

Dacon 3회 게임 행동 데이터 분석 경진대회

허은정 이승훈 조윤영

팀 프로젝트 (3인, 경진대회 2위/94팀)

수행 기간 : 20.03.01 ~ 20.04.15 (약 1.5개월)

내용 : 이진 분류 / 역할 : 데이터 전처리, 소스코드 커스터마이징

기술 : 언어 - Python / 툴 - Google Colab

0. 개요

주 제

Blizzard 스타크래프트2 경기의
행동 데이터로 승패 예측

평가 지표

AUC

배 경

E-Sports 속 한국이란 나라의 위용에
걸맞은 알고리즘을 개발하여
게이머들의 전략 발전 도모

주최, 주관

- 주최 : (사)한국인공지능협회
- 주관 : DACON

분석 환경

사용 언어 : Python

작업 환경 : Google Colab

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

1

데이터 전처리

2

모델 구축&검증

3

결과 및 결론

1

데이터 전처리

- raw 데이터 탐색
- Baseline 모델링
- 변수 추출
- player 0,1 반전

2

모델 구축 & 검증

- CatBoost 모델링
- 파라미터 최적화
- 앙상블

3

결과 및 결론

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 활용할 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	???								X				
		game_id	winner	time	player	species	event	event_contents		feature_1	feature_2	feature_3	feature_4	y
0	0	1	0.00	0	T	Camera	at (145.25, 21.5078125)	at (145.25, 21.5078125)	0	-	-	-	-	0
1	0	1	0.00	1	T	Camera	at (22.75, 147.0078125)	at (22.75, 147.0078125)	1	-	-	-	-	0
2	0	1	0.02	0	T	Selection	['OrbitalCommand [3080001]']	['OrbitalCommand [3080001]']	2	-	-	-	-	0
3	0	1	0.02	0	T	Ability	(1360) - TrainSCV	(1360) - TrainSCV	3	-	-	-	-	0
4	0	1	0.14	0	T	Camera	at (142.99609375, 24.50390625)	at (142.99609375, 24.50390625)	4	-	-	-	-	0
...
67091771	38871	0	8.51	0	Z	Camera	at (139.578125, 62.58203125)	at (139.578125, 62.58203125)	38867	-	-	-	-	0
67091772	38871	0	8.52	1	T	GetControlGroup			38868	-	-	-	-	0
67091773	38871	0	8.52	0	Z	Camera	at (122.42578125, 45.4296875)	at (122.42578125, 45.4296875)	38869	-	-	-	-	0
67091774	38871	0	8.52	0	Z	Camera	at (122.42578125, 43.25390625)	at (122.42578125, 43.25390625)	38870	-	-	-	-	0
67091775	38871	0	8.52	1	T	Ability	(1360) - TrainSCV	(1360) - TrainSCV	38871	-	-	-	-	0

67091776 rows x 7 columns

38872 rows x 4 columns

1. 2) Baseline 모델링

lightGBM 패키지 활용

1. game_id 개수만큼의 index를 가진 DataFrame X 생성
2. train 데이터에서 game_id별 winner 추출하여 Series y 생성
3. train, valid 데이터로 분할
4. 모델링

