KHUDA

경희대학교 데이터 분석 동아리

KHUDA 2023

XGBoost 기반 모바일 광고 클릭률 (CTR) 예측 알고리즘 개발

이영석 이채린 정혜주

KHUDA

KHUDA

Contents

KHUDA

프로젝트 개요

KHUDA

프로젝트 배경

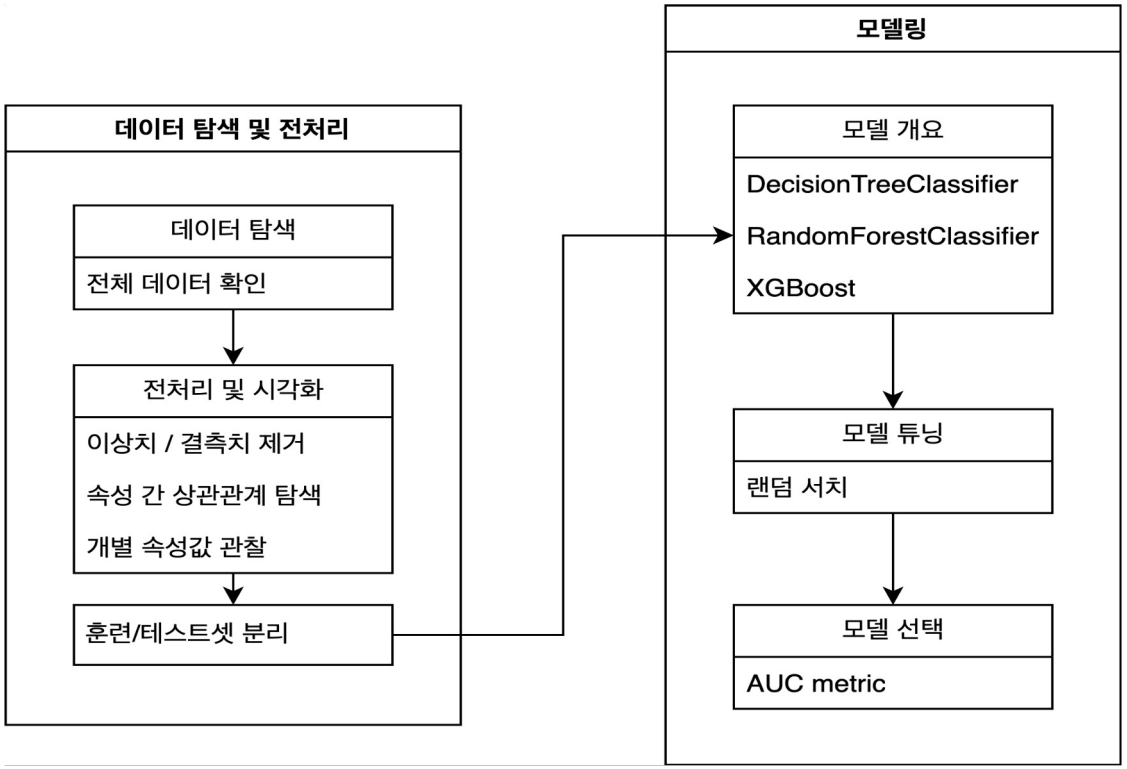
프로젝트 목표

유저, 퍼블리셔, 앱 정보, 클릭 등의 기본적인 광고 정보를 바탕으로 모델링해
데이터 샘플 모바일 광고의 클릭률(CTR)을 예측하는 알고리즘 고안

- 추천시스템과 같은 고도화 기술이 발전하는 것과 더불어 광고의 효율을 높이기 위한 Click Prediction 기술 또한 비슷한 기법들로 발전

KHUDA

프로젝트 진행 과정



KHUDA

데이터셋

데이터셋 개요

- 2020 DIGIX Global AI Challenge에서 화웨이가 제공한 데이터셋 사용
- 총 36가지 칼럼의 다양한 사용자 데이터
- 익명화된 데이터, 불균형 데이터

KHUDA

데이터 탐색

	label	uid	task_id	adv_id	creat_type_cd	adv_prim_id	dev_id	inter_type_cd	slot_id	spread_app_id	...
0	0	1638254	2112	6869	7	207	17	5	11	13	...
1	0	1161786	3104	3247	7	183	29	5	17	86	...
2	0	1814783	5890	4183	7	178	17	5	11	70	...
3	0	1468996	1993	5405	7	207	17	5	21	13	...
4	0	2164010	5439	4677	2	138	24	5	12	33	...
...	...	...	...	...	...	...	...	...	...	...	...

