Kbo FA
선수별 FA 금액 예측

디자인공학전공 - 4조

20191043 정민규
20190294 김주환
20190356 김호성
20191019 전민욱
20191182 최재준

CONTENTS 1

주제 선정

- 배경
- 목적

CONTENTS 2

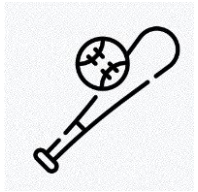
데이터

- 수집

CONTENTS 3

모델 정의

- 상관관계
- 데이터 전처리
- 알고리즘
- 알고리즘 설계
- 테스트



CONTENTS 4

데이터 분석

- 예측결과

CONTENTS 5

결론 및 제언

- 결과
- 한계점

주제 선정 - 배경


동아일보 | 오피니언

‘몸값 쏠림’ 프로야구 FA제도… 계약시 보수 상한제 등 개선책 필요[인사이드&인사이트]

강동웅 스포츠부 기자

입력 2023-03-13 03:00 | 업데이트 2023-03-13 04:39

읽기모드 가

빈익빈 부익부 FA 제도

WBC 1·2차전 졸전으로 FA 몸값 거품 논란 재점화

A등급 FA만 몸값 치솟아… 찬밥 C등급 강제 은퇴 우려도

NBA,‘맥시멈 계약’ 조건 명시… 자격 따라 상한액 미리 정해놔

2023 하동세계 茶 엑스포

WORLD TEA EXPO 2023 HADONG, KOREA

성원에 감사드립니다.


이대호, 롯데와 4년 총액 150억 FA 계약 “팀 동료 후배들과 우승하는 것이 마지막 소원”

머니투데이방송 백승기 기자

입력 2017.01.24 09:39:33

가

“실력에 비해 연봉을 너무 많이 받는 것 아니냐’ 하는 지적이 나온다.”

“배가 너무 불렀어... 대충해도 FA 때 돈 많이 주니까 실력이 안 느는 것 같다.”

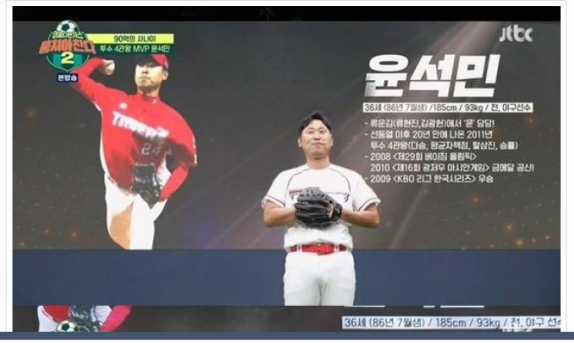
주제 선정 - 목적

홈 > 연예 > 방송 · TV

윤석민 "90억 먹튀? 큰 계약 후 못했기 때문...겸허히 받아들여"

(서울=뉴스1) 고승아 기자 | 2021-08-15 21:17 송고

댓글 가



[김수인의 직격 야구] 'FA 먹튀' 대책은 없는 걸까

권정식 | 입력 2022.07.04 10:20 | 댓글 0

삼성 투수 박정현이 역투하고 있다.

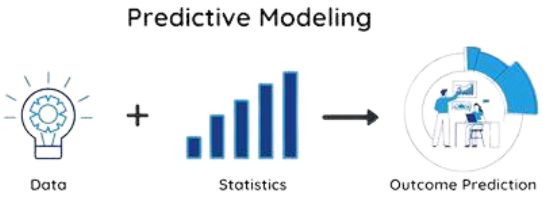
지난해 14승(5패)을 거둬 2007년 데뷔후 팀의 에이스로 우뚝 선 삼성 백정현(35). 시즌후 4년

