Ad Tracking Fraud Detection Challenge

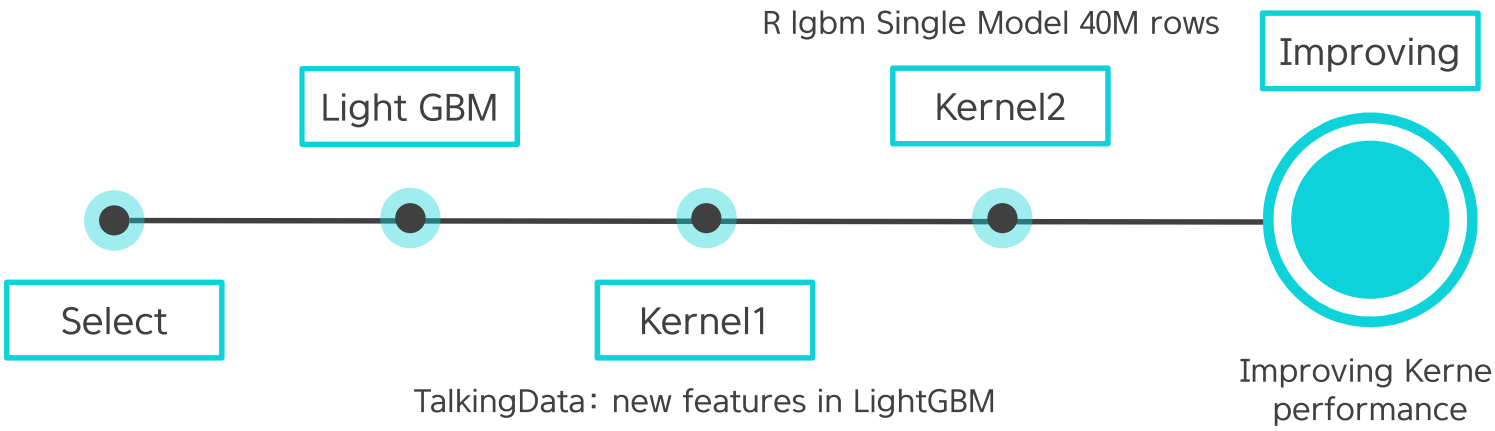
Using LightGBM

Contents

- 온라인 광고를 하는 회사의 경우 **Click Fraud**가 발생
- 'TalkingData'라는 회사는 Click Fraud를 방지하기 위해 IP and Device BlackList를 Build
- 이번 Competition의 목적은 광고를 클릭한 후 앱을 다운 받을지 여부를 예측

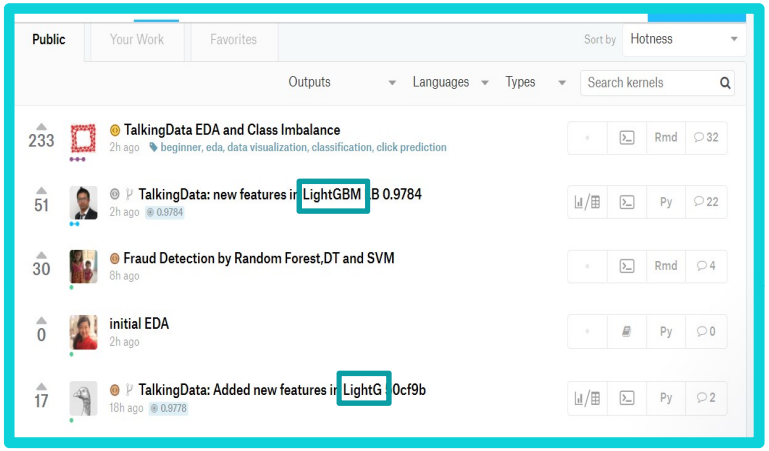
1. Select Kernel and Study
2. Kernel Implementation and Improving Performance

