인적사항

- 성명 : 이성범
- 이메일 : 2712qwer@naver.com

KBO 팀별 경기 성적 예측

한신대학교_이성범

목차

1 분석 개요

- 주어진 데이터와 수집된 데이터의 정보의 차이가 존재
- 연도별 KBO 리그의 특성 차이가 존재
- 타율, 방어율, 승률에 따라 중요한 변수의 차이가 존재
- 예측된 결과는 리그 최종 성적목표
- 우리가 원하는 결과는 9월 28일 이후의 KBO리그의 팀별 성적

1 분석 개요

따라서 데이터 세트와 목표 변수에 따라 총 3종류의 모델을 구축하고

모델 간의 성능을 비교 후 최적의 모델을 선택

2020년 데이터를 바탕으로 모델을 구축하고

예측된 결과에 공식을 적용하여

9월 28일 이후의 KBO리그의 팀별 성적을 최종적으로 예측

2 데이터 수집

- 주어진 데이터는 2016년 ~ 2019년의 각 팀 또는 선수들의 일일 경기 성적
- 따라서 누적과 일일 데이터를 각각 구할 수 있다.

	게임키	일자	팀코드	상대팀코드	더블헤더코드	초말	선수코드	선발	타순	타자	타수	타점	득점	안타	2루타	3루타	홈런	도루	도루실패	희타	희비	4구	고4	사구	삼진	병살타	실책	잔루	득점권타율	득점권타수	득점권안타
0	20160401HHLG0	20160401	HH	LG	0	T	60404	0	3	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.0	0	0
1	20160401HHLG0	20160401	LG	HH	0	B	61102	1	8	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0.0	0	0
2	20160401HHLG0	20160401	LG	HH	0	B	61186	1	7	4	4	2	2	3	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1.0	1	1
3	20160401HHLG0	20160401	LG	HH	0	B	62164	0	9	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0.0	0	0
4	20160401HHLG0	20160401	HH	LG	0	T	62700	1	9	2	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0.0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18679	20161009WOLT0	20161009	WO	LT	0	T	78168	1	1	4	3	0	1	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1.0	1	1
18680	20161009WOLT0	20161009	LT	WO	0	B	78513	1	1	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0.0	0	0
18681	20161009WOLT0	20161009	WO	LT	0	T	79130	0	5	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.0	1	0
18682	20161009WOLT0	20161009	WO	LT	0	T	79300	1	7	4	4	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.0	0	0
18683	20161009WOLT0	20161009	WO	LT	0	T	79365	1	8	3	3	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0	1	0

18684 rows × 31 columns

선수 데이터

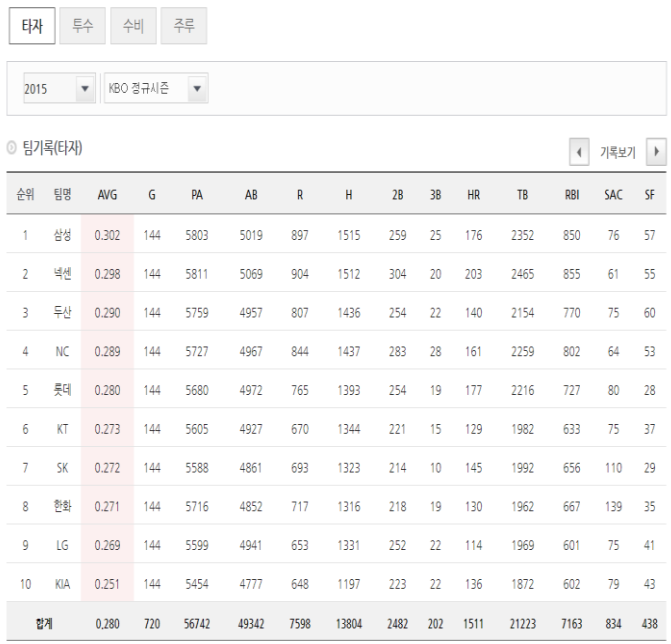
	게임키	일자	팀코드	상대팀코드	더블헤더코드	초말	타자	타수	타점	득점	안타	2루타	3루타	홈런	도루	도루실패	희타	희비	4구	고4	사구	삼진	병살타	실책	잔루	득점권타율	득점권타수	득점권안타
0	20160401HHLG0	20160401	LG	HH	0	B	47	42	4	5	9	2	0	1	2	1	1	0	4	0	0	11	0	0	8	0.333333	12	4
1	20160401HHLG0	20160401	HH	LG	0	T	52	46	4	4	13	2	0	0	0	0	3	0	3	0	0	10	1	2	12	0.200000	15	3
2	20160401HTNC0	20160401	NC	HT	0	B	36	30	5	5	9	2	0	2	0	0	1	0	5	0	0	9	1	1	7	0.142857	7	1
3	20160401HTNC0	20160401	HT	NC	0	T	38	34	3	4	8	3	0	1	0	0	0	0	3	0	1	10	1	0	7	0.100000	10	1
4	20160401KTSK0	20160401	SK	KT	0	B	36	36	4	4	10	5	0	1	0	0	0	0	0	0	0	7	1	1	5	0.375000	8	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1435	20161008SSSK0	20161008	SS	SK	0	T	39	36	5	6	10	2	0	2	1	0	0	0	3	0	0	4	1	2	6	0.250000	8	2
1436	20161009KTNC0	20161009	NC	KT	0	B	41	39	3	4	12	4	0	0	4	1	0	0	1	0	1	13	0	0	10	0.111111	18	2
1437	20161009KTNC0	20161009	KT	NC	0	T	43	38	5	7	14	3	0	0	1	0	1	1	3	0	0	7	0	0	9	0.100000	10	1
1438	20161009WOLT0	20161009	LT	WO	0	B	38	36	8	8	11	2	0	2	1	0	0	0	1	0	1	4	2	0	6	0.416667	12	5
1439	20161009WOLT0	20161009	WO	LT	0	T	38	33	5	5	10	0	0	0	0	1	0	2	3	0	0	6	1	3	6	0.428571	7	3

1440 rows × 28 columns

팀 데이터

2 데이터 수집

- 2001년 ~ 2015년의 각 팀의 해당년도 누적 경기 성적을 수집
- 주어진 데이터와 달리 수집한 데이터는 일일 데이터를 알 수 없다.



타자 | 투수 | 수비 | 주루

2015 | KBO 정규시즌

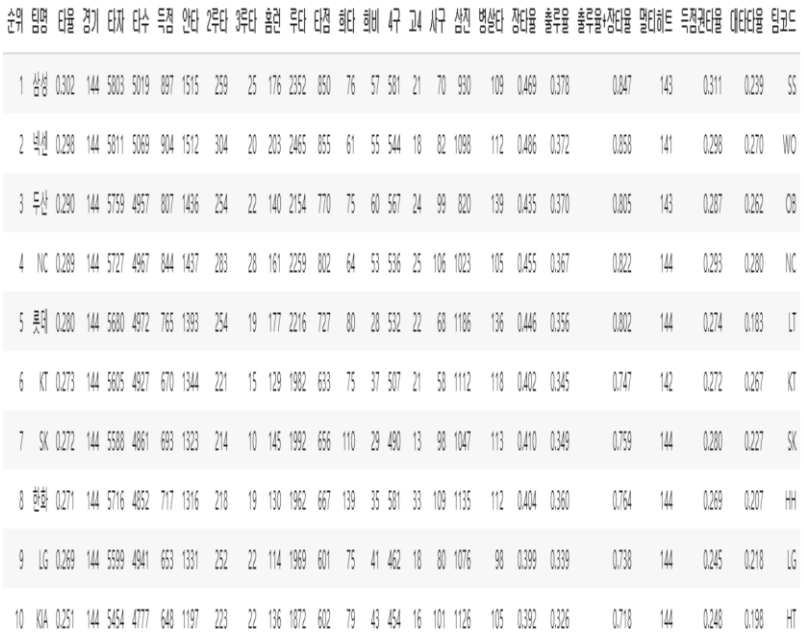
팀기록(타자) 기록보기

순위	팀명	AVG	G	PA	AB	R	H	2B	3B	HR	TB	RBI	SAC	SF
1	삼성	0.302	144	5803	5019	897	1515	259	25	176	2352	850	76	57
2	넥센	0.298	144	5811	5069	904	1512	304	20	203	2465	855	61	55
3	두산	0.290	144	5759	4957	807	1436	254	22	140	2154	770	75	60
4	NC	0.289	144	5727	4967	844	1437	283	28	161	2259	802	64	53
5	롯데	0.280	144	5680	4972	765	1393	254	19	177	2216	727	80	28
6	KT	0.273	144	5605	4927	670	1344	221	15	129	1982	633	75	37
7	SK	0.272	144	5588	4861	693	1323	214	10	145	1992	656	110	29
8	한화	0.271	144	5716	4852	717	1316	218	19	130	1962	667	139	35
9	LG	0.269	144	5599	4941	653	1331	252	22	114	1969	601	75	41
10	KIA	0.251	144	5454	4777	648	1197	223	22	136	1872	602	79	43
합계		0.280	720	56742	49342	7598	13804	2482	202	1511	21223	7163	834	438

KBO 홈페이지 자료
(https://www.koreabaseball.com/)

```python
from selenium import webdriver
import pandas as pd
from bs4 import BeautifulSoup as bs
import time

path = "C:/chromedriver" #크롬 드라이버 위치
driver = webdriver.Chrome(path) #크롬 드라이버 키기

button_H = "https://www.koreabaseball.com/Record/Team/Hitter/Basic1.aspx" # 타자 선택
button_P = 'https://www.koreabaseball.com/Record/Team/Pitcher/Basic1.aspx' # 투수 선택

buttons = [button_H, button_P]

for button in buttons:
    driver.get(button) #크롬 드라이버 KBO 사이트 접속
    time.sleep(3)

    for i in range(1,16):

        if (i < 10) & (button == button_H):
            file_name = "팀타자_200"+str(i)+".csv"
        if (i >= 10) & (button == button_H):
            file_name = "팀타자_20" + str(i)+".csv"

        if (i < 10) & (button == button_P):
            file_name = "팀투수_200"+str(i)+".csv"
        if (i >= 10) & (button == button_P):
            file_name = "팀투수_20" + str(i)+".csv"

        driver.find_element_by_xpath("""//*[@id="cphContents_cphContents_cphContents_ddlSe
```