- train set에는 약 4190만 여 개의 행 존재
→ 메모리 부족으로 50만 개 행만 샘플링해 분석 수행

- 대부분의 변수는 암호화되어 익명화(비식별화)된 변수
→ 시각화를 통한 EDA에 어려움이 있음
→ 데이터 간의 관계 이상의 것들을 추론하기 어려움

- 범주형 변수인 communication_onlinerate를 제외하고 모두 수치형 변수

```
edited_file.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500000 entries, 0 to 499999
Data columns (total 36 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   label                       500000 non-null  int64
 1   uid                         500000 non-null  int64
 2   task_id                     500000 non-null  int64
 3   adv_id                      500000 non-null  int64
 4   creat_type_cd               500000 non-null  int64
 5   adv_prim_id                 500000 non-null  int64
 6   dev_id                      500000 non-null  int64
 7   inter_type_cd               500000 non-null  int64
 8   slot_id                     500000 non-null  int64
 9   spread_app_id               500000 non-null  int64
 10  tags                        500000 non-null  int64
 11  app_first_class             500000 non-null  int64
 12  app_second_class            500000 non-null  int64
 13  age                         500000 non-null  int64
 14  city                        500000 non-null  int64
 15  city_rank                   500000 non-null  int64
 16  device_name                 500000 non-null  int64
 17  device_size                 500000 non-null  int64
 18  career                      500000 non-null  int64
 19  gender                      500000 non-null  int64
 20  net_type                    500000 non-null  int64
 21  residence                   500000 non-null  int64
 22  his_app_size                500000 non-null  int64
 23  his_on_shelf_time           500000 non-null  int64
 24  app_score                   500000 non-null  int64
 25  emui_dev                    500000 non-null  int64
 26  list_time                   500000 non-null  int64
 27  device_price                500000 non-null  int64
 28  up_life_duration            500000 non-null  int64
 29  up_membership_grade         500000 non-null  int64
 30  membership_life_duration    500000 non-null  int64
 31  consume_purchase            500000 non-null  int64
 32  communication_onlinerate    500000 non-null  object
 33  communication_avgonline_30d 500000 non-null  int64
 34  indu_name                   500000 non-null  int64
 35  pt_d                        500000 non-null  int64
dtypes: int64(35), object(1)
memory usage: 137.3+ MB
```

KHUDA

데이터 전처리

```
edited_file.isnull().sum()
```

```
label                          0
uid                            0
task_id                        0
adv_id                         0
creat_type_cd                  0
adv_prim_id                    0
dev_id                         0
inter_type_cd                  0
slot_id                        0
spread_app_id                  0
tags                           0
app_first_class                0
app_second_class               0
age                            0
city                           0
city_rank                      0
device_name                    0
device_size                    0
career                         0
gender                         0
net_type                       0
residence                      0
his_app_size                   0
his_on_shelf_time              0
app_score                      0
emui_dev                       0
list_time                      0
device_price                   0
up_life_duration               0
up_membership_grade            0
membership_life_duration       0
consume_purchase               0
communication_onlinerate       0
communication_avgonline_30d    0
indu_name                      0
pt_d                           0
dtype: int64
```

결측치 확인 결과 없음 확인

KHUDA

데이터 전처리

```
[13] communication_onlinerate=edited_file['communication_onlinerate'].value_counts()[edited_file['communication_onlinerate'].value_counts()>50]
     communication_onlinerate=pd.DataFrame(communication_onlinerate.index)
```

```
[14] cat_encoder=OneHotEncoder(sparse=False, handle_unknown='ignore')
     cat_encoder.fit(communication_onlinerate.values.reshape(-1,1))
     communication_onlinerate_encoded=pd.DataFrame(cat_encoder.transform(edited_file['communication_onlinerate'].values.reshape(-1,1)))
     features=list(edited_file.columns)
     features=features+cat_encoder.categories_[0].tolist()
     features.remove('communication_onlinerate')
     engineered_data=edited_file.join(communication_onlinerate_encoded)
     engineered_data=engineered_data.drop(columns=['communication_onlinerate'])
     engineered_data.columns = engineered_data.columns.astype(str)
```

➡ 범주형 변수 communication_onlinerate를 수치형으로 변환(원-핫 인코딩)