야구

박동원 65억, 오버페이라 생각했다...이제 보니 '특급 혜자'였다

기사입력 2023-05-19 11:27:27

Google Colaboratory

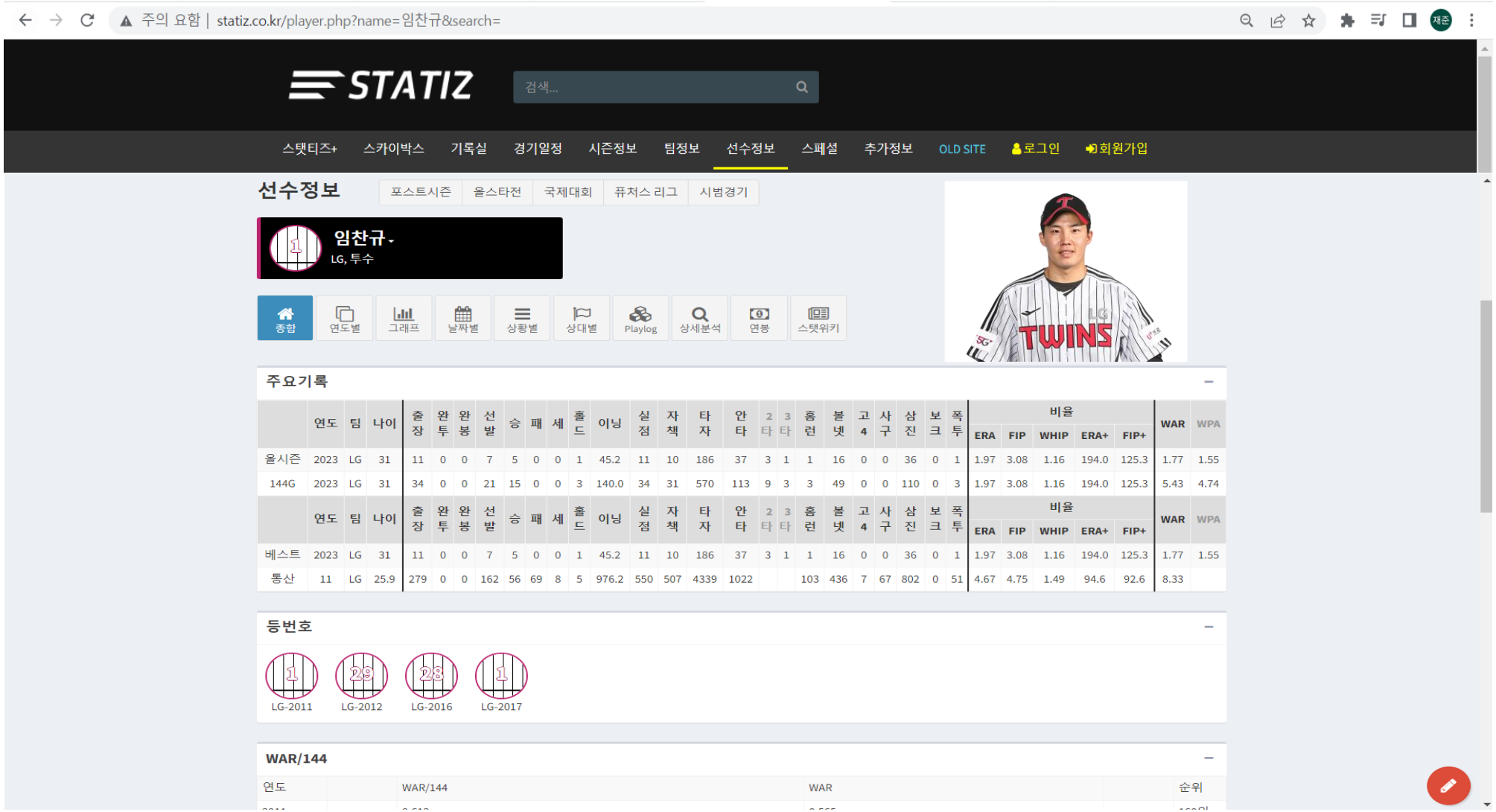


Q : Purpose of Model

A : 24년 KBO 선수들의 FA 연봉 예측

데이터 - 수집

데이터 추출 - STATIZ - 2014년에 설립된 데이터베이스 및 온라인정보 제공사



데이터 - 수집

타자 데이터 추출 인자 = 92개

1. '주요'지표 - 타율, 출루율, 장타율, OPS을 포함한 총 48개
2. '확장'지표 - , wOBA, wRC, wRC/27, wRAA, wRC+을 포함한 총 18개
3. '세부'지표 - 파크펙터 조정 wOBA , wRC, wRC/27, wRAA, wRC+을 포함한 총 26개

투수 데이터 추출 인자 = 72개

1. '주요'기록 - 승, 패, 홀드, 세이브, 이닝, ERA, FIP을 포함한 총 37개
2. '확장'기록 - K/9, BB/9, K/BB, WHIP을 포함한 총 16개
3. '세부'기록 - 파크펙터 조정 ERA, FIP, RA을 포함한 총 19개

모델 학습에 필요한 데이터

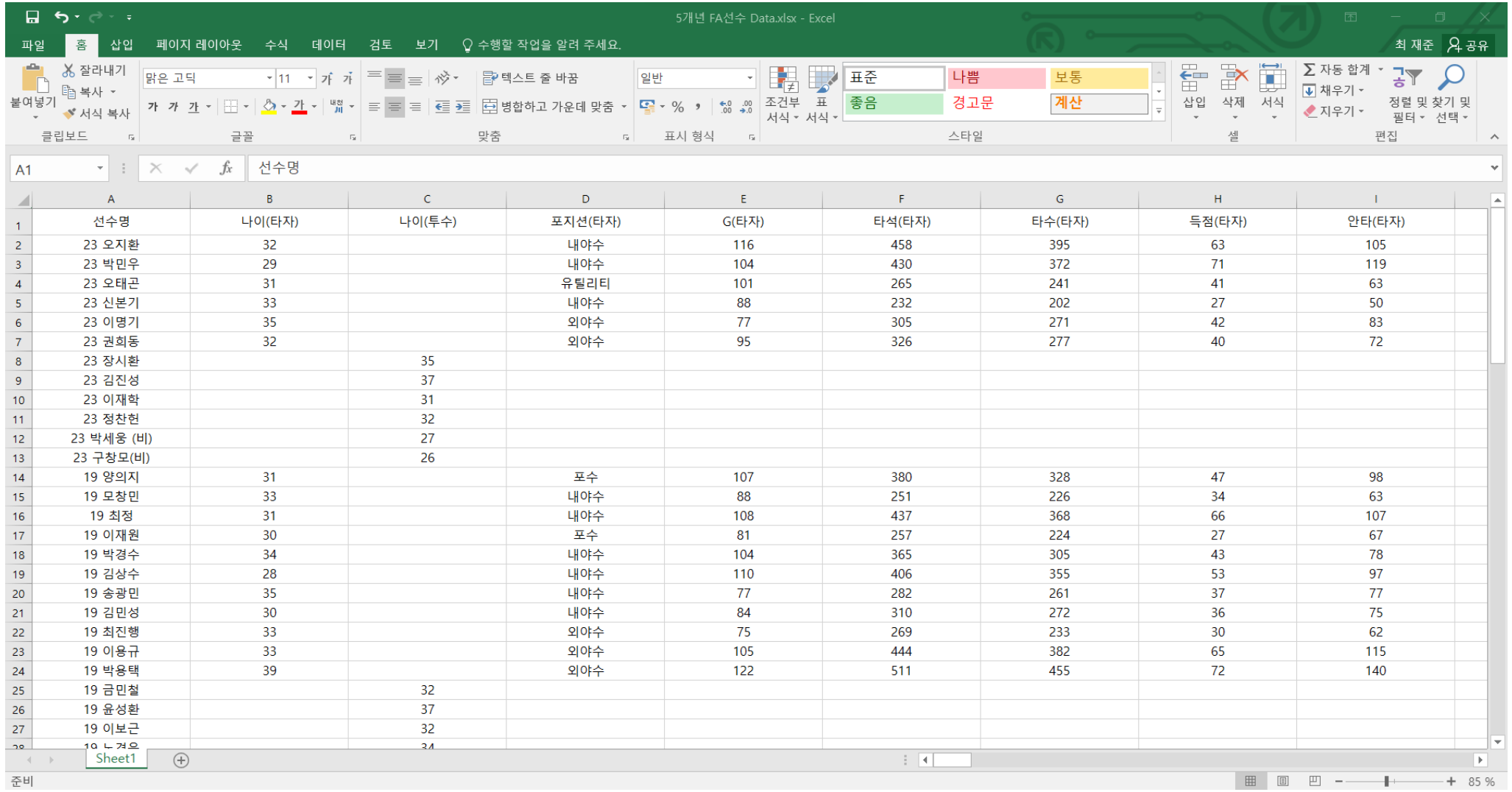
2019~2023년까지 FA권리를 행사한 실제 야구선수(83명) + 비FA계약(7명)을 맺은 총 90명의 기록지표

