DACON

Dacon 3회 게임 행동 데이터 분석 경진대회

팀 : 도발하려던건 아니었습니다만

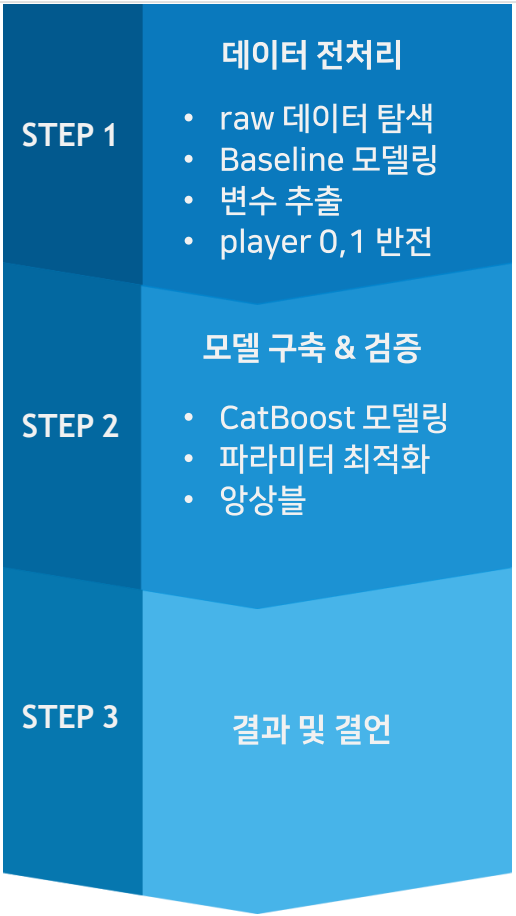
허은정 이승훈 조윤영

0. 분석 수행 환경

분석 수행 환경

- 사용 언어 : Python
- Jupyter Notebook 환경의 Google Colab에서 실행
- CPU : Intel® Xeon®
- RAM : 약 25GB
- Disk : 약 68GB
- GPU : Tesla P100
- Colab에서 할당받는 하드웨어에 따라 성능이 변경될 수 있음
- GPU 사용 시 실행 결과마다 미세한 차이 발생

목차



1.1) Raw 데이터 탐색

Raw 데이터

로그 형태 반정형 데이터 (67,091,776건)

유저의 행동 = 한 건의 데이터

time, player, species, event, event_contents

변수를 활용하여 winner를 예측

모델 학습에 사용할 데이터 X, y

-> 정형 데이터로 변형 필요 (38,872건)

game_id 변수로 그룹화, 집계 함수 활용

(38,872 × n)의 설명변수 X와

(38,872 × 1)의 반응변수 y 생성

index y ---------- ??? ----------

	game_id	winner	time	player	species	event	event_contents
0	0	1	0.00	0	T	Camera	at (145.25, 21.5078125)
1	0	1	0.00	1	T	Camera	at (22.75, 147.0078125)
2	0	1	0.02	0	T	Selection	['OrbitalCommand [3080001]']
3	0	1	0.02	0	T	Ability	(1360) - TrainSCV
4	0	1	0.14	0	T	Camera	at (142.99609375, 24.50390625)
...	...	...	...	...	...	...	...
67091771	38871	0	8.51	0	Z	Camera	at (139.578125, 62.58203125)
67091772	38871	0	8.52	1	T	GetControlGroup	NaN
67091773	38871	0	8.52	0	Z	Camera	at (122.42578125, 45.4296875)
67091774	38871	0	8.52	0	Z	Camera	at (122.42578125, 43.25390625)
67091775	38871	0	8.52	1	T	Ability	(1360) - TrainSCV

67091776 rows × 7 columns

---------- X ---------- y

	feature_1	feature_2	feature_3	feature_4	y
0	-	-	-	-	0
1	-	-	-	-	0
2	-	-	-	-	0
3	-	-	-	-	0
4	-	-	-	-	0
...	...	...	...	...	...
38867	-	-	-	-	0
38868	-	-	-	-	0
38869	-	-	-	-	0
38870	-	-	-	-	0
38871	-	-	-	-	0