Crawling



순위	팀명	타율	경기	타자	타수	득점	안타	2루타	3루타	홈런	루타	타점	희타	희비	4구	고4	사구	삼진	병살타	장타율	출루율	출루율+장타율	멀티히트	득점권타율	대타타율	팀코드
1	삼성	0.302	144	5803	5019	897	1515	259	25	176	2352	850	76	57	581	21	70	930	109	0.469	0.378	0.847	143	0.311	0.259	SS
2	넥센	0.298	144	5811	5069	904	1512	304	20	203	2465	855	61	55	544	18	82	1098	112	0.486	0.372	0.858	141	0.298	0.270	WO
3	두산	0.290	144	5759	4957	807	1436	254	22	140	2154	770	75	60	567	24	99	820	139	0.455	0.370	0.805	143	0.287	0.262	OB
4	NC	0.289	144	5727	4967	844	1437	283	28	161	2259	802	64	53	536	25	106	1023	105	0.455	0.367	0.822	144	0.293	0.280	NC
5	롯데	0.280	144	5680	4972	765	1393	254	19	177	2216	727	80	28	532	22	68	1186	136	0.446	0.356	0.802	144	0.274	0.183	LT
6	KT	0.273	144	5605	4927	670	1344	221	15	129	1982	633	75	37	507	21	58	1112	118	0.402	0.345	0.747	142	0.272	0.267	KT
7	SK	0.272	144	5588	4861	693	1323	214	10	145	1992	656	110	29	490	13	98	1047	113	0.410	0.349	0.759	144	0.280	0.227	SK
8	한화	0.271	144	5716	4852	717	1316	218	19	130	1962	667	139	35	581	33	109	1135	112	0.404	0.360	0.764	144	0.269	0.207	HH
9	LG	0.269	144	5599	4941	653	1331	252	22	114	1969	601	75	41	462	18	80	1076	98	0.398	0.339	0.738	144	0.245	0.218	LG
10	KIA	0.251	144	5454	4777	648	1197	223	22	136	1872	602	79	43	454	16	101	1126	105	0.392	0.326	0.718	144	0.248	0.198	HT

수집 데이터

2 데이터 수집

따라서 2001년 ~ 2019년 까지의 팀별 경기 성적 누적 데이터,

2016년 ~ 2019년 까지의 팀별 일일 경기 성적 데이터를

최종적으로 사용할 수 있다.

3 데이터 전처리

- 현재 주어진 지표로는 모델을 구축하기 부족하여 조금 더 다양한 지표를 구함
- 야구를 통계학적으로 분석하는 방법론인 Sabermetrics에서 사용하는 지표를 구함
- 타자의 지표를 투수 입장의 지표로도 구함
- 각 지표는 팀별 성적으로 구함
- 누적 값으로 사용되는 지표는 리그경기 수로 나눠 평균 값으로 대체

3 데이터 전처리

주어진 지표

출루율, 장타율, 득점권 타율, 평균 삼진,

평균 타자, 평균 타수, 평균 도루실패,

평균 도루성공, 평균 홈런, 평균 4구,

평균 안타, 평균 득점 등

Sabermetrics 지표

타자 평가 지표

OPS, GPA, RC, RawEqA, wOBA, BABIP

투수 평가 지표

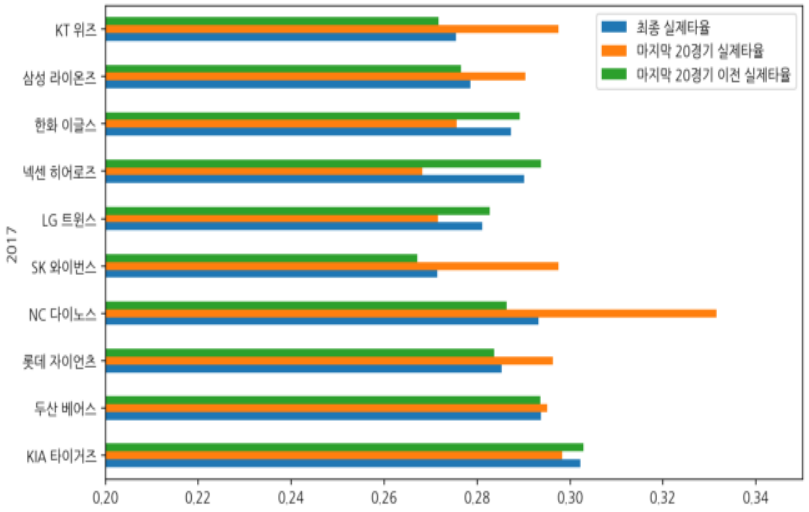
DIPS, ERC, WHIP

3 데이터 전처리

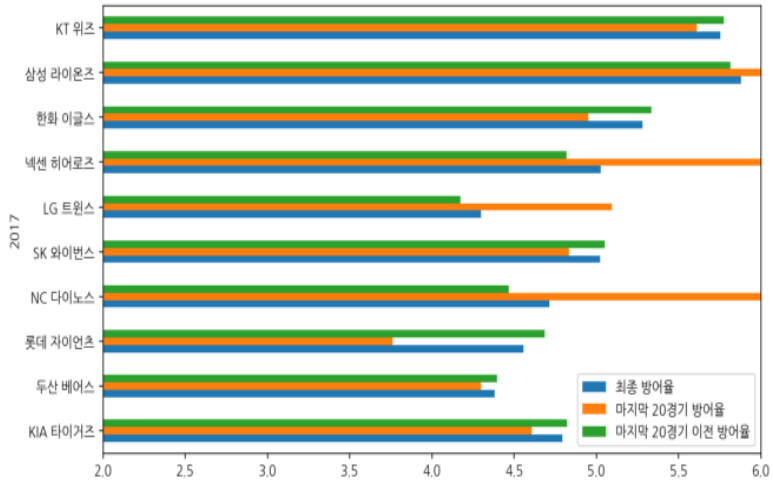
- KBO 홈페이지 기준 9월 28일 이후의 경기 수는 약 20경기
- 따라서 총 3가지 종류의 데이터 세트를 구함
- 2016년 ~ 2019년 리그 마지막 20경기 이전 성적 데이터
- 2016년 ~ 2019년 리그 전체 경기 성적 데이터
- 2001년 ~ 2019년 리그 전체 경기 성적 데이터
- 비교를 위해 2016년 ~ 2019년의 마지막 20경기 성적 데이터도 구함
- 데이터의 목표 변수는 리그 최종 타율, 방어율, 승률

4 EDA

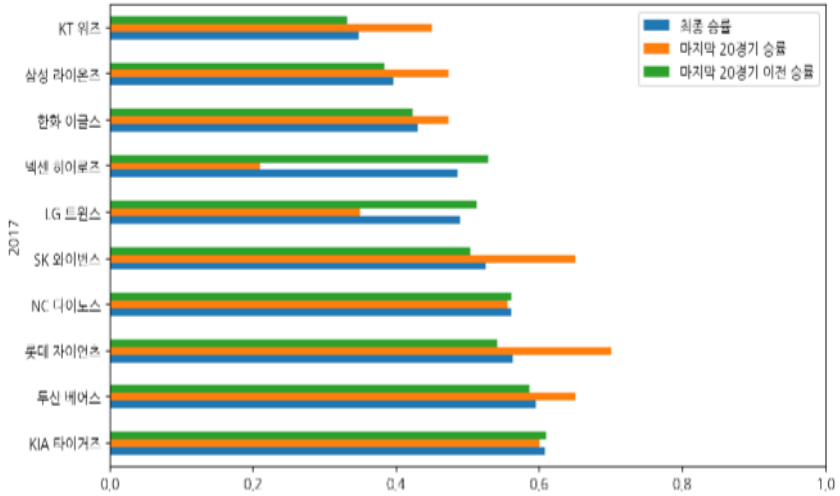
- 최종 성적과 마지막 20경기 이전 성적을 비교한 결과 두 값은 서로 유사함
- 따라서 현재 까지의 성적으로 최종 성적을 정확하게 예측하는 것이 중요함



