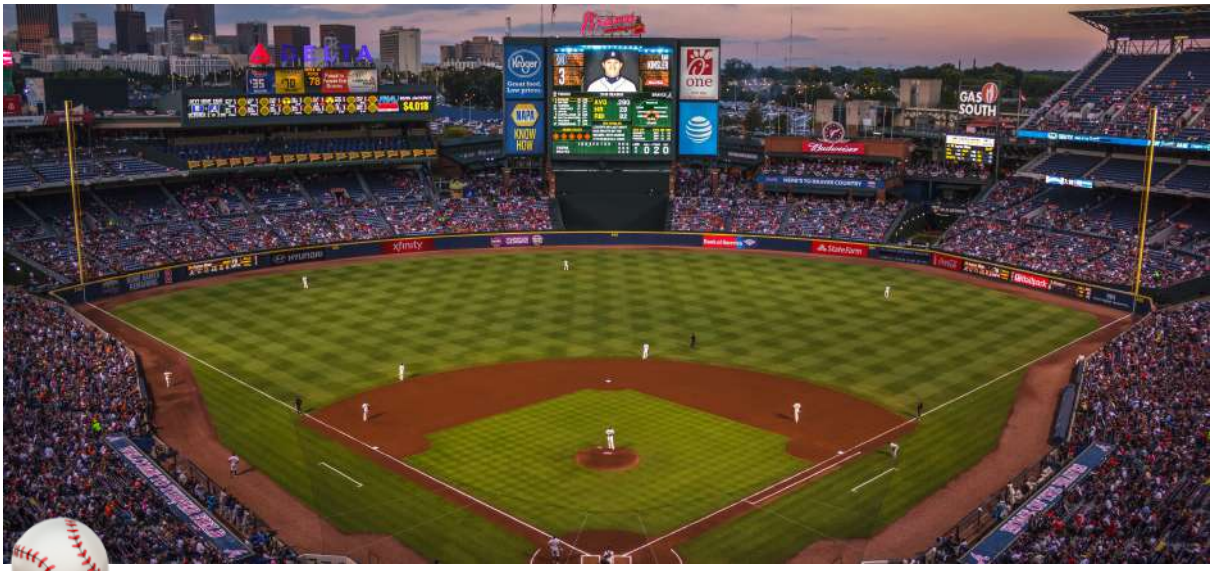


보고서_퍼펙트게임

추정 득점 예측 모델을 통한 국내 프로 야구 팀 별 최적 라인업 추천

1. 프로젝트 개요

1-1. 주제

- 데이터 분석 기반 국내 프로 야구 타순 최적화에 따른 득점 예측 모델 구축

1-2. 분석 목적

- 데이터 분석을 통해 최적의 타순을 도출하여 팀의 경기 승률을 높임
- 코칭스태프의 직관에 의존하는 기존 타순 결정 방식의 문제점에 따른 데이터 기반 타순 결정의 필요성
- 야구팬들에게 데이터 기반의 야구 분석에 대한 관심과 이해도를 증진 시키고자 함

1-3. 타겟 및 활용방안

- 구단에서 실제 타순을 결정할 때 활용
    - 각 선수의 강점을 극대화 할 수 있는 타순 배치

    - 상대 팀 투수와 수비의 데이터를 분석해 특정 경기에서 효과적인 타순 결정

- 교육 및 훈련 자료로 활용
    - 선수, 코치들에게 데이터에 기반한 경기 전략을 이해하고 적용할 수 있도록 함

- 야구 팬들의 흥미 및 궁금증 해결에 활용
    - 가상 라인업 구성에 관심이 있는 팬들에게 데이터 분석을 통한 객관적이고 과학적인 기반을 제공해 팬들도 효율적이고 최적화된 타순을 구성해 볼 수 있도록 함
    - 데이터 기반의 야구 분석에 대한 관심과 이해도를 증진 시킬 수 있음

2. 프로젝트 수행 절차 및 방법

2-1. 검토사항

1. 앞선 타자의 출루율은 뒷 타자의 장타율에 긍정적인 영향을 준다
2. 도루능력이 높은 타자가 앞선 타순에 배치되면, 뒤 타자의 타율이 증가한다
3. 득점권 타율이 높은 타자를 중심타선에 배치하면 팀의 추정득점이 증가한다
4. 하위타선에 수비율이 높은 선수들을 배치하면 팀의 성적이 향상된다

2-2. 분석방법

- **EDA 데이터분석**
    - 각 지표와 팀의 추정득점 간의 상관관계를 분석하여 데이터를 탐색함
    - 상관계수를 계산하여, 각 지표가 팀의 추정득점에 미치는 영향을 평가함

- **상관관계 분석 및 라인업 배치 순서**
    - 상관계수가 높은 지표는 팀의 추정득점에 큰 영향을 미친다고 판단하여 분석함
    - 분석을 통해 타자 배치 및 타순 결정에 중요한 지표를 식별함

- **타자 조합 최적화**
    - 분석 결과를 바탕으로, 팀의 득점을 극대화할 수 있는 타자 조합 최적화 알고리즘을 적용함
    - 최적화 알고리즘을 통해 각 타자의 배치 및 타순을 결정함

- **추정득점 예측 모델 구축**
  - 최적화된 타순을 기반으로 추정득점 예측 모델을 구축함
  - 예측 모델을 통해 다양한 타순 시나리오에 따른 팀의 추정득점을 계산하고 2024.06.24일 기준 라인업 추정득점과 비교함

3. 데이터 전처리

3-1 데이터 설명

< 컬럼 설명 >

타율	타율은 안타를 친 비율, 안타 수를 타수로 나눈 값
경기	선수가 참여한 경기 수
타석	선수가 타자로 나온 총 횟수
타수	타석 중 실제로 타격을 시도한 횟수
득점	선수가 홈을 밟아 득점한 횟수
안타	선수가 타격 후 출루한 횟수
2루타	타자가 2루까지 진루한 안타 수
3루타	타자가 3루까지 진루한 안타 수
홈런	타자가 공을 쳐서 홈런을 친 횟수
루타	타자가 얻은 총 베이스 수
타점	타자가 타격으로 인해 다른 주자를 홈으로 불러들인 횟수
희생번트	타자가 번트로 주자를 전루시키고 자신은 아웃되는 경우
볼넷	타자가 4개의 볼을 얻어 1루로 진 루한 경우
고의4구	투수가 고의적으로 볼넷을 주어 1루로 진루시키는 경우
사구	타자가 투수가 던진 공에 맞아 1루로 진루하는 경우
삼진	타자가 3스트라이크를 받아 아웃 되는 경우
병살타	타자가 친 공으로 인해 두명의 주 자가 아웃되는 경우
장타율	타자의 장타능력을 나타내는 지표 로 총루타수를 타수로 나눈값
출루율	타자가 출루한 비율로, 안타, 볼넷, 사구를 포함한 출루 수를 타석 수 로 나눈 값
출루율+ 장타율	출루율과 장타율을 합한 값으로 타 자의 공격을 종합적으로 평가
멀티히트	한 경기에서 두 번 이상의 안타를 친 경우
득점권 타율	득점권에 주자가 있을 때의 타율
대타타율	대타로 나왔을 때의 타율

장타	2루타, 3루타, 홈런 등의 장타 수
땅볼	선수가 친 땅볼의 수
뜬공	선수가 친 뜬공의 수
땅볼/뜬공	땅볼 수를 뜬공 수로 나눈 비율
결승타	결승타의 수 즉 경기에서 승리를 결정짓는 안타 수
볼넷/삼진	볼넷 수를 삼진 수로 나눈 비율
투구수/타석	타석 당 투구 수의 평균
순수장타율	장타만 고려한 비율
추정득점	추정된 득점 수
(1.8x출루율 +장타율)/4	출루율과 장타율을 결합한 지표
타순	타자가 타석에 들어가는 순서
포지션	선수의 수비 위치
수비승리 기여도	수비에서 팀 승리에 기여한 정도
선발경기	선수가 선발로 출전한 경기 수
수비이닝	선수가 수비한 총 이닝 수
실책	선수가 수비 중 저지른 실책 수
견제사	투수에 의해 견제된 주자의 수
풋아웃	수비수가 타자나 주자를 아웃시 킨 횟수
어시스트	수비수가 다른 수비수에게 공을 전달하여 아웃시킨 횟수
수비율	수비 성공률
포일	포수가 놓친 공의 수
도루허용	도루성공 개수
도루저지	도루 실패 개수
도루시도	도루 성공과 도루 저지를 합한 총 시도 횟수
도루저지율	도루저지/(도루허용+도루저지)