38872 rows × 4 columns

1. 2) Baseline 모델링

lightGBM 패키지 활용

```
# game_id 개수만큼의 index를 가진 DataFrame X 생성
n = train.game_id.max()+1
X = pd.DataFrame(index=range(n))
X['temp'] = 1
```

```
# train 데이터에서 game_id별 winner 추출
y = train.drop_duplicates(['game_id', 'winner']).winner.reset_index(drop=True)
```

```
# train, valid 데이터로 분할
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.33, random_state=2020)
```

```
# 모델링
train_data = lgb.Dataset(train_X, label=train_y)
valid_data = lgb.Dataset(valid_X, label=valid_y)

params = {'boosting_type': 'gbdt',
          'learning_rate': 0.02,
          'seed': 2020,
          'is_training_metric': True,
          'metric': 'auc',
          'early_stopping_rounds': 10,
          }

start = time.time()
model = lgb.train(params=params, train_set=train_data, num_boost_round=5000,
                  valid_sets=[train_data, valid_data], verbose_eval=5000)
print("time :", time.time() - start)
```

```
Training until validation scores don't improve for 10 rounds.
Early stopping, best iteration is:
[1]     training's auc: 0.5     valid_1's auc: 0.5
time : 0.03282022476196289
```

X

	temp
0	1
1	1
2	1
3	1
4	1
...	...
38867	1
38868	1
38869	1
38870	1
38871	1

38872 rows × 1 columns

y

```
0        1
1        1
2        0
3        0
4        0
        ..
38867    1
38868    0
38869    0
38870    1
38871    0
Name: winner
```

AUC : 0.5인
Baseline 모델 완성

1. 3) 변수 추출 (species, time, event 활용)

	game_id	player	time	species	event
0	0	0	0.00	T	Camera
1	0	1	0.00	T	Camera
2	0	0	0.02	T	Selection
3	0	0	0.02	T	Ability
4	0	0	0.14	T	Camera
...	...	...	...	...	...
1137	0	1	7.24	T	Ability
1138	0	0	7.24	T	Right Click
1139	0	0	7.24	T	Camera
1140	1	1	0.00	T	Camera
1141	1	0	0.00	P	Camera

species

- game_id별 player 0, 1의 종족을 더미 변수로 변환

```
training's auc: 0.525746        valid_1's auc: 0.504763
```

time

- game_id별 마지막 time 값 추출
- 1.23은 1분 23초를 의미, 초 단위로 변환

```
training's auc: 0.575    valid_1's auc: 0.553378
```

event

- game_id, player별 event 횟수 count
- time 변수로 나눈 값도 변수로 추가

```
training's auc: 0.634788        valid_1's auc: 0.575306
```

Event value

- game_id, player별 event 속성값 각각 count

```
train.event.unique()      8 종류의 속성값

array(['Camera', 'Selection', 'Ability', 'Right Click', 'SetControlGroup',
       'GetControlGroup', 'AddToControlGroup', 'ControlGroup'],
      dtype=object)
```

```
training's auc: 0.752508        valid_1's auc: 0.656082
```

1. 3) 변수 추출 (event, event_contents 활용)

	player	event	event_contents
0	0	Camera	at (145.25, 21.5078125)
4	0	Camera	at (142.99609375, 24.50390625)
5	0	Camera	at (142.5078125, 24.98828125)
6	0	Camera	at (139.6171875, 27.8828125)
7	0	Camera	at (138.3359375, 29.1640625)

거리

move

- event=Camera인 경우 event_contents의 x, y 좌표 추출
- game_id, player별 x, y 좌표 이동 거리(Euclidean distance)의 sum, min, median, max 변수 추가
- 30초 이내의 move_sum 변수 추가적으로 생성

```
training's auc: 0.74423 valid_1's auc: 0.663062
```

	event	event_contents
3	Ability	(1360) - TrainSCV
19	Ability	(1021) - BuildSupplyDepot; Location: (135.0, 4...
27	Ability	(480) - Stop
37	Ability	(1360) - TrainSCV
87	Ability	(1023) - BuildBarracks; Location: (135.5, 39.5...

Ability

- event=Ability인 경우 event_contents 값 일부분 추출

```
len(train.event_contents[(train.event == 'Ability')].
    map(lambda x: x[x.find('(')+1:x.find(')')]).unique())
```

```
384
```

train에는 384 종류의 속성값

```
len(test.event_contents[(test.event == 'Ability')].
    map(lambda x: x[x.find('(')+1:x.find(')')]).unique())
```

```
368
```

test에는 368 종류의 속성값

- 속성값을 더미 변수로 변환하여 game_id, player별 count
- time 변수로 나눈 값도 변수로 추가

```
training's auc: 0.834214        valid_1's auc: 0.70131
```

