1. Introduction

현대 야구는 통계의 바다라고 불립니다. 수 많은 통계 자료들이 쏟아지고 다양한 수치들이 만들어져 선수들을 평가하고 있습니다. 이 통계들을 세이버매트릭스라고 부릅니다. 미국과 일본에서는 이에 대한 연구활동이 활발하지만 현재 한국에서는 측정 장비들과 함께 도입 단계에 이르고 있습니다. 한국 KBO 리그의 세이버매트릭스가 많이 정립되어 있지 않기 때문에 이번 과제에 대해서 직접 데이터들을 다뤄보고 해석해볼 것입니다. 그래서 이번 과제에서 다룰 내용은 많은 데이터들 중에서 가장 대표적인 세이버매트릭스 OPS 입니다. On-base Plus Slugging, 말 뜻 그대로 출루율과 장타율을 합친 이 통계는 타자의 실력을 할 때 wrc+와 함께 매우 중요하게 판단되는 지표로서 2018 년도까지의 선수들의 통계를 RandomForest Regression 을 통해 학습시켜 2019 년도 선수들의 OPS 를 예측하고 실제 2019 년 성적과 어느정도 일치하는 지를 확인하는 것이 이번 과제의 목표입니다.

2. Results

예측 OPS 0.706896008

실제 OPS 0.73334 일치율 0.96393

0.12569804918964889 MSE 값

결과 값으로 받은 csv 파일에 19 년도 때의 실제 성적을 넣고 2018 년도까지 선수 생활을 했지만 2019 년도에 뛰지 않은 선수들을 제외하고 가중치 AB(타석 수)를 적용시켜 평균을 내어 계산한 결과 약 96.393%가 측정되었습니다. 실제 BABIP 으로 예측하는 방법은 널리 통용되어 있는 방법으로 추가적으로 더 고려해야 할 점들은 트리 개수를 바꾸거나 BABIP 과 더 복잡한 세이버매트릭스를 추가 변수로 넣는 방법들이 있습니다.

Ex) war(대체 선수 대비 승리 기여도), wOBA(가중 출루율), OPS+(조정 OPS), 연도별 공 반발계수 등

이러한 통계들도 있지만 매우 복잡한 계산 공식들이기에 표현하지 못했습니다. 추후에 직접 통계 사이트를 크롤링해서 데이터 전처리 이후에 OPS 예측에 대해서 다시 한번 더 분석 예정입니다.

3. How to run

1. 수식 설명

$$OPS = OBP(출루율) + SLG(장타율)$$
$$OBP = \{H(안타) + BB(볼넷) + HBP(사구)\}/PA(타석\ 수)$$
$$SLG = \{1B(1\ 루타) + 2B(2\ 루타)*2 + 3B(3\ 루타)*3 + HR(홈런)*4\}/AB(타수)$$
$$AVG(타율) = H/AB$$

2. 요소들 간의 상관 관계 파악
   위 통계들을 이해했다는 가정하에 통계들의 연관성 그리고 그 외의 요소들 ex) 연도, 타석 수, 포지션 등을 그래프를 통해 상관 관계에 대해서 파악을 먼저 합니다.

3. 변수에 대한 차단
   예측을 할 때 값들의 편차가 매우 클 수 있기 때문에 야구에서 운적인 요소를 배제하기 위한 작업을 거친다. 각각의 안타들의 자기상관계수 측정을 통해서 얼만큼 운적인 요소인지를 파악한다. (삼진과 볼넷, 홈런의 경우는 타자와 투수 1 대 1 이기에 운적인 요소가 없다.)

4. 새로운 지표 생성
   운적인 요소와 운적이지 않은 요소들을 조합해서 만들 수 있는 지표인 BABIP 을 생성한다.

5. 새로운 지표 리스트 추가 후 train set 과 test set 나누기
   train 과 test 는 n-1 년까지의 데이터를 바탕으로 n 년도의 성적을 예측하기 위한 과정이기에 train 에 n-1 년까지의 데이터, test 에 n 년도 즉 2017 년도까지와 2018 년도까지의 데이터를 각각 저장한다.

6. 랜덤포레스트 회귀 분석
   랜덤포레스트 회귀 분석을 사용하는 이유는 현재 데이터셋에 있는 수많은 요소들이 있기에 트리들이 랜덤하게 요소들을 선택해서 학습하면 최적의 방법을 찾도록 하기 위해서 사용했습니다. 랜덤으로 샘플링해서 학습하지만 결국에는 OPS 를 결정하는 요소들은 정해져 있기에 그 데이터들을 바탕으로 학습합니다.

7. 예측값 저장 후 일치율 분석
   크롤링에 실패하여 2019 년도 성적을 직접 추가해서 비교 후 계산하여 예측 OPS 와 실제 OPS 간의 비교를 진행한다.