3-2. 데이터 샘플

- 2019 ~ 2024.06.24 10구단 중 2023년 키움 히어로즈 데이터 예시

<kiwoom_hitter_2023.csv> → 10경기이상, 31타수이상 데이터만 사용

```
df1= pd.read_csv('/content/drive/MyDrive/DATATHON/data/키움/kiwoom_hitter_2023.csv')
df1.head(5)
```

	선수명	팀명	타율	경기	타석	타수	득점	안타	2루타	3루타	...	고의4구	사구	삼진	병살타	장타율	출루율	출루율+장타율	멀티히트	득점권타율	대타타율
0	박수종	키움	0.422	23	50	45	7	19	1	2	...	0	1	6	1	0.533	0.460	0.993	6	0.222	0.000
1	도슨	키움	0.336	57	261	229	37	77	14	2	...	0	9	41	3	0.454	0.398	0.852	25	0.279	0.000
2	김혜성	키움	0.335	137	621	556	104	186	29	6	...	3	3	77	6	0.446	0.396	0.842	50	0.314	0.000
3	송재선	키움	0.333	5	3	3	0	1	0	0	...	0	0	1	0	0.333	0.333	0.666	0	0.000	0.000
4	이주형	키움	0.326	69	243	215	32	70	13	4	...	2	5	53	4	0.507	0.390	0.897	20	0.339	0.167

```
Data volume : ( 48, 26 )
```

<kiwoom_hitter_detail_2023.csv>

```
df1= pd.read_csv('/content/drive/MyDrive/DATATHON/data/키움/kiwoom_hitter_detail_2023.csv')
df1.head(5)
```

	선수명	팀명	타율	장타	땅볼	뜬공	땅볼/뜬공	결승타	볼넷/삼진	투구수/타석	순수장타율	추정득점	(1.8x출루율+장타율)/4
0	박수종	키움	0.422	3	11	10	1.10	0	0.50	3.46	0.111	9.8	0.340
1	도슨	키움	0.336	19	60	56	1.07	2	0.44	3.55	0.118	42.4	0.293
2	김혜성	키움	0.335	42	135	163	0.83	7	0.74	3.77	0.112	98.6	0.290
3	송재선	키움	0.333	0	1	0	-	0	0.00	2.67	0.000	0.3	0.233
4	이주형	키움	0.326	23	40	54	0.74	4	0.36	3.81	0.181	39.7	0.302

```
Data volume : ( 48, 13 )
```

<kiwoom_batting_order_2023.csv> → 10타수 이상 데이터만 사용

```
df1= pd.read_csv('/content/drive/MyDrive/DATATHON/data/키움/kiwoom_batting_order_2023.csv',encoding='cp949')
df1.head(5)
```

	선수명	팀명	타율	타수	안타	2루타	3루타	홈런	타점	볼넷	사구	삼진	병살타	타순
0	임지열	키움	0.500	12	6	1	0	1	5	3	0	2	0	1번
1	김혜성	키움	0.400	90	36	6	3	1	10	11	0	14	0	1번
2	박수종	키움	0.364	22	8	1	1	0	1	1	1	2	1	1번
3	김태진	키움	0.360	25	9	0	0	0	1	0	0	2	0	1번
4	송성문	키움	0.290	31	9	1	0	1	4	4	0	2	0	1번

```
Data volume : ( 289, 14 )
```

<kiwoom_defense_2023.csv>

```
df1= pd.read_csv('/content/drive/MyDrive/DATATHON/data/키움/kiwoom_defense_2023.csv')
df1.head(5)
```

	선수명	포지션	수비	승리	기여도	년도	팀명	경기	선발경기	수비이닝	실책	견제사	풋아웃	어시스트	병살	수비율	포일	도루허용	도루저지	도루저지율
0	이정후	중견수			1.282	2023	키움	81	80	698	1	0	204	5	1	0.995	0	0	0	-
1	이지영	포수			0.944	2023	키움	78	69	539	4	0	467	32	3	0.992	0	52	14	21.2
2	김혜성	2루수			0.869	2023	키움	126	124	1067	15	0	247	374	79	0.976	0	0	0	-
3	박수종	우익수			0.648	2023	키움	14	11	102 2/3	0	0	35	1	0	1.000	0	0	0	-
4	송성문	3루수			0.504	2023	키움	88	74	685 2/3	8	0	73	128	16	0.962	0	0	0	-

```
Data volume : (72, 18)
```

<kiwoom_runner_2023.csv>

```
df1= pd.read_csv('/content/drive/MyDrive/DATATHON/data/키움/kiwoom_runner_2023.csv',encoding='cp949')
df1.head(5)
```

	선수명	팀명	경기	도루시도	도루허용	도루저지	도루성공률	주루사	견제사
0	김혜성	키움	137	28	25	3	89.3	6	2
1	도슨	키움	57	11	9	2	81.8	2	0
2	이정후	키움	86	9	6	3	66.7	1	1
3	임병욱	키움	80	4	4	0	100	4	0
4	이주형	키움	69	4	3	1	75.0	0	1

```
Data volume : ( 48, 9 )
```

- 모델링 학습에 사용한 데이터(2019~2023년 5년간 상위5개팀)

<KBO_top5.csv>

```
df1= pd.read_csv('/content/drive/MyDrive/DATATHON/data/@KBO_5개구단/KBO_top5.csv',encoding='cp949')
df1.head(5)
```

	선수명	팀명	년도	타율	경기	타수	득점	홈런	출루율	장타율	출루율+장타율	득점권타율	추정득점	포지션	수비	승리	기여도	수비이닝	합계	도루시도	도루허용	도루성공률
0	페르난데스	두산	2019	0.344	144	572	87	15	0.409	0.483	0.892	0.313	104.2	['1루수']			-0.3600	117	3.0	1.0	33.3
1	박건우	두산	2019	0.319	127	458	83	10	0.397	0.465	0.862	0.284	82.1	['우익수', '중견수']			0.0455	1002 1/3	15.0	12.0	80.0
2	이흥련	두산	2019	0.310	27	42	3	0	0.341	0.405	0.746	0.222	5.1	['포수']			0.2920	91	0.0	0.0	-
3	오재일	두산	2019	0.293	130	467	76	21	0.369	0.495	0.864	0.344	84.0	['1루수']			-0.1580	1044	3.0	2.0	66.7
4	허경민	두산	2019	0.288	133	475	71	4	0.350	0.371	0.721	0.324	63.4	['3루수', '2루수']			0.1315	1082	16.0	11.0	68.8

```
Data volume : ( 896, 19 )
```

3-3. 데이터 수집 및 전처리

- 데이터 수집
  - KBO 기록실의 팀별 (2019 - 2024.06.24) 타자/주루/수비 기록을 selenium을 이용하여 크롤링

    - 스탯티즈의 WAAwithPOS (평균 대비 수비 승리 기여-포지션 조정 포함)을 selenium을 이용하여 크롤링

- 데이터 전처리
    - 각 팀의 타자/주루/수비 기록 중 필요한 컬럼들을 선정함
    - 1루타에 대한 기록이 없어 안타-(2루타+3루타+홈런)으로 1루타 컬럼을 생성
    - 출루율 및 장타율을 계산하여 추가함
    - 추정 득점 공식에 따라 추정득점 컬럼 추가함
    - 특정 타순의 기록이 없는 경우에는 가중치를 반영하여 각 선수의 1-9번 기록을 채움

3-4. 활용한 라이브러리 등 기술적 요소

- ☑ ~~numpy~~
- ☑ ~~pandas~~
- ☑ ~~sklearn~~
- ☑ ~~matplotlib~~
- ☑ ~~seaborn~~
- ☑ ~~selenium~~
- ☑ ~~pycaret, optuna~~

4. 분석결과

4-1. 검토사항

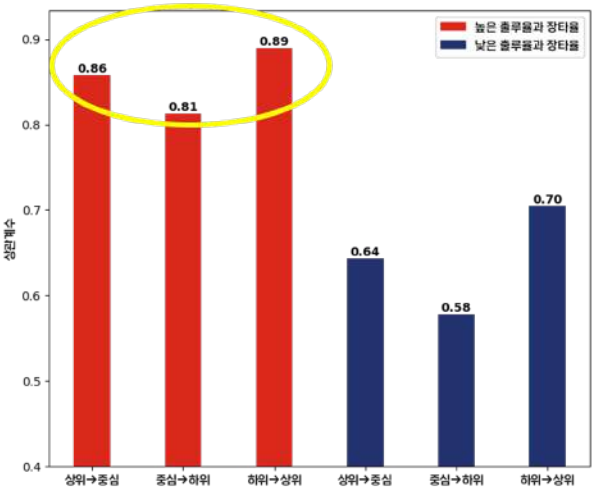
1. 앞 타자가 잘 치면 뒷 타자는 더 잘 칠까?

