팀명

Paul the Foul

팀장

김내히(1999121@yonsei.ac.kr)

팀원

조유림(rim6@naver.com)

권혁(kh1204350@gmail.com)

홍익선(dlrtjs512@naver.com)

임선우(96limtotoro@gmail.com)

최종 결과 보고서

2020 빅콘테스트 데이터분석 퓨처스리그(야구)

목차

대회 설명

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

팀명 유래

“Paul the Foul”

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

Paul은 2010 남아공 월드컵 결과를 예측하여 화제가 된 문어이다. 이 문어의 이름에 운율을 맞추기 위해 야구 용어 **foul**을 덧붙임.

야구 경기 결과를 정확히 예측하고자 하는 열망

대회 설명

- 목적 : KBO 팀의 정규시즌 잔여기간 팀별

1) 방어율 2) 타율 3) 승률 예측

- 데이터 제공 : 스포츠투아이 (https://www.sports2i.com/)

- 2016년부터 2020.7.20까지의 야구 데이터 제공
- 이를 토대로 2020.9.28 ~ 종료시점까지의 승률 , 타율 , 방어율 예측

데이터 설명

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

시트 NO	데이터	설명
시트 1	팀	팀을 식별하기 위한 코드, 팀명
시트 2	경기	게임에 관한 설명; 게임KEY 경기일자 홈/원정팀 더블헤더 여부 등
시트 3	선수	선수 정보; 시즌 선수코드 선수명 팀코드 포지션 나이 연봉
시트 4	등록선수	일자(등록/말소) 팀코드 선수코드 등록/말소 여부
시트 5	팀 투수	각 경기 별 양팀 투수들의 종합적인 정보; 상대타자 수 , 삼진, 이닝, 자책점
시트 6	팀 타자	각 경기 별 양팀 타자들의 종합적인 성적 ; 타수, 안타, 도루, 4구, 병살타 등
시트 7	개인 투수	각 경기에 등판한 투수들의 성적
시트 8	개인 타자	각 경기에 출전한 타자들의 성적

개인 성적을 통해 타율 방어율 예측 ➔ 시트 7 8 활용

2020년 팀 성적을 통해 승률 예측 ➔ 시트 5 6 활용

데이터 설명

대회설명

EDA 및 변수선택

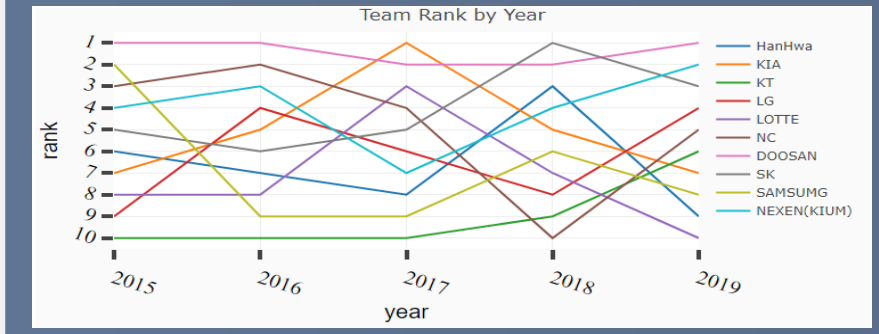
모델링방식

타율 방어율 승률 예측

개인 성적을 통해 타율 방어율 예측한 이유

근거 1

팀 성적은 시간에 따른 변동성이 매우 크다 → 변동성이 덜한 개인 성적으로 모델링



근거 2

정보량 : 팀 데이터는 개인 선수 별 데이터 압축

- 시트 5~6만은 정보 손실이 크며 데이터의 행 개수가 매우 적다

- 야구는 동료의 성적이 개인성적에 미치는 영향이 적어 “나무 → 숲”의 접근이 가능하다!

데이터 설명

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

2020년 팀 성적을 통해 승률 예측한 이유

"피타고리안 승률"