타율



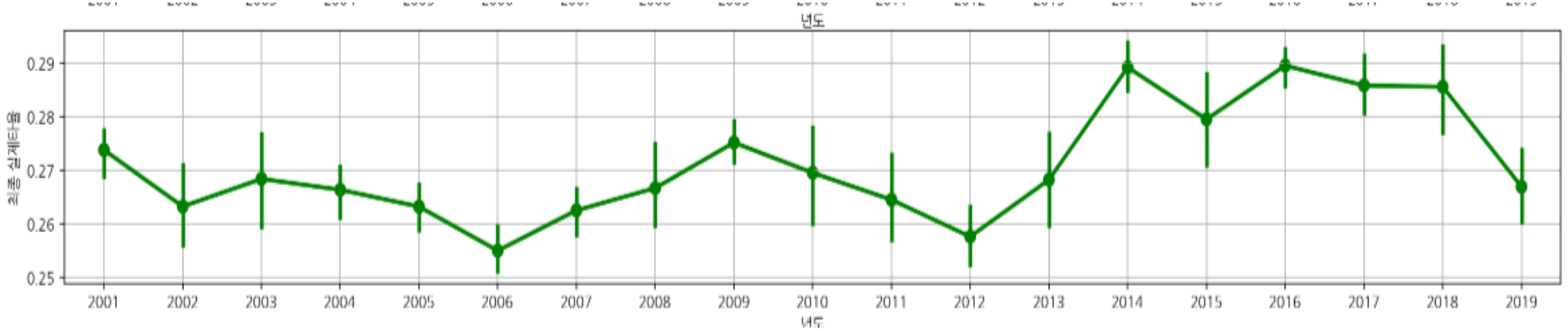
방어율



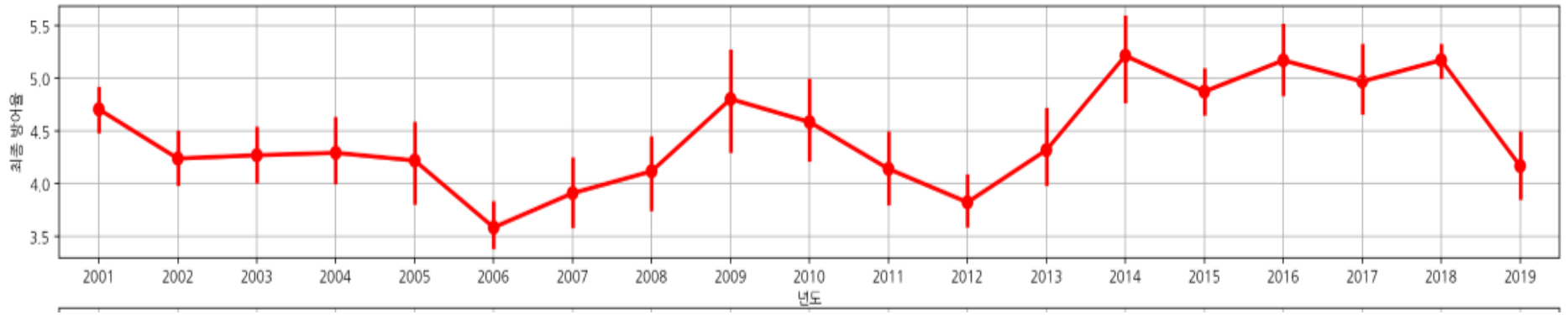
승률

4 EDA

- 각 연도마다 지표의 값의 차이가 존재
- 해당 년도의 지표들의 값 차이는 지표들 모두가 유사함
- 따라서 각 연도마다 리그의 성향 차이가 존재함 (ex 타고투저, 투고타저 등)



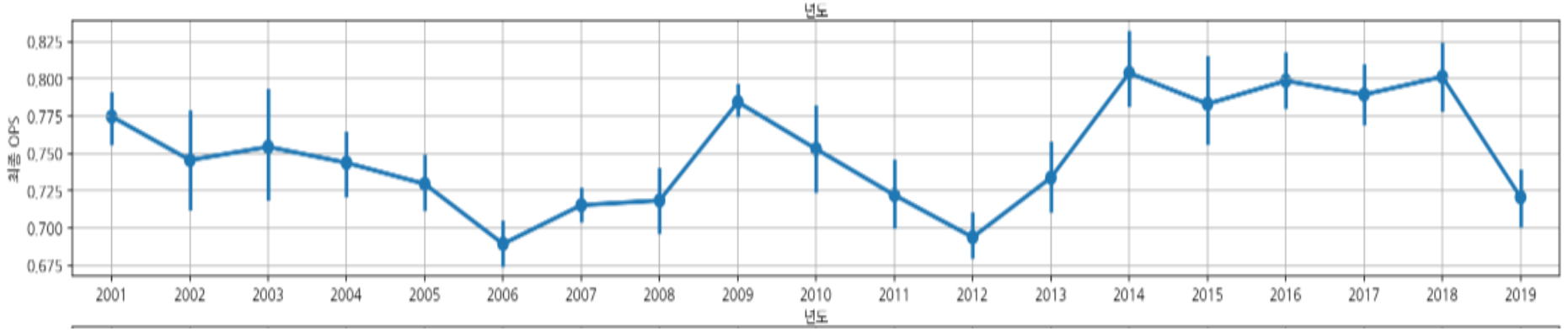
타율



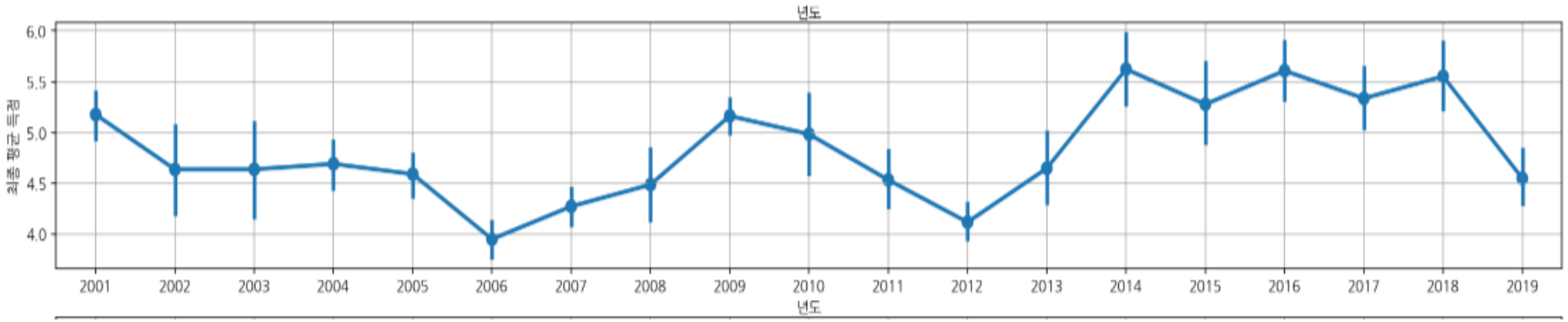
방어율

4 EDA

- 각 연도마다 지표의 값의 차이가 존재
- 해당 년도의 지표들의 값 차이는 지표들 모두가 유사함
- 따라서 각 연도마다 리그의 성향 차이가 존재함 (ex 타고투저, 투고타저 등)



OPS



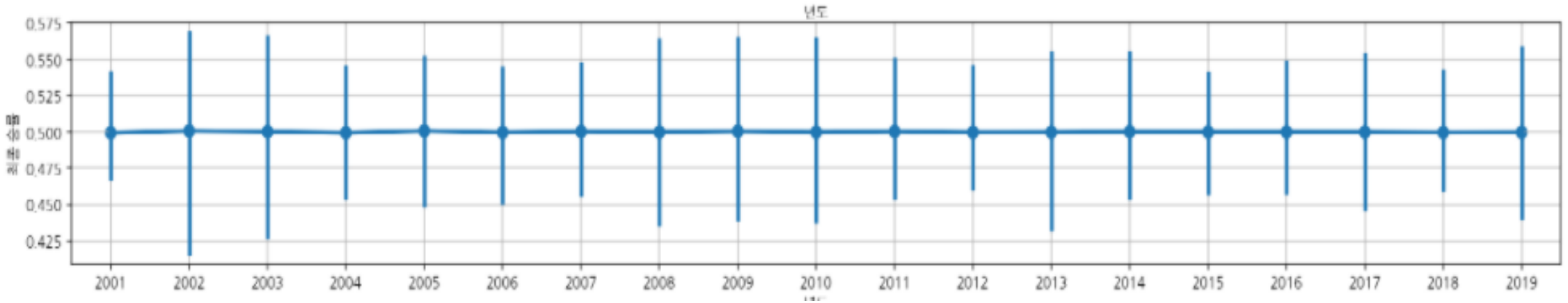
평균 득점

4 EDA

따라서 지표 값의 크기 차이는

타율과 방어율 예측에는 영향을 주지 않으나

승률 예측에는 영향을 준다.



승률

4 EDA

- 검증 데이터셋을 구하고자 평균 득점을 기준으로 유사한 성향의 리그를 구함
- 마지막 20경기 성적의 유무 + 유사한 리그의 수 + 2020년 데이터와의 유사성을 고려
- 최종적으로 2017년 데이터를 검증 데이터로 선택

```python
# 95% 신뢰구간을 구하는 함수
import numpy as np
import scipy.stats
def mean_confidence_interval(data, confidence=0.95):
    a = 1.0 * np.array(data)
    n = len(a)
    m, se = np.mean(a), scipy.stats.sem(a)
    h = se * scipy.stats.t.ppf((1 + confidence) / 2., n-1)
    return m, m-h, m+h
```

유사 성향의 리그 판단은
95% 신뢰구간을 이용

4 EDA

- 2019년 리그
  - 2019년 리그의 평균 득점 95% 신뢰 구간를 기준으로 비슷한 성향의 리그 :
    - [2007, 2008, 2011, 2005, 2002, 2003, 2013, 2004] - 8개
  - 2019년 리그의 평균 실점 95% 신뢰 구간를 기준으로 비슷한 성향의 리그 :
    - [2007, 2008, 2011, 2005, 2002, 2003, 2013, 2004] - 8개
- 2018년 리그
  - 2018년 리그의 평균 득점 95% 신뢰 구간를 기준으로 비슷한 성향의 리그 :
    - [2009, 2001, 2015, 2017, 2016, 2014] - 6개
  - 2018년 리그의 평균 실점 95% 신뢰 구간를 기준으로 비슷한 성향의 리그 :
    - [2016, 2014] - 2개
- 2017년 리그
  - 2017년 리그의 평균 득점 95% 신뢰 구간를 기준으로 비슷한 성향의 리그 :
    - [2010, 2009, 2001, 2015, 2018, 2016, 2014] - 7개
  - 2017년 리그의 평균 실점 95% 신뢰 구간를 기준으로 비슷한 성향의 리그 :
    - [2010, 2009, 2001, 2015, 2018, 2016, 2014] - 7개
- 2016년 리그
  - 2016년 리그의 평균 득점 95% 신뢰 구간를 기준으로 비슷한 성향의 리그 :
    - [2015, 2017, 2018, 2014] - 4개
  - 2016년 리그의 평균 실점 95% 신뢰 구간를 기준으로 비슷한 성향의 리그 :
    - [2015, 2017, 2018, 2014] - 4개