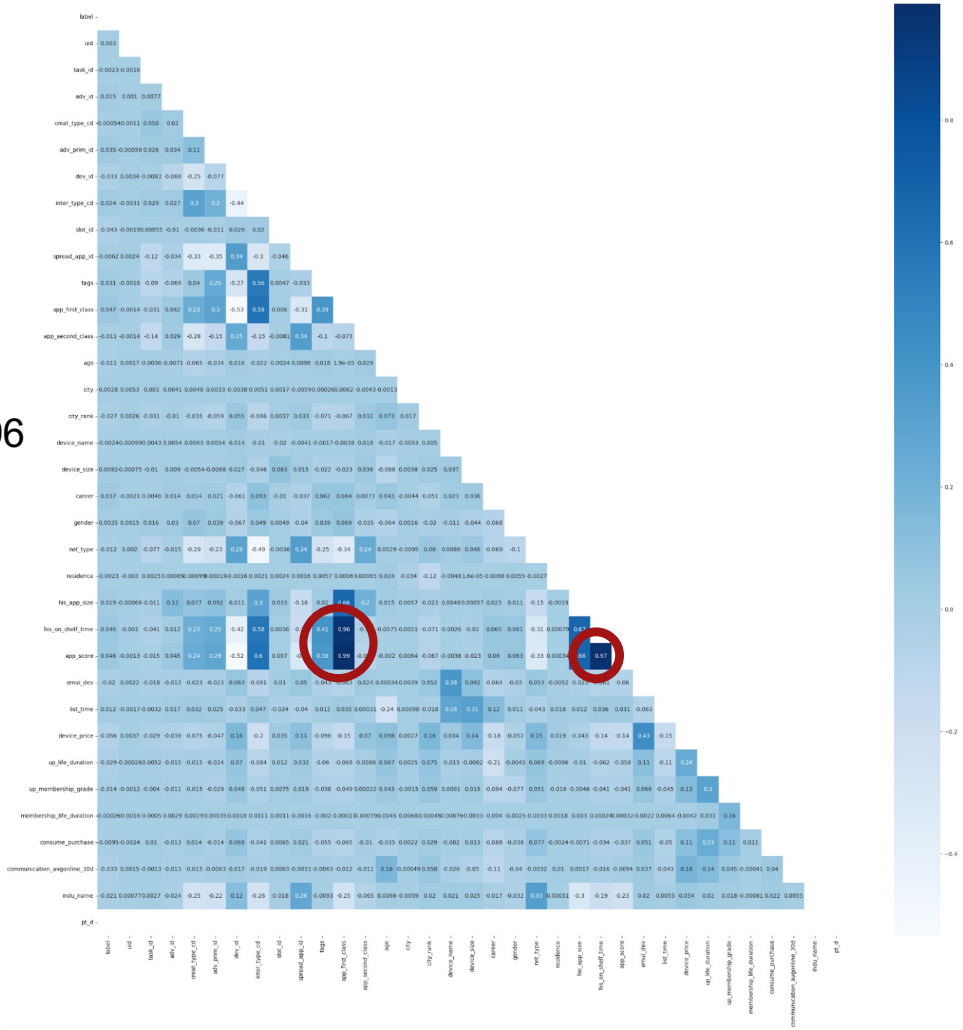
KHUDA

EDA

히트맵을 통해 변수 간 상관관계 탐색

상관관계가 높은 변수들 확인

- corr(his_on_shelf_time, app_first_class) = 0.96
- corr(app_score, app_first_class) = 0.99
- corr(app_score, his_on_shelf_time) = 0.97



그러나 의사결정트리 기반 모델 특성상 다중 공선성 문제는 고려하지 않아도 OK!

KHUDA

EDA

개별 속성값 관찰

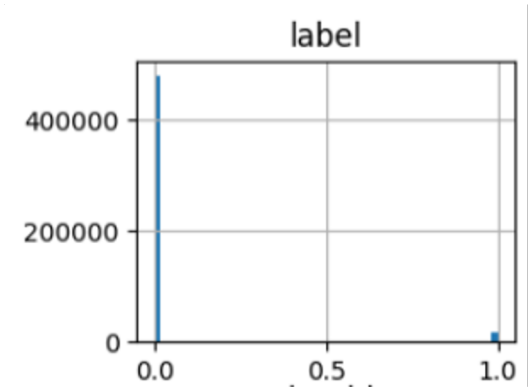
- 히스토그램을 그려 다양한 변수들의 분포를 관찰하고 데이터의 왜곡이나 편향을 확인해 데이터에 대한 이해를 높임

➡ label 변수를 통한 데이터 불균형 확인!