TimeLine

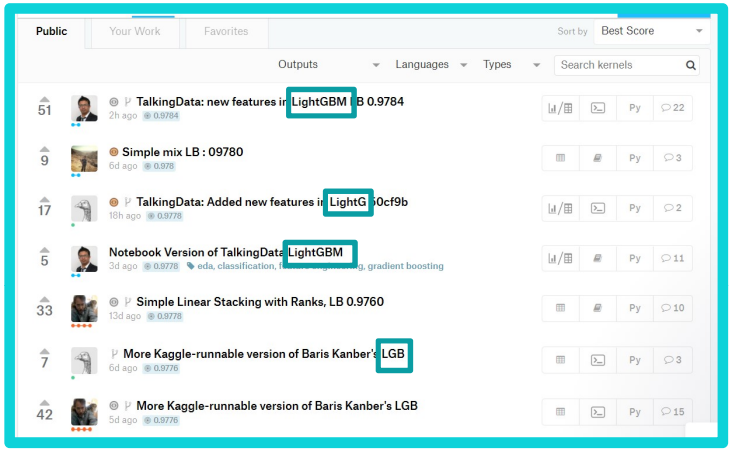

R lgbm Single Model 40M rows
Improving
Light GBM
Kernel2
Select
Kernel1
TalkingData：new features in LightGBM
Improving Kernel performance

Select Kernel and Study

Select Kernel



Hotness

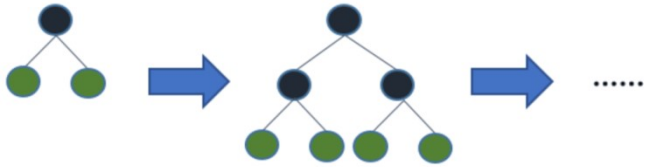


Best Score

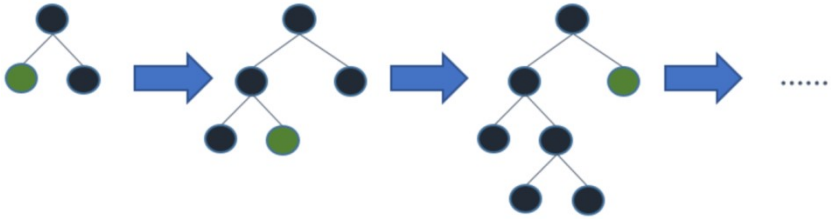
What is LightGBM

Light GBM 이란?

- 트리 기반 학습 알고리즘을 사용하는 gradient boosting framework
- Light GBM의 구현은 쉽지만 parameter tuning이 복잡하며 중요함



Level-wise tree growth



Leaf-wise tree growth

Source : Medium.com/@pushkarmandot “what is LightGBM, How to implement it? How to fine tune the parameters?”

Control Parameters

Parameters	Meaning
max_depth	• tree의 최대 깊이 • model이 overfitting 되었다고 느낄 때 max_depth를 낮춤
min_data_in_leaf	• leaf가 가질 수 있는 최소 record 수 • default 20이며 최적값임, overfitting을 처리하는데 사용
feature_fraction	• boosting이 임의의 forest일 때 사용 • 0.8이라면, parameter의 80%를 무작위로 선택한다는 것을 의미
bagging_fraction	• 각 반복에 사용할 data의 비율을 지정함. • 일반적으로 training 속도를 높이고 overfitting을 피하는데 사용
early_stopping_round	• 분석속도를 높일 수 있음 • 마지막 early_stopping_round에서 하나의 validation data 중에서 하나의 metric이 개선되지 않으면 model이 training을 중지함. 이렇게 하면 과도한 iteration을 줄일 수 있음
lambda	• 정규화를 지정, 일반적인 값의 범위는 0에서 1사이
min_gain_to_split	• split을 수행하기 위한 최소 이득을 의미. tree에서 유용한 split수를 제어하는데 사용
max_cat_group	• category수가 많으면 split point를 쉽게 찾을 수 없음. 그래서 LightGBM은 이들을 'max_cat_group'에 merge하고 그룹 경계에서 split point를 찾음