유사한 리그의 수

년도	최종 방어율	최종 실제타율	최종 OPS	최종 평균 득점	최종 평균 실점
2001	4.7065150	0.2738149	0.7745485	5.1757519	5.1757519
2002	4.2389029	0.2632807	0.7454064	4.6353383	4.6353383
2003	4.2698541	0.2684325	0.7542397	4.6362782	4.6362782
2004	4.2928188	0.2663959	0.7436038	4.6898496	4.6898496
2005	4.2195920	0.2632569	0.7294032	4.5892857	4.5892857
2006	3.5839998	0.2550425	0.6891859	3.9494048	3.9494048
2007	3.9095592	0.2625572	0.7151850	4.2718254	4.2718254
2008	4.1177325	0.2667178	0.7183113	4.4861111	4.4861111
2009	4.8037617	0.2751953	0.7842733	5.1616541	5.1616541
2010	4.5860489	0.2695553	0.7528063	4.9821429	4.9821429
2011	4.1403926	0.2645358	0.7219377	4.5319549	4.5319549
2012	3.8226949	0.2576511	0.6936426	4.1165414	4.1165414
2013	4.3190814	0.2682721	0.7336107	4.6467014	4.6467014
2014	5.2152647	0.2892349	0.8037463	5.6223958	5.6223958
2015	4.8743074	0.2795438	0.7830240	5.2763889	5.2763889
2016	5.1712240	0.2895544	0.7985421	5.6069444	5.6069444
2017	4.9710338	0.2858524	0.7892422	5.3347222	5.3347222
2018	5.1730565	0.2856011	0.8012478	5.5513889	5.5513889
2019	4.1679978	0.2669754	0.7205315	4.5472222	4.5472222

연도별 평균성적

```
방어율       4.8175847
실제타율     0.2738823
OPS          0.7605374
평균 득점    5.2117115
평균 실점    5.2136074
dtype: float64
```

2020년 평균성적
(9월 5일 기준)

4 EDA

- 검증 데이터는 2017년 데이터
- 총 3가지 유형의 데이터셋을 학습용 데이터에 이용
  - 2016, 2018년 마지막 20경기 이전 성적 데이터
  - 2016, 2018년 리그 전체 성적 데이터
  - 2001, 2009, 2010, 2014, 2015, 2016, 2018년 리그 전체 성적 데이터
- 리그 전체 성적 데이터의 경우 연도에 상관 없이 사용할 독립변수가 같다고 가정
- 따라서 마지막 20경기 이전 성적 데이터, 리그 전체 성적 데이터에 최적의 독립변수를 찾음

4 EDA

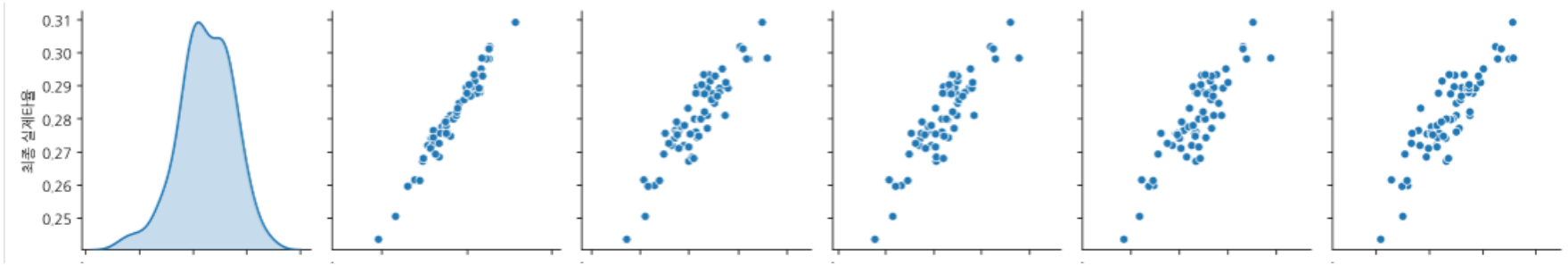
- 최적의 독립변수를 찾고자 1차적으로 수가 많고 특정 상관계수를 넘는 변수를 선택

- 타율 (타자 관련 지표와 상관성이 높음)
  - 리그 전체 성적 데이터는 상관계수 0.8 이상
  - 리그 마지막 20경기 이전 성적 데이터는 상관계수 0.75 이상

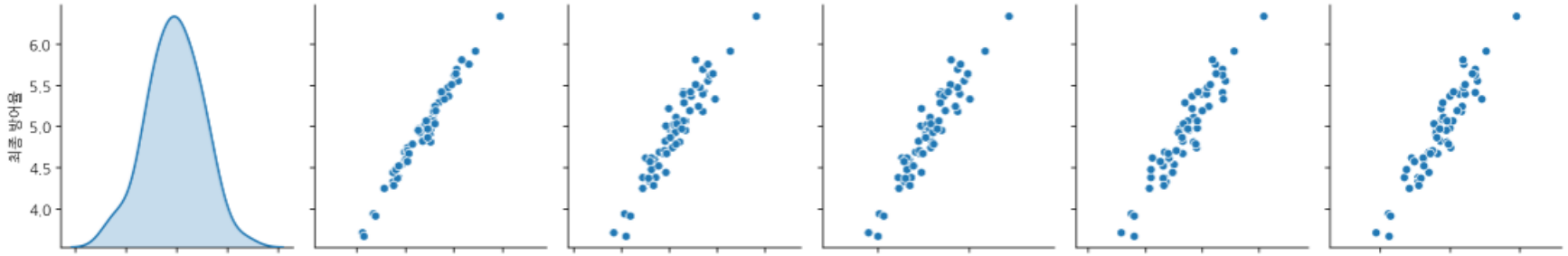
- 방어율 (투수 관련 지표와 상관성이 높음)
  - 리그 전체 성적 데이터는 상관계수 0.8 이상
  - 리그 마지막 20경기 이전 성적 데이터는 상관계수 0.8 이상

- 승률 (두 지표 모두 상관성이 높음)
  - 리그 전체 성적 데이터는 상관계수 0.6 이상
  - 리그 마지막 20경기 이전 성적 데이터는 상관계수 0.7 이상

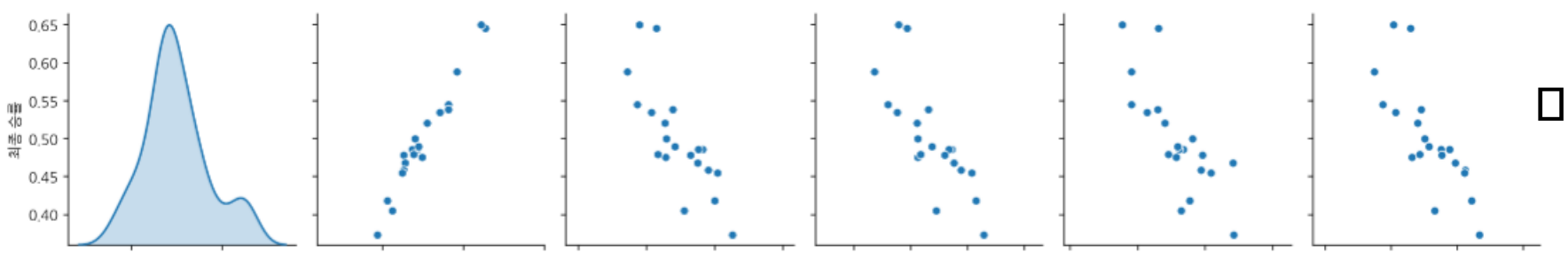
4 EDA



타율
리그 전체 성적
상관계수 상위 5개



방어율
리그 전체 성적
상관계수 상위 5개



승률
마지막 20경기 이전 성적
상관계수 상위 5개

4 EDA

- 최적의 독립변수를 찾고자 2차적으로 PCA와 회귀분석을 활용
- PCA를 활용하여 변수들 간의 다중공선성을 판단

- 상관계수가 작은 변수를 우선으로 VIF 값이 10이 넘어가면 제거
- 위 과정을 반복하여 VIF 값이 최대한 10보다 작아질 때 까지 반복

- 회귀분석의 결정계수 값과 AIC 값을 활용하여 변수를 판단

4 EDA

```python
from patsy import dmatrices
import statsmodels.api as sm # 다중회귀분석 모델
from statsmodels.stats.outliers_influence import variance_inflation_factor # VIF 계산 모델
a = pd.DataFrame()

columns = ['최종 실제타율', '최종 평균 안타','최종 GPA', '최종 BABIP', '최종 평균 타수']

columns_set = ['A', 'B', 'C', 'D', 'E']

b = 0
for i in columns:
  a[columns_set[b]] = Score_League_df_2017[columns[b]].values
  b = b + 1

columns = ['Intercept', '최종 평균 안타', '최종 GPA', '최종 BABIP', '최종 평균 타수']

columns_set = ['B', 'C', 'D', 'E']

features = '+'.join(columns_set)
features = 'A ~'+features

y, X = dmatrices(features, data = a, return_type='dataframe')

result = sm.OLS(y, X).fit()

X.columns = columns
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      A   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 2.095e+04
Date:                Sun, 06 Sep 2020   Prob (F-statistic):           6.82e-91
Time:                        12:40:27   Log-Likelihood:                 416.38
No. Observations:                  63   AIC:                            -822.8
Df Residuals:                      58   BIC:                            -812.0
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.2755      0.005     57.081      0.000       0.266       0.285
최종 평균 안타       0.0290      0.000     85.092      0.000       0.028       0.030
최종 GPA         0.0058      0.009      0.682      0.498      -0.011       0.023
최종 BABIP      -0.0002      0.006     -0.036      0.971      -0.013       0.012
최종 평균 타수       -0.0080      0.000    -47.360      0.000      -0.008      -0.008
==============================================================================
Omnibus:                       59.724   Durbin-Watson:                   2.091
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              313.838
Skew:                          -2.779   Prob(JB):                     7.10e-69
Kurtosis:                      12.416   Cond. No.                     8.09e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.09e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

타율
리그 전체 성적
회귀 분석