2019 두산 양의지, LG 박용택을 포함한 15명
2020 기아 안치홍, LG 오지환을 포함한 18명
2021 롯데 이대호, 두산 허경민을 포함한 15명
2022 LG 김현수, NC 나성범을 포함한 15명 + 비FA 삼성 구자욱을 포함한 4명
2023 NC 한현희, LG 유강남을 포함한 20명 + 비FA 롯데 박세웅을 포함한 3명

데이터 - 수집

2019~2022년 FA계약 맺은 선수 Data

Data Frame = 90*166



	A	B	C	D	E	F	G	H	I
1	선수명	나이(타자)	나이(투수)	포지션(타자)	G(타자)	타석(타자)	타수(타자)	득점(타자)	안타(타자)
2	23 오지환	32		내야수	116	458	395	63	105
3	23 박민우	29		내야수	104	430	372	71	119
4	23 오태곤	31		유틸리티	101	265	241	41	63
5	23 신본기	33		내야수	88	232	202	27	50
6	23 이명기	35		외야수	77	305	271	42	83
7	23 권희동	32		외야수	95	326	277	40	72
8	23 장시환		35						
9	23 김진성		37						
10	23 이재학		31						
11	23 정찬헌		32						
12	23 박세웅 (비)		27						
13	23 구창모(비)		26						
14	19 양의지	31		포수	107	380	328	47	98
15	19 모창민	33		내야수	88	251	226	34	63
16	19 최정	31		내야수	108	437	368	66	107
17	19 이재원	30		포수	81	257	224	27	67
18	19 박경수	34		내야수	104	365	305	43	78
19	19 김상수	28		내야수	110	406	355	53	97
20	19 송광민	35		내야수	77	282	261	37	77
21	19 김민성	30		내야수	84	310	272	36	75
22	19 최진행	33		외야수	75	269	233	30	62
23	19 이용규	33		외야수	105	444	382	65	115
24	19 박용택	39		외야수	122	511	455	72	140
25	19 금민철		32						
26	19 윤성환		37						
27	19 이보근		32						

데이터 - 수집

예측할 데이터 추출

2024 구단별 FA 취득 예정 선수명단

SSG LANDERS	KIWOOM HEROES	LG TWINS	kt wiz suwon	TIGERS	NC DINOS	SAMSUNG LIONS	LOTTE GIANTS	DOOSAN BEARS	HANWHA EAGLES
이재원	임창민	임찬규	김재윤	김선빈	심창민	김대우	윤명준	장원준	장민재
김민식	이지영	진해수	주권	고종욱		오승환	신정락	김강률	정우람
	이용규	함덕주	박경수			김태군	안치홍	홍건희	노수광
		서건창				강한울	전준우	양석환	
		김민성				김헌곤			