<출루율과 장타율의 상관관계>

앞 타자의 **출루율 높을** 때 뒷 타자가 **장타 확률**이 큼
- 높은 출루율 - 장타율 : **80% 이상**
- 낮은 출루율 - 장타율 : 50~70%

Q. 하위타자의 출루율과 상위타자의 장타율의 상관관계가 가장 크다?

<출루율과 장타율의 상관관계>



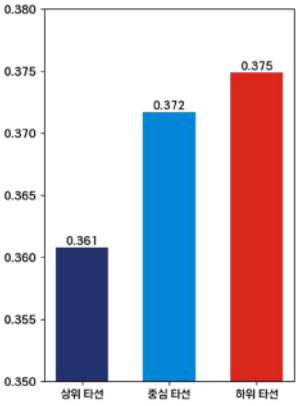
1. 앞 타자가 잘 치면 뒷 타자는 더 잘 칠까?

<타순별 출루율>

년도	상위타선	중심타선	하위타선
2019	0.352	0.365	0.366
2020	0.370	0.381	0.383
2021	0.362	0.379	0.375
2022	0.362	0.359	0.378
2023	0.358	0.375	0.372
평균	0.361	0.372	0.375

하위타선의 평균 출루율이 가장 높음

<타순 유형에 따른 평균 출루율 비교>



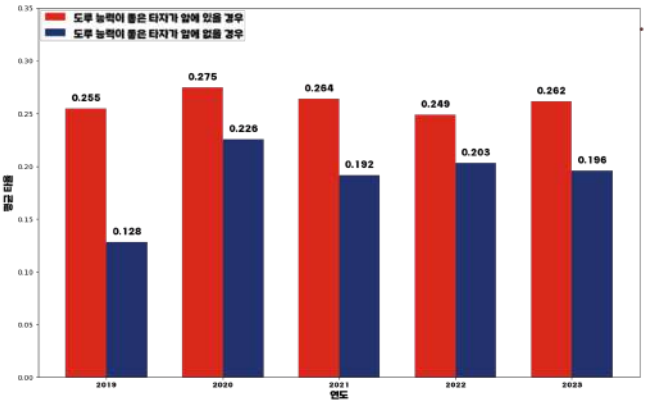
상위타선에 장타력이 좋은 선수 배치하면 시너지 효과 기대

2. 앞 타자가 도루를 잘 하면 뒷 타자의 타율이 증가할까?

<앞 타자의 도루 능력에 따른 평균 타율>

년도	앞의 도루능력	앞의 도루능력
2019	0.255	0.128
2020	0.275	0.226
2021	0.264	0.192
2022	0.249	0.203
2023	0.262	0.196
평균	0.261	0.189

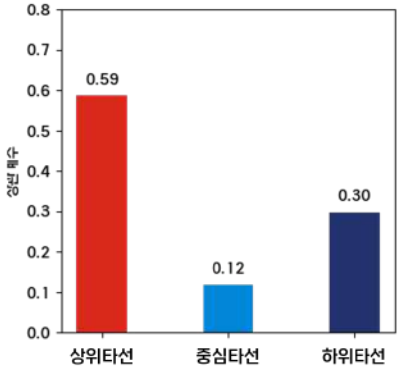
<연도별 평균타율 비교>



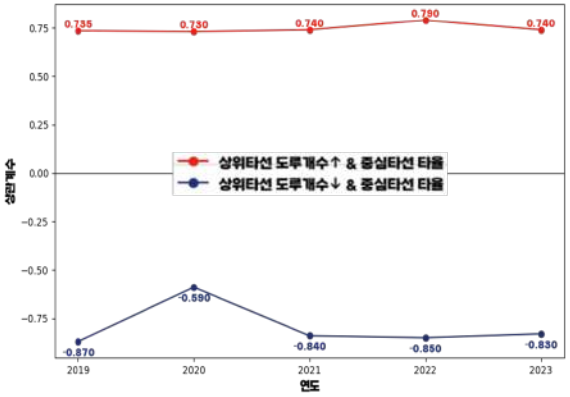
도루능력이 좋은 타자들이 앞에 배치될 경우 뒤 타자의 평균 타율 증가

2. 앞 타자가 도루를 잘 하면 뒷 타자의 타율이 증가할까?

<타선별 도루능력과 추정득점 간 상관관계>
도루 능력은 상위타선의 중요한 지표



<연도별 중심타선 타율 비교>



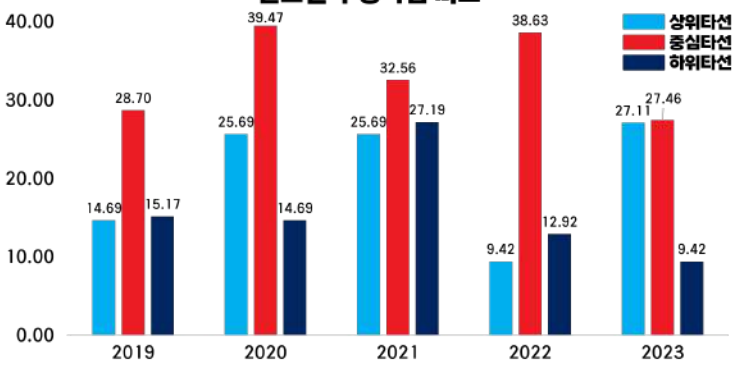
상위타선의 도루능력이 좋으면 중심타선의 타율이 증가

3-1. 득점권 타율이 높은 타자를 중심타선에 배치하면 총 추정득점이 증가할까?

<타순 배치에 따른 추정득점>

년도	상위타선	중심타선	하위타선
2019	14.69	28.70	15.17
2020	25.69	39.47	14.69
2021	25.69	32.56	27.19
2022	9.42	38.63	12.92
2023	27.11	27.46	9.42
평균	20.52	33.36	15.88

<연도별 추정득점 비교>



주 : 상위타선, 중심타선, 하위타선은 득점권 타율이 높은 타자들을 배치한 위치

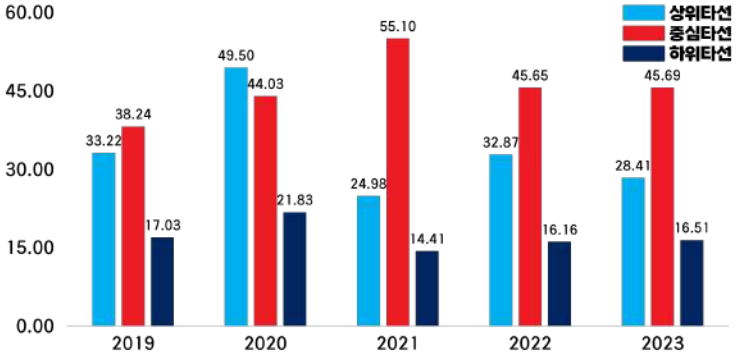
득점권 타율이 높은 선수가 중심타선에 배치 될 경우 추정득점이 가장 높음

3-2. 장타율이 높은 타자를 중심타선에 배치하면 총 추정득점이 증가할까?