```python
from patsy import dmatrices
import statsmodels.api as sm # 다중회귀분석 모델
from statsmodels.stats.outliers_influence import variance_inflation_factor # VIF 계산 모델
a = pd.DataFrame()

columns = ['최종 실제타율', '최종 평균 안타','최종 GPA', '최종 BABIP', '최종 평균 타수']

columns_set = ['A', 'B', 'C', 'D', 'E']

b = 0
for i in columns:
  a[columns_set[b]] = Score_League_df_2017[columns[b]].values
  b = b + 1

columns = ['Intercept', '최종 평균 안타','최종 GPA', '최종 BABIP', '최종 평균 타수']

columns_set = ['B', 'C', 'D', 'E']

features = '+'.join(columns_set)
features = 'A ~'+features

y, X = dmatrices(features, data = a, return_type='dataframe')

vif = pd.DataFrame()
vif['Vif Factor'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
X.columns = columns
vif['features'] = X.columns

vif
```

	Vif Factor	features
0	12705.6401019	Intercept
1	22.6236507	최종 평균 안타
2	5.2799014	최종 GPA
3	4.3557894	최종 BABIP
4	7.9804703	최종 평균 타수

타율
리그 전체 성적
PCA 분석

4 EDA

- PCA와 회귀분석을 통해 구해진 변수에 개인적 판단을 넣어 최종적으로 변수 선택

- 타율
- 리그 전체 성적 데이터 - ['최종 평균 안타', '최종 GPA', '최종 BABIP', '최종 평균 타수']
- 리그 마지막 20경기 이전 성적 데이터 - ['마지막 20경기 이전 실제타율', '마지막 20경기 이전 평균 안타','마지막 20경기 이전 OPS', '마지막 20경기 이전 평균 득점', '마지막 20경기 이전 GPA']
- 방어율
- 리그 전체 성적 데이터 - ['최종 평균 자책점', '최종 평균 실점', '최종 피GPA', '최종 피OPS', '최종 피타율', '최종 DIPS']
- 리그 마지막 20경기 이전 성적 데이터 - ['마지막 20경기 이전 방어율', '마지막 20경기 이전 피RC/27']
- 승률
- 리그 전체 성적 데이터 - ['최종 피GPA', '최종 평균 실점','최종 평균 득점', '최종 피OPS', '최종 GPA', '최종 OPS']
- 리그 마지막 20경기 이전 성적 데이터 - ['마지막 20경기 이전 승률', '마지막 20경기 이전 피RC/27', '마지막 20경기 이전 평균 실점', '마지막 20경기 이전 RC/27', '마지막 20경기 이전 평균 득점']

5 모델 구축 및 검증

XGBRegressor

타율 오류: 0.0019195715729660634
방어율 오류: 0.08097397924777153
승률 오류: 0.027590018909832637

LGBRegressor

타율 오류: 0.003420942325017632 8
방어율 오류: 0.17313490841910 74
승률 오류: 0.035310191474658 03

RandomForestRegressor

타율 오류: 0.0022483225202338 74
방어율 오류: 0.07829373665545972
승률 오류: 0.032670941460158164

가장 오류가 적은 모델 XGBRegressor, RandomForestRegressor 중
Outlier에 취약하지 않고 과적합이 적고 일반화가 잘 된 모델로
RandomForestRegressor 으로 판단

최종적으로 RandomForestRegressor 모델 선택

5 모델 구축 및 검증

- 총 3가지 유형의 모델 구축

- 모델1

- 2001, 2009, 2010, 2014, 2015, 2016, 2018년 리그 전체 성적 데이터로 학습
- 목표 변수는 리그 최종 타율, 방어율, 승률

- 모델2

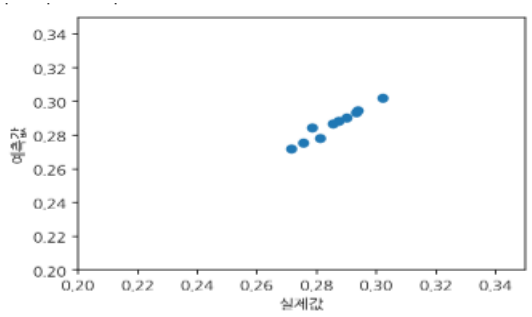
- 2016, 2018년 리그 전체 성적 데이터로 학습
- 목표 변수는 리그 최종 타율, 방어율, 승률

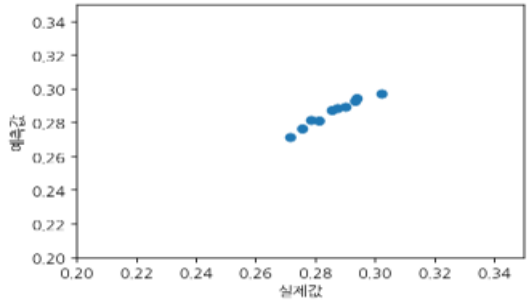
- 모델3

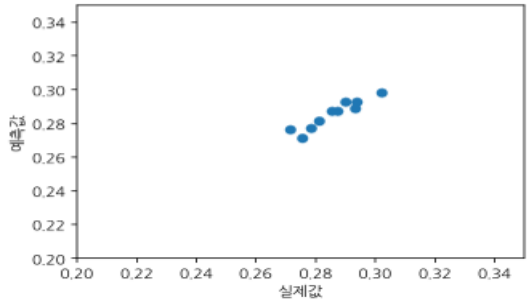
- 2016, 2018년 마지막 20경기 이전 성적 데이터로 학습
- 목표 변수는 리그 최종 타율, 방어율, 승률

5 모델 구축 및 검증

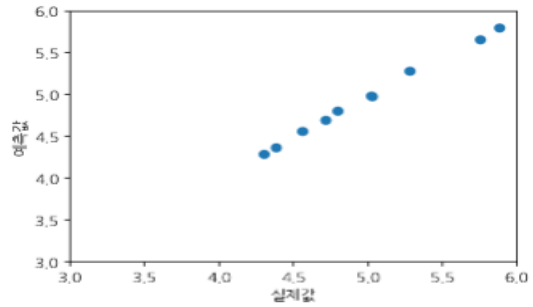
- 2017년 리그 최종 타율, 방어율, 승률과 예측된 리그 최종 타율, 방어율, 승률과 오류 비교 (그래프)

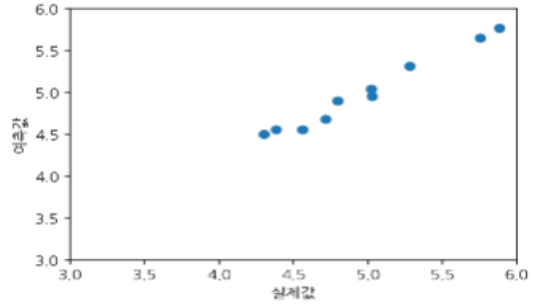


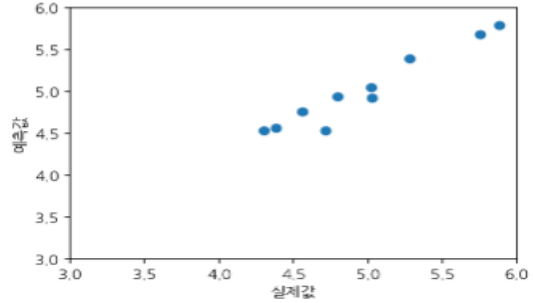




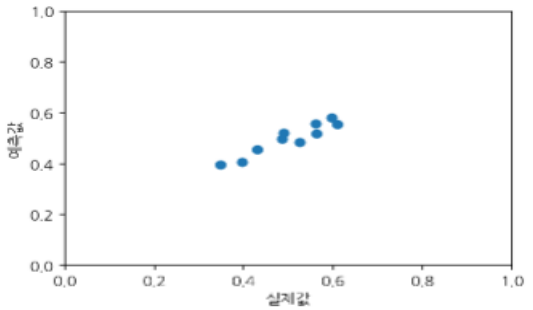
타율

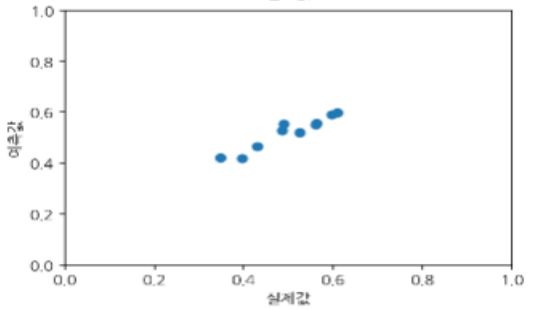


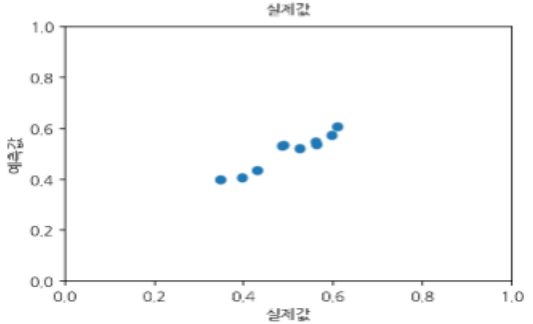




방어율







승률

모델1

모델2

모델3

5 모델 구축 및 검증

- 2017년 리그 최종 타율, 방어율, 승률과 예측된 리그 최종 타율, 방어율, 승률과 오류 비교 (RMSE)

1번 모델의 오류

타율 오류: 0.0022483225202338743

방어율 오류: 0.043478051957044293

승률 오류: 0.032670941460158164

2번 모델의 오류

타율 오류: 0.0021175176489442835

방어율 오류: 0.10670634218779229

승률 오류: 0.036659624853944364

3번 모델의 오류

타율 오류: 0.0029916845452766576

방어율 오류: 0.14748614006416366

승률 오류: 0.029419378186603014

5 모델 구축 및 검증

- 현재 구한 값은 리그 최종 타율, 방어율, 승률 값
- 우리가 구하고자 하는 값은 9월 28일 이후의 리그 성적 (KBO 홈페이지 기준 약 20경기)
- 리그 마지막 20경기 성적을 예측하고자 공식을 적용
- 최종적으로 리그 마지막 20경기 성적을 예측

5 모델 구축 및 검증

마지막 20경기 예측 공식

공식의 가정은 '야구는 통계적으로 평균에 근접해지는 성향을 보인다.

따라서 마지막 20경기는 최종 리그 성적에 도움이 되는 값에 근접해질 것이다' 이다.