1. 3) 변수 추출 (event, event_contents 활용)

	event	event_contents
2	Selection	['OrbitalCommand [3080001]']
28	Selection	['OrbitalCommand [3080001]']
68	Selection	['SCV [3280001]']
102	Selection	['CreepOnlyBlocker4x4 [1B00001]']
103	Selection	['SCV [30C0001]', 'SCV [3780001]']

Selection

- event=Selection인 경우 event_contents 값 일부분 추출
- 여러 개체를 Selection 하였기 때문에 “,” 기준으로 분리
- 한 행에 한 속성값을 갖도록 제1정규화
- 속성값을 더미 변수로 변환하여 game_id, player별 count
- time 변수로 나눈 값도 변수로 추가
- 30초 이내의 count 변수 추가적으로 생성

```
contents = (train[train.event == 'Selection'].event_contents.
            map(lambda x: re.sub('₩s₩[.....₩]', '', re.sub('₩s₩[......₩]', '', re.sub('₩s₩[.......₩]', '', x))).
                          replace('[', '').replace(']', '').replace(' ', '').replace('₩'', '')))
contents = contents.str.split(',')
max_num = max(contents.map(lambda x: len(x)))
t = [0 for x in range(max_num)]
for i in range(max_num):
  t[i] = pd.DataFrame(contents[contents.map(lambda x: len(x) > i)].map(lambda x: x[i]))
contents = pd.concat([t[i] for i in range(max_num)])
len(contents.event_contents.unique())
```

202 train에는 202 종류의 속성값

```
contents = (test[test.event == 'Selection'].event_contents.
            map(lambda x: re.sub('₩s₩[.....₩]', '', re.sub('₩s₩[......₩]', '', re.sub('₩s₩[.......₩]', '', x))).
                          replace('[', '').replace(']', '').replace(' ', '').replace('₩'', '')))
contents = contents.str.split(',')
max_num = max(contents.map(lambda x: len(x)))
t = [0 for x in range(max_num)]
for i in range(max_num):
  t[i] = pd.DataFrame(contents[contents.map(lambda x: len(x) > i)].map(lambda x: x[i]))
contents = pd.concat([t[i] for i in range(max_num)])
len(contents.event_contents.unique())
```

190 test에는 190 종류의 속성값

```
training's auc: 0.845891
valid_1's auc: 0.730074
```

1. 3) 변수 추출 (event, event_contents 활용)

	event	event_contents
3492	Right Click	Location: (52.0, 34.0, 49094)
3538	Right Click	Target: Assimilator [04B80001]; Location: (145...
3540	Right Click	Location: (57.978759765625, 22.35400390625, 40...
3551	Right Click	Location: (58.572265625, 22.791748046875, 40913)
3562	Right Click	Location: (61.143310546875, 25.4287109375, 40913)

Right Click

- event=Right Click인 경우 event_contents 값 중 Target을 포함하는 행만 추출

```
len(train.event_contents[(train.event == 'Right Click') &
                         (train.event_contents.map(lambda x: str(x)[:6]) == 'Target')].
    map(lambda x: x[x.find(':')+2:x.find(' [')]).unique())
```

```
175
```

train에는 175 종류의 속성값

```
len(test.event_contents[(test.event == 'Right Click') &
                        (test.event_contents.map(lambda x: str(x)[:6]) == 'Target')].
    map(lambda x: x[x.find(':')+2:x.find(' [')]).unique())
```

```
171
```

test에는 171 종류의 속성값

- 속성값을 더미 변수로 변환하여 game_id, player별 count

```
training's auc: 0.871089        valid_1's auc: 0.732055
```

1. 4) player 0, 1 반전

기존 학습 데이터에 player 0, 1을 반전시킨 데이터를 병합
2배의 데이터로 학습 가능 (77,744건)

```
training's auc: 0.85888
valid_1's auc: 0.741633
```

	player0_feature1	player0_feature2	player0_feature3	player1_feature1	player1_feature2	player0_feature3
0	-	-	-	-	-	-
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-

	player0_feature1	player0_feature2	player0_feature3	player1_feature1	player1_feature2	player0_feature3
0	-	-	-	-	-	-
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	-	-	-	-	-	-