2024FA 선수들의 2022년 까지의 기록

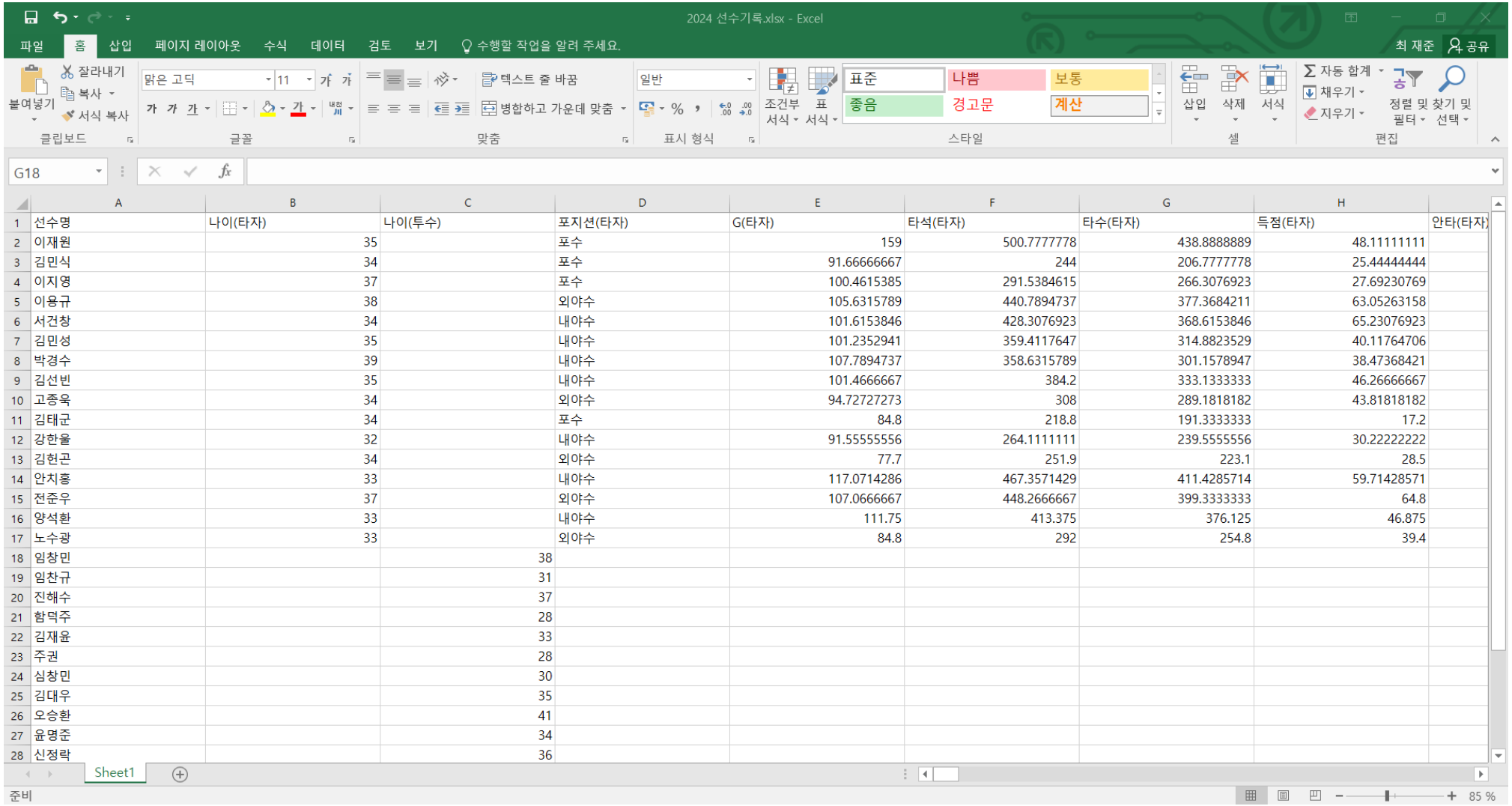
＋

2024FA 선수들의 2023년 현재까지의 기록을 한 시즌인 144경기를 기준으로 만든 데이터

데이터 - 수집

2024년 FA조건을 완성한 선수 Data

Data Frame = 32*166



2024 선수기록.xlsx - Excel

	A	B	C	D	E	F	G	H	
1	선수명	나이(타자)	나이(투수)	포지션(타자)	G(타자)	타석(타자)	타수(타자)	득점(타자)	안타(타자)
2	이재원	35		포수	159	500.7777778	438.8888889	48.11111111	
3	김민식	34		포수	91.66666667	244	206.7777778	25.44444444	
4	이지영	37		포수	100.4615385	291.5384615	266.3076923	27.69230769	
5	이용규	38		외야수	105.6315789	440.7894737	377.3684211	63.05263158	
6	서건창	34		내야수	101.6153846	428.3076923	368.6153846	65.23076923	
7	김민성	35		내야수	101.2352941	359.4117647	314.8823529	40.11764706	
8	박경수	39		내야수	107.7894737	358.6315789	301.1578947	38.47368421	
9	김선빈	35		내야수	101.4666667	384.2	333.1333333	46.26666667	
10	고종욱	34		외야수	94.72727273	308	289.1818182	43.81818182	
11	김태군	34		포수	84.8	218.8	191.3333333	17.2	
12	강한울	32		내야수	91.55555556	264.1111111	239.5555556	30.22222222	
13	김헌곤	34		외야수	77.7	251.9	223.1	28.5	
14	안치홍	33		내야수	117.0714286	467.3571429	411.4285714	59.71428571	
15	전준우	37		외야수	107.0666667	448.2666667	399.3333333	64.8	
16	양석환	33		내야수	111.75	413.375	376.125	46.875	
17	노수광	33		외야수	84.8	292	254.8	39.4	
18	임창민		38						
19	임찬규		31						
20	진해수		37						
21	함덕주		28						
22	김재윤		33						
23	주권		28						
24	심창민		30						
25	김대우		35						
26	오승환		41						
27	윤명준		34						
28	신정락		36						

Sheet1

준비

85 %

모델 정의 – Correlation(상관관계)

상관계수를 위한 데이터 전처리
타자 X인자, 투수 X인자, 연봉(Y인자) 분리

```
import numpy as np
import pandas as pd
from google.colab import drive
from sklearn.preprocessing import LabelEncoder

# 5개년 FA선수 데이터 불러오기
filename = '/content/5개년 FA선수 Data.xlsx'
df = pd.read_excel(filename)

#선수명 제거
df.drop('선수명', axis=1, inplace=True)

# 포지션(타자) 열 레이블 인코딩
label_encoder = LabelEncoder()
df['포지션(타자)'] = label_encoder.fit_transform(df['포지션(타자)'])

# 선수명과 포지션(타자) 열 데이터 유형을 숫자형으로 변환
df['포지션(타자)'] = df['포지션(타자)'].astype(int)

# 타자 데이터와 투수 데이터를 저장할 변수 초기화
타자_df = pd.DataFrame()
투수_df = pd.DataFrame()

# 열 이름을 확인하면서 "(타자)"를 포함하는 열은 타자 데이터로, "(투수)"를 포함하는 열은 투수 데이터로 분류
for column in df.columns:
    if "(타자)" in column:
        타자_df[column] = df[column]
    if "(투수)" in column:
        투수_df[column] = df[column]

# 연봉 열 넣기
타자_df["FA 연봉(억)"] = df["FA 연봉(억)"]
투수_df["FA 연봉(억)"] = df["FA 연봉(억)"]

#결측치 제거
타자_df = 타자_df.dropna().reset_index(drop=True) #재정렬
투수_df = 투수_df.dropna().reset_index(drop=True)

# "FA 연봉(억)" 열 제거 및 따로 정의
타자_salary = 타자_df.pop("FA 연봉(억)") #이제 df_removed에는 연봉이 제거됨
투수_salary = 투수_df.pop("FA 연봉(억)")
```

타자 X인자와 타자 연봉 상관관계

```
import pandas as pd
# X값과 y값 간의 상관 관계 계산
타자correlation_matrix = 타자_df.corrwith(타자_salary)

# 상관 관계 출력
print(타자correlation_matrix)
```

투수 X인자와 투수 연봉 상관관계

```
import pandas as pd
# X값과 y값 간의 상관 관계 계산
투수correlation_matrix = 투수_df.corrwith(투수_salary)

# 상관 관계 출력
print(투수correlation_matrix)
```

모델 정의 – Correlation(상관관계)