Core Parameters

Parameters	Meaning
Task	• data에서 수행할 작업을 지정. train일수도 predict일수도 있음.
application	• 가장 중요한 parameter. regression 문제인지 classification문제인지 관계 없이 모델 적용을 지정. • default는 regression model • regression, binary, multiclass
boosting	• 실행할 algorithm 유형을 정의 default = gdbt • gdbt : 전통적인 Gradient Boosting Decision Tree • rf : random forest • dart : Dropouts이 추가된 Multiple Additive Regression Trees • gross: Gradient-based One-Side Sampling
num_boost_round	• boosting iterations, 일반적으로 100+
learning_rate	• 최종 output에 대한 각 tree의 impact를 결정함. GBM은 각 tree의 output을 사용하여 update되는 initial estimate으로 시작하여 작동함. learning_rate는 추정치의 변화량을 제어함. • 일반적인 값 : 0.1, 0.001, 0.003 …
num_leaves	• 전체 tree의 잎 수(number of leaves), default:31
device	• default : cpu, gpu도 가능함

Metric and IO Parameters

◆ Metric Parameters

• Metric :중요한 Parameter는 Model building에서 loss를 지정하는 것임. Regression과 Classification에 대한 loss는 거의 없음.

Parameters	Meaning
mae	• mean absolute error
mse	• mean squared error
binary_logloss	• loss for binary classification
multi_logloss	• loss for multi classification

◆ IO Parameters

Parameters	Meaning
max_bin	• feature value가 bucket에 들어갈 수 있는 최대 bin 수를 의미.
categorical_feature	• categorical feature의 index를 의미. categoricla_features = 0, 1, 20이면 column1, column2, column3은 categorical variables임
ignore_column	• categorical_feature와 동일하지만 특정 column을 category로 간주하는 대신 완전히 무시함
save_binary	• data file의 memory size를 그대로 다루려면 True로 지정. • dataset을 binary file로 저장하므로 다음에 data file을 불러올 때 data reading 시간을 단축함

Kernel1. New Features

Python

Kernel1. TalkingData:new features in lightGBM

Feature extraction

- 다른 Kernel 'Feature Engineering & Importance Testing'의 4가지 방법(Confidence Rates for is_attributed, Group-By-Aggregation, Time till next click, Clicks on app ad before & after) 중 Time till next click와 Group-By_Aggregation을 변형, 조합하여 사용함

  (app, device, os, channel, hour)

- Feature extraction 방법 중 성능이 가장 좋게 나타난 Confidence Rates for is_attributed 소개

  1. 일부 ip, app, device에서 is_attributed의 빈도가 더 높을 수 있으므로 그 정보를 "attributed rates"로 계산

     P(is_attributed|category)

     경우에 따라 2개 또는 여러 쌍으로 된 조합을 계산함

     P(is_attributed|category_1, category_2)

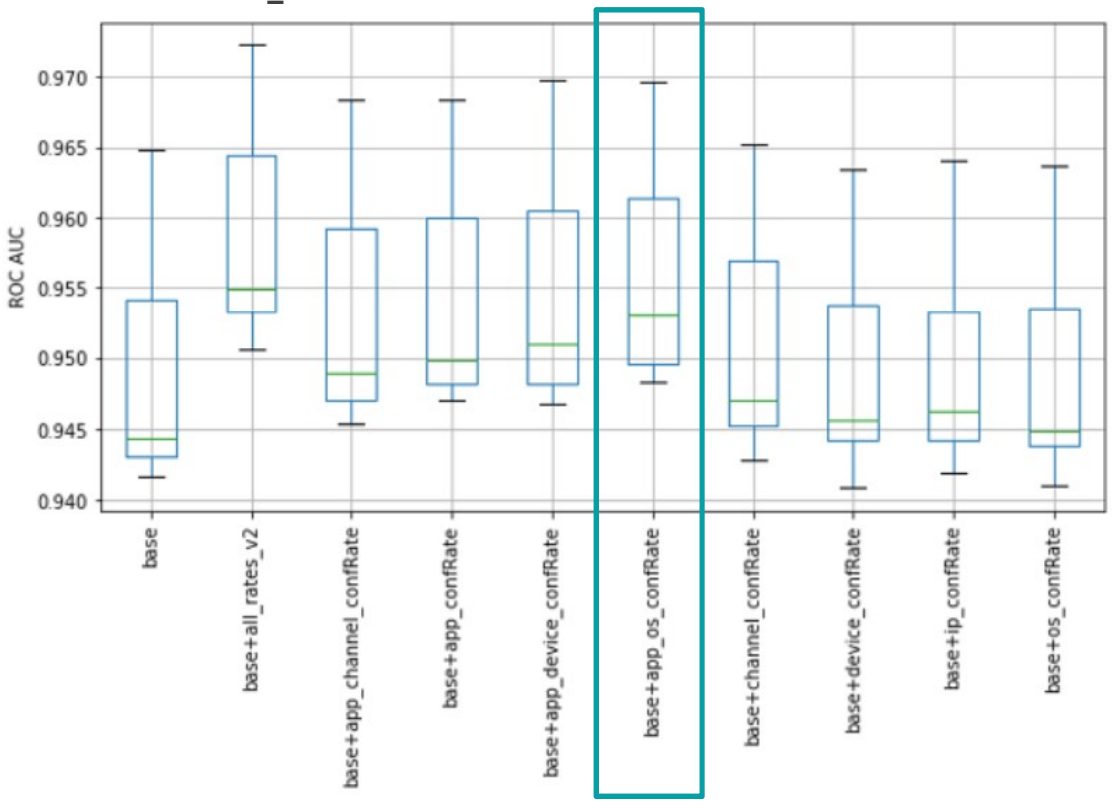
  2. 주어진 category-combination의 클릭 수가 매우 적으면 위 방정식의 통계적 중요성을 신뢰할 수 없다는 것을 의미 log(views_category_i)/log(100,000)로 계산하여 가중치로 환산

