**[Dacon] 행동 데이터 분석 인공지능 AI 경진대회 2등 코드 분석 — 나무늘보의 개발 블로그**

**노트북:** 첫 번째 노트북
**만든 날짜:** 2021-07-04 오후 10:51
**URL:** https://continuous-development.tistory.com/253

Data scientist/Data Science

# [Dacon] 행동 데이터 분석 인공지능 AI 경진대회 2등 코드 분석

2021. 7. 4. 20:45 수정 삭제 공개


## 행동 데이터 분석 인공지능 AI 경진대회

## 주제

대회는 Blizzard 스타크래프트2 경기의 행동 데이터로 승패를 예측

# 배경

게임을 잘하는 나라', 'E-sports의 성지'라는 호칭을 얻게 된 요인에 게이머들의 탁월한 전략이 함께 합니다. 그리고 여러분은 데이터를 분석하여 전략을 발전시킬 수 있는 능력을 갖추고 있습니다. E-Sports 속 한국이란 나라의 위용에 걸맞은 알고리즘을 만들어주세요! 여러분이 만든 알고리즘이 우리의 게임 실력을 한층 더 발전시킬 수 있습니다.

# 평가

- AUC

# 소스

**[2등][도발하려던건 아니었습니다만]Ensembled CatBoost Model**

## 1.Library & Data

```
# 라이브러리 설치
import os                              # 디렉토리 설정
os.chdir("/data")
import warnings                          # 경고 메세지 무시
warnings.filterwarnings('ignore')
import pandas as pd                        # 데이터 조작, 분석
import numpy as np                         # 행렬 연산
import random                            # 난수 생성
random.seed(2020)
random_seed = 2020
import time                             # 시간 측정
import re                              # 정규표현식

from sklearn.model_selection import train_test_split # train, validation 데이터 나누기
from sklearn import metrics                  # AUC 측정
!pip install catboost
from catboost import CatBoostClassifier, Pool      # CatBoost 모델링
import lightgbm as lgb                       # lightGBM 모델링
from sklearn.model_selection import KFold         # K-fold CV
!pip install bayesian-optimization
```

```
from bayes_opt import BayesianOptimization        # 베이지안 최적화 라이브러리
from functools import partial                     # 함수 변수 고정

# 데이터 불러오기
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
```