<타순 배치에 따른 추정득점>

년도	상위타선	중심타선	하위타선
2019	33.22	38.24	17.03
2020	49.50	44.03	21.83
2021	24.98	55.10	14.41
2022	32.87	45.65	16.16
2023	28.41	45.69	16.51
평균	33.80	45.74	17.19

<연도별 추정득점 비교>

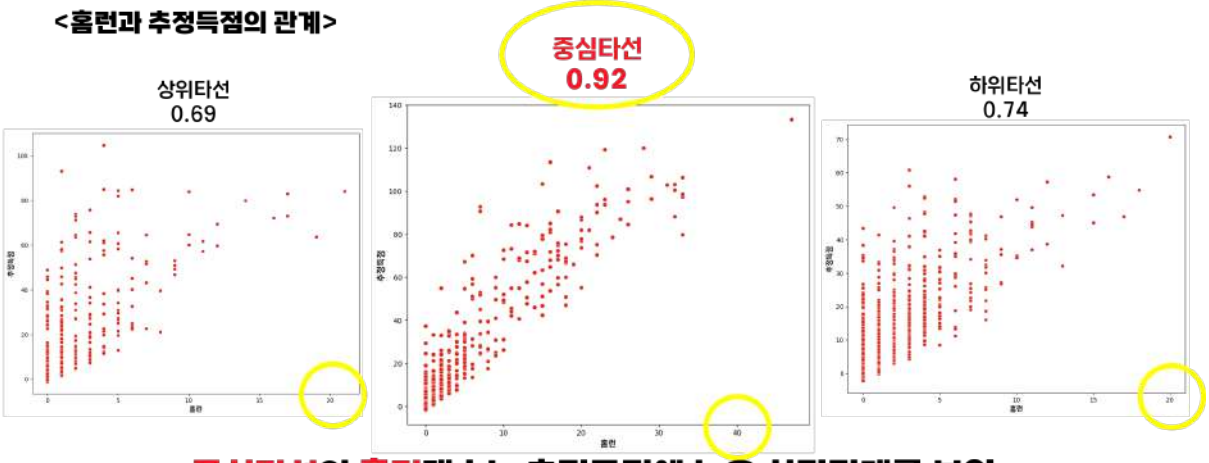


주 : 상위타선, 중심타선, 하위타선은 득점권 타율이 높은 타자들을 배치한 위치 / 타선당 상위 10% 선수들만 선정하였음

장타율이 높은 선수가 중심타선에 배치 될 경우 추정득점이 가장 높음

3-3. 홈런이 많은 타자를 중심타선에 배치하면 총 추정득점이 증가할까?

<홈런과 추정득점의 관계>



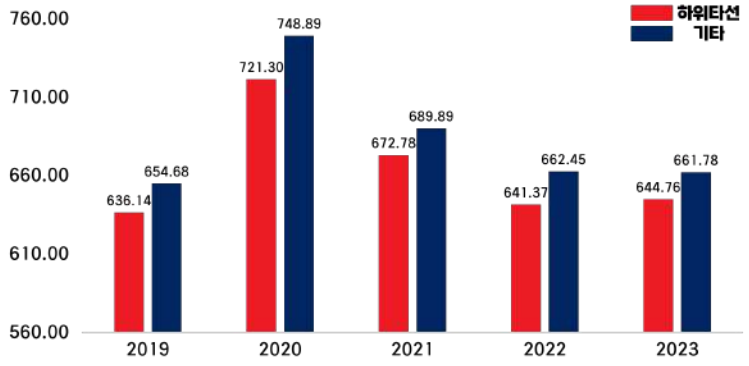
중심타선의 홈런개수는 추정득점에 높은 상관관계를 보임

4. 수비승리기여도가 높은 선수를 하위타선 배치 시 추정실점이 낮아질까?

< 추정 실점 공식 : 추정실점 = (리그 평균 실점 *경기수) – (수비승리기여도*조정계수)/경기수*시즌 총 경기수 >

<KBO 10구단 연도별 평균 추정실점 비교>

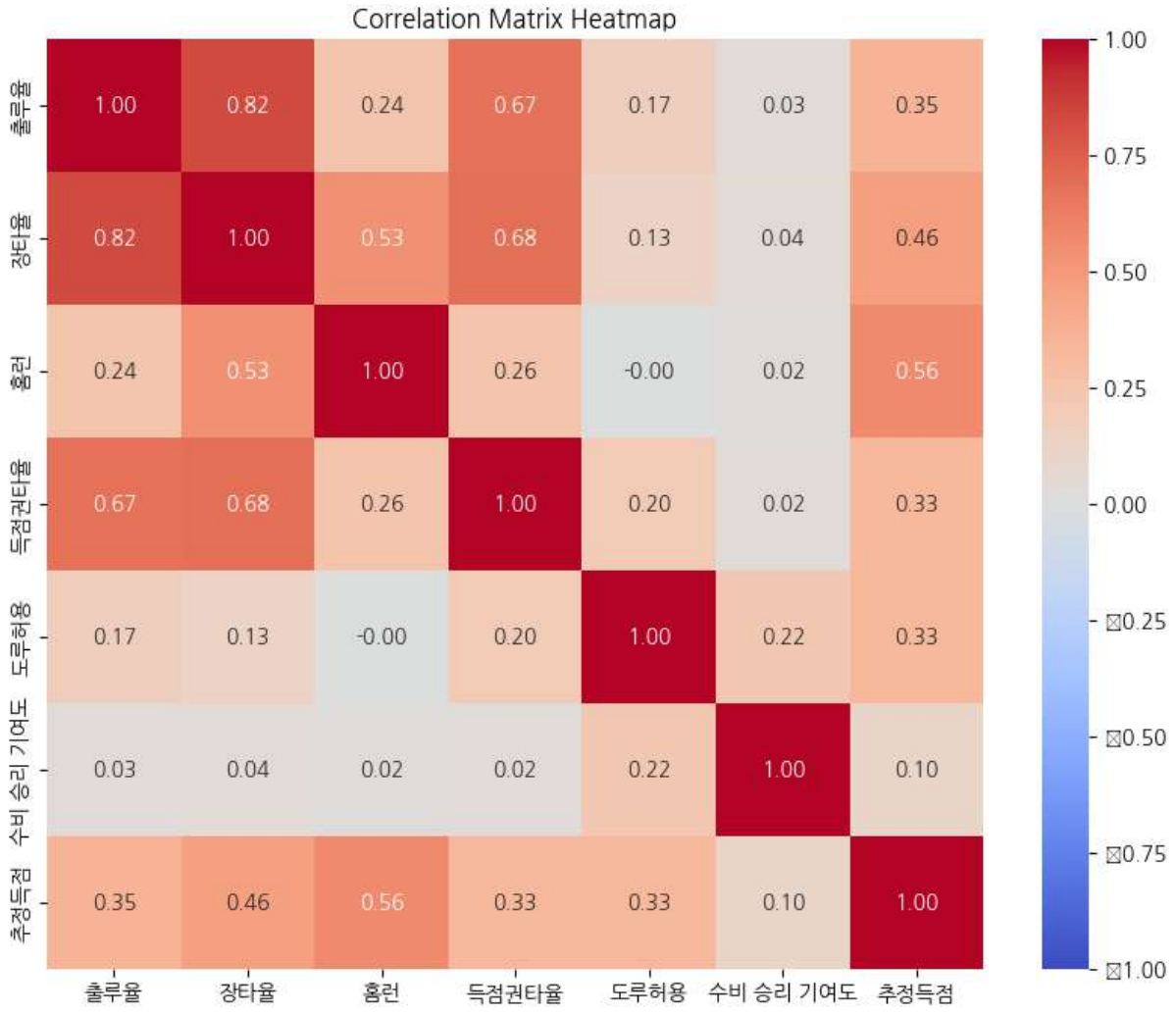
년도	하위타선	기타
2019	636.14	654.68
2020	721.30	748.89
2021	672.78	689.89
2022	641.37	662.45
2023	644.76	661.78
평균	663.27	683.54