타자 Correlation

명칭	지표
RAA(타자)	0.773412666
RAR(타자)	0.728549929
WAR*(타자)	0.709344339
장타(타자)	0.666545859
WAA(타자)	0.6645754
파크팩터 조정 wRC(타자)	0.658656651
RAA-Adj(타자)	0.657293546
타점(타자)	0.650139769
루타(타자)	0.646107434
2루타(타자)	0.639518653
공격 RAA 종합(타자)	0.637928465
파크팩터 조정 wRAA(타자)	0.622920047
공격RAA 타격(타자)	0.618143802
wRC(타자)	0.609508391
WAR Bat(타자)	0.603984167
WPA(타자)	0.596685131
XH/AB(타자)	0.58585281
wRAA(타자)	0.585053834
안타(타자)	0.58014446
장타율(타자)	0.575532792
파크팩터 조정 wRC/27(타자)	0.568949263
희비(타자)	0.56350678
홈런(타자)	0.560028219
득점(타자)	0.555520575
타석(타자)	0.55108159
타수(타자)	0.549872121
wRC/27(타자)	0.546551447
IsoP(타자)	0.537594306
OPS(타자)	0.521376354
파크팩터 조정 wOBA(타자)	0.514073271

투수 Correlation

명칭	지표
fWAR(투수)	0.74128944
RAR 종합(투수)	0.725365387
WAR(투수)	0.714490529
탈삼진(투수)	0.70431935
패(투수)	0.7042098
CVP(투수)	0.698639938
RA9-WAR(투수)	0.684544704
승(투수)	0.671672017
RAA 종합(투수)	0.667747864
폭투(투수)	0.664566315
선발WAR(투수)	0.65869957
이닝(투수)	0.652509996
대체 Run 종합(투수)	0.641322541
상대타자(투수)	0.635611442
보크(투수)	0.634408161
피안타(투수)	0.608871041
IP/G(투수)	0.595931881
선발(투수)	0.587190079
K/BB(투수)	0.584991038
WAA 종합(투수)	0.573573929
피홈런(투수)	0.571424591
ERA+(투수)	0.565423353
파크펙터조정 ERA+(투수)	0.565244754
K-BB%(투수)	0.549811882
자책(투수)	0.534619792
실점(투수)	0.531862911

Correlaion 분석결과

전통적인 기록보다는
세이버매트릭스 관련 지표들이 높은
Correlation을 가지는 것을 볼 수 있다.

세이버매트릭스(sabermetrics)란?

1971년 8월 밥 데이비스가 창시한
SABR라는 모임에서 만들어진
야구를 통계학적/수학적으로 분석하는 방법론

모델 정의 – 데이터 전처리

```
import numpy as np
import pandas as pd
from google.colab import drive
from sklearn.preprocessing import LabelEncoder

# 5개년 FA선수 데이터 불러오기
filename = '/content/5개년 FA선수 Data.xlsx'
df = pd.read_excel(filename)

#선수명 제거
df.drop('선수명', axis=1, inplace=True)

# 포지션(타자) 열 레이블 인코딩
label_encoder = LabelEncoder()
df['포지션(타자)'] = label_encoder.fit_transform(df['포지션(타자)'])

# 선수명과 포지션(타자) 열 데이터 유형을 숫자형으로 변환
df['포지션(타자)'] = df['포지션(타자)'].astype(int)

# 타자 데이터와 투수 데이터를 저장할 변수 초기화
타자_df = pd.DataFrame()
투수_df = pd.DataFrame()

# 열 이름을 확인하면서 "(타자)"를 포함하는 열은 타자 데이터로, "(투수)"를 포함하는 열은 투수 데이터로 분류
for column in df.columns:
    if "(타자)" in column:
        타자_df[column] = df[column]
    if "(투수)" in column:
        투수_df[column] = df[column]

# 연봉 열 넣기
타자_df["FA 연봉(억)"] = df["FA 연봉(억)"]

#결측치 제거
타자_df = 타자_df.dropna().reset_index(drop=True) #재정렬

#X인자 재설정
타자_df_1 = 타자_df[['타석(타자)', '타수(타자)', '득점(타자)', '안타(타자)', '2루타(타자)', '홈런(타자)', '루타(타자)', '타점(타자)', '희비(타자)', '장타율(타자)', 'OPS(타자)',
'wRC+(타자)', 'WAR*(타자)', 'WPA(타자)', 'IsoP(타자)', 'wRC(타자)', 'wRC/27(타자)', 'wRAA(타자)', '파크팩터 조정 wOBA(타자)', '파크팩터 조정 wRC(타자)', '파크팩터 조정 wRC/27(타자)',
'파크팩터 조정 wRAA(타자)', '파크팩터 조정 wRC+(타자)', '공격RAA 타격(타자)', '공격 RAA 종합(타자)', 'RAA-Adj(타자)', 'RAA(타자)', 'RAR(타자)', 'WAA(타자)', 'WAR Bat(타자)',
'장타(타자)', 'XH/AB(타자)', 'FA 연봉(억)']]

#이대호 행 분리
이대호_row = 타자_df_1.loc[38]
타자_df_1 = 타자_df_1.drop(38)

# 이대호 행 "FA 연봉(억)" 열 삭제
이대호_row.drop("FA 연봉(억)", inplace=True)

# "FA 연봉(억)" 열 제거 및 따로 정의
타자_salary = 타자_df_1.pop("FA 연봉(억)") #이제 df_removed에는 연봉이 제거됨
투수_salary = 투수_df.pop("FA 연봉(억)")
```

5개년 데이터를 투타자를 나눠서 데이터를 전처리

모델 학습 시 사용 X인자

상관계수 크기 ≥ 0.5

모델 정의 - 알고리즘

타자 모델

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
X = 타자_df_1
y = 타자_salary
# 테스트 세트 분할
train_input, test_input, train_target, test_target = train_test_split(X, y, test_size=0.3, random_state=42)

