submit.to_csv('submit_lgbm_ap.csv', index=False,header=False)
```

In [ ]:

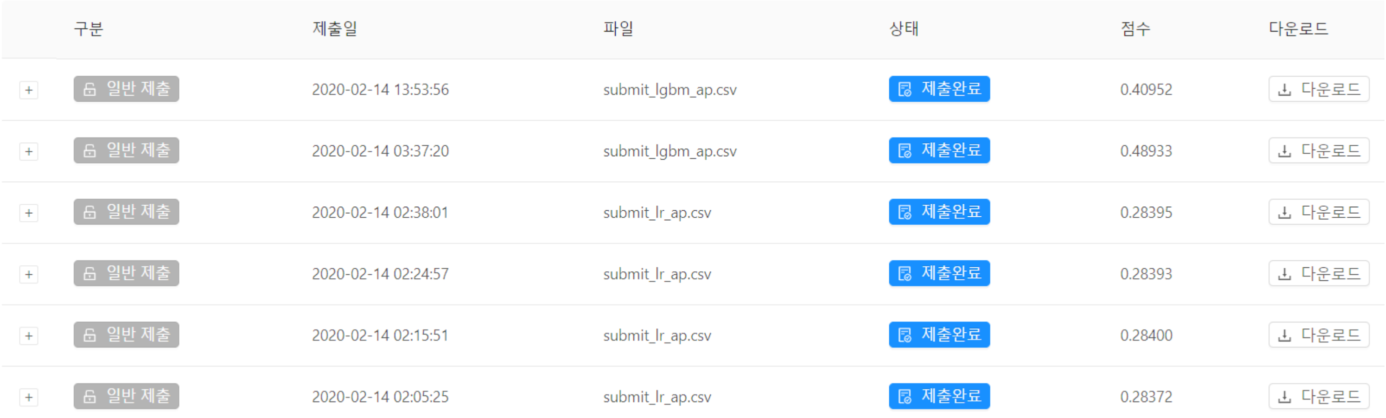
```

```

리더보드 결과



	구분	제출일	파일	상태	점수	다운로드
+	일반 제출	2020-02-14 13:37:39	submit_gbc_ap.csv	제출완료	0.40710	다운로드
+	일반 제출	2020-02-14 13:37:02	submit_adaMod_ap.csv	제출완료	0.63848	다운로드
+	일반 제출	2020-02-14 13:36:01	submit_xgbC_ap.csv	제출완료	1.05699	다운로드
+	일반 제출	2020-02-14 13:35:12	submit_RFC_ap.csv	제출완료	0.27199	다운로드



	구분	제출일	파일	상태	점수	다운로드
+	일반 제출	2020-02-14 13:53:56	submit_lgbm_ap.csv	제출완료	0.40952	다운로드
+	일반 제출	2020-02-14 03:37:20	submit_lgbm_ap.csv	제출완료	0.48933	다운로드
+	일반 제출	2020-02-14 02:38:01	submit_lr_ap.csv	제출완료	0.28395	다운로드
+	일반 제출	2020-02-14 02:24:57	submit_lr_ap.csv	제출완료	0.28393	다운로드
+	일반 제출	2020-02-14 02:15:51	submit_lr_ap.csv	제출완료	0.28400	다운로드
+	일반 제출	2020-02-14 02:05:25	submit_lr_ap.csv	제출완료	0.28372	다운로드

리더보드 스코어와 validation score를 종합적으로 비교했을때 RandomForestClassifier가 가장 좋은 성능을 내는 것으로 나와 최종적으로 RandomForestClassifier를 모델로 결정.

보완해야 할 점

1. RandomForestClassifier를 제외한 나머지 머신러닝 모델의 경우 하이퍼 파라미터의 값이 성능에 영향을 많이 끼친다. 하지만 시간적, 자원적 제한으로 하이퍼 파라미터 튜닝과정을 거치지 못하였다. 각 모델마다 Grid Search 를 통한 하이퍼 파라미터 튜닝과정을 거쳤다면 RandomForest가 아닌 다른 모델을 선정했을 가능성이 있다.
2. Feature selection 과정에서 threshold 값 설정에 대한 논리적인 결정 과정이 없었다.
3. Outlier 제거, Scaling, feature seleting 과정을 동시에 진행하였는데 이 세과정을 거치지 않았을때의 모델의 정확도(리더보드 기준)가 오히려 더 좋았다.세 과정을 하나하나 거치며 어느부분에서 문제가 있었는지에 대한 조사가 필요하다.
4. neural network, knn, Naive Bayse 등등 더 많은 모델에 대한 시도가 부족했다.
5. audience profile에 포함된 instal_pack, Cate_code feature를 가공하여 새로운 의미있는 feature를 생성해 내지 못했다.
6. Target encoding외에 다른 방법으로 categorical data를 encoding 하는 방법을 시도해볼 필요가 있다.

```
In [ ]:
```