## 2. 데이터 전처리

```
# 반응변수 전처리
def preprocess_y(data, exchange_player=False):
 y = data.drop_duplicates(['game_id', 'winner']).winner.reset_index(drop=True)
 if (exchange_player == True):
   y = y.append(-(y - 1)).reset_index(drop=True)
 return y
```

```
# 설명변수 전처리
def preprocess_X(data, exchange_player=False):

 # game_id 개수만큼의 index를 가진 DataFrame X 생성
 n = data.game_id.max() + 1
 X = pd.DataFrame(index=range(n)[data.game_id.min():])

 # time 변수
 X['time'] = data.drop_duplicates(['game_id'],keep='last').set_index('game
 X['time'] = (X.time*100//100*60 + X.time*100%100).astype(int)

 # species 더미 변수
 X = pd.concat([pd.get_dummies(data[data.player == 0].drop_duplicates(['g
           pd.get_dummies(data[data.player == 1].drop_duplicates(['game_i
           X],axis=1)

 # event 카운트
 contents = data.loc[:,['player','game_id','time']].groupby(['player', 'game
 contents.columns = ['0_event', '1_event']
 X['0_event'], X['1_event'] = contents['0_event'], contents['1_event']

 # event 카운트 / time
 X['0_event_per_sec'], X['1_event_per_sec'] = X['0_event'] /X.time, X['1_

 # event == Ability, AddToControlGroup, Camera, ControlGroup, Get
```

```
contents = data.loc[:,['player','event','game_id','time']].groupby(['player
contents.columns = ['0_'+x for x in sorted(data.event.unique())] + ['1_'+x f
for i in contents.columns:
  X[i] = contents[i]

# event == Camera 일 때 event_contents의 2차원 좌표 간 euclidean d
def move_sum(i):
  return sum(np.sqrt(np.diff(i.map(lambda x: x[4:x.find(',')]).astype(float)) **
                np.diff(i.map(lambda x: x[x.find(',')+2:len(x)-1]).astype(float))
def move_min(i):
  if len(i) == 1:
    return 0
  return min(np.sqrt(np.diff(i.map(lambda x: x[4:x.find(',')]).astype(float)) **
                np.diff(i.map(lambda x: x[x.find(',')+2:len(x)-1]).astype(float))
def move_median(i):
  if len(i) == 1:
    return 0
  return np.median(np.sqrt(np.diff(i.map(lambda x: x[4:x.find(',')]).astype(fl
                    np.diff(i.map(lambda x: x[x.find(',')+2:len(x)-1]).astype(flo
def move_max(i):
  if len(i) == 1:
    return 0
  return max(np.sqrt(np.diff(i.map(lambda x: x[4:x.find(',')]).astype(float)) **
                np.diff(i.map(lambda x: x[x.find(',')+2:len(x)-1]).astype(float))
contents = (data[data.event == 'Camera'].loc[:,['player','game_id','event
        groupby(['player','game_id'])).agg([move_sum,move_min,move_m
contents.columns = [y+x for x in ['sum','min','median','max'] for y in ['0
for i in contents.columns:
  X[i] = contents[i].fillna(0)

# 30초 이내 move_sum
contents = (data[(data.time < (data.event == 'Camera')].loc[:,['player','ga
        groupby(['player','game_id'])).agg(move_sum).unstack(level=0)
contents.columns = ['0_move_sum_30sec','1_move_sum_30sec']
for i in contents.columns:
  X[i] = contents[i]

# event == Ability 일 때 event_contents 더미 변수 생성, 카운트
contents = pd.DataFrame(data.event_contents[(data.event == 'Ability')].m
contents['game_id'], contents['player'], contents['count'] = data.game_id
contents_X = pd.DataFrame(columns=[x+y for x in ['0_','1_'] for y in conte
contents = contents.groupby(['player','event_contents','game_id']).cour
contents.columns = contents.columns.map(lambda x: str(x[1])+'_'+x[2])
```

```
contents_X = pd.concat([contents_X, contents])
for i in contents_X.columns:
  X[i] = contents_X[i]
  X[i] = X[i].fillna(0).astype(int)

# event == Ability 일 때 event_contents 더미 변수 생성 / time
for i in contents_X.columns:
  X[i+'_div_time'] = X[i] /X.time

# event == Selection 일 때 event_contents 더미 변수 생성, 카운트
contents = data[data.event == 'Selection'].event_contents.map(lambda x
                                        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(lar
contents = pd.concat([t[i] for i in range(max_num)])
contents['game_id'], contents['player'], contents['count'] = data.game_id
contents_X = pd.DataFrame(columns=[x+y for x in ['0_','1_'] for y in conte
contents = contents.groupby(['player','event_contents','game_id']).cour
contents.columns = contents.columns.map(lambda x: str(x[1])+'_'+x[2])
contents_X = pd.concat([contents_X, contents])
for i in contents_X.columns:
  X[i] = contents_X[i]
  X[i] = X[i].fillna(0).astype(int)

# event == Selection 일 때 event_contents 더미 변수 생성 / time
for i in contents_X.columns:
  X[i+'_div_time'] = X[i] /X.time

# 30초 이내 event == Selection 일 때 event_contents 더미 변수 생성,
contents = data[(data.time < (data.event == 'Selection')].event_contents.r
                                        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(lar
contents = pd.concat([t[i] for i in range(max_num)])
contents['game_id'], contents['player'], contents['count'] = data.game_id
contents_X = pd.DataFrame(columns=[x+y for x in ['0_','1_'] for y in conte
contents = contents.groupby(['player','event_contents','game_id']).cour
contents.columns = contents.columns.map(lambda x: str(x[1])+'_'+x[2])
```

```
    contents_X = pd.concat([contents_X, contents])
    for i in contents_X.columns:
      X[i+'_30sec'] = contents_X[i]
      X[i+'_30sec'] = X[i+'_30sec'].fillna(0).astype(int)

    # event == Right Click 일 때 Target 이름 더미 변수 생성, 카운트
    contents = pd.DataFrame(data.event_contents[(data.event == 'Right Click
    contents['game_id'], contents['player'], contents['count'] = data.game_id
    contents_X = pd.DataFrame(columns=[x+y for x in ['0_Target_','1_Target_
    contents = contents.groupby(['player','event_contents','game_id']).cour
    contents.columns = contents.columns.map(lambda x: str(x[1])+'_Target_'+
    contents_X = pd.concat([contents_X, contents])
    for i in contents_X.columns:
      X[i] = contents_X[i]
      X[i] = X[i].fillna(0).astype(int)

    # 컬럼 이름 순서로 정렬
    X = X[sorted(X.columns)]

    # player 0,1 자리 바꾼 X1 생성, X와 행 병합해 데이터 2배로 만들기
    if (exchange_player == True):
      c = X.shape[1]//2
      X1 = X.copy()
      X1.columns = list(X.columns[c:2*c])+list(X.columns[:c])+['time']
      X1.index = [x+n for x in range(n)]
      X = pd.concat([X, X1])

    return X
```

EDA 를 통한 Feature Engineering 보다는, 가공되지 않은 raw 데이터에 포함 된 정보를 최대한 정형화된 형태로 피쳐를 생성하는 것에 집중

모델 학습 단계에서 모든 observation의 정보를 이용

player1 과 player2 를 스왑하여 observation을 두배로 만들어 주는것이 성능 향상에 도움

```
# train, test 전처리 수행, y(반응변수), X,test_X(설명변수) 생성
y = preprocess_y(train, True)
X = preprocess_X(train, True)
```

```
test_X = preprocess_X(test, False)

# 메모리 효율 위해 train, test raw data 삭제
del train, test

# X, test_X에만 있는 컬럼 삭제
X.drop(set(X.columns) - set(test_X.columns), axis=1, inplace=True)
test_X.drop(set(test_X.columns) - set(X.columns), axis=1, inplace=True)
```