2.1) CatBoost 모델링

단일 CatBoost 모델링

```
# catBoost
model = CatBoostClassifier(eval_metric = 'AUC',              # AUC로 성능 측정
                           iterations = 25000,               # 반복횟수 최대 25000
                           metric_period = 25000,            # 중간결과 출력X
                           early_stopping_rounds = 1000,     # 1000iteration 동안 AUC 증가 없으면 학습 중단
                           task_type = 'GPU',                # GPU 사용
                           grow_policy = 'Depthwise',        # 트리 노드 생성 방식
                                                             # 1) Depthwise(지정한 depth에 이를 때까지 level 순으로 노드 분할)
                                                             # 2) Lossguide(loss 변화가 큰 순으로 노드 분할)
                           depth = 10,                       # 트리 깊이
                           learning_rate = 0.03,             # 러닝레이트
                           l2_leaf_reg = 0,                  # L2 정규화
                           random_seed = 2020,               # 랜덤시드 고정
                           )
# 모델 학습
model.fit(train_X, train_y, eval_set=(valid_X, valid_y))
```

```
bestTest = 0.7450751066
```

3-fold Cross Validation

```
score = 0
kf = KFold(n_splits=3, random_state=2020)
for train_index, valid_index in kf.split(X):
    train_X, train_y = X.iloc[train_index], y[train_index]
    valid_X, valid_y = X.iloc[valid_index], y[valid_index]
```

⋮

```
valid_pred = model.predict_proba(valid_y)[:,1]
score += metrics.roc_auc_score(valid_y, valid_pred) /3
```

```
score
```

```
0.7412856971457716
```

2. 2) 파라미터 최적화

Bayesian Optimization을 통해 하이퍼 파라미터 최적화
- 한 가지 경우의 랜덤 파라미터 값으로 3-fold Cross Validation 수행, 성능 측정
- 처음 5회 랜덤 값으로 score 계산 후 45회 최적화
- 최적점으로 추정되는 네 가지 경우의 파라미터 값 구함

grow_policy (노드 생성 방식)	depth (트리 깊이)	learning_rate (러닝 레이트)	l2_leaf_reg (L2 정규화 lambda값)
Depthwise	10	0.02423	20.35 ①
	12	0.01564	49.99 ②
Lossguide	8	0.01063	5.127 ③
	16	0.01213	5.027 ④

2. 3) 앙상블

lightGBM도 합치려 했으나 CatBoost만으로 합친 게 좋아서 뺌
10-fold로 데이터 0.9만 각각 다른 조합으로 사용, 랜덤시드 2개 -> 한 가지 파라미터로 20개 모델 생성 +
(①+③)/2 *1/3 + (②+④)/2 *2/3
-> 최종 결과물 생성

3. 결과 및 결언

결론

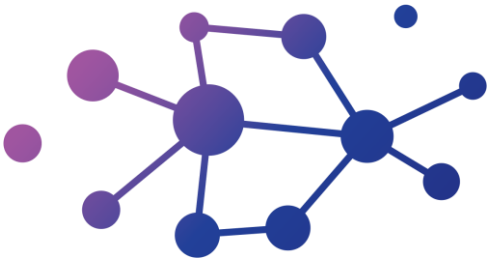
최종 순위 : 2등
Public Score : 0.7687
Private Score : 0.76542

- 이번 대회의 행동데이터를 전처리를 하여 만들어질 데이터는 sparse할 것이라 예상하여, EDA 를 통한 Feature Engineering 보다는, 가공되지 않은 raw 데이터에 포함 된 정보를 최대한 정형화된 형태로 피쳐를 생성하는 것에 집중함.
- player1 과 player2 를 스왑하여 observation을 두 배로 만들어 주는 것이 성능 향상에 도움이 됨.
- 베이지안 최적화를 통해 최적 파라미터를 구할 수 있었음.
- 파라미터의 다양성이 모델의 성능을 향상 시키므로, 총 네가지 조합의 파라미터로 앙상블
- 최종 모델 학습 단계에서 모든 observation의 정보를 이용하고, 과적합 방지를 위해서 10 - fold 로 모델을 학습 시켜 성능을 높임
- 과적합 방지를 위해 각 조합마다 랜덤 시드를 2번 바꿔줌.(총 8개의 모델을 학습시킴.)

3. 결과 및 결언

한계

- 도메인 지식 부족하여 피쳐 생성 제한적, 피쳐 생성의 논리적 근거 부족.
- 피쳐 선택, 드롭 안해서 너무 많은 차원의 데이터로 학습.
- catboost 모델 depth 깊게 준 모델을 추가함으로써 성능은 좋아졌지만 학습시간 너무 많이 소요되어 비효율적
- 앙상블 할 때 stack 시도했으나 오히려 성능하락 경험. 그러나 시간이 충분했다면 다양하게 실험하여 좋은 결과 얻었을 것이라 예상함.



THANK YOU