하위타선의 추정실점이 기타(상위/중심타선)의 추정실점보다 낮음

4-2. 상관관계 분석 및 라인업 배치 순서

- 추정득점과 상관관계



→ 상관계수가 높은 순: 홈런, 장타율, 출루율, 득점권 타율, 도루허용, 수비승리 기여도

- 1번타자 지표 : 출루율, 도루허용
  상관계수 평균 : (0.35+0.33)/2 = 0.34
- 2번타자 지표 : 홈런, 장타율, 출루율, 도루허용
  상관계수 평균 : (0.56+0.46+0.35+0.33)/4 = 0.425
- 중심타선 지표 : 홈런, 장타율, 득점권타율
  상관계수 평균 : (0.56+0.46+0.33)/3 = 0.45
- 하위타선 지표 : 출루율, 수비승리기여도
  상관계수 평균 : (0.35+0.10)/2 = 0.225

📣 중심타선 - 2번타자 - 1번타자 - 하위타선 순으로 라인업 배치 결정

4-3. 타자 조합 최적화

4-3-1. 라인업 만들기

중심타선(3~5번) 배치

- 장타율, 득점권 타율, 홈런 지표를 표준화하여 선수별 성적을 비교 하고, 표준화된 지표들에 가중치(지표별 추정득점과의 상관계수)를 부여하여 합산 지표를 계산함
- 중심타선 합산 지표 기준으로 상위 3명의 선수를 선택

```
# 표준화 스케일러 생성
scaler = StandardScaler()

# 장타율과 득점권 타율 표준화
df[['장타율_표준화', '득점권 타율_표준화', '홈런_표준화']] = scaler.fit_transf
                                                                '홈

# 표준화된 지표 합산
df['중심타선 합산 지표'] = df['장타율_표준화']*0.46 + df['득점권 타율_표준화']*

# 표준화된 합산 지표 기준으로 상위 3명 선택
center_hitters_1 = df

# 결과 출력
center_hitters_1 = center_hitters_1.nlargest(3, '중심타선 합산 지표')
center_hitters_1.nlargest(3, '중심타선 합산 지표')
```

- 선택된 3명의 선수의 3번, 4번, 5번 타순 기록을 데이터프레임에서 추출하고, 각 선수의 포지션 정보를 병합하여 최종 데이터프레임을 구성함

```
# center_hitters의 선수명을 df2에서 찾기
center_hitters_names = center_hitters_1['선수명'].values
filtered_df2 = df2[df2['선수명'].isin(center_hitters_names)]

# '타순'이 '3번', '4번', '5번'인 것만 남기기
filtered_df2 = filtered_df2[filtered_df2['타순'].isin(['3번', '4번', '5

center_hitters = filtered_df2

# center_hitters_1에서 '포지션' 컬럼만 선택
center_hitters_1_position = center_hitters_1[['선수명', '포지션']]
```

```
# '선수명'을 기준으로 병합
center_hitters = pd.merge(center_hitters, center_hitters_1_position,

center_hitters
```

- 각 선수명에 따른 타순 값을 리스트로 모아 타순 조합 생성함

```
# 각 선수명에 따른 타순 값을 리스트로 모음
center_per_player = filtered_df2.groupby('선수명')['타순'].apply(list).
center_per_player
```

	선수명	타순
0	김혜성	[3번, 4번, 5번]
1	도슨	[3번, 4번, 5번]
2	송성문	[3번, 4번, 5번]

2번타자 배치

- 표준화 및 지표 합산- 장타율, 출루율, 도루허용(=도루 성공 개수)

```
# 표준화 스케일러 생성
scaler = StandardScaler()

# 장타율, 출루율, 도루허용 표준화
df[['장타율_표준화', '출루율_표준화', '도루_표준화', '홈런_표준화']] = scaler.f


# 2번타자 지표 합산 (가중치는 예시로 설정, 필요에 따라 조정 가능)
df['2번타자 합산 지표'] = df['장타율_표준화']*0.46 + df['출루율_표준화']*0.35
+ df['도루_표준화']*0.33 + df['홈런_표준화']*0.56
```

- 포지션을 고려하기 위해 중심 타자들의 주 포지션을 받아온 뒤 필터링

```
# center_hitters에 있는 선수명 제외 (중심타선 3명 제외)
exclude_names = center_hitters['선수명'].tolist()
```

```
filtered_df = df[~df['선수명'].isin(exclude_names)]

# center_hitters의 주 포지션 확인
center_main_positions = center_hitters['포지션'].apply(lambda x: x.spli

# 포지션별 필터링 함수
def filter_by_position(df, position, min_count, max_count, excluded_p
    if position in excluded_positions:
        return pd.DataFrame()  # 제외된 포지션이면 빈 데이터프레임 반환
    pos_df = df[df['포지션'].str.contains(position)]
    if len(pos_df) > max_count:
        pos_df = pos_df.nlargest(max_count, '2번타자 합산 지표')
    elif len(pos_df) < min_count:
        pos_df = df[df['포지션'].str.contains(position)].nlargest(min_c
    return pos_df

# 포지션별로 필터링
catcher = filter_by_position(filtered_df, '포수', 1, 1, center_main_po
outfielders = filter_by_position(filtered_df, '좌익수|우익수|중견수', 1,
infielders = filter_by_position(filtered_df, '1루수|2루수|3루수|유격수',
```

```
# 2번타자 합산 지표 기준으로 상위 1명 선택
second_hitter_1 = final_candidates.nlargest(1, '2번타자 합산 지표')
second_hitter_1
```

	선수명	팀명	타율	경기	타수	홈런	득점	장타율	출루율	득점권타율	...	포지션	수비	승리 기여도	수비이닝	장타율_표준화	득점권 타율_표준화	홈런_표준화	중심타선 합산 지표	출루율_표준화	도루_표준화	2번타자 합산 지표
8	이형종	키움	0.268	21	71	4	18	0.479	0.402	0.45	...	우익수		-0.041	160.0	1.183148	1.455817	0.418472	1.259012	1.075184	-0.100732	0.920563

1 rows × 25 columns

1번타자 배치

- 표준화 및 지표 합산- 장타율, 출루율, 도루허용(=도루 성공 개수)

```
# 표준화 스케일러 생성
scaler = StandardScaler()

# 장타율, 출루율, 도루허용 표준화
df[['출루율_표준화', '도루_표준화']] = scaler.fit_transform(df[['출루율', '
```

```
# 1번타자 지표 합산 (가중치는 예시로 설정, 필요에 따라 조정 가능)
df['1번타자 합산 지표'] = df['출루율_표준화']*0.35 + df['도루_표준화']*0.33
```

- 4-3-2의 2번 타자 선정과 같이 앞서 정한 타자들의 주 포지션을 받아온 뒤 필터링

```
# 1번타자 합산 지표 기준으로 상위 1명 선택
first_hitter_1 = final_candidates.nlargest(1, '1번타자 합산 지표')
first_hitter_1
```

	선수명	팀명	타율	타수	안타	2루타	3루타	홈런	타점	볼넷	...	도루저지	고의4구	희생플라이	희생번트	수비	승리	기여도	득점권타율	추정득점	출루율	장타율	포지션
0	이용규	키움	0.304	92	28	3	0	1	5	14	...	1	0	1	2			0.031	0.207	15.596	0.409	0.37	좌익수, 중견수, 우익수

1 rows × 26 columns

하위타선 배치

- 표준화 및 지표 합산- 출루율, 수비 승리 기여도

```
# 출루율과 수비 승리 기여도 표준화
scaler = StandardScaler()
df[['출루율_표준화', '수비_표준화']] = scaler.fit_transform(df[['출루율', '

# 하위타선 지표 합산
df['하위타선 합산 지표'] = df['출루율_표준화']*0.35 + df['수비_표준화']*0.10
```

- 위에서 시행했던 방법 대로 상위/중심타선 선수의 포지션을 필터링
- 이후, 상위/중심타선에서 포수가 포함되었는지 여부를 확인 후 포수를 추가/삭제

```
# 포수가 center_hitters, second_hitter_1, first_hitter_1에 없는지 확인
if not any('포수' in pos for pos in center_main_positions) and \
   not any('포수' in pos for pos in second_hitter_main_positions) and
   not any('포수' in pos for pos in first_hitter_main_positions):
    if '포수' not in bottom_candidates['포지션'].values:
        # 포수를 제외한 나머지 선수들 중 상위 3명 선택
        bottom_candidates = bottom_candidates.iloc[:-1]
        # filtered_df에서 포지션이 포수인 선수 추가
```

```
        catcher = filtered_df[filtered_df['포지션'].str.contains('포수'
        bottom_candidates = pd.concat([bottom_candidates, catcher])

# 포수가 2명 이상 포함된 경우 처리
if bottom_candidates['포지션'].str.contains('포수').sum() > 1:
    # 포수들 중에서 하위타선 합산 지표가 가장 높은 선수 선택
    best_catcher = bottom_candidates[bottom_candidates['포지션'].str.co
    # 포수를 제외한 나머지 선수들 중 상위 3명 선택
    bottom_candidates = bottom_candidates[~bottom_candidates['포지션']
    # 다시 포수 추가하여 상위 4명 선택
    bottom_candidates = pd.concat([bottom_candidates, best_catcher])
```