## 3. 변수 선택 및 모델 구축

```
# CatBoost 모델링
def catboost_modeling(x_train, y_train, x_test, grow_policy, depth, le

 # 빈 Series인 test_pred 생성
 test_pred = pd.Series([0 for x in range(len(x_test))], index=x_test.index)

 # 10-fold 모델링을 n회 반복할 것
 for i in range(n):
  kf = KFold(n_splits=10, random_state=random_seed+i)
  for train_index, valid_index in kf.split(x_train):
   train_X, train_y = x_train.iloc[train_index], y_train[train_index]
   valid_X, valid_y = x_train.iloc[valid_index], y_train[valid_index]

   # catBoost(grow_policy='Depthwise')
   model = CatBoostClassifier(eval_metric = 'AUC',          # AUC로 성능
                              iterations = 25000,           # 반복횟수 최대 25000
                              metric_period = 25000,        # 중간결과 출력X
                              early_stopping_rounds = 1000,    # 1000iteration 동
                              task_type = 'GPU',            # GPU 사용
                              grow_policy = grow_policy,       # 트리 노드 생성 방식
                                                  # 1) Depthwise(지정한 depth에
                                                  # 2) Lossguide(loss 변화가 큰 순
                              depth = depth,                # 트리 깊이
                              learning_rate = learning_rate,    # 러닝레이트
                              l2_leaf_reg = l2_leaf_reg,       # L2 정규화
                              random_seed = random_seed+i,     # 랜덤시드 고정
                              )
```

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

        # 모델 적용
        test_pred += model.predict_proba(x_test)[:,1] /(10*n)

    # 설정된 디렉토리에 결과물 저장
    sample_submission = pd.read_csv('sample_submission.csv', index_col=0
    submission = pd.DataFrame(data=test_pred, columns=sample_submission.
    submission.to_csv('CatBoost_'+grow_policy+'_'+str(depth)+'.csv', index=

    return test_pred
```

10 - fold 로 모델을 학습 시키고 평균을 내는 것이 성능을 많이 높일 수 있었음

조합마다 랜덤시드를 2번 바꿔주었기에 (catboost_modeling 사용자 함수에서의 n 파라미터) 총 8개의 모델을 학습

## 4. 모델 학습 및 검증

```
data1 = catboost_modeling(X, y, test_X, 'Depthwise', 10, 0.02423, 20.35, 2
data2 = catboost_modeling(X, y, test_X, 'Lossguide', 8, 0.01063, 5.127, 20
data3 = catboost_modeling(X, y, test_X, 'Depthwise', 12, 0.01564, 49.99, 2
data4 = catboost_modeling(X, y, test_X, 'Lossguide', 16, 0.01213, 5.027, 2
```

catboost알고리듬의 growplicy 옵션의 파라미터 값( "Lossgiude" 와 "Depthwise") 에 따라 베이지안 최적화를 팀원과 함께 해본 결과, depth와 l2_leaf_reg 등의 최적값은 다르게 나왔음

파라미터의 다양성이 모델의 성능을 향상시킨다는 것을 알게 되어, 총 네가지 조합의 파라미터로 앙상블

## 5. 결과 및 결언

```
# 최종 모델 앙상블
sample_submission = pd.read_csv('sample_submission.csv', index_col=0)
data_final = pd.DataFrame((data1+data2)/2 *1/3 + (data3+data4)/2 *2/3)
data_final.columns = sample_submission.columns
data_final.to_csv('data_final.csv', index =True)
data_final
```

# 정리

play 1 / 2 바꿔서 데이터양 2배로 늘린 것

한가지 모델의 파라미터의 다양성을 앙상블 하는 방법도 있음

10 fold 평균을 사용한 것이 성능에 있어 도움 됨(최소 5는 하는게 좋다고 한다.)

# 후기

앙상블을 하는데 있어서 한가지 모델로 여러가지 파라미터를 넣은 걸 한다는게 신기했다. 이렇게 앙상블도 한다는 것도 배웠다. 그것고 데이터양을 늘린것도 신기했다. 예전에 이미지 관련 딥러닝에서 데이터를 늘려 학습하는 것을 얼핏 들었었는데 여기서도 그렇게 한다는 것을 배웠다.

**'Data scientist > Data Science'** 카테고리의 다른 글

**[Dacon] 행동 데이터 분석 인공지능 AI 경진대회 2등 코드 분석**

[Data Science] 데이터 사이언스 개념 - 10.딥러닝

[Data Science] 데이터 사이언스 개념 - 9.신경망이 기초

[Data Science] 데이터 사이언스 개념 - 8.토픽 모델 / 네트워크 분석□

[Data Science] 데이터 사이언스 개념 - 7.비지도 학습□

[Data Science] 데이터 사이언스 개념 - 6.분류문제□

나아무늘보

혼자 끄적끄적하는 블로그 입니다.