4. Code description

1. 사용한 데이터 패키지

```
#데이터 패키지
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
plt.style.use('fivethirtyeight') #파이브써티에잇
import warnings
warnings.filterwarnings('ignore') #워닝 무시
```

2. 컬럼 딕셔너리 만들기

```
[7] agg={}
    for i in regular.columns:
        agg[i]=[]
```

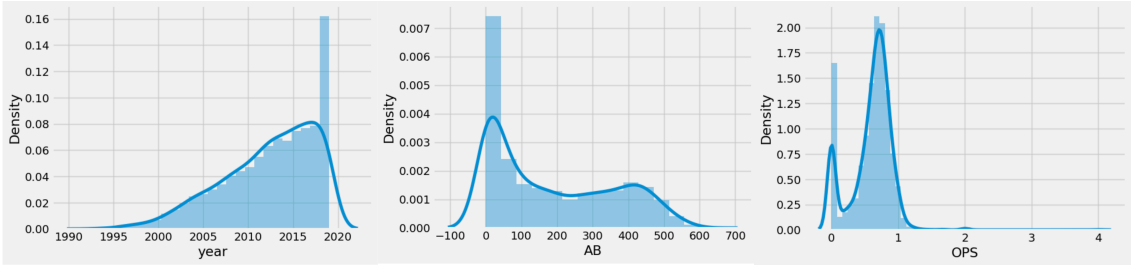
```
[8] for i in submission['batter_name'].unique():
        for j in regular.columns:
            if j in ['batter_id','batter_name','height/weight','year_born','position','starting_s
                agg[j].append(regular.loc[regular['batter_name']==i,j].iloc[0])
            elif j=='year':
                agg[j].append(2019)
            else:
                agg[j].append(0)
    regular=pd.concat([regular,pd.DataFrame(agg)])
```

3. 요소 간의 상관관계 파악하기

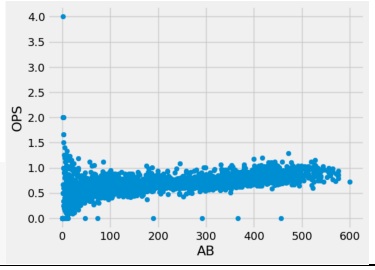
```
sns.distplot(regular['year'])
```

```
sns.distplot(regular['AB'])
```

```
sns.distplot(regular['OPS'].dropna())
```




```
plt.scatter(regular['AB'],regular['OPS'])
plt.xlabel('AB')
plt.ylabel('OPS')
```



4. 자기상관계수 측정하기

```
[25] def get_self_corr(var,regular=regular):
         x=[]
         y=[]
         regular1=regular.loc[regular['AB']>=50,]
         for name in regular1['batter_name'].unique():
             a=regular1.loc[regular1['batter_name']==name,].sort_values('year')
             k=[]
             for i in a['year'].unique():
                 if (a['year']==i+1).sum()==1:
                     k.append(i)
             for i in k:
                 x.append(a.loc[a['year']==i,var].iloc[0])
                 y.append(a.loc[a['year']==i+1,var].iloc[0])
         plt.scatter(x,y)
         plt.title(var)
         plt.show()
         print(pd.Series(x).corr(pd.Series(y))**2)

    regular['1B']=regular['H']-regular['2B']-regular['3B']-regular['HR']
```

```
for i in ['avg','1B','2B','3B']:
    get_self_corr(i)
```

```
0.17987194251531383
0.3579451034966973 0.3216523588655752
0.2001112514921482
```

5. BABIP 만들고 리스트 만들기

```
[28] regular['avg_luck']=(regular['H']-regular['HR'])/(regular['AB']-regular['HR']-regular['SO'])
```

```
for j in ['avg', 'G', 'AB', 'R', 'H','2B', '3B', 'HR', 'TB', 'RBI', 'SB', 'CS', 'BB', 'HBP', 'SO', 'GDP','SLG', 'OBP', 'E','avg_luck']:
    lag_1_avg=[]
    for i in range(len(regular)):
        if len(regular.loc[(regular['batter_name']==regular['batter_name'].iloc[i])&(regular['year']==regular['year'].iloc[i]-1)][j])==0:
            lag_1_avg.append(np.nan)
        else:
            lag_1_avg.append(regular.loc[(regular['batter_name']==regular['batter_name'].iloc[i])&(regular['year']==regular['year'].iloc[i]-1)][j].iloc[0])

    regular['lag_1_'+j]=lag_1_avg
    print(j)
```

6. 랜덤포레스트 회귀 분석

```
[34] #로지스틱 회귀 분석
     rf=RandomForestRegressor(n_estimators=500)
     rf.fit(X_train.fillna(-1),y_train,sample_weight=train['AB'])
```

```
▼ RandomForestRegressor
RandomForestRegressor(n_estimators=500)
```

7. MSE 구하기

```
[36] #MSE값 구하기
     real=test['OPS']
     ab=test['AB']

     from sklearn.metrics import mean_squared_error
     mean_squared_error(real,pred,sample_weight=ab)**0.5
```

```
0.12569804918964889
```

8. 2019 년 OPS 예측 학습

```
[37] train=regular.loc[regular['year']<=2018,]
     test=regular.loc[regular['year']==2019,]
     y_train=train['OPS']
     X_train=train[[x for x in regular.columns if ('lag' in x)|('total' in x)]]


     rf=RandomForestRegressor(n_estimators=500)
     rf.fit(X_train.fillna(-1),y_train,sample_weight=train['AB'])
```

```
▾ RandomForestRegressor
RandomForestRegressor(n_estimators=500)
```

5. References

1. https://dacon.io/competitions/official/235546/codeshare/578?page=1&dtype=recent (데이콘 KBO 타자 OPS 시각화 경진대회)
2. http://www.statiz.co.kr/stat.php (스탯티즈 기록실)
3. https://chat.openai.com/ (Chat GPT)
4. https://everyissues.tistory.com/7 (세이버매트릭스 BABIP)
5. https://wooono.tistory.com/115 (랜덤 포레스트란?)