X = 전체 리그의 성적 (모델로 예측 된 값)

Y = 마지막 20경기 이전의 성적

Z = 마지막 20경기의 성적 (우리가 구하고자 하는 값)

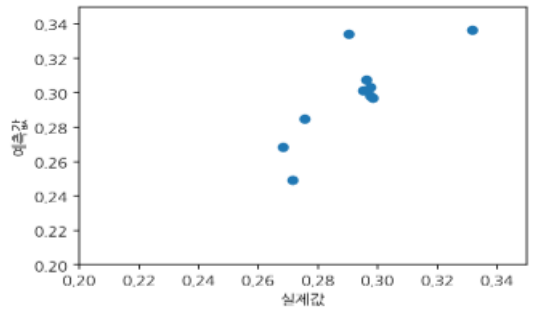
X * 전체 리그의 경기 수 = Y * (전체 리그의 경기 수 – 20) + Z * 20

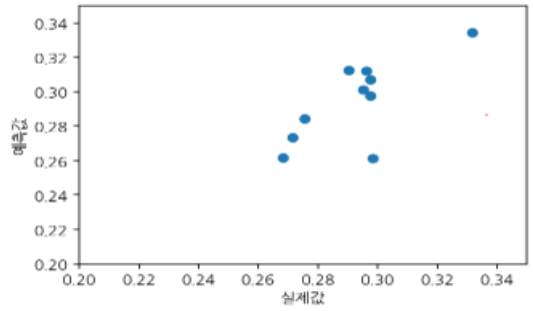
따라서 Z는

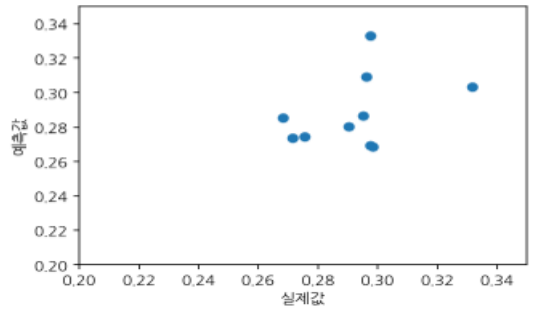
Z = (X * 전체 리그의 경기 수 - Y * (전체 리그의 경기 수 – 20)) / 20

5 모델 구축 및 검증

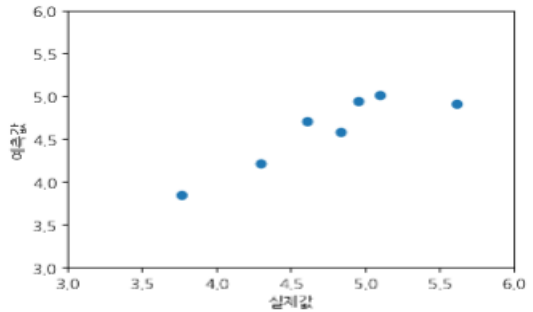
- 2017년 마지막 20경기 타율, 방어율, 승률과 예측된 마지막 20경기 타율, 방어율, 승률과 오류 비교

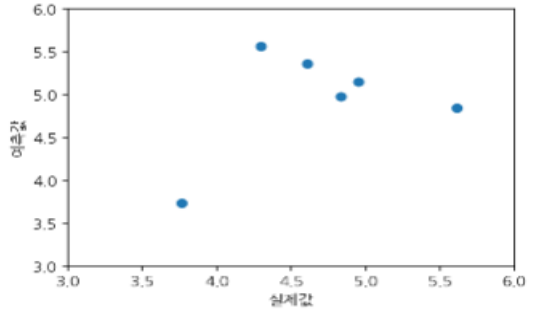


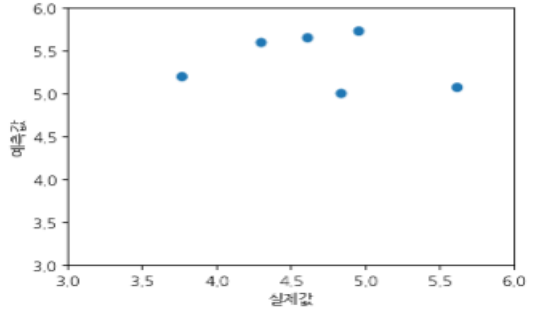




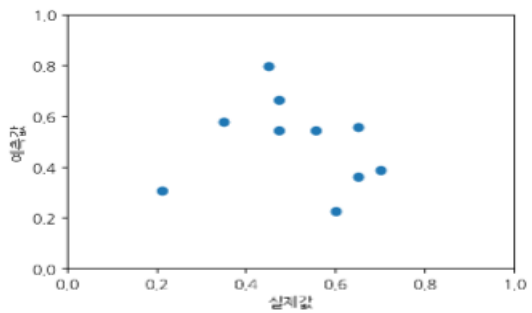
타율



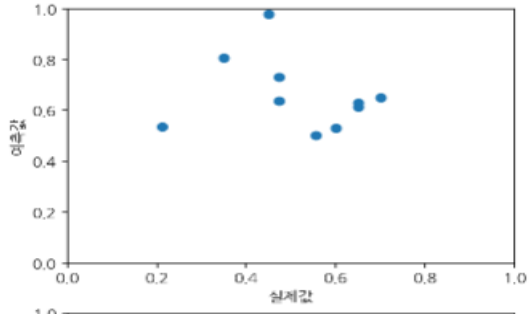




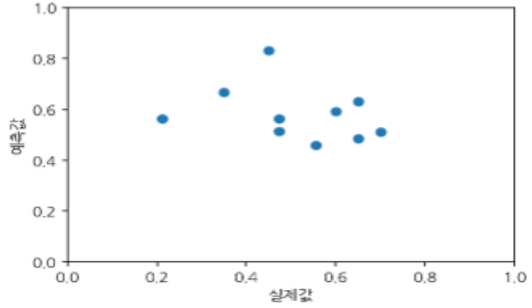
방어율



모델1



모델2



모델3

승률

5 모델 구축 및 검증

- 2017년 마지막 20경기 타율, 방어율, 승률과 예측된 마지막 20경기 타율, 방어율, 승률과 오류 비교

1번 모델의 오류

타율 오류: 0.016441942402535676

방어율 오류: 0.31194193676306636

승률 오류: 0.23467610186119603

2번 모델의 오류

타율 오류: 0.015404899593681205

방어율 오류: 0.7665786357457052

승률 오류: 0.26441202332040866

3번 모델의 오류

타율 오류: 0.020931691289736258

방어율 오류: 1.0648538688000182

승률 오류: 0.21277496507841136

5 모델 구축 및 검증

- 리그 최종 성적 예측의 경우 모델들의 정확성 차이가 크지 않음
- 마지막 20경기 성적 예측의 경우 모델들의 정확성 차이가 큼
- 타율, 방어율 예측에는 1번 모델이 다른 모델에 비해서 상대적으로 우수
- 승률 예측에는 3번 모델이 다른 모델에 비해서 상대적으로 우수

따라서

타율, 방어율 예측에는 1번 모델을

승률 예측에는 3번 모델을

최종적으로 선택

6 최종 결과

- 각 과정은 검증 방법과 동일하게 진행
- 2020년 데이터의 평균 득점을 기준으로 유사 성향의 리그 데이터를 구함
- 각 데이터 세트에는 2017년 데이터가 추가
- 상관계수, PCA, 회귀분석을 통해서 최적의 변수를 찾음
- 타율, 방어율 예측 모델은 모델1과 최적의 변수가 동일
- 승률 예측 모델은 모델3이 아닌 모델1과 최적의 변수가 동일 (표본의 증가에 따른 정확도 상승)
- 따라서 데이터가 많아진다면 모델 1의 변수가 모든 모델의 최적의 변수가 될 것임

6 최종 결과

- 목표 변수에 따른 모델의 사용 데이터 세트와 변수

- 타율 예측 모델
  - 2001, 2009, 2010, 2014, 2015, 2016, 2017, 2018년 리그 전체 성적 데이터로 학습
  - 사용 변수 : `['평균 안타', 'GPA', 'BABIP', '평균 타수']`
- 방어율 예측 모델
  - 2001, 2009, 2010, 2014, 2015, 2016, 2017, 2018년 리그 전체 성적 데이터로 학습
  - 사용 변수 : `['경기당 평균 자책점','평균 실점', '피GPA', '피OPS', '피타율', 'DIPS']`
- 승률 예측 모델
  - 2016, 2017, 2018년 마지막 20경기 이전 성적 데이터로 학습
  - 사용 변수 : `['마지막 20경기 이전 승률', '마지막 20경기 이전 GPA','마지막 20경기 이전 평균 실점', '마지막 20경기 이전 OPS', '마지막 20경기 이전 평균 득점', '마지막 20경기 이전 피OPS', '마지막 20경기 이전 피GPA']`

6 최종 결과

- Hyperparameter Tuning 과정

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score

n_estimators = 300

num_epoch = 100

coarse_hyperparameters_list = []

for epoch in range(num_epoch):
    max_depth = np.random.randint(low=1, high=50)
    max_features = np.random.uniform(low=0.5, high=0.9)

    model = RandomForestRegressor(n_estimators=n_estimators,
                                  max_depth=max_depth,
                                  max_features=max_features,
                                  n_jobs=-1,
                                  random_state=37)

    score = cross_val_score(model, X_train_E, Y_train_E, cv=20, scoring=rmse_score).mean() # X_train_E, Y_train_E 을 변경하여 각 모델마다의 최고의 파라미터를 찾아감

    hyperparameters = {
        'epoch': epoch,
        'score': score,
        'n_estimators': n_estimators,
        'max_depth': max_depth,
        'max_features': max_features,
    }

    coarse_hyperparameters_list.append(hyperparameters)
    print(f"{epoch:2} n_estimators = {n_estimators}, max_depth = {max_depth:2}, max_features = {max_features:.6f}, Score = {score:.5f}")

coarse_hyperparameters_list = pd.DataFrame.from_dict(coarse_hyperparameters_list)

coarse_hyperparameters_list = coarse_hyperparameters_list.sort_values(by="score")

print(coarse_hyperparameters_list.shape)

coarse_hyperparameters_list.head(10)
```