# 최적의 CV 값을 탐색하기 위한 범위 설정
cv_values = [3, 5, 7, 10]
best_cv = None
best_score = -1
for cv in cv_values:
    # GridSearchCV를 사용하여 파라미터 탐색
    param_grid = {
        'n_estimators': [100, 200, 300],
        'max_depth': [None, 10, 20],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }
    grid_search = GridSearchCV(RandomForestRegressor(random_state=42), param_grid, cv=cv)
    grid_search.fit(train_input, train_target)

    # GridSearchCV의 최고 점수와 파라미터 조합 확인
    score = grid_search.best_score_
    print(f"CV={cv}: 최고 점수={score}, 최적의 파라미터 조합={grid_search.best_params_}")

    # 현재 CV 값의 점수가 더 높으면 최적의 CV 값 및 점수 업데이트
    if score > best_score:
        best_cv = cv
        best_score = score

print("최적의 CV 값:", best_cv)

# 최적의 CV 값을 기반으로 파라미터 탐색
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# GridSearchCV를 사용하여 파라미터 탐색
grid_search = GridSearchCV(RandomForestRegressor(random_state=42), param_grid, cv=best_cv)
grid_search.fit(train_input, train_target)

# 최적의 파라미터 조합 출력
best_params = grid_search.best_params_
print("최적의 파라미터 조합:", best_params)

# 최적의 파라미터로 모델 생성
best_model = grid_search.best_estimator_
```

투수 모델

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
X = 투수_df_1
y = 투수_salary
# 테스트 세트 분할
train_input, test_input, train_target, test_target = train_test_split(X, y, test_size=0.3, random_state=42)

# 최적의 CV 값을 탐색하기 위한 범위 설정
cv_values = [3, 5, 7, 10]
best_cv = None
best_score = -1
for cv in cv_values:
    # GridSearchCV를 사용하여 파라미터 탐색
    param_grid = {
        'n_estimators': [100, 200, 300],
        'max_depth': [None, 10, 20],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }
    grid_search = GridSearchCV(RandomForestRegressor(random_state=42), param_grid, cv=cv)
    grid_search.fit(train_input, train_target)

    # GridSearchCV의 최고 점수와 파라미터 조합 확인
    score = grid_search.best_score_
    print(f"CV={cv}: 최고 점수={score}, 최적의 파라미터 조합={grid_search.best_params_}")

    # 현재 CV 값의 점수가 더 높으면 최적의 CV 값 및 점수 업데이트
    if score > best_score:
        best_cv = cv
        best_score = score

print("최적의 CV 값:", best_cv)

# 최적의 CV 값을 기반으로 파라미터 탐색
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# GridSearchCV를 사용하여 파라미터 탐색
grid_search = GridSearchCV(RandomForestRegressor(random_state=42), param_grid, cv=best_cv)
grid_search.fit(train_input, train_target)

# 최적의 파라미터 조합 출력
best_params = grid_search.best_params_
print("최적의 파라미터 조합:", best_params)

# 최적의 파라미터로 모델 생성
best_model = grid_search.best_estimator_
```

사용 알고리즘

RandomForestRegressor

최적의 하이퍼파라미터값

1. n_estimators
2. max_depth
3. min_samples_split
4. min_samples_leaf

위의 4가지 값을 찾아 모델 생성

파라미터 값을 찾을 때 GridSearchCV 사용

모델 정의 – 알고리즘 설계 & 테스트

타자 모델

```python
import matplotlib.pyplot as plt

# 테스트 데이터에 대한 예측
predictions = best_model.predict(test_input)

# 정확도 평가
accuracy = best_model.score(test_input, test_target)
print("정확도:", accuracy)

# 예측 결과와 실제 답 사이의 오차 계산
errors = test_target - predictions

# 평균 절대 오차(MAE) 계산
mae = np.mean(np.abs(errors))

# 평균 제곱 오차(MSE) 계산
mse = np.mean(errors ** 2)

# 평균 제곱근 오차(RMSE) 계산
rmse = np.sqrt(mse)

# 결정 계수(R2 score) 계산
r2_score = best_model.score(test_input, test_target)

# 통계적 수치 출력
print("평균 절대 오차(MAE):", mae)
print("평균 제곱 오차(MSE):", mse)
print("평균 제곱근 오차(RMSE):", rmse)
print("결정 계수(R2 score):", r2_score)

# 예측 오차 그래프
plt.scatter(predictions, errors)
plt.xlabel("Predictions")
plt.ylabel("Errors")
plt.title("Prediction Errors")
plt.show()
```

투수 모델

```python
import matplotlib.pyplot as plt

# 테스트 데이터에 대한 예측
predictions = best_model.predict(test_input)

# 정확도 평가
accuracy = best_model.score(test_input, test_target)
print("정확도:", accuracy)

# 예측 결과와 실제 답 사이의 오차 계산
errors = test_target - predictions

# 평균 절대 오차(MAE) 계산
mae = np.mean(np.abs(errors))

# 평균 제곱 오차(MSE) 계산
mse = np.mean(errors ** 2)

# 평균 제곱근 오차(RMSE) 계산
rmse = np.sqrt(mse)

# 결정 계수(R2 score) 계산
r2_score = best_model.score(test_input, test_target)

# 통계적 수치 출력
print("평균 절대 오차(MAE):", mae)
print("평균 제곱 오차(MSE):", mse)
print("평균 제곱근 오차(RMSE):", rmse)
print("결정 계수(R2 score):", r2_score)

# 예측 오차 그래프
plt.scatter(predictions, errors)
plt.xlabel("Predictions")
plt.ylabel("Errors")
plt.title("Prediction Errors")
plt.show()
```

모델 테스트

```python
이대호_row = 이대호_row.values.reshape(1, -1)
이대호_predictions = best_model.predict(이대호_row)

# 예측 결과 출력
print("이대호 데이터 예측 결과:", 이대호_predictions)
```

```
이대호 데이터 예측 결과: [15.83373541]
```