KHUDA

EDA

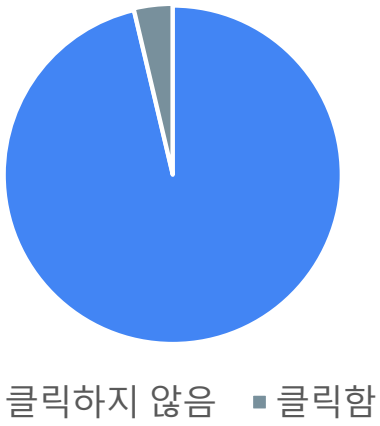
심한 데이터 불균형




```
[11] edited_file['label'].value_counts()[1] / 500000

     0.036942
```

실제 광고 클릭수



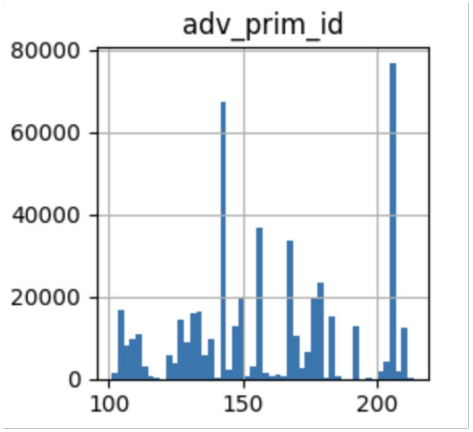
KHUDA

데이터셋 분리

```
split=StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=0)
for train_index, test_index in split.split(engineered_data, engineered_data['adv_prim_id']):
    strat_train_set=engineered_data.loc[train_index]
    strat_test_set=engineered_data.loc[test_index]

train_labels=strat_train_set['label']
test_labels=strat_test_set['label']
strat_train_set=strat_train_set.drop(columns=['label'])
strat_test_set=strat_test_set.drop(columns=['label'])

scaler=StandardScaler()
strat_train_set=pd.DataFrame(scaler.fit_transform(strat_train_set))
strat_test_set=pd.DataFrame(scaler.fit_transform(strat_test_set))
```

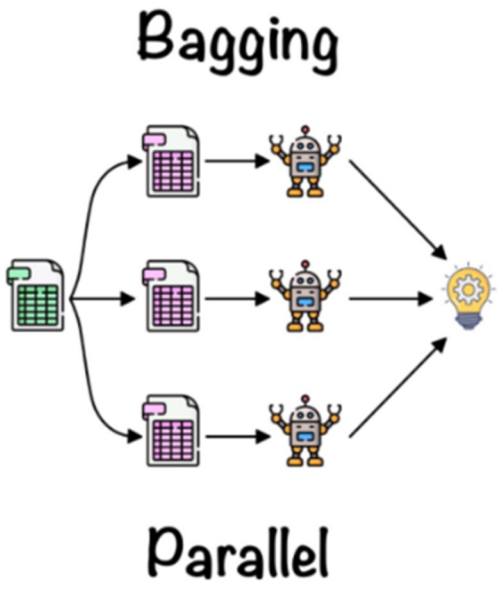


정확도를 높이기 위해 계층적 샘플링 사용
훈련 세트와 테스트 세트를 분리하고 데이터 표준화

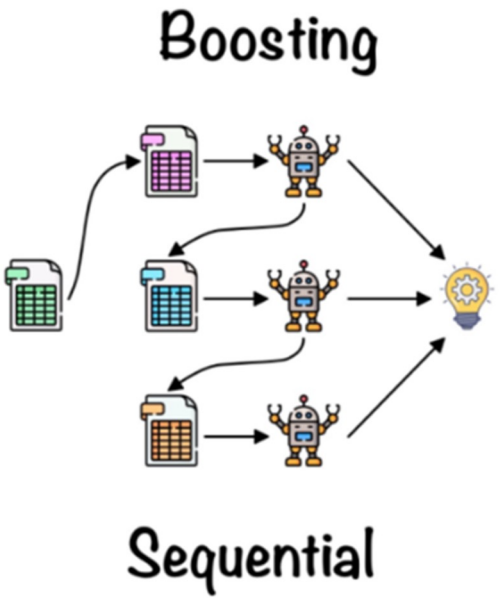
KHUDA

알고리즘 개발

모델 선정



ex. 랜덤포레스트



ex. GBM, XGBoost, LightGBM

XGBoost

기존 GBM 모델의 단점

학습이 순차적으로 이루어지기 때문에 계산 시간이 오래 걸리며, 과적합을 방지하는 규제 없음.