     (ex. 전체 data가 100,000개인 경우 category1에 1,000개의 관측치가 있는 경우 60%의 가중치를 얻고 100개의 관측치가 있는 경우 40%의 가중치를 얻음

  3. xgBoost를 사용하여 천 만개의 샘플로 10-fold cross-validation score로 계산, 0.955에서 0.9624로 향상됨

Kernel1. TalkingData：new features in lightGBM

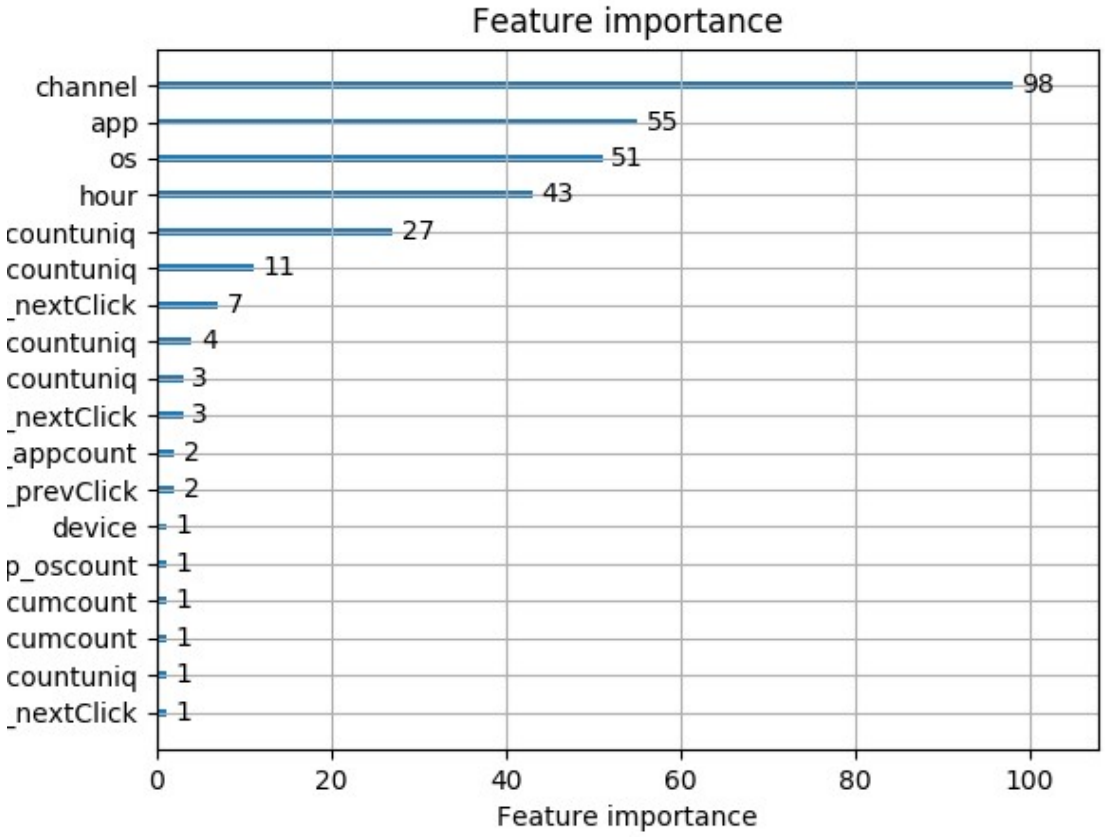
Confidence Rates for is_attributed



Kernel1. TalkingData:new features in lightGBM

```
def lgb_modelfit_nocv(params, dtrain, dvalid, predictors, target='target', objective='binary', metrics='auc',
                      feval=None, early_stopping_rounds=50, num_boost_round=3000, verbose_eval=10,
                      categorical_features=None):
    lgb_params = {
        'boosting_type': 'gbdt',
        'objective': objective,
        'metric': metrics,
        'learning_rate': 0.05,
        # 'is_unbalance': 'true',  #because training data is unbalance (replaced with scale_pos_weight)
        'num_leaves': 31,  # we should let it be smaller than 2^(max_depth)
        'max_depth': -1,  # -1 means no limit
        'min_child_samples': 20,  # Minimum number of data need in a child(min_data_in_leaf)
        'max_bin': 255,  # Number of bucketed bin for feature values
        'subsample': 0.6,  # Subsample ratio of the training instance.
        'subsample_freq': 0,  # frequence of subsample, <=0 means no enable
        'colsample_bytree': 0.3,  # Subsample ratio of columns when constructing each tree.
        'min_child_weight': 5,  # Minimum sum of instance weight(hessian) needed in a child(leaf)
        'subsample_for_bin': 200000,  # Number of samples for constructing bin
        'min_split_gain': 0,  # lambda_l1, lambda_l2 and min_gain_to_split to regularization
        'reg_alpha': 0,  # L1 regularization term on weights
        'reg_lambda': 0,  # L2 regularization term on weights