앞에 만든 모델로 이대호 선수 데이터로 실제 계약한 연봉과 예측 연봉이 얼마나 다른지 보여줌

실제 21년도 이대호 선수 기록 데이터 적용

실제 21년도 이대호 선수는 2년 26억 계약했기에 개발한 예측 모델에서는 26/2=13억이 나와야 100% 일치

15.8천만원이 나왔으므로 82.1% 일치

데이터 분석 - 타자

```
# 2024 FA선수 데이터 불러오기
filename1 = '/content/2024 선수기록.xlsx'
df1 = pd.read_excel(filename1)

# 선수명 제거
df1.drop('선수명', axis=1, inplace=True)

# 포지션(타자) 열 레이블 인코딩
label_encoder = LabelEncoder()
df1['포지션(타자)'] = label_encoder.fit_transform(df1['포지션(타자)'])

# 선수명과 포지션(타자) 열 데이터 유형을 숫자형으로 변환
df1['포지션(타자)'] = df1['포지션(타자)'].astype(int)

# 타자 데이터와 투수 데이터를 저장할 변수 초기화
new타자_df = pd.DataFrame()
new투수_df = pd.DataFrame()

# 열 이름을 확인하면서 "(타자)"를 포함하는 열은 타자 데이터로, "(투수)"를 포함하는 열은 투수 데이터로 분류
for column in df1.columns:
    if "(타자)" in column:
        new타자_df[column] = df1[column]
    if "(투수)" in column:
        new투수_df[column] = df1[column]

# 결측치 제거
new타자_df_1 = new타자_df.dropna().reset_index(drop=True)  # 재정렬
new투수_df_1 = new투수_df.dropna().reset_index(drop=True)  # 재정렬

#X인자 재설정
new_FA_df = new타자_df_1[['타석(타자)', '타수(타자)', '득점(타자)', '안타(타자)', '2루타(타자)', '홈런(타자)', '루타(타자)', '타점(타자)', '희비(타자)', '장타율(타자)', 'OPS(타자)',
'wRC+(타자)', 'WAR*(타자)', 'WPA(타자)', 'IsoP(타자)', 'wRC(타자)', 'wRC/27(타자)', 'wRAA(타자)', '파크팩터 조정 wOBA(타자)', '파크팩터 조정 wRC(타자)', '파크팩터 조정 wRC/27(타자)',
'파크팩터 조정 wRAA(타자)', '파크팩터 조정 wRC+(타자)', '공격RAA 타격(타자)', '공격 RAA 종합(타자)', 'RAA-Adj(타자)', 'RAA(타자)', 'RAR(타자)', 'WAA(타자)', 'WAR Bat(타자)',
'장타(타자)', 'XH/AB(타자)']]

# 테스트 데이터에 대한 예측 수행
new_FA_predictions = best_model.predict(new_FA_df)

# 예측 결과 출력
print("2024년도 데이터 예측 결과:", new_FA_predictions)
```

24년 데이터 전처리 후
타자 모델로 24년 타자 FA 선수 금액 예측

타자 총액

```
8.575886971687725
4.417194159674029
4.635469206849077
9.820632527657523
11.936947442930158
8.065947758018838
9.044612483062311
10.669431751697445
5.358284342391639
4.339985417120288
4.326614491194362
4.534659639472009
13.086199076974216
12.748619220188841
8.100524381438992
4.5636459862733565
```

데이터 분석 - 투수

```
# 2024 FA선수 데이터 불러오기
filename1 = '/content/2024 선수기록.xlsx'
df1 = pd.read_excel(filename1)

# 선수명 제거
df1.drop('선수명', axis=1, inplace=True)

# 포지션(타자) 열 레이블 인코딩
label_encoder = LabelEncoder()
df1['포지션(타자)'] = label_encoder.fit_transform(df1['포지션(타자)'])

# 선수명과 포지션(타자) 열 데이터 유형을 숫자형으로 변환
df1['포지션(타자)'] = df1['포지션(타자)'].astype(int)

# 타자 데이터와 투수 데이터를 저장할 변수 초기화
new타자_df = pd.DataFrame()
new투수_df = pd.DataFrame()

# 열 이름을 확인하면서 "(타자)"를 포함하는 열은 타자 데이터로, "(투수)"를 포함하는 열은 투수 데이터로 분류
for column in df1.columns:
    if "(타자)" in column:
        new타자_df[column] = df1[column]
    if "(투수)" in column:
        new투수_df[column] = df1[column]

# 결측치 제거
new타자_df_1 = new타자_df.dropna().reset_index(drop=True)  # 재정렬
new투수_df_1 = new투수_df.dropna().reset_index(drop=True)  # 재정렬

#X인자 재설정
new_FA_df = new타자_df_1[['타석(타자)', '타수(타자)', '득점(타자)', '안타(타자)', '2루타(타자)', '홈런(타자)', '루타(타자)', '타점(타자)', '희비(타자)', '장타율(타자)', 'OPS(타자)',
'wRC+(타자)', 'WAR*(타자)', 'WPA(타자)', 'IsoP(타자)', 'wRC(타자)', 'wRC/27(타자)', 'wRAA(타자)', '파크팩터 조정 wOBA(타자)', '파크팩터 조정 wRC(타자)', '파크팩터 조정 wRC/27(타자)',
'파크팩터 조정 wRAA(타자)', '파크팩터 조정 wRC+(타자)', '공격RAA 타격(타자)', '공격 RAA 종합(타자)', 'RAA-Adj(타자)', 'RAA(타자)', 'RAR(타자)', 'WAA(타자)', 'WAR Bat(타자)',
'장타(타자)', 'XH/AB(타자)']]

# 테스트 데이터에 대한 예측 수행
new_FA_predictions = best_model.predict(new_FA_df)

# 예측 결과 출력
print("2024년도 데이터 예측 결과:", new_FA_predictions)
```

24년 데이터 전처리 후
투수 모델로 24년 투수 FA 선수 금액 예측

투수 총액

```
4.014383571428571
4.273642460317459
3.5389561904761906
4.246996031746032
11.788611031746036
5.00924095238095
3.9011940476190468
3.60453119047619
10.667011388888895
3.4390896031746037
3.5173395238095235
11.226085198412699
4.078530714285716
3.257157936507936
3.4137103174603163
10.56065305555556
```

