Introduction:

The thing of beauty in baseball is that each year we have a chance to see players making a leap. Like Jose Bautista, Jose Ramirez, Ben Zobrist, etc. This research aims to find out of these breakout players, the improvement of what stats are more responsible for their WAR and wRC+ gain.

Methods:

Data include MLB players stats from 2000-2020

During that period, 63 players who had a leap year were selected

Not only look at players who had a leap(defined as having a 4 war differential from last season), but also verify that leap by looking at the two years average before that leap year

For example: 2007 Ben Zobrist had a -1.1 WAR season, jumped to 1.5 in 2011 then 8.7 in 2012. He had a 7.2(8.7-1.5) WAR jump as well as verified by 8.7-(-1.1+1.5)/2=8.5(marked as jumpfromavg in column below) Batting categories include G(games played), PA, HR, R, SB, BBp(BB percentage), Kp(K percentage), ISO, BABIP, AVG, OBP, SLG, wOBA, wRCp(wRC+), BsR(Base Running), GBFB(GB/FB), Batted ball type(LDp, GBp, FBp), IFFBp(Infield Fly Ball percentage), HRFB(HR/FB), IFHp(Infield hit percentage), Batted ball direction(Pullp, Centp, Oppop), Quality of contact(Softp, Medp, Hardp).

Step1: Import the data of the year before jump

In [1]:

```
import pandas as pd
import pyodbc
#connect with sql server and retrieve the data we want
sql conn = pyodbc.connect('''DRIVER={ODBC Driver 13 for SQL Server);
                         SERVER=ALLENHO\MSSOLSERVER002;
                        DATABASE=WAR Jump:
                        Trusted Connection=yes''')
query = '''
select*
from BeforeLeap
order by NextWAR desc
#convert the data into dataframe
df beforeleap = pd.read sql(query, sql conn)
#take a look at the data
print(df beforeleap.head())
  Season
                            Team G PA HR R RBI SB
                   Name
                                                          ВВр ...
0
    2011
           Buster Posey
                           Giants
                                   45
                                       185
                                               17
                                                   21
                                                          0.097
                        Dodgers 158 608 23
         Adrian Beltre
1
    2003
                                               50
                                                   80
                                                        2 0.061
   2010 Jacoby Ellsbury
                        Red Sox
                                           0 10
                                  18
                                       8.4
                                                   5 7 0.048
2
         Bryce Harper Nationals 100 395 13 41 32 2 0.096
   2014
   2008
            Ben Zobrist
                            Rays 62 227 12 32 30 3 0.110 ...
                           Medp Hardp WAR NextWAR jumpfromavg jump
  Pullp Centp Oppop Softp
0 0.406 0.346 0.248 0.233 0.534 0.233 1.8
                                            10.1
                                                          7 20
                                                                 8 3
1 0.434 0.288 0.277 0.161 0.630 0.209 3.2
                                               9.7
                                                          6.10
                                                                6.5
2 0.333 0.406 0.261 0.261 0.609 0.130 -0.2
                                               9.5
                                                          8.55
                                                                9.7
3 0.389 0.353 0.258 0.179 0.520 0.302 1.6
                                               9.3
                                                          6.45
                                                                 7.7
4 0.491 0.307 0.203 0.129 0.558 0.313 1.5
                                               8.7
[5 rows x 36 columns]
```

Step2: Import the data of the year that the player made the jump

```
In [2]:
```

```
#connect with sql server and retrieve the data we want
sql_conn = pyodbc.connect('''DRIVER={ODBC Driver 13 for SQL Server};
SERVED=ALLENHO\MSSOLSERVEDOO2.
```

```
OENVEN-ALLENIIO/MOOQLOENVENUUZ,
                            DATABASE=WAR Jump;
                            Trusted Connection=yes''')
query = '''
select*
from LeapYear
order by WAR desc
#convert the data into dataframe
df leapyear = pd.read sql(query, sql conn)
#take a look at the data
print(df leapyear.head())
                                                                     ВВр ...
                                 Team G PA HR
                                                     R RBI SB
   Season
                      Name
Λ
   2012
                              Giants 148 610 24
                                                      78 103
            Buster Posey
                                                               1 0.113
          Adrian Beltre Dodgers 156 657 48
Jacoby Ellsbury Red Sox 158 732 32
Bryce Harper Nationals 153 654 42
     2004
                                                     104
                                                          121
                                                                   0.081
1
                                                                          . . .
2
     2011 Jacoby Ellsbury
                                                     119
                                                          105
                                                               39 0.071
                                                                          . . .
                                                     118
                                                               6 0.190
    2015
                                                          99
                                Rays 152 599 27
                                                           91 17 0.152 ...
             Ben Zobrist
                                                     91
                IFHp Pullp Centp Oppop Softp Medp 0.083 0.382 0.363 0.255 0.108 0.576
   IFFBp
         HRFB
                                                    Medp Hardp
                                                                   WAR
  0.039 0.188 0.083
                                                           0.316
                                                                  10.1
  0.145  0.240  0.052  0.361  0.305  0.334  0.132  0.528  0.340
                                                                   9.7
2 0.104 0.167 0.091 0.397 0.353 0.251 0.240 0.495 0.265
                                                                  9.5
3 0.058 0.273 0.046 0.454 0.338 0.208 0.119 0.472 0.409
4 0.052 0.175 0.066 0.488 0.306 0.207 0.124 0.557 0.318
                                                                  8.7
[5 rows x 33 columns]
Step3: The correlation table and the data of the two year differential
In [3]:
df beforeleaptrim=df beforeleap.loc[:,'G':'WAR']
df leapyeartrim=df leapyear.loc[:,'G':'WAR']
df_diff=df_leapyeartrim-df_beforeleaptrim
df corr diff = df diff.corr()
print(df corr diff.loc[:,'WAR'])
print(df diff)
```

```
0.180927
PΑ
        0.182269
HR
        0.133373
        0.381860
        0.242242
RBT
        0.289972
SB
BBp
        0.117997
        0.009606
Kρ
        0.132658
TSO
BABIP
        0.307701
AVG
        0.306928
OBP
        0.325248
        0.247819
SLG
wOBA
        0.304482
wRCp
        0.341955
        0.241788
BsR
GBFB
       -0.065691
LDp
        0.139518
       -0.108654
GBp
        0.032023
FBp
        0.181872
IFFBp
        0.052006
HRFB
IFHp
       0.031965
       -0.105371
Pullp
Centp
        0.037522
        0.098905
Oppop
       -0.016719
Softp
Medp
       -0.031591
Hardp
       0.044388
        1.000000
WAR
Name: WAR, dtype: float64
                                                                      HBEB /
     C DV HB
                   R RRT CR
                                RRn
                                              TGO RARTP
                                                              TFFRn