Oaklands - Python

May 26, 2020

1 Finding Replacement Players for Oakland A's

Faraz Khan,

May 18, 2020

1.1 Introduction

In this project I will be providing a solution to the actual problem a baseball team Oakland A's faced in 2001 when three of their key players left the team. To find the replacement players,the general manager of Oakland A's took sports analytics to the next level by inventing some new KPIs for finding undervalued but brilliant players . This strategy was so successful that they went on the win 20 consecutive games. This strategy was brought to the world in a book called 'Moneyball: The Art of Winning an Unfair Game' by Michel Lewis. Later, this book was turned into a movie called Moneyball. In this project we will analyze the KPIs that better account for the offence's success such as Slugging and On Base Percentage. While finding replacement players we have following three constraints:

1. Combined salaries should not exceed 15 million USD
2. Combined At Bat(AB) should be more than the combined AB of lost players
3. Mean On Base Percentage(OBP) should be more than the mean OBP of lost players

PlayerID of lost players : giambja01 , damonjo01 , saenzol01

Before we start, Let's import some packages that might help us in this task

[12]:
```
import pandas as pd
import itertools
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter
import matplotlib.ticker as ticker
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import jaccard_similarity_score
from sklearn import metrics
```

1.2 Loading the Datasets and analyzing it's structure.

[13]:
```
batting = pd.read_csv("Batting.csv")
salaries = pd.read_csv("Salaries.csv")
batting.head()
```

[13]:
```
    playerID  yearID  stint teamID lgID   G  G_batting   AB    R    H  ...  \
0  aardsda01    2004      1    SFN   NL  11       11.0  0.0  0.0  0.0  ...
1  aardsda01    2006      1    CHN   NL  45       43.0  2.0  0.0  0.0  ...
2  aardsda01    2007      1    CHA   AL  25        2.0  0.0  0.0  0.0  ...
3  aardsda01    2008      1    BOS   AL  47        5.0  1.0  0.0  0.0  ...
4  aardsda01    2009      1    SEA   AL  73        3.0  0.0  0.0  0.0  ...

    SB   CS   BB   SO  IBB  HBP   SH   SF  GIDP  G_old
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0   11.0
1  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0   0.0   45.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0    2.0
3  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0   0.0    5.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0    NaN

[5 rows x 24 columns]
```

[14]:
```
salaries.head()
```

[14]:
```
   yearID teamID lgID   playerID   salary
0    1985    BAL   AL  murraed02  1472819
1    1985    BAL   AL   lynnfr01  1090000
2    1985    BAL   AL  ripkeca01   800000
3    1985    BAL   AL   lacyle01   725000
4    1985    BAL   AL  flanami01   641667
```

Let's assess the distribution of variables in these 2 datasets.

[15]:
```
print('Summary of Batting')
print(batting.describe())
print('Summary of Salaries')
print(salaries.describe())
```

```
Summary of Batting
             yearID         stint              G     G_batting             AB  \
count  97889.000000  97889.000000  97889.000000  96483.000000  91476.000000
mean    1961.732922      1.076873     51.654078     49.130790    154.067766
std       38.104588      0.282653     47.267487     48.869353    187.374936
min     1871.000000      1.000000      1.000000      0.000000      0.000000
25%     1931.000000      1.000000     13.000000      7.000000      9.000000
50%     1970.000000      1.000000     35.000000     32.000000     61.000000
75%     1995.000000      1.000000     81.000000     81.000000    260.000000
max     2013.000000      5.000000    165.000000    165.000000    716.000000
```

```
                R             H            2B            3B            HR  \
count  91476.000000  91476.000000  91476.000000  91476.000000  91476.000000
mean      20.468615     40.366883      6.799543      1.424483      3.001640
std       28.935336     53.674482      9.855328      2.755706      6.455669
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.000000      1.000000      0.000000      0.000000      0.000000
50%        5.000000     12.000000      2.000000      0.000000      0.000000
75%       31.000000     66.000000     10.000000      2.000000      3.000000
max      192.000000    262.000000     67.000000     36.000000     73.000000

       ...            SB            CS            BB            SO  \
count  ...  90176.000000  68022.000000  91476.000000  83638.000000
mean   ...      3.264616      1.384802     14.215073     21.951768
std    ...      8.055966      2.900107     21.298814     28.141424
min    ...      0.000000      0.000000      0.000000      0.000000
25%    ...      0.000000      0.000000      0.000000      2.000000
50%    ...      0.000000      0.000000      4.000000     11.000000
75%    ...      2.000000      1.000000     21.000000     31.000000
max    ...    138.000000     42.000000    232.000000    223.000000

                IBB           HBP           SH           SF          GIDP  \
count  54912.000000  88656.000000  85138.000000  55443.000000  65368.000000
mean       1.281960      1.135919      2.563920      1.202532      3.329045
std        2.968605      2.333229      4.427705      2.057438      4.878830
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      1.000000      0.000000      1.000000
75%        1.000000      1.000000      3.000000      2.000000      5.000000
max      120.000000     51.000000     67.000000     19.000000     36.000000

             G_old
count  92700.000000
mean      50.994725
std       47.699666
min        0.000000
25%       11.000000
50%       34.000000
75%       82.000000
max      165.000000

[8 rows x 21 columns]
Summary of Salaries
             yearID        salary
count  23956.000000  2.395600e+04
mean    1999.419436  1.864357e+06
std        8.045512  3.079812e+06
min     1985.000000  0.000000e+00
```

```
25%      1993.000000  2.500000e+05
50%      1999.000000  5.079500e+05
75%      2006.000000  2.100000e+06
max      2013.000000  3.300000e+07
```