Coding

```
 0 n_estimators = 300, max_depth = 45, max_features = 0.537697, Score = 0.01911
 1 n_estimators = 300, max_depth = 44, max_features = 0.793732, Score = 0.01820
 2 n_estimators = 300, max_depth = 39, max_features = 0.856540, Score = 0.01820
 3 n_estimators = 300, max_depth = 22, max_features = 0.822532, Score = 0.01820
 4 n_estimators = 300, max_depth = 36, max_features = 0.880929, Score = 0.01809
 5 n_estimators = 300, max_depth = 46, max_features = 0.691174, Score = 0.01874
 6 n_estimators = 300, max_depth =  1, max_features = 0.585575, Score = 0.03687
 7 n_estimators = 300, max_depth = 40, max_features = 0.536897, Score = 0.01911
 8 n_estimators = 300, max_depth = 37, max_features = 0.747637, Score = 0.01820
 9 n_estimators = 300, max_depth = 32, max_features = 0.760016, Score = 0.01820
10 n_estimators = 300, max_depth = 16, max_features = 0.644152, Score = 0.01874
11 n_estimators = 300, max_depth = 10, max_features = 0.506547, Score = 0.01911
12 n_estimators = 300, max_depth =  9, max_features = 0.646299, Score = 0.01873
13 n_estimators = 300, max_depth = 21, max_features = 0.664134, Score = 0.01874
14 n_estimators = 300, max_depth = 20, max_features = 0.631248, Score = 0.01874
15 n_estimators = 300, max_depth = 11, max_features = 0.569254, Score = 0.01911
16 n_estimators = 300, max_depth = 26, max_features = 0.772434, Score = 0.01820
17 n_estimators = 300, max_depth = 19, max_features = 0.776087, Score = 0.01820
18 n_estimators = 300, max_depth = 16, max_features = 0.636020, Score = 0.01874
19 n_estimators = 300, max_depth = 40, max_features = 0.810229, Score = 0.01820
20 n_estimators = 300, max_depth = 12, max_features = 0.878334, Score = 0.01809
21 n_estimators = 300, max_depth =  7, max_features = 0.521890, Score = 0.01913
22 n_estimators = 300, max_depth = 25, max_features = 0.562648, Score = 0.01911
23 n_estimators = 300, max_depth = 46, max_features = 0.524676, Score = 0.01911
24 n_estimators = 300, max_depth = 31, max_features = 0.507254, Score = 0.01911
25 n_estimators = 300, max_depth = 21, max_features = 0.585052, Score = 0.01874
26 n_estimators = 300, max_depth = 18, max_features = 0.661160, Score = 0.01874
```

진행 과정

6 최종 결과

- Hyperparameter Tuning 진행 후 만들어진 모델의 최종 파라미터

- 타율 예측 모델

```
# 타율예측 모델 -> n_estimators를 제외하고 기본 옵션 사용
model_B = RandomForestRegressor(n_estimators=best_n_estimators,
                                random_state=37,
                                n_jobs=-1)
```

- 방어율 예측 모델

```
# 방어율예측 모델 -> n_estimators를 제외하고 기본 옵션 사용
model_E = RandomForestRegressor(n_estimators=best_n_estimators,
                                random_state=37,
                                n_jobs=-1)
```

- 승률 예측 모델

```
# 승률예측 모델 -> Hyperparameter Tuning 옵션 사용
model_W = RandomForestRegressor(n_estimators=best_n_estimators,
                                max_depth=best_max_depth_W,
                                max_features=best_max_features_W,
                                random_state=37,
                                n_jobs=-1)
```

```
best_n_estimators = 3000

best_max_depth_W = 78
best_max_features_W = 0.8836279
```

6 최종 결과

- 모델의 오류 체크를 위한 RMSE 값 (훈련용 데이터 내 예측 정확도)

```python
import numpy as np

from sklearn.metrics import make_scorer

# RMSE 공식
def rmse(predict, actual):
    predict = np.array(predict)
    actual = np.array(actual)

    distance = predict - actual

    square_distance = distance ** 2

    mean_square_distance = square_distance.mean()

    score = np.sqrt(mean_square_distance)

    return score

rmse_score = make_scorer(rmse)
rmse_score
```

```python
# 훈련용 데이터 RMSE 구하기
score_B = cross_val_score(model_B, X_train_B, Y_train_B, cv=20, scoring=rmse_score).mean()
print('타율 오류: ',score_B)

score_E = cross_val_score(model_E, X_train_E, Y_train_E, cv=20, scoring=rmse_score).mean()
print('방어율 오류: ',score_E)

score_W = cross_val_score(model_W, X_train_W, Y_train_W, cv=20, scoring=rmse_score).mean()
print('승률 오류: ',score_W)
```

Coding

```
타율 오류:  0.0024172428788928205
방어율 오류:  0.06102082283369911
승률 오류:  0.018094190701567895
```

결과

6 최종 결과

- 모델로 예측 된 결과는 리그 최종 경기 성적
- 모델 예측 결과와 현재까지 경기 결과를 활용해 공식을 적용 (9월 21일 12시 기준)
- 최종적으로 남은 경기 결과를 예측

팀명	예측 최종 순위	리그 전체 예측 타율	리그 전체 예측 방어율	리그 전체 예측 승률	현재 순위	현재 진행 경기 수	현재 까지 타율	현재 까지 방어율	현재 까지 승률	남은 경기 수
NC	1	0.2932353	4.6580176	0.5925517	1	110	0.2907928	4.6376471	0.6074766	34
키움	2	0.2708269	4.5240488	0.5647739	2	117	0.2719582	4.5028754	0.5775862	27
KT	3	0.2855245	4.6425968	0.5630784	3	111	0.2828101	4.6011406	0.5727273	33
LG	4	0.2864602	4.5481122	0.5614713	4	113	0.2835859	4.5014798	0.5636364	31
두산	5	0.2937843	4.7135508	0.5470314	5	112	0.2946945	4.6196013	0.5462963	32
KIA	6	0.2774558	4.6880365	0.5414151	6	109	0.2758621	4.7348643	0.5412844	35
롯데	7	0.2758088	4.6871211	0.4974064	7	110	0.2748615	4.6732538	0.5137615	34
삼성	8	0.2722256	4.9297745	0.4789626	8	112	0.2731081	5.0155932	0.4545455	32
한화	9	0.2517212	5.2575168	0.3836645	10	112	0.2375735	5.2240326	0.2909091	32
SK	10	0.2535276	5.6046117	0.3779123	9	114	0.2496067	5.6570567	0.3362832	30

모델 예측 결과 + 현재까지 경기 결과

X = 전체 리그의 성적 (모델로 예측 된 값)

Y = 현재까지 진행한 경기의 성적

Z = 남은 경기의 성적 (우리가 구하고자 하는 값)

X * 전체 리그의 경기 수 = Y * (전체 리그의 경기 수 – 현재 진행한 경기 수) + Z * 남은 경기의 수

따라서 Z는

Z = (X * 전체 리그의 경기 수 - Y * (전체 리그의 경기 수 – 현재 진행한 경기 수)) / 남은 경기의 수

공식

6 최종 결과

- 9월 28일 이후 남은 경기에 대한 최종 예측 결과 (9월 21일 12시 기준)
- 팀별 약 30경기가 남았지만 남은 경기 결과도 평균에 가까워질 것으로 예상됨
- 따라서 9월 28일 이후 남은 경기 결과도 현재 예측 결과와 매우 유사할 것으로 예상됨

팀명	남은 경기 수	예측 타율	예측 방어율	예측 승률
NC	34	0.3011373	4.7239223	0.5442652
키움	27	0.2659246	4.6158000	0.5092537
KT	33	0.2946548	4.7820406	0.5306231
LG	31	0.2969374	4.7180950	0.5535792
두산	32	0.2905985	5.0423738	0.5496043
KIA	35	0.2824190	4.5422012	0.5418221
롯데	34	0.2788735	4.7319860	0.4444930
삼성	32	0.2691369	4.6294089	0.5644226
한화	32	0.3012383	5.3747115	0.7083086
SK	30	0.2684270	5.4053209	0.5361030

최종 예측 결과

6 최종 결과

- 종합 (9월 21일 12시 기준)