$$\text{기대승률} = \frac{(\textbf{득점})^2}{(\textbf{득점})^2+(\textbf{실점})^2}$$

각 팀의 2020년 득점 실점 총 합을 집어넣어
피타고리안 승률 공식을 통해 승률 예측

EDA 및 변수 선택

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

DATA Concatenation

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

```
# 개인투수
private_pitcher_2016 = pd.read_csv('2020빅콘테스트_스포츠투아이_제공데이터_개인투수_2016.csv', engine='python', encoding="CP949")
private_pitcher_2017 = pd.read_csv('2020빅콘테스트_스포츠투아이_제공데이터_개인투수_2017.csv', engine='python', encoding="CP949")
private_pitcher_2018 = pd.read_csv('2020빅콘테스트_스포츠투아이_제공데이터_개인투수_2018.csv', engine='python', encoding="CP949")
private_pitcher_2019 = pd.read_csv('2020빅콘테스트_스포츠투아이_제공데이터_개인투수_2019.csv', engine='python', encoding="CP949")
private_pitcher_2020 = pd.read_csv('2020빅콘테스트_스포츠투아이_제공데이터_개인투수_2020.csv', engine='python', encoding="CP949")

# 개인타자
private_batter_2016 = pd.read_csv('2020빅콘테스트_스포츠투아이_제공데이터_개인타자_2016.csv', engine='python', encoding="CP949")
private_batter_2017 = pd.read_csv('2020빅콘테스트_스포츠투아이_제공데이터_개인타자_2017.csv', engine='python', encoding="CP949")
private_batter_2018 = pd.read_csv('2020빅콘테스트_스포츠투아이_제공데이터_개인타자_2018.csv', engine='python', encoding="CP949")
private_batter_2019 = pd.read_csv('2020빅콘테스트_스포츠투아이_제공데이터_개인타자_2019.csv', engine='python', encoding="CP949")
private_batter_2020 = pd.read_csv('2020빅콘테스트_스포츠투아이_제공데이터_개인타자_2020.csv', engine='python', encoding="CP949")

private_pitcher = pd.concat([private_pitcher_2016, private_pitcher_2017, private_pitcher_2018, private_pitcher_2019, private_pitcher
private_batter = pd.concat([private_batter_2016, private_batter_2017, private_batter_2018, private_batter_2019, private_batter_2020]
```

연도별로 저장된 기존 데이터를 통합하며 개인 투수, 개인 타자 데이터 생성

NA 확인

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

```
print(private_pitcher.shape)
print(private_pitcher.isna().sum())
print(np.sum(private_pitcher=="?", axis=1).value_counts())
```

```
(27804, 38)
G_ID          0
GDAY_DS       0
T_ID          0
VS_T_ID       0
HEADER_NO     0
TB_SC         0
P_ID          0
START_CK      0
RELIEF_CK     0
CG_CK         0
QUIT_CK       0
WLS           0
HOLD          0
INN2          0
BF            0
PA            0
AB            0
HIT           0
H2            0
H3            0
HR            0
SB            0
CS            0
SH            0
SF            0
BB            0
IB            0
HP            0
KK            0
GD            0
WP            0
BK            0
ERR           0
R             0
ER            0
P_WHIP_RT     0
P2_WHIP_RT    0
CB_WHIP_RT    0
dtype: int64
0    27804
dtype: int64
```

```
print(private_batter.shape)
print(private_batter.isna().sum())
print(np.sum(private_batter=="?", axis=1).value_counts())
```

```
(81102, 31)
G_ID            0
GDAY_DS         0
T_ID            0
VS_T_ID         0
HEADER_NO       0
TB_SC           0
P_ID            0
START_CK        0
BAT_ORDER_NO    0
PA              0
AB              0
RBI             0
RUN             0
HIT             0
H2              0
H3              0
HR              0
SB              0
CS              0
SH              0
SF              0
BB              0
IB              0
HP              0
KK              0
GD              0
ERR             0
LOB             0
P_HRA_RT        0
P_AB_CN         0
P_HIT_CN        0
dtype: int64
0    81102
dtype: int64
```

NA값 없다

새 변수 생성

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

변수명	설명	공식
AVG	타율	HIT/AB
ERA	방어율	(ER*9)/(INN2/3)
SB_trial	도루 시도 횟수	SB+CS
SB_SR	도루 성공률	SB/(SB+CS)
PA-PB	타석수 - 타수	
SH+SF	희생타 + 희생플라이	
BABIP		(HIT-HR)/(AB-KK-HR+SF)
KK9	9이닝 당 삼진 수	KK/INN2*27
BB9	9이닝 당 볼넷 수	(BB+HP)/INN2*27
SLG	장타 허용	H2+H3+HR
H1	단타 허용	HIT-SLG

변수 1차 제거 (투수)

- PA 대신 (PA - AB) 사용 : 단순히 타석 수보다 투수 허용 사사구 수 (BB, IB, HP 총합) 이 연관성 있음
- SB, CS 대신 SB_trial (도루 시도 횟수) , SB_SR(도루 성공률) 변수 생성