        'nthread': 8,
        'verbose': 0,
    }
```

Kernel1. TalkingData：new features in lightGBM



Kernel1. TalkingData:new features in lightGBM

```
preparing validation datasets
C:₩Python35₩lib₩site-packages₩lightgbm₩basic.py:1036: UserWarning: Using categorical_feature in Dataset.
  warnings.warn('Using categorical_feature in Dataset.')
C:₩Python35₩lib₩site-packages₩lightgbm₩basic.py:681: UserWarning: categorical_feature in param dict is overrided.
  warnings.warn('categorical_feature in param dict is overrided.')
Training until validation scores don't improve for 30 rounds.
[10]    valid's auc: 0.984957
[20]    valid's auc: 0.987053
[30]    valid's auc: 0.987685
[40]    valid's auc: 0.990644
[50]    valid's auc: 0.992448
[60]    valid's auc: 0.990923
[70]    valid's auc: 0.991524
[80]    valid's auc: 0.991267
Early stopping, best iteration is:
[52]    valid's auc: 0.992757

Model Report
bst1.best_iteration:  52
auc: 0.9927572547293558
[0.7615234851837158]: model training time
```

Kernel2. Single Model 40M

R

Kernel2. R lgbm Single Model 40M rows

1. Sample data : 90만개

1-1 Train 모델 평가 결과 (AUC 이용)

- 10%는 test data, 90% train data

[LightGBM] [Info] Number of data: 832068, number of used features: 15
[1] : validation's auc:0.942413
[26] : validation's auc:0.968536
[51] : validation's auc:0.967458
[76] : validation's auc:0.967904
Validation AUC @ best iter: 0.9696

Kaggle 결과:
0.9471
04/28 약 2643/3540

Kernel2. R lgbm Single Model 40M rows

1. Sample data : 90만개

1-2 변수의 중요도

Feature	Gain	Cover	Frequency
app	**0.7810**	**0.4120**	**0.2414**
channel	**0.1068**	**0.2573**	**0.3621**
n_app	0.0598	0.0895	0.0575
os	0.0241	0.0774	0.1494
nip_day_test_hh	0.0200	0.1524	0.1264
hour	0.0068	0.0098	0.0517
device	0.0009	0.0002	0.0057
UsrCount	0.0007	0.0014	0.0057