1.3 Adding new statistical measures to the dataset:

[16]:
```
#Adding a new variable OBP for calculating On Base Percentage:
batting['OBP'] = (batting['H']+batting['BB']+batting['HBP'])/
 ↪(batting['AB']+batting['BB']+batting['HBP']+batting['SF'])

#Adding a new variable SLG for calculating Slugging:
#Creating a new variable 1xB to calculate Singles as it is not already in the␣
 ↪batting table.
batting['1B']=batting['H']-(batting['2B']+batting['3B']+batting['HR'])

#Creating new column for slugging using following formula
batting['SLG']=(batting['1B']+(2*batting['2B'])+(3*batting['3B'])+(4*batting['HR']))/
 ↪batting['AB']

#Quick look at the final batting table:
batting.head()
```

[16]:

	playerID	yearID	stint	teamID	lgID	G	G_batting	AB	R	H	...	\
0	aardsda01	2004	1	SFN	NL	11	11.0	0.0	0.0	0.0	...	
1	aardsda01	2006	1	CHN	NL	45	43.0	2.0	0.0	0.0	...	
2	aardsda01	2007	1	CHA	AL	25	2.0	0.0	0.0	0.0	...	
3	aardsda01	2008	1	BOS	AL	47	5.0	1.0	0.0	0.0	...	
4	aardsda01	2009	1	SEA	AL	73	3.0	0.0	0.0	0.0	...	

	SO	IBB	HBP	SH	SF	GIDP	G_old	OBP	1B	SLG
0	0.0	0.0	0.0	0.0	0.0	0.0	11.0	NaN	0.0	NaN
1	0.0	0.0	0.0	1.0	0.0	0.0	45.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	2.0	NaN	0.0	NaN
3	1.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	NaN	NaN	0.0	NaN

[5 rows x 27 columns]

1.4 Cleaning and merging datasets:

Since our Salary data starts from year 1985, we will only select data after 1984 from batting table

[17]:
```
batting=batting[batting["yearID"] > 1984]

#Checking if we applied subset function correctly and only have data after 1984
batting["yearID"].describe()
```

[17]:
```
count    35652.000000
mean      1999.724391
std          8.178206
min       1985.000000
25%       1993.000000
50%       2000.000000
75%       2007.000000
max       2013.000000
Name: yearID, dtype: float64
```

[18]:
```
#Merging batting and salary data for further analysis on selecting replacement players
batsal=batting.merge(salaries,how='outer',left_on=['playerID','yearID'],right_on=['playerID','yearID'])

#Removing NaN values.
batsal.dropna(inplace=True)

#Checking if everything is in order.
batsal.head(25)
```

[18]:

	playerID	yearID	stint	teamID_x	lgID_x	G	G_batting	AB	R	\
3	aardsda01	2008	1.0	BOS	AL	47.0	5.0	1.0	0.0	
11	aasedo01	1989	1.0	NYN	NL	49.0	49.0	5.0	0.0	
15	abadan01	2006	1.0	CIN	NL	5.0	5.0	3.0	0.0	
20	abbotje01	1998	1.0	CHA	AL	89.0	89.0	244.0	33.0	
21	abbotje01	1999	1.0	CHA	AL	17.0	17.0	57.0	5.0	
22	abbotje01	2000	1.0	CHA	AL	80.0	80.0	215.0	31.0	
23	abbotje01	2001	1.0	FLO	NL	28.0	28.0	42.0	5.0	
34	abbotji01	1999	1.0	MIL	NL	20.0	18.0	21.0	0.0	
35	abbotku01	1993	1.0	OAK	AL	20.0	20.0	61.0	11.0	
36	abbotku01	1994	1.0	FLO	NL	101.0	101.0	345.0	41.0	
37	abbotku01	1995	1.0	FLO	NL	120.0	120.0	420.0	60.0	
38	abbotku01	1996	1.0	FLO	NL	109.0	109.0	320.0	37.0	
39	abbotku01	1997	1.0	FLO	NL	94.0	94.0	252.0	35.0	
40	abbotku01	1998	1.0	OAK	AL	35.0	35.0	123.0	17.0	
41	abbotku01	1998	2.0	COL	NL	42.0	42.0	71.0	9.0	
42	abbotku01	1999	1.0	COL	NL	96.0	96.0	286.0	41.0	
43	abbotku01	2000	1.0	NYN	NL	79.0	79.0	157.0	22.0	
44	abbotku01	2001	1.0	ATL	NL	6.0	6.0	9.0	0.0	
46	abbotky01	1992	1.0	PHI	NL	31.0	31.0	29.0	1.0	
47	abbotky01	1995	1.0	PHI	NL	18.0	18.0	2.0	1.0	
55	abbotpa01	2000	1.0	SEA	AL	35.0	2.0	5.0	1.0	
56	abbotpa01	2001	1.0	SEA	AL	28.0	2.0	4.0	0.0	
60	abbotpa01	2004	2.0	PHI	NL	10.0	8.0	11.0	1.0	
61	abercre01	2006	1.0	FLO	NL	111.0	111.0	255.0	39.0	
65	abernbr01	2002	1.0	TBA	AL	117.0	117.0	463.0	46.0	

```
        H  ...   SH   SF  GIDP  G_old       OBP    1B       SLG  teamID_y  \
3     0.0  ...  0.0  0.0   0.0    5.0  0.000000   0.0  0.000000       BOS
11    0.0  ...  0.0  0.0   0.0   49.0  0.000000   0.0  0.000000       NYN
15    0.0  ...  0.0  0.0   0.0    5.0  0.400000   0.0  0.000000       CIN
20   68.0  ...  2.0  5.0   2.0   89.0  0.298450  41.0  0.491803       CHA
21    9.0  ...  1.0  1.0   4.0   17.0  0.222222   7.0  0.263158       CHA
22   59.0  ...  2.0  1.0   2.0   80.0  0.343096  40.0  0.395349       CHA
23   11.0  ...  0.0  0.0   1.0   28.0  0.326087   8.0  0.333333       FLO
34    2.0  ...  3.0  0.0   1.0   18.0  0.095238   2.0  0.095238       MIL
35   15.0  ...  3.0  0.0   3.0   20.0  0.281250  11.0  0.409836       OAK
36   86.0  ...  3.0  2.0   5.0  101.0  0.290761  57.0  0.394203       FLO
37  107.0  ...  2.0  5.0   6.0  120.0  0.317597  65.0  0.452381       FLO
38   81.0  ...  4.0  0.0   7.0  109.0  0.307246  48.0  0.428125       FLO
39   69.0  ...  6.0  0.0   5.0   94.0  0.314607  43.0  0.432540       FLO
40   33.0  ...  1.0  1.0   3.0   35.0  0.325926  23.0  0.390244       OAK
41   18.0  ...  0.0  2.0   2.0   42.0  0.276316   9.0  0.464789       OAK
42   78.0  ...  2.0  1.0   4.0   96.0  0.310231  51.0  0.430070       COL
43   34.0  ...  0.0  1.0   2.0   79.0  0.283237  20.0  0.388535       NYN
44    2.0  ...  0.0  0.0   0.0    6.0  0.222222   2.0  0.222222       ATL
46    2.0  ...  6.0  0.0   0.0   31.0  0.100000   1.0  0.103448       PHI
47    1.0  ...  0.0  0.0   0.0   18.0  0.500000   1.0  0.500000       PHI
55    2.0  ...  1.0  0.0   0.0    2.0  0.400000   1.0  0.600000       SEA
56    1.0  ...  1.0  0.0   0.0    2.0  0.250000   1.0  0.250000       SEA
60    2.0  ...  3.0  0.0   0.0    8.0  0.181818   2.0  0.181818       TBA
61   54.0  ...  4.0  1.0   2.0  111.0  0.270758  35.0  0.333333       FLO
65  112.0  ...  8.0  2.0   8.0  117.0  0.288306  88.0  0.311015       TBA

    lgID_y     salary
3       AL   403250.0
11      NL   400000.0
15      NL   327000.0
20      AL   175000.0
21      AL   255000.0
22      AL   255000.0
23      NL   300000.0
34      NL   400000.0
35      AL   109000.0
36      NL   109000.0
37      NL   119000.0
38      NL   250000.0
39      NL   650000.0
40      AL  1000000.0
41      AL  1000000.0
42      NL   900000.0
43      NL   500000.0
44      NL   600000.0
```

```
46      NL   109000.0
47      NL   150000.0
55      AL   285000.0
56      AL  1700000.0
60      AL   600000.0
61      NL   327000.0
65      AL   215000.0

[25 rows x 30 columns]
```