변수 1차 제거 (타자)

- H1, H2, H3, H4 등 타율의 분자, 즉 안타의 구성요소를 제거 : 장타율을 예측하는 게 아니니까!
- SB, CS 대신 **SB_trial** (도루 시도 횟수) , **SB_SR** (도루 성공률) 사용
- BB, IB, HP 대신 이들을 모두 고려하는 **(PA - AB)** 사용
- SH, SF의 차이는 중요하지 않다! 자신의 희생으로 동료 선수의 진루를 돕는 것 : **SH + SF** 변수 사용

변수 2차 제거

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

- 투수 데이터에서 높은 상관관계를 보이는 변수 제거 ➔ SLG(장타), KK(삼진), HIT(안타), AB(타수), BF(투구 수) 제거
- 타자 데이터의 x변수는 높은 상관관계를 보이는 변수가 없었다 ➔ 그대로 사용

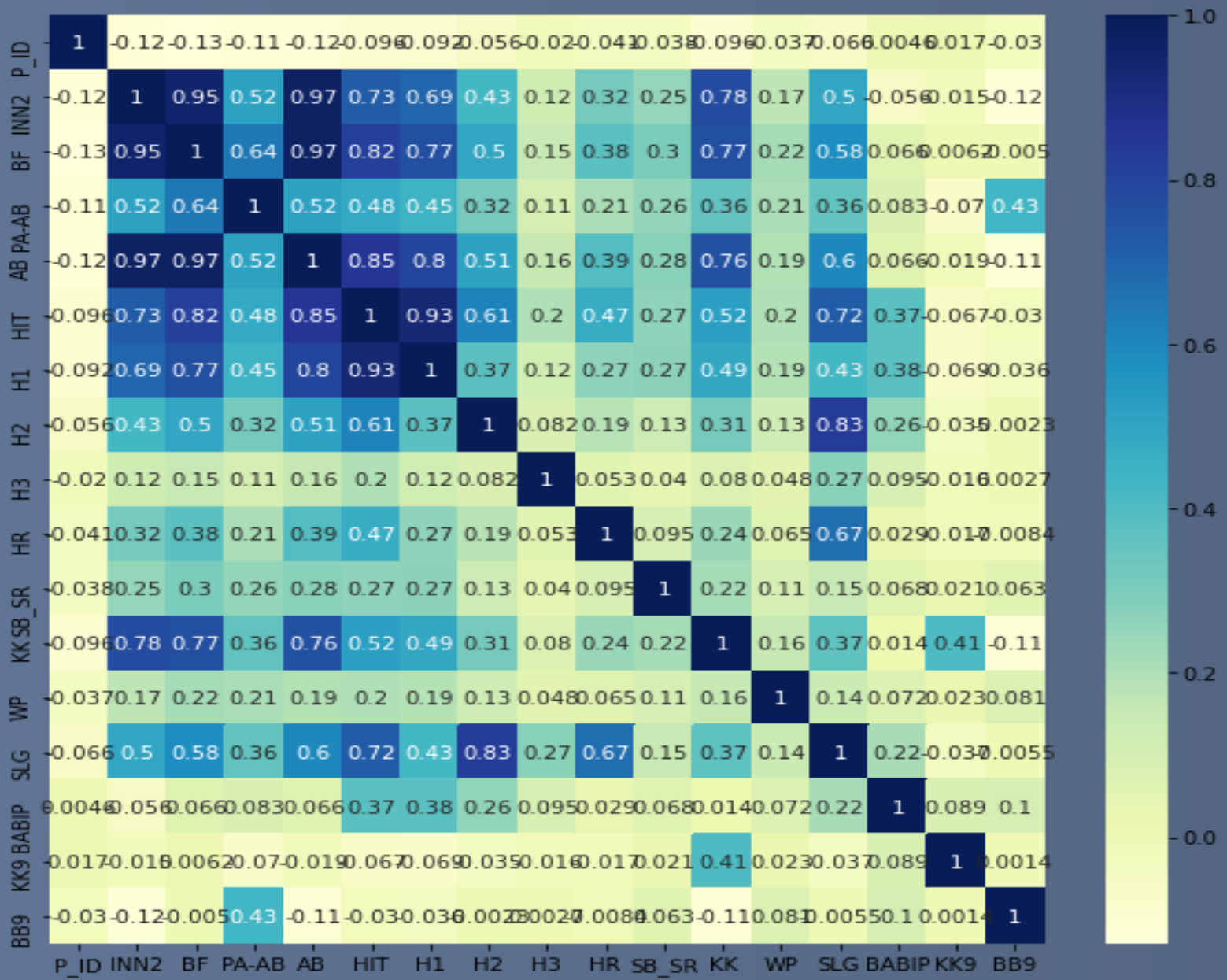
X변수 간 correlation (투수)