Kernel2. R lgbm Single Model 40M rows

2. Sample data : 180만개

2-1 Train 모델 평가 결과 (AUC 이용)

- 10%는 test data, 90% train data

[LightGBM] [Info] Number of data: 1664136, number of used features: 15
[1] : validation's auc:0.954189
[26] : validation's auc:0.969948
[51] : validation's auc:0.972988
[76] : validation's auc:0.975103
[101] : validation's auc:0.976007
[126] : validation's auc:0.976754
[151] : validation's auc:0.97707
[176] : validation's auc:0.97727
[201] : validation's auc:0.977393
[226] : validation's auc:0.976856
[251] : validation's auc:0.976859
Validation AUC @ best iter: 0.9774

Kaggle 결과:
0.9517

Kernel2. R lgbm Single Model 40M rows

2. Sample data : 180만개

2-2 변수의 중요도

Feature	Gain	Cover	Frequency
app	**0.6497**	**0.2101**	**0.1609**
channel	**0.1586**	**0.3830**	**0.3366**
n_app	0.0571	0.0317	0.0286
os	0.0465	0.1431	0.1732
nip_day_test_hh	0.0326	0.0585	0.0507
hour	0.0290	0.0842	0.1324
UsrCount	0.0107	0.0201	0.0343
UsrNewness	0.0034	0.0096	0.0172
device	0.0030	0.0219	0.0123
n_ip	0.0027	0.0071	0.0139
UsrappCount	0.0019	0.0045	0.0098
UsrappNewness	0.0016	0.0072	0.0098
n_ip_app_os	0.0014	0.0086	0.0082
n_ip_app	0.0010	0.0032	0.0065
n_ip_os	0.0007	0.0071	0.0057

Kernel2. R lgbm Single Model 40M rows

3. Sample data : 540만개

3-1 Train 모델 평가 결과(AUC 이용)

- 10%는 test data, 90% train data

[LightGBM] [Info] Number of data: 4992406, number of used features: 15
[1]: validation's auc:0.951946
[26]: validation's auc:0.963926
[51]: validation's auc:0.96791
[76]: validation's auc:0.969479
[101]: validation's auc:0.970068
[126]: validation's auc:0.970404
[151]: validation's auc:0.970614
[176]: validation's auc:0.970642
[201]: validation's auc:0.970616
[226]: validation's auc:0.970776
[251]: validation's auc:0.970656
Validation AUC @ best iter: 0.9708

Kaggle 결과:
0.9584

Kernel2. R lgbm Single Model 40M rows

3. Sample data : 540만개

3-2 변수의 중요도

Feature	Gain	Cover	Frequency
app	**0.7015**	**0.2263**	**0.1776**
channel	**0.1140**	**0.3174**	**0.3084**
n_app	0.0673	0.0456	0.0421
nip_day_test_hh	0.0529	0.0791	0.0662
os	0.0283	0.1222	0.1706
hour	0.0122	0.0769	0.1020
UsrCount	0.0084	0.0304	0.0366
device	0.0057	0.0260	0.0164
n_ip	0.0039	0.0219	0.0218
UsrappCount	0.0018	0.0198	0.0164
n_ip_app_os	0.0015	0.0222	0.0171
UsrNewness	0.0013	0.0036	0.0125
n_ip_app	0.0008	0.0033	0.0070
UsrappNewness	0.0003	0.0021	0.0031
n_ip_os	0.0002	0.0031	0.0023

Kernel2. R lgbm Single Model 40M rows

4. Sample data : 900만개

Train 모델 평가 결과(AUC 이용)

- 10%는 test data, 90% train data

[176] : validation's auc:0.971962
[201] : validation's auc:0.972145
[226] : validation's auc:0.972241
[251] : validation's auc:0.972342
[276] : validation's auc:0.97259
[301] : validation's auc:0.972478
Validation AUC @ best iter: 0.9726

Kaggle 결과:
0.9625

Thank you