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

38872 rows x 1 columns

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

valid_1's auc: 0.504763

- game_id별 player 0, 1의 종족 -> 더미 변수

time

valid_1's auc: 0.553378

- game_id별 마지막 값 추출 (플레이 시간)
- 1.23은 1분 23초를 의미 -> 초 단위로 변환

event

valid_1's auc: 0.575306

- game_id, player별 event 횟수 count
- time 변수로 나눈 값 -> 파생 변수 추가

event value

valid_1's auc: 0.656082

- 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)
```

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

1. 데이터 전처리 & EDA

	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



species

valid_1's auc: 0.663062

- event=Camera인 경우
event_contents의 x, y 좌표 추출
- game_id, player별 좌표 이동 거리의
sum, min, median, max 변수 추가
- 30초 이내의 레코드 -> 파생 변수 생성

	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...

(1021) - BuildSupplyDepot; Location: (135.0, 4...

(1360) - TrainSCV

(480) - Stop

(1360) - TrainSCV

(1023) - BuildBarracks; Location: (135.5, 39.5...

Ability

valid_1's auc: 0.70131

- event=Ability인 경우
event_contents 일부 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

- 속성값 -> 더미 변수 변환 후
game_id, player별 count
- time 변수로 나눈 값 -> 파생 변수 추가

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

1. 데이터 전처리 & EDA

	event	event_contents
2	Selection	['OrbitalCommand' [3080001]]
28	Selection	['OrbitalCommand' [3080001]]
68	Selection	['SCV' [3280001]]
102	Selection	['CreepOnlyBlocker4x4' [1800001]]
103	Selection	['SCV' [30C0001] 'SCV' [3780001]]

Selection

valid_1's auc: 0.730074

- event=Selection인 경우 event_contents 값 일부 추출
- 다중 속성값 존재 (구분자 ';') -> 제1정규화
- 속성값 -> 더미 변수 변환 후 game_id, player별 count
- time 변수로 나눈 값 -> 파생 변수 추가
- 30초 이내의 레코드 -> 파생 변수 추가

```
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

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

valid_1's auc: 0.732055

- 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

1. 4) player 0, 1 반전

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

단일 CatBoost 모델링

```
# catBoost
model = CatBoostClassifier(eval_metric = 'AUC',
                           iterations = 25000,
                           metric_period = 25000,
                           early_stopping_rounds = 1000,
                           task_type = 'GPU',
                           grow_policy = 'Depthwise',

                           depth = 10,
                           learning_rate = 0.03,
                           l2_leaf_reg = 0,
                           random_seed = 2020,
                           )

# 모델 학습
model.fit(train_X, train_y, eval_set=(valid_X, valid_y))
```

AUC로 성능 측정
반복횟수 최대 25000
중간결과 출력X
1000iteration 동안 AUC 증가 없으면 학습 중단
GPU 사용
트리 노드 생성 방식
1) Depthwise(지정한 depth에 이를 때까지 level 순으로 노드 분할)
2) Lossguide(loss 변화가 큰 순으로 노드 분할)
트리 깊이
러닝레이트
L2 정규화
랜덤시드 고정

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) 하이퍼 파라미터 최적화

Bayesian Optimization을 통해 하이퍼 파라미터 최적화

- 한 가지 경우의 랜덤 파라미터 값으로 3-fold Cross Validation 수행, 성능 측정
- 처음 5회 score 값을 바탕으로 45회 최적화
- 최적점으로 추정되는 네 가지 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

Sampling

- 데이터를 10분할하여 (10-fold) 10가지 모델 생성
- 랜덤시드를 바꾸어 2가지 모델 생성
-> 하나의 파라미터 설정값 별로 20개 모델 앙상블

Stacking

- $(①+③)/2 * 1/3 + (②+④)/2 * 2/3$ 로 가중평균
-> 파라미터 설정값 별 4가지 모델 생성

결론

최종 순위 : 2위

Public Score : 0.7687 / Private Score : 0.76542

- 전처리 후의 데이터가 sparse할 것이라 예상하여, EDA 를 통한 Feature Engineering 보다는 raw 데이터에 포함 된 정보를 반영하는 피처를 최대한 생성하는 것에 집중함.
- player0 과 player1 를 스왑하여 레코드를 2배로 늘린 것이 성능 향상에 도움이 됨.
- 베이지안 최적화를 통해 최적 파라미터를 구하는 시간을 단축함.
- 파라미터의 다양성이 모델의 성능을 향상 시키므로, 총 네가지 조합의 파라미터로 앙상블
- 최종 모델 학습 단계에서 모든 레코드의 정보를 이용하고, 과적합 방지를 위해서 10 - fold 로 모델을 학습 시켜 성능을 높임

한계

- 도메인 지식 부족하여 피처 생성 제한적, 피처 생성의 논리적 근거 부족.
- 트리 모델의 depth 깊게 설정해 학습 시간이 비효율적

- 끝 -