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측



X변수 간 correlation (타자)

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

최종 변수

투수	
X변수	TB_SC , PA-AB , H1 , H2 , H3 , HR , SB_SR , WP, BABIP , KK9, BB9
Y변수	INN2 , ER
식별코드	GDAY_DS , T_ID , P_ID

타자	
X변수	TB_SC , PA-AB , AB , RUN , RBI , H1T, SH+SF, KK, SB_trial, BABIP
Y변수	HIT , AB
식별코드	GDAY_DS , T_ID , P_ID

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

모델링

타율/방어율

모델링

타율은 HIT AB로 구성되고, 방어율은 ER INN2로 구성된다. 따라서 Y값을 HIT AB ER INN2로 놓고 나머지 변수들을(X) 통해 Y값을 예측하려고 한다. 예측 방식에서 크게 두 가지 방법을 시도했다

방식 1

시계열로 x값들을 예측하고 그 x값을 머신러닝 모델에 넣어 y값을 예측하는 방식

방식 2

기존의 데이터프레임에서 x값은 그대로 두고, y값은 해당 선수가 출전한 다음 경기의 값으로 채워주는 방식으로 변환. 즉 각 선수가 출전한 마지막 경기의 x값을 통해 다음 경기의 y값을 예측하는 방식. 독립변수를 예측할 필요가 없어진다.

최종적으로 **방식2**를 채택. 방식1과 방식2를 뒤 슬라이드에서 자세히 설명하려고 한다.

모델링 방식 1

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

1. 독립변수의 부재

우리가 제공받은 데이터(train data)에는 독립변수 종속변수가 모두 존재하지만, 예측 시점(2020.09.28~2020.10.18)에는 독립변수 종속변수 모두 미지수다. 미래시점에서 각 선수가 몇 개의 홈런을 칠지 알 수 없기 때문에!

→ 독립변수를 예측해야 한다.

2. 독립변수 예측

시계열 trend 예측방식 ➔ " Exponential smoothing(지수평활법)"
최신 데이터에 가중치를 더 부여하는 방식으로 x값을 예측

- 지수 평활법에는 simple exponential smoothing, holt's exponential smoothing 방식이 존재 ➔ grid search 방식으로 파라미터 튜닝

모델링 방식 1

3. Grid-search: 각 X값에 대해서 파라미터 튜닝

- 각 변수 별 데이터를 7:3의 비율로 train set, validation set으로 나눈다. 각 선수에 대한 MSE를 구하고 전체 선수에 대한 평균 MSE가 가장 작은 것을 최종 파라미터로 선정

- Train data가 10개 이하인 경우에는 단순 평균 값을 예측 값으로 사용

- 모든 x변수에 대해서(투수.타자 모두) **simple exponential smoothing(smoothing_level=0.1)** 모델을 사용할 때 예측력이 가장 높게 나왔다


```
def x_variable(col_name):
    best_score = 100000000
    best_level = 0
    for level in tqdm_notebook([0.1,0.2,0.4,0.6,0.8]):
        total_mse=0
        avg_mse = 0
        for i in temp.P_ID.unique():

            df = temp[(temp['P_ID']==i)&(temp['GDAY_DS'].dt.year<2020)]
            df = df[['GDAY_DS',col_name]]

            train_num = int(round(df.count()[1]*0.7))
            train = df[0:train_num]
            test = df[train_num:]

            fit1 = SimpleExpSmoothing(np.array(train[col_name])).fit(smoothing_level=level)
            pred = fit1.forecast(len(test))
            pred = pd.DataFrame(pred)

            pred.index = test['GDAY_DS']
            pred= pred.rename(columns={0:'prediction'})
            result = pd.merge(test,pred,on='GDAY_DS')
            mse = mean_squared_error(result[col_name], result.prediction)
            total_mse = total_mse + mse

        avg_mse = total_mse/temp.P_ID.nunique()
        if avg_mse < best_score:
            best_score = avg_mse
            best_level = level
    print(col_name, '(First-method) level: ',best_level,' MSE: ',best_score)
```

```
def x_variable2(col_name):
    best_score = 100000000
    best_level = 0
    best_slope = 0
    for level in tqdm_notebook([0.1,0.2,0.4,0.6,0.8]):
        for slope in [0.1,0.2,0.4,0.6,0.8]:
            total_mse=0
            avg_mse = 0
            for i in temp.P_ID.unique():
                time.sleep(0.01)

                df = temp[(temp['P_ID']==i)&(temp['GDAY_DS'].dt.year<2020)]
                df = df[['GDAY_DS',col_name]]

                train_num = int(round(df.count()[1]*0.7))
                train = df[0:train_num]
                test = df[train_num:]

                fit2 = Holt(np.array(train[col_name])).fit(smoothing_level=level, smoothing_slope=slope)
                pred = fit2.forecast(len(test))
                pred = pd.DataFrame(pred)

                pred.index = test['GDAY_DS']
                pred= pred.rename(columns={0:'prediction2'})
                result = pd.merge(test,pred,on='GDAY_DS')
                mse = mean_squared_error(result[col_name], result.prediction2)
                total_mse = total_mse + mse

            avg_mse = total_mse/temp.P_ID.nunique()
            if avg_mse < best_score:
                best_score = avg_mse
                best_level = level
                best_slope = slope
    print(col_name,'(Second-method) level: ',best_level,' slope: ',best_slope,' MSE: ',best_score)
```