- 부족한 포지션이 있는지 체크 후 부족한 포지션이 있다면 포지션을 조정

```
# 부족한 포지션이 있는지 확인
missing_positions = {'포수', '좌익수', '우익수', '중견수', '1루수', '2루수',

# 부족한 포지션이 있다면 처리
if missing_positions:
    for position in missing_positions:
        if not any(position in pos for pos in bottom_hitters_1['포지션
            # 점수가 낮은 선수 제거
            lowest_score_player = bottom_hitters_1.nsmallest(1, '하위타
            bottom_hitters_1 = bottom_hitters_1.drop(lowest_score_pla
            # 해당 포지션의 선수 추가
            position_candidate = filtered_df[filtered_df['포지션'].str
            bottom_hitters_1 = pd.concat([bottom_hitters_1, position_
```

	선수명	팀명	타율	경기	타수	홈런	득점	장타율	출루율	득점권타율	...	장타율_표준화	득점권 타율_표준화	홈런_표준화	중심타선 합산 지표	출루율_표준화	도루_표준화	2번타자 합산 지표	1번타자 합산 지표	수비_표준화	하위타선 합산 지표
2	김건희	키움	0.321	20	56	1	5	0.411	0.381	0.412	...	0.569775	1.141493	-0.485428	0.366950	0.769835	-0.359759	0.140979	0.150722	0.382532	0.307695
6	이주형	키움	0.280	39	157	3	28	0.395	0.369	0.217	...	0.425452	-0.471486	0.117172	0.105734	0.595350	-0.359759	0.350977	0.089652	-0.002790	0.208093
5	고영우	키움	0.307	46	127	0	12	0.354	0.375	0.400	...	0.055625	1.042232	-0.786728	-0.071044	0.682592	-0.359759	-0.294793	0.120187	-0.630723	0.175835
18	최주환	키움	0.187	58	214	5	20	0.285	0.252	0.212	...	-0.566769	-0.512844	0.719772	-0.026880	-1.105880	-0.359759	-0.363420	-0.505779	-0.305779	-0.417636

4 rows × 28 columns

- 선정된 선수들의 각 타순 기록을 불러오기

```
# bottom_hitters의 선수명을 df2에서 찾기
bottom_hitters_names = bottom_hitters_1['선수명'].values
filtered_df2 = df2[df2['선수명'].isin(bottom_hitters_names)]
```

```
# '타순'이 '6-9번' 인 것만 남기기
filtered_df2 = filtered_df2[filtered_df2['타순'].isin(['6번', '7번', '8

bottom_hitters = filtered_df2
# bottom_hitters_1에서 '포지션' 컬럼만 선택
bottom_hitters_1_position = bottom_hitters_1[['선수명', '포지션']]

# '선수명'을 기준으로 병합
bottom_hitters = pd.merge(bottom_hitters, bottom_hitters_1_position,

bottom_hitters
```

```
# 각 선수명에 따른 타순 값을 리스트로 모음
bottom_per_player = filtered_df2.groupby('선수명')['타순'].apply(list).
bottom_per_player
```

	선수명	타순
0	고영우	[6번, 7번, 9번, 8번]
1	김건희	[8번, 9번, 6번, 7번]
2	이주형	[6번, 7번, 8번, 9번]
3	최주환	[6번, 9번, 7번, 8번]

📣 타자선발 KEY POINT

- 수비 포지션을 고려하기 위해 상위, 중심타선에서 포수가 없으면 하위타선에 포수 중 점수가 가장 높은 선수를 선택

- 포수 외 포지션 누락이 있다면 하위타선에서 선택된 4명 중 가장 점수가 낮은 선수를 제외하고 누락된 포지션의 선수 중 점수가 가장 높은 선수를 추가

4-3-2. 라인업 결합

상위타선 결합

- 상위타선을 결합하기 위해 first_hitter, second_hitter df병합

```
# first_hitter와 second_hitter 병합
df = pd.concat([first_hitter, second_hitter], ignore_index=True)
```

중심타선 결합

- 중심타선의 가능한 타순 조합을 생성하고, 각 조합을 df저장

```
# 선수별 가능한 타순을 딕셔너리로 변환
players = center_per_player.set_index('선수명')['타순'].to_dict()

# 가능한 타순 조합 생성
possible_orders = list(itertools.product(*players.values()))

# 타순이 중복되지 않는 조합 필터링
valid_orders = [order for order in possible_orders if len(set(order))

# 타순과 선수 이름을 매칭하여 조합 생성
combinations = []
for order in valid_orders:
    order_dict = {order[i]: player for i, player in enumerate(players
    combinations.append(order_dict)

# 각 조합을 center_hitters에서 찾아오기
results = []
for comb in combinations:
    filtered_center_hitters = center_hitters[center_hitters.apply(lam
    if len(filtered_center_hitters) == len(players):  # 모든 타순이 일치
        results.append(filtered_center_hitters)

# 결과를 각각 center_1, center_2 등의 DataFrame으로 저장
center_dfs = []
for i, result in enumerate(results):
    center_df = result.reset_index(drop=True)
    center_dfs.append(center_df)
    globals()[f'center_{i+1}'] = center_df
```

```
중심타선, 상위타선 df 병합

# center_dfs와 df를 각각 concat
```

```
concat_dfs = []
for i, center_df in enumerate(center_dfs):
    concat_df = pd.concat([df, center_df], ignore_index=True)
    concat_dfs.append(concat_df)
    globals()[f'df_{i+1}'] = concat_df
```

하위타선 결합

- 하위타선의 가능한 타순 조합을 생성하고 , 각 조합 df 저장

```
# 선수별 가능한 타순을 딕셔너리로 변환
players = bottom_per_player.set_index('선수명')['타순'].to_dict()

# 가능한 타순 조합 생성
possible_orders = list(itertools.product(*players.values()))

# 타순이 중복되지 않는 조합 필터링
valid_orders = [order for order in possible_orders if len(set(order))

# 타순과 선수 이름을 매칭하여 조합 생성
combinations = []
for order in valid_orders:
    order_dict = {order[i]: player for i, player in enumerate(players
    combinations.append(order_dict)

# 각 조합을 bottom_hitters에서 찾아오기
results = []
for comb in combinations:
    filtered_bottom_hitters = bottom_hitters[bottom_hitters.apply(lam
    if len(filtered_bottom_hitters) == len(players):  # 모든 타순이 일치
        results.append(filtered_bottom_hitters)

# 결과를 각각 bottom_1, bottom_2 등의 DataFrame으로 저장
bottom_dfs = []
for i, result in enumerate(results):
    bottom_df = result.reset_index(drop=True)
    bottom_dfs.append(bottom_df)
    globals()[f'bottom_{i+1}'] = bottom_df
```

→ 가능한 조합 중심타선(6가지 경우의 수) X 하위타선(24가지 경우의 수) = 총 144가지 라인업 경우의 수 생김

4-4. 추정득점 예측 모델 학습 및 평가

4-4-1. 지표 평가

- RMSE (Root Mean Square Error)
  - 평균 제곱근 오차
  - 추정 값 또는 모델이 예측한 값과 실제 환경에서 관찰되는 값의 차이를 다룰 때 사용함

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(pred_i - target_i)^2}$$

- MSE (Mean Square Error)
  - 예측 값과 실제 값 사이의 오차의 제곱의 평균

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(pred_i - target_i)^2$$

- MAE (Mean Absolute Error)
  - 예측 값과 실제 값 사이의 절대 오차의 평균

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|pred_i - target_i|$$

- R2 (R- Squared)
  - 예측 값이 실제 값에 얼마나 잘 맞는지를 나타내는 지표로, 설명된 분산의 비율

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(target_i - pred_i)^2}{\sum_{i=1}^{N}(target_i - \bar{target})^2}$$