팀명	남은 경기 수	예측 타율	예측 방어율	예측 승률	현재 순위	현재 진행 경기 수	현재 까지 타율	현재 까지 방어율	현재 까지 승률	예측 최종 순위	리그 전체 예측 타율	리그 전체 예측 방어율	리그 전체 예측 승률
NC	34	0.3011373	4.7239223	0.5442652	1	110	0.2907928	4.6376471	0.6074766	1	0.2932353	4.6580176	0.5925517
키움	27	0.2659246	4.6158000	0.5092537	2	117	0.2719582	4.5028754	0.5775862	2	0.2708269	4.5240488	0.5647739
KT	33	0.2946548	4.7820406	0.5306231	3	111	0.2828101	4.6011406	0.5727273	3	0.2855245	4.6425968	0.5630784
LG	31	0.2969374	4.7180950	0.5535792	4	113	0.2835859	4.5014798	0.5636364	4	0.2864602	4.5481122	0.5614713
두산	32	0.2905985	5.0423738	0.5496043	5	112	0.2946945	4.6196013	0.5462963	5	0.2937843	4.7135508	0.5470314
KIA	35	0.2824190	4.5422012	0.5418221	6	109	0.2758621	4.7348643	0.5412844	6	0.2774558	4.6880365	0.5414151
롯데	34	0.2788735	4.7319860	0.4444930	7	110	0.2748615	4.6732538	0.5137615	7	0.2758088	4.6871211	0.4974064
삼성	32	0.2691369	4.6294089	0.5644226	8	112	0.2731081	5.0155932	0.4545455	8	0.2722256	4.9297745	0.4789626
한화	32	0.3012383	5.3747115	0.7083086	10	112	0.2375735	5.2240326	0.2909091	9	0.2517212	5.2575168	0.3836645
SK	30	0.2684270	5.4053209	0.5361030	9	114	0.2496067	5.6570567	0.3362832	10	0.2535276	5.6046117	0.3779123

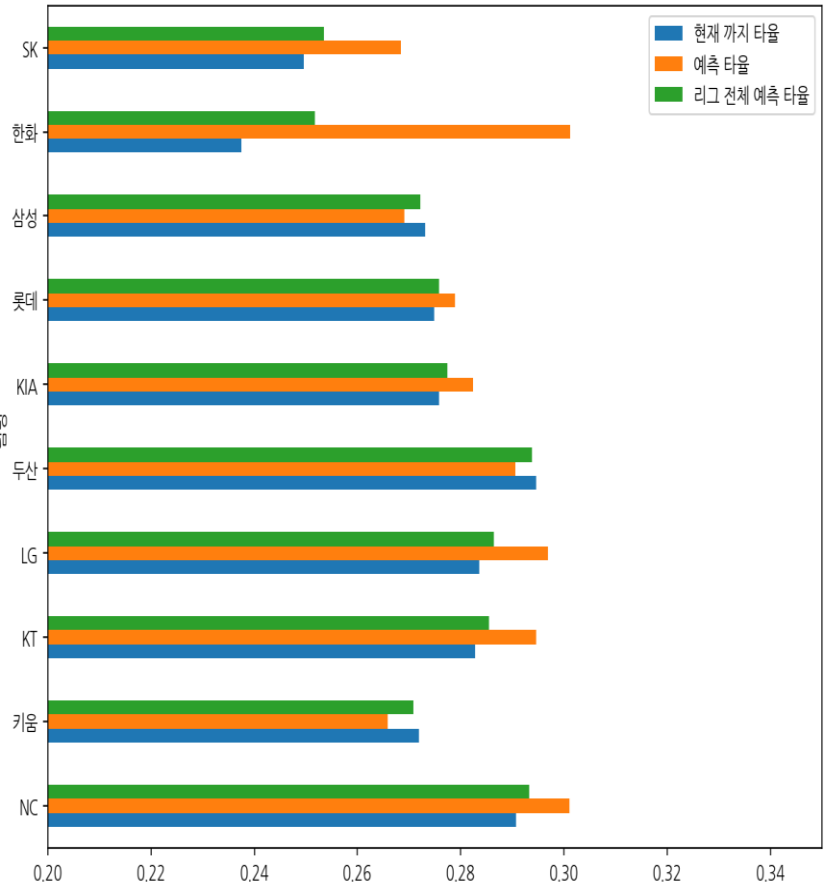
최종 예측 결과

현재까지 결과

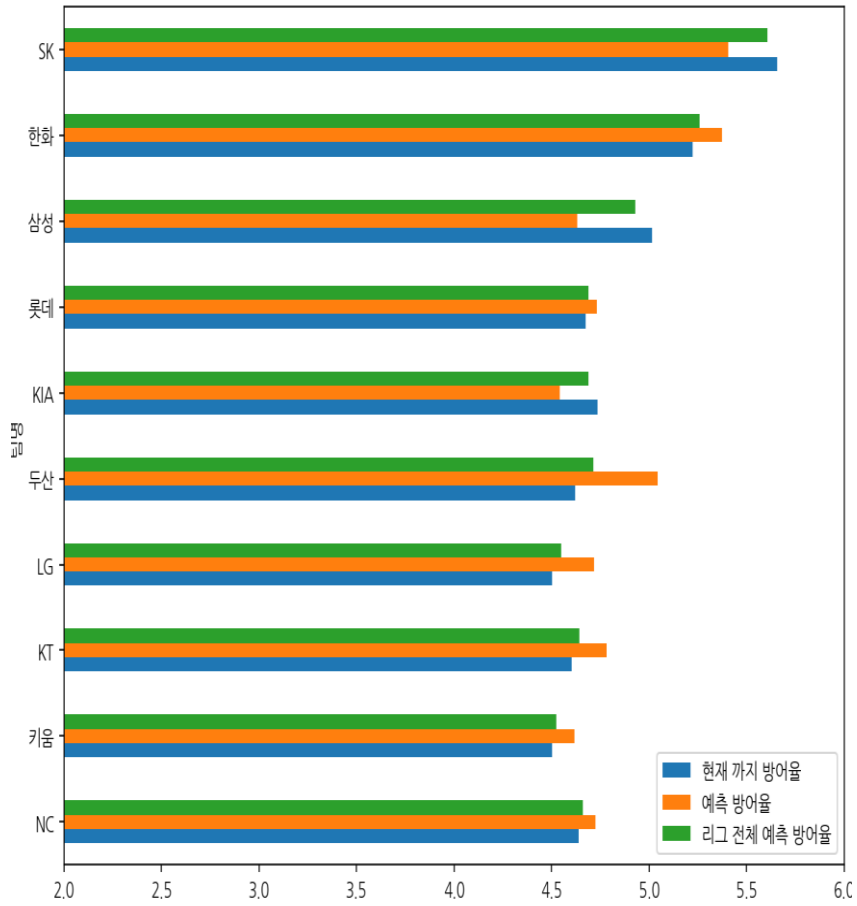
모델 예측 결과

6 최종 결과

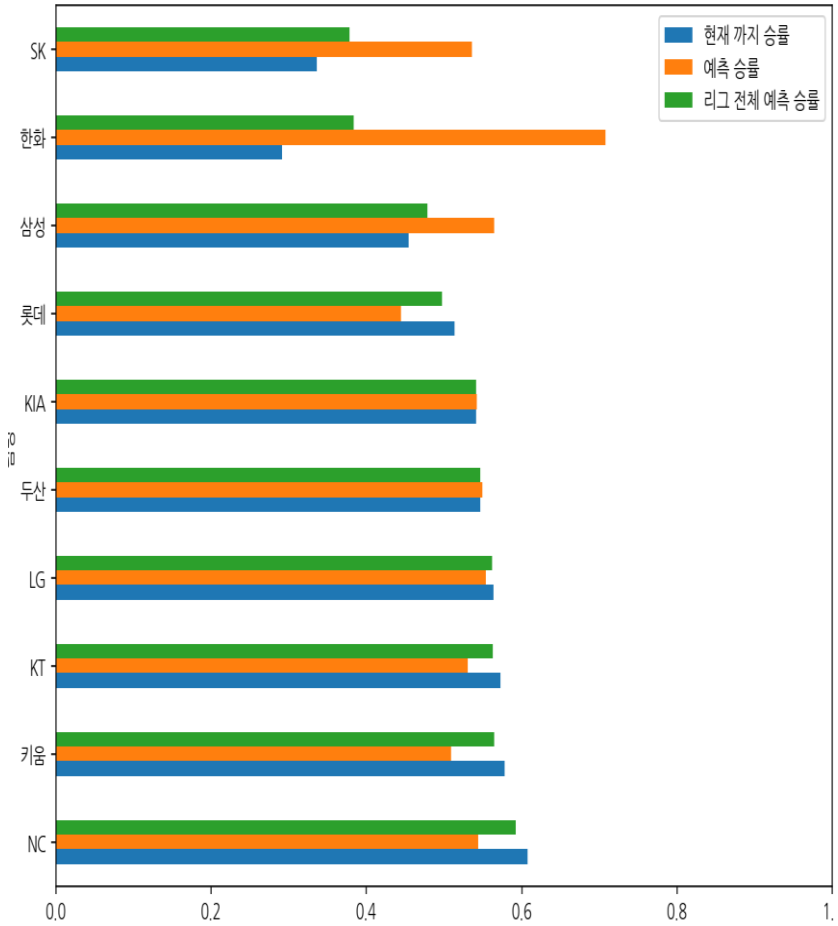
- 종합 (9월 21일 12시 기준)



타율



방어율



승률

6 최종 결과

- 모델의 장단점

- 타율, 방어율 예측 모델

- 리그 최종 성적을 기준으로 학습된 모델 (2020년 데이터와 유사하지 않은 데이터)
- 2020년 데이터 기준으로 최종 성적을 예측하기 때문에 일반화의 문제점이 존재
- 학습 데이터의 수가 많아 예측의 정확도는 높음

- 승률 예측 모델

- 마지막 20경기 이전 성적을 기준으로 학습된 모델 (2020년 데이터와 유사한 데이터)
- 타율, 방어율 예측 모델 보다 일반화가 잘 되어 있음
- 학습 데이터의 수가 매우 적은 단점이 존재

6 최종 결과

따라서 2001년 ~ 2015년의 마지막 20경기 이전 성적 데이터를

수집할 수 있다면

승률 예측에 활용된 모델3에

모델1에 적용된 학습 변수를 적용해

남은 경기의 타율, 방어율, 승률을 예측하는 것이

최적의 모델이 될 것이라고 판단됨.

감사합니다