모델링 방식 1

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

4. 예측된 x값을 통해 y값 예측

- Xgboost lgbm random forest 3가지 모델을 사용하여 INN2 ER AB HIT 값을 예측하고, 예측 값으로 최종 타율 방어율을 계산함

5. 한계

- 지수평활법으로 진행한 방식의 타율 방어율 예측 값이 비현실적으로 나옴 ➔ 다른방식(방식2)

T_ID	타율	방어율
NC	0.374757	3.321016
LG	0.509124	3.341757
SS	0.427517	4.459596
HT	0.398210	3.422274
WO	0.383782	2.998738
LT	0.393120	4.101977
SK	0.417878	3.836402
HH	0.466362	4.485550
OB	0.536851	3.121050
KT	0.443416	5.302426

모델링 방식 2
➔ 최종 방식

1. 데이터 프레임 변형

- (T-1)시점의 독립변수로 T시점의 종속변수를 예측하자! 데이터셋의 X값이 (T-1)시점의 값이면 Y값은 (T-1) 시점의 Y 대신 T시점의 Y

	GDAY_DS	T_ID	P_ID	TB_SC	PA-AB	AB	RUN	RBI	HIT	SH+SF	KK	AVG	SB_trial	BABIP
76138	2020-06-03	KT	50054	B	0	3	0	0	2	0	0	0.666667	0	0.666667
76262	2020-06-04	KT	50054	B	1	3	1	0	0	0	1	0.000000	0	0.000000
76522	2020-06-06	KT	50054	T	0	0	0	0	0	0	0	0.000000	0	0.000000
76647	2020-06-07	KT	50054	T	0	3	0	0	1	0	2	0.333333	0	1.000000
76902	2020-06-10	KT	50054	B	0	2	0	0	1	0	0	0.500000	0	0.500000

원래 데이터 : 같은 시점의 X와 Y

모델링 방식 2
➔ 최종 방식

1. 데이터 프레임 변형

- (T-1)시점의 독립변수로 T시점의 종속변수를 예측하자! 데이터셋의 X값이 (T-1)시점의 값이면 Y값은 (T-1) 시점의 Y 대신 T시점의 Y

	GDAY_DS	T_ID	P_ID	TB_SC	PA-AB	RUN	RBI	SH+SF	KK	SB_trial	BABIP	AB	HIT
0	2020-06-03	KT	50054.0	B	0.0	0.0	0.0	0.0	0.0	0.0	0.666667	3	0
1	2020-06-04	KT	50054.0	B	1.0	1.0	0.0	0.0	1.0	0.0	0.000000	0	0
2	2020-06-06	KT	50054.0	T	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	3	1
3	2020-06-07	KT	50054.0	T	0.0	0.0	0.0	0.0	2.0	0.0	1.000000	2	1
4	2020-06-10	KT	50054.0	B	0.0	0.0	0.0	0.0	0.0	0.0	0.500000	4	3

변경 데이터: **(T-1)시점의 X와 T시점의 Y**

모델링 방식 2
➔ 최종 방식

2-1 Y값 예측 : 모델 선정

- 모델링 후보

1 : Linear regression
2 : Regression Tree
3 : Ensemble method

최종: 후보3 - Ensemble Method

후보1 Linear regression

아무리 bias를 확장한다고 해도 자책점, 안타에 대한 회귀선을 찾기가 어렵다
가용 변수들과 종속 변수들 간의 선형 상관관계가 높지 않다
한계가 명확하기 때문에 활용하지 않기로 결정