1.5 Having a look at lost players statistics:

Let's make a subset of lost players

[19]:
```
#Subselecting data of players lost in 2001
lost_players=batting[(batting['playerID']=='giambja01')| \

 ↪(batting['playerID']=='damonjo01') | (batting['playerID']=='saenzol01')]
lost_players=lost_players[lost_players['yearID']==2001]
lost_players
```

[19]:
```
        playerID  yearID  stint teamID lgID    G  G_batting     AB      R  \
19745  damonjo01    2001      1    OAK   AL  155      155.0  644.0  108.0
30725  giambja01    2001      1    OAK   AL  154      154.0  520.0  109.0
76245  saenzol01    2001      1    OAK   AL  106      106.0  305.0   33.0

           H  ...    SO   IBB   HBP   SH   SF  GIDP  G_old       OBP     1B  \
19745  165.0  ...  70.0   1.0   5.0  5.0  4.0   7.0  155.0  0.323529  118.0
30725  178.0  ...  83.0  24.0  13.0  0.0  9.0  17.0  154.0  0.476900   91.0
76245   67.0  ...  64.0   1.0  13.0  1.0  3.0   9.0  106.0  0.291176   36.0

            SLG
19745  0.363354
30725  0.659615
76245  0.383607

[3 rows x 27 columns]
```

Let's calculate some statistics for lost players:

[20]:
```
print('OBP mean of lost players:',lost_players['OBP'].mean())
print('Combined AB of lost players:',lost_players['AB'].sum())
```

```
OBP mean of lost players: 0.36386867712807924
Combined AB of lost players: 1469.0
```

1.6 Excluding lost players from the dataset and selecting replacements:

[25]:
```
#Excluding lost players from the analysis
delete_row_1 = batsal[batsal["playerID"]=='giambja01'].index
delete_row_2 = batsal[batsal["playerID"]=='damonjo01'].index
delete_row_3 = batsal[batsal["playerID"]=='saenzol01'].index
remaining_players = batsal.drop(delete_row_1)
remaining_players = remaining_players.drop(delete_row_2)
remaining_players = remaining_players.drop(delete_row_3)

#Selecting replacement players according to the defined criteria
replacement_players=remaining_players[remaining_players['AB']>300]
replacement_players=replacement_players[replacement_players['yearID']==2001]
replacement_players=replacement_players[replacement_players['OBP']>0.37]
replacement_players=replacement_players[replacement_players['salary']<=5000000]

#Arraning for highest OBP
replacement_players.sort_values(by='OBP',ascending=False,inplace=True)

#displaying top 3 choices
Top_3=replacement_players.head(3)
Top_3
```

[25]:

	playerID	yearID	stint	teamID_x	lgID_x	G	G_batting	AB	R	H	...	SH	SF	GIDP	G_old	OBP	1B	SLG	teamID_y	lgID_y	salary
13287	heltoto01	2001	1.0	COL	NL	159.0	159.0	587.0	132.0	197.0	...	1.0	5.0	14.0	159.0	0.431655	92.0	0.684838	COL	NL	4950000.0
2446	berkmla01	2001	1.0	HOU	NL	156.0	156.0	577.0	110.0	191.0	...	0.0	6.0	8.0	156.0	0.430233	97.0	0.620451	HOU	NL	305000.0
11336	gonzalu01	2001	1.0	ARI	NL	162.0	162.0	609.0	128.0	198.0	...	0.0	5.0	14.0	162.0	0.428571	98.0	0.688013	ARI	NL	4833333.0

[3 rows x 30 columns]

Let's confirm our choices by checking if they meet our constraints.

[30]:
```
print('OBP mean of Replacement:',Top_3['OBP'].mean())
print('Combined AB of Replacement:',Top_3['AB'].sum())
print('Combined Salaries of Replacement:',Top_3['salary'].sum())
```

```
OBP mean of Replacement: 0.43015288765665205
Combined AB of Replacement: 1773.0
```

```
Combined Salaries of Replacement: 10088333.0
```

1.7 Conclusion:

In the end we have successfully chosen three palyers who meet our original criteria we set in the introduction. These replacement players are some of the great players who are highly undervalued in the market. We have unearthed them thanks to the innovative features.

Further details about the players is provided in the links with their names below.

1.Todd Helton

https://www.baseball-reference.com/players/h/heltoto01.shtml

2.Lance Berkman

https://www.baseball-reference.com/players/b/berkmla01.shtml

3.Luis Gonzalez

https://www.baseball-reference.com/players/g/gonzalu01.shtml