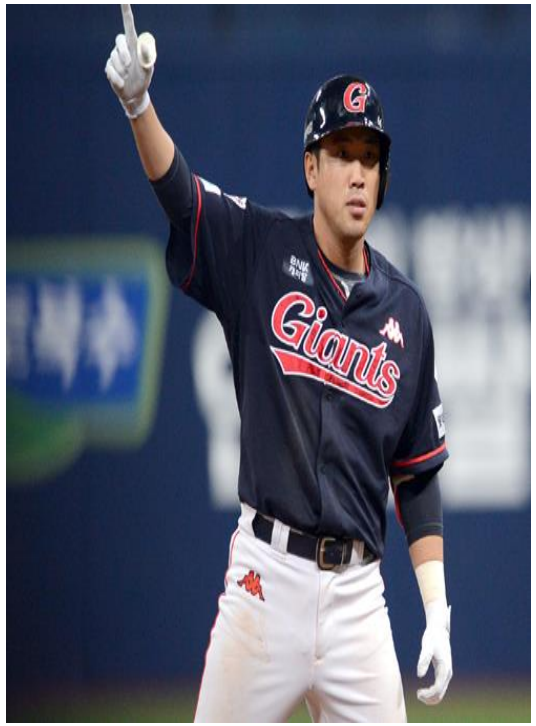
데이터 분석 – 예측결과

모델이 예측하는 2024FA 총액 TOP 4 !

안치홍
현 롯데 자이언츠
내야수

13.08

약 13.1억
4년 52.4억



전준우
현 롯데 자이언츠
외야수

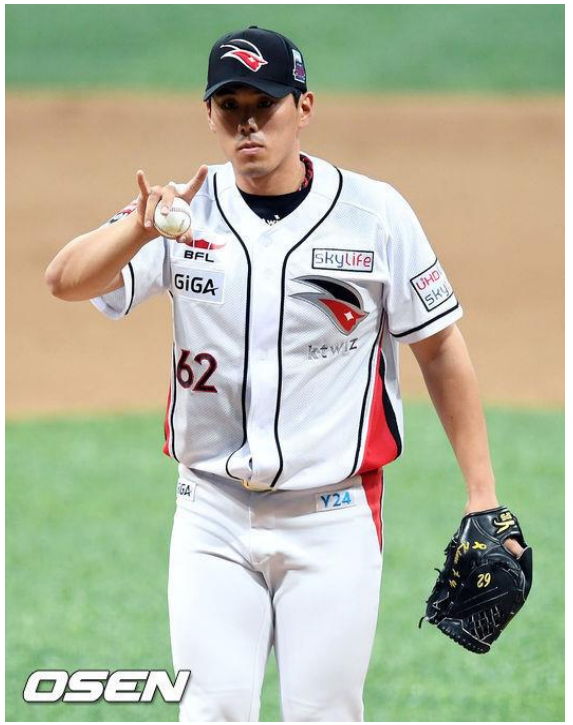
12.74

약 12.7억
4년 50.8억

서건창
현 LG 트윈스
내야수

11.93

약 11.9억
4년 47.6억



김재윤
현 KT 위즈
투수

11.78

약 11.8억
4년 47.2억

데이터 분석 – 예측 결과

모델이 예측한 2024FA 연봉 순위

선수명	소속팀	포지션	FA금액(억)	4년
장원준	두산 베어스	투수	11.23	44.92
김선빈	기아 타이거즈	내야수	10.67	42.68
오승환	삼성 라이온즈	투수	10.67	42.68
정우람	한화 이글스	투수	10.56	42.24
이용규	키움 히어로즈	외야수	9.82	39.28
박경수	KT 위즈	내야수	9.04	36.16
이재원	SSG 랜더스	포수	8.58	34.32
양석환	두산 베어스	내야수	8.10	32.4
김민성	LG 트윈스	내야수	8.07	32.28
고종욱	기아 타이거즈	외야수	5.36	21.44
주권	KT 위즈	투수	5.01	20.04
이지영	키움 히어로즈	포수	4.64	18.56
노수광	한화 이글스	외야수	4.56	18.24
김헌곤	삼성 라이온즈	외야수	4.53	18.12

선수명	소속팀	포지션	FA금액(억)	4년
김민식	SSG 랜더스	포수	4.42	17.68
김태군	삼성 라이온즈	포수	4.34	17.36
강한울	삼성 라이온즈	내야수	4.33	17.32
임찬규	LG 트윈스	투수	4.27	17.08
함덕주	LG 트윈스	투수	4.25	17
김강률	두산 베어스	투수	4.08	16.32
임창민	키움 히어로즈	투수	4.01	16.04
심창민	NC 다이노스	투수	3.90	15.6
김대우	삼성 라이온즈	투수	3.60	14.4
진해수	LG 트윈스	투수	3.54	14.16
윤명준	롯데 자이언츠	투수	3.44	13.76
윤명준	롯데 자이언츠	투수	3.44	13.76
장민재	한화 이글스	투수	3.41	13.64
홍건희	두산 베어스	투수	3.26	13.04

결론 및 제언

결과

여러가지 알고리즘 분석을 했을 때, 랜덤포레스트 모델이 가장 정확도가 높은 것을 알 수 있었다.

데이터의 수가 적어서 오차가 크게 나왔다.

한계점

1. 결정계수(R2 Score)값이 타자의 경우에는 약 0.52로 나왔지만 투수의 경우에는 약 0.31로 나와 전체적으로 봤을 때, 좋은 모델이라고 보기에는 어렵다.
2. Correlation 분석에서 세이버매트릭스 지표가 FA금액을 결정하는데 있어 많은 연관성을 가졌다는 점에서 통계적으로 세부적이고 과학적 지표가 효과가 있음을 확인할 수 있었다.
3. 투수의 데이터 수가 타자에 비해서 적었다는 점에서 투수의 모델을 구성하고 분석하는데 있어서 아쉬움이 존재했다.
4. 선수들의 FA금액을 판단하는데 있어서 지표만으로 판단하는 것을 섣부른 판단이라는 것을 알게 되었다.

THANK YOU

HAVE A NICE DAY!