4-4-2. 모델 선정 및 학습

- 데이터 전처리 수행

```
# 모델 생성
df = pd.read_csv('/content/drive/MyDrive/DATATHON/data/@KBO_5개구단/KBO

df = df.drop(columns=['선수명', '년도'])

# '타율' 열에서 '-' 값을 가진 행을 제거
df = df[df['타율'] != '-']

# 나머지 열에서 '-' 값을 0으로 변경
df.replace('-', 0, inplace=True)

# 특징과 타겟 설정
features = ['출루율', '장타율', '홈런', '수비 승리 기여도', '득점권타율', '도루
target = '추정득점'

X = df[features]
y = df[target]

# 데이터 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0
```

- 관련 연구자료에서 언급된 모델 대상으로 실험

```
# 모델 목록
models = {
    'XGB': XGBRegressor(),
    'SVR': SVR(),
    'Ridge': Ridge(),
    'RandomForest': RandomForestRegressor(random_state=42),
    'GradientBoosting': GradientBoostingRegressor(),
    'KNN': KNeighborsRegressor(),
    'AdaBoost': AdaBoostRegressor(random_state=42),
    'LGBM': LGBMRegressor(),
    'LinearRegression': LinearRegression(),
    'Lasso': Lasso()
}

# 각 모델에 대한 예측 및 결과 출력
```

```
results = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)

    # MSE, RMSE, MAE, R2 계산
    mse = mean_squared_error(y_test, predictions)
    rmse = mean_squared_error(y_test, predictions, squared=False)
    mae = mean_absolute_error(y_test, predictions)
    r2 = r2_score(y_test, predictions)

    results[name] = {
        'MSE': mse,
        'RMSE': rmse,
        'MAE': mae,
        'R2 Score': r2
    }

results_df = pd.DataFrame(results).T

results_df_sorted = results_df.sort_values
```

모델	MSE	RMSE	MAE	R2
XGB	294.06	17.15	11.77	0.69
SVR	992.15	31.50	19.78	-0.05
Ridge	591.58	24.32	17.93	0.38
GradientBoosting	228.05	15.10	9.85	0.76
RandomForest	223.41	14.95	9.44	0.76
KNN	506.76	22.51	14.52	0.46
AdaBoost	722.84	26.89	19.14	0.24
LGBM	226.66	15.06	10.06	0.76
LinearRegression	581.90	24.12	17.96	0.39
Lasso	589.28	24.28	18.07	0.38

- Random Search & Grid Search → RandomForest 최적의 파라미터 값 찾기

```
# 랜덤 포레스트 파라미터 그리드 설정
param_grid = {
    'n_estimators': [50, 100, 150, 200],
    'max_depth': [10, 20, 30, 40],
    'max_features': [0.3, 0.5, 0.7, 1.0],
}

# 랜덤 포레스트 모델
model = RandomForestRegressor(random_state=42)

# 랜덤 서치 수행
random_search = RandomizedSearchCV(model, param_distributions=param_g
random_search.fit(X_train, y_train)
best_params_random = random_search.best_params_

# 그리드 서치 수행
grid_search = GridSearchCV(model, param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
predictions = grid_search.predict(X_test)
mae = mean_absolute_error(y_test, predictions)
mse = mean_squared_error(y_test, predictions)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, predictions)

# 결과 저장
results = {
    'Best Params': grid_search.best_params_,
    'MAE': mae,
    'MSE': mse,
    'RMSE': rmse,
    'R2': r2
}
```

```
Best Params: {'n_estimators': 50, 'max_features': 0.3, 'max_depth': 20}
MAE: 9.642742290574695
MSE: 225.52818014843683
RMSE: 15.017595684677252
R2: 0.7617872841897814
```

- RandomForest 모델 학습 및 라인업 추정득점 예측

```
# 모델 생성
df = pd.read_csv('/content/drive/MyDrive/DATATHON/data/@KBO_5개구단/KBO

df = df.drop(columns=['선수명', '년도'])

# '타율' 열에서 '-' 값을 가진 행을 제거
df = df[df['타율'] != '-']

# 나머지 열에서 '-' 값을 0으로 변경
df.replace('-', 0, inplace=True)

# 특징과 타겟 설정
features = ['출루율', '장타율', '홈런', '수비 승리 기여도', '득점권타율', '도루
target = '추정득점'

X = df[features]
y = df[target]

# 데이터 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

# 모델 학습
model = RandomForestRegressor()
model.fit(X_train, y_train)

# 모든 df_1부터 df_144까지의 라인업에 대한 추정득점 예측
predicted_scores = {}

for i in range(1, 145):
    lineup_df = globals()[f'df_{i}']
    lineup_features = lineup_df[features]
    predicted_score = model.predict(lineup_features)
    total_predicted_score = predicted_score.mean()  # 각 선수의 예측 점수
    predicted_scores[f'df_{i}'] = total_predicted_score
```

5. 최종 평가

5-1. 키움 히어로즈 팀 선정 이유

- 2024년 6월 24일 기준, 키움 히어로즈 팀이 10위에 머물러 있음. 성적을 올리기 위해 라인업 개선 방향을 제시하고자 함

5-2. 타자 선발 기준

기존라인업 선발

- 상위타선 : 출루율, 주루 능력
- 중심타선 : 장타율
- 하위타선 : 출루율

최적라인업 (퍼펙트게임 라인업) 선발

- 상위타선 : 출루율, 도루, 장타율
- 중심타선 : 장타율, 득점권타율, 홈런
- 하위타선 : 출루율, 수비 능력

5-3. 기준 라인업 추정득점 예측

- KBO 공식 사이트에서 기준 라인업 확인 (2024.06.24 기준)
  - 타순별 타수가 가장 높은선수로 구성 하였음
- 키움 히어로즈

```
# 기준 라인업의 선수와 타순을 기준으로 데이터를 필터링하여 standard_df 생성
standard_lineup = [
    ('이용규', '1번'),
    ('도슨'  , '2번'),
    ('김혜성', '3번'),
    ('이주형', '4번'),
    ('최주환', '5번'),
    ('송성문', '6번'),
    ('변상권', '7번'),
    ('김재현', '8번'),
    ('이재상', '9번')
]

standard_df = pd.DataFrame()

for player, order in standard_lineup:
    filtered_data = df2[(df2['선수명'] == player) & (df2['타순'] == ord
    standard_df = pd.concat([standard_df, filtered_data])
```

```
standard_df.reset_index(drop=True, inplace=True)

X = df[features]
y = df[target]

# 데이터 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

# 모델 학습
model = RandomForestRegressor(n_estimators=50, max_depth=20, max_feat
model.fit(X_train, y_train)

# standard_df에 대한 추정득점 예측
standard_features = standard_df[features]
predicted_score = model.predict(standard_features)
total_predicted_score = predicted_score.mean()  # 각 선수의 예측 점수의 평

print(f'standard_df에 대한 총 추정득점: {total_predicted_score}')
```

```
standard_df에 대한 총 추정득점: 29.110666666666663
```

	선수명	팀명	타율	타수	안타	2루타	3루타	홈런	타점	볼넷	...	도루허용	도루저지	고의4구	희생플라이	희생번트	수비	승리	기여도	득점권타율	추정득점	출루율	장타율
0	이용규	키움	0.304	92	28	3	0	1	5	14	...	2	1	0	1	3			0.040	0.182	15.636	0.409	0.370
1	도슨	키움	0.356	284	101	23	2	9	37	25	...	2	2	0	1	0			-0.145	0.291	55.322	0.412	0.546
2	김혜성	키움	0.348	247	86	13	2	9	39	29	...	18	4	1	3	0			1.088	0.383	53.462	0.420	0.526
3	이주형	키움	0.225	80	18	2	0	3	14	8	...	0	0	1	0	0			0.017	0.220	9.426	0.303	0.362
4	최주환	키움	0.228	79	18	2	0	0	9	5	...	0	1	1	4	0			-0.128	0.213	6.200	0.270	0.253
5	송성문	키움	0.469	32	15	3	0	3	10	5	...	2	0	0	7	0			0.236	0.333	14.392	0.467	0.844
6	변상권	키움	0.294	34	10	1	0	0	1	4	...	0	0	0	1	0			0.186	0.250	4.348	0.359	0.324
7	김재현	키움	0.246	114	28	4	0	0	11	4	...	0	0	0	3	6			0.353	0.255	9.848	0.299	0.281
8	이재상	키움	0.220	41	9	0	0	1	4	0	...	0	0	0	0	1			-0.537	0.143	2.480	0.220	0.293