후보2 Regression Tree

변수들을 기준으로 이진 결정 경계를 나누는 방식
세분화 기준: 분류로 인해 각 node 안의 이질성(Entropy)가 얼마나 변화하였는가
해석력이 높고 직관적 이해가 가능하다는 장점이 있지만, 훈련 데이터에
overfitting 되는 경향이 있다. 따라서 활용하지 않기로 결정

모델링 방식 2
➔ 최종 방식

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

2-1 Y값 예측 : 모델선정

Ensemble method

1) Bagging : Bootstrap Aggregating의 준말. M개의 Booststrap Sample에서의 결정 Tree 결과를 혼합

2) Random Forest : Bootstrap Sample의 특성상 Bagging 하위 모델들은 공분산이 크다는 점 보완

3) Boosting : 모델을 여러 번 훈련하는데 이전에 오분류된 데이터에 더 큰 가중치 부여!

앙상블 모델 중 Random Forest, Light Gradient Boosting Method (LGBM) , XGBoost 최종 선정

모델링 방식 2
➔ 최종 방식

2-2 Y값 예측 : Grid search cross validation

- 세 모델 모두 leaf의 개수, learning rate, 각 leaf 안의 데이터 수 등의 hyperparameter가 중요한 역할!

이 hyperparameter의 값에 따라 모델링의 결과가 달라진다.

- 최적의 parameter를 선정하기 위해 다수의 parameter의 grid를 구성하고 최적의 조합 찾기!
- MSE를 기준으로 해서 값이 가장 작은 모델을 최종 모델로 선정

모델링 방식 2
➔ 최종 방식

2-2 Y값 예측 : Grid search cross validation

투수데이터 MSE비교

	INN2	ER
LGBM	12.506	2.021
Xgboost	12.528	2.037
Random Forest	12.708	2.019

타자데이터 MSE비교

	AB	HIT
LGBM	2.04	0.792
Xgboost	2.04	0.791
Random Forest	2.04	0.792

모델링 방식 2
➔ 최종 방식

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

2-3 Y값 예측 : 최종 모델 선정

- 투수 모델, 타자 모델 모두 예측할 대상이 2개! 두 MSE 값의 산술 평균이 가장 작은 모델로 선택
- **투수 LGBM 타자 Xgboost**

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

모델링

승률

승률 예측 방식

"피타고리안 승률"

$$\text{기대승률} = \frac{(\textbf{득점})^2}{(\textbf{득점})^2+(\textbf{실점})^2}$$

야구의 승률을 추정하기 위해 James(1982)는 승률은 득점의 제곱을 득점의 제곱과 실정의 제곱의 합으로 나누어 추정할 수 있음을 제안하였고, 이를 야구 경기의 피타고라스 정리라고 불렀다.

2020년 팀별 투수 타자 데이터를 통해 득점과 실점의 합을 구하였고, 피타고리안 공식에 대입을 하여 승률을 예측하였다.

```
run = batter_T['RUN'].groupby(batter_T['T_ID']).sum()
R = pitcher_T['R'].groupby(pitcher_T['T_ID']).sum()
WR = (run**2)/((run**2)+(R**2))
```

최종 예측 값 보고

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

예측 값 보고

대회설명

EDA 및 변수선택

모델링방식

타율 방어율 승률 예측

팀 명	방어율	타율	승률
HH (한화)	4.965384	0.273720	0.294683
HT (KIA)	5.172462	0.279354	0.583043
KT	4.609663	0.270775	0.539710
LG	4.802338	0.277884	0.583521
LT (롯데)	4.692812	0.276886	0.521620
NC	4.759503	0.282182	0.671297
OB (두산)	4.910974	0.271753	0.609756
SK	4.893906	0.272903	0.402553
SS (삼성)	4.861336	0.276818	0.602566
WO (키움)	4.830148	0.274706	0.590687

출처

- 지수평활법 :
https://otexts.com/fpp2/ses.html

- 4장 Regression Tree 그림 :
https://allmodelsarewrong.github.io/trees.html

- 6장 논문 :
Kim, S. K., & Lee, Y. H. (2016). The estimation of winning rate in Korean professional baseball league. *Journal of the Korean Data and Information Science Society*, *27*(3), 653-661.