XGBoost는 이러한 GBM의 문제를 해결하기 위해 고안된 알고리즘!

- 병렬처리 지원
- 과적합 방지 규제 시행
- 회귀&분류 둘 다 적용 가능

KHUDA

XGBoost 모델 생성

```
scale_pos_weight = edited_file['label'].value_counts()[0] / edited_file['label'].value_counts()[1]
xg_class= xgb.XGBClassifier(random_state=0, use_label_encoder=False)
params=  {'n_estimators': randint(50, 400),
              'learning_rate': uniform(0.01, 0.59),
              'subsample': uniform(0.3, 0.6),
              'max_depth': [3, 4, 5, 6, 7, 8, 9],
              'colsample_bytree': uniform(0.4, 0.5),
              'min_child_weight': [1, 2, 3, 4],
              'scale_pos_weight': [scale_pos_weight] }
rnd_search=RandomizedSearchCV(xg_class, param_distributions = params, cv = 5,n_iter = 10,
                              scoring = 'roc_auc', error_score = 0, verbose = 3, n_jobs = -1)
rnd_search.fit(strat_train_set, train_labels, eval_metric='mlogloss')
```

- Scale_pos_weight - 하이퍼 파라미터, 가중치 조절
  ex) 5라면, 5배의 가중치를 가짐. 불균형한 클래스에 집중하여 학습하게 되므로, 예측 성능이 개선됨
- rnd_search - 모델을 여러 번 학습하고, 최적의 하이퍼파라미터를 찾음
- rnd_serach.fit - RandomizedSearchCV 객체를 통해 하이퍼파라미터 탐색

주요 하이퍼파라미터

- learning_rate (기본값 0.3): 학습률. 각 트리 모델이 개선될 때마다 적용되는 가중치 조정값
- subsample (기본값 1): 각 트리 모델 학습 시 사용되는 샘플 데이터의 비율
- n_estimators (기본값 100): 학습에 사용되는 결정 트리의 개수
- max_depth (기본값 6): 결정 트리의 최대 깊이
- colsample_bytree (기본값 1): 각 트리 모델 학습 시 사용되는 특성(feature) 데이터의 비율
- scale_pos_weight (기본값 1): 데이터 레이블 불균형이 있을 때 레이블 가중치 조절
  → 권장값: summ(negative instances) / sum(positive instances)
  → 데이터 불균형이 심한 경우 반드시 설정해주면 좋음!

KHUDA

예측 결과

모델 예측 결과

```
0.7093734348108509 {'colsample_bytree': 0.6068723369033346, 'learning_rate': 0.19941397199852942,
                    'max_depth': 4, 'min_child_weight': 3, 'n_estimators': 140,
                    'scale_pos_weight': 26.069460234962914, 'subsample': 0.7229330198367354}

0.6754994983137546 {'colsample_bytree': 0.5360612529803217, 'learning_rate': 0.3505512369068798,
                    'max_depth': 5, 'min_child_weight': 1, 'n_estimators': 382,
                    'scale_pos_weight': 26.069460234962914, 'subsample': 0.7212188565774438}

0.7104439229558988 {'colsample_bytree': 0.5790466130880881, 'learning_rate': 0.17182493244122665,
                    'max_depth': 3, 'min_child_weight': 2, 'n_estimators': 379,
                    'scale_pos_weight': 26.069460234962914, 'subsample': 0.7202891277887382}
```

- 처음 숫자 - 모델의 성능
- Scale_pos_weight - 양성 클래스에 대한 가중치
- min_child_weight - 리프 노드(마지막 노드)에 필요한 최소 가중치를 정의

최종 결과

```
y_predict2 = pd.DataFrame(rnd_search.best_estimator_.predict(strat_test_set))
y_predict2.value_counts()[1]/500000
0.051338
```

- 양성 클래스(1)의 비율, 즉, 전체 테스트 세트에서 양성 클래스로 분류된 샘플의 비율은 약 0.051338 (약 5.13%)입니다.

한계점 및 보완점

- 개인정보 보호를 위해 데이터가 익명화되어 있어서 해석에 어려움이 있었음.
  → ex) [0,1,2…6]이 아니라 [Mon,Tue, … Sun]으로 되어있다면 해석하기 편했을 것
- 데이터의 용량이 너무 커서(train의 칼럼은 4000만개) 일부만 사용
- 시간 부족으로 LightGBM, Catboost 등 다른 모델 사용해 보지 못함

Thank You

Any Questions?

KHODA