5-4. 최적 라인업 추정득점 예측

- 키움 히어로즈

```
# 모델 생성
df = pd.read_csv('/content/drive/MyDrive/DATATHON/data/@KBO_5개구단/KBO

df = df.drop(columns=['선수명', '년도'])

# '타율' 열에서 '-' 값을 가진 행을 제거
df = df[df['타율'] != '-']
```

```
# 나머지 열에서 '-' 값을 0으로 변경
df.replace('-', 0, inplace=True)

# 특징과 타겟 설정
features = ['출루율', '장타율', '홈런', '수비 승리 기여도', '득점권타율', '도루
target = '추정득점'

X = df[features]
y = df[target]

# 데이터 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

# 모델 학습
model = RandomForestRegressor(n_estimators=50, max_depth=20, max_feat
model.fit(X_train, y_train)

# 모든 df_1부터 df_144까지의 라인업에 대한 추정득점 예측
predicted_scores = {}

for i in range(1, 145):
    lineup_df = globals()[f'df_{i}']
    lineup_features = lineup_df[features]
    predicted_score = model.predict(lineup_features)
    total_predicted_score = predicted_score.mean()  # 각 선수의 예측 점수
    predicted_scores[f'df_{i}'] = total_predicted_score
```

```
가장 높은 점수를 받은 라인업: df_38 with score 38.52511111111111
79.20599999999999
```

	선수명	팀명	타율	타수	안타	2루타	3루타	홈런	타점	볼넷	...	도루저지	고의4구	희생플라이	희생번트	수비 승리 기여도	득점권타율	추정득점	출루율	장타율	포지션
1	이용규	키움	0.304	92	28	3	0	1	5	14	...	1	0	1	3	0.040	0.182	15.636	0.409	0.370	좌익수, 우익수, 중견수
2	이정중	키움	0.252	30	7	1	0	2	7	7	...	0	0	0	0	-0.017	0.450	6.852	0.400	0.500	우익수
3	김혜성	키움	0.348	247	86	13	2	9	39	29	...	4	1	3	0	1.088	0.383	53.462	0.420	0.526	2루수
4	송성문	키움	0.385	78	30	4	1	2	21	12	...	0	0	7	0	0.236	0.333	20.938	0.433	0.538	3루수, 2루수, 1루수
5	도슨	키움	0.356	284	101	23	2	9	37	25	...	2	0	1	0	-0.145	0.291	55.322	0.412	0.546	좌익수, 중견수
6	고영우	키움	0.421	19	8	2	0	0	0	1	...	0	0	2	1	-0.156	0.382	4.522	0.409	0.526	3루수, 유격수, 2루수
7	김건희	키움	0.333	12	4	1	0	0	1	3	...	0	0	1	0	0.146	0.368	2.874	0.438	0.417	포수
8	이주형	키움	0.280	92	25	3	0	2	11	10	...	0	1	0	0	0.017	0.220	14.258	0.361	0.407	우익수, 중견수
9	최주환	키움	0.201	91	18	2	0	2	11	7	...	1	1	4	0	-0.128	0.213	7.917	0.249	0.294	1루수

5-5. 기준 라인업 추정득점 VS 최적 라인업 추정득점

- 키움 히어로즈 추정득점 비교 결과

- LG 트윈스 추정득점 비교 결과

5-5. 결론

하위권 팀인 키움 히어로즈의 경우, 우리의 모델을 적용했을 때 기준 라인업에 비해 득점이 약 9.4점 향상됨 즉, 팀 성적 향상에 큰 도움이 될 수 있음

반면 상위권 팀인 LG 트윈스는 이미 최적화된 라인업을 갖추고 있어 득점 차이가 상대적으로 적은 약 0.1점의 득점 향상이 있었음

6. 스토리텔링

6-1. 문제원인 리뷰

프로젝트의 출발점은 코칭스태프의 직관에 의존하는 기존 타순 결정 방식의 한계에서 비롯됨

경기에서의 타순 결정은 팀 성적에 큰 영향을 미치지만, 기존 방식은 과학적 데이터에 기반하지 않아 최적의 타순을 찾기 어려움

이를 해결하기 위해 데이터 분석을 통해 최적의 타순을 도출하고자 함

6-2. 분석결과 요약

우리의 모델은 각 타자의 주요 지표를 표준화하고, 이 지표들에 가중치를 부여하여 최적의 타순을 제안하며 이를 통해 팀의 득점력을 최대한으로 끌어올릴 수 있음

특히, 키움 히어로즈와 LG 트윈스 팀을 예로 들며 기준 라인업과 최적 라인업의 추정 득점을 비교한 결과, 다음과 같은 결론을 도출할 수 있었음.

- **키움 히어로즈**
  - 기준 라인업 추정 득점: 29.11점
  - 최적 라인업 추정 득점: 38.53점
  - 득점 차이: 약 9.4점

- **LG 트윈스**
  - 기준 라인업 추정 득점: 52.83점
  - 최적 라인업 추정 득점: 52.93점
  - 득점 차이: 약 0.1점

6-3. 인사이트 제안

- **데이터 기반 의사결정**
  - 코칭스태프의 직관에 의존하는 기존의 타순 결정 방식을 벗어나, 데이터에 기반한 과학적인 접근 방식을 채택함으로써 보다 신뢰성 있는 결과를 도출할 수 있음.

- **더 큰 득점 개선 가능성**
    - 하위권 팀에게는 더욱 큰 득점 향상을 제공하여 성적을 개선할 수 있으며, 상위권 팀에게도 유의미한 득점 증가를 가져올 수 있음.

- **전략적 활용**
    - 우리 모델은 특정 경기 상황에 맞추어 전략적으로 타순을 조정할 수 있는 유연성을 가지며 상대 팀의 투수 및 수비 데이터를 분석하여 최적의 타순을 결정함으로써 경기에서 더 나은 성과를 기대할 수 있음.

7. 프로젝트 회고 및 개선점

7-1. Q&A

7-2 회고

7-3. 개선점

7-4. 추후 개선 계획

8. 업무 및 일정 관리

8-1. 업무분장

8-2. 프로젝트 일정표

9. 부록

9-1. 참고자료

▼ 타순별 통계와 기계학습의 회귀 모델을 활용한 한국 프로야구에서 팀 득점력 개선

KCI_FI003024370 (1).pdf

▼ MARKOV CHAIN을 이용한 한국프로야구 분석 및 최적 타순 계산을 위한 모델 연구

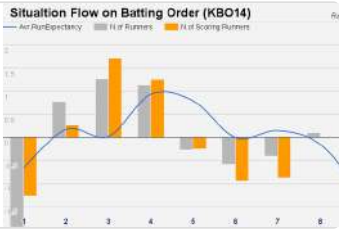
MARKOV_CHAIN_ (1).pdf

▼ 최적 타순에 관한 통계적기준 - 세이버메트릭스 가이드

"최적 타순"에 관한 통계적 기준 - 라인업 놀이를 위한 세이버메트릭스 가이드

야구의 매력 중 상당부분이 “전략게임”에 있는 바, 내 맘대로 감독 노릇하며 즐기는 라인업 놀이야 말로 “팬질” 중의 “팬질”이라 할 수 있겠습니다. 거기에 방대한 야구통계에 기반한 각종 데이터를 첨가하면 금상첨화라 할 수 있겠네요. 세이버메트릭스 또는 최근의

https://baseball-in-play.com/220



▼ 라인업, 타순의 미학

[2020년 9월호] Baseball Team Fight Tactic – 라인업, 타순의 미학

Baseball Team Fight Tactic – 라인업, 타순의 미학 [SPORTSKU=글 천유진 기자, 사진 김...

https://blog.naver.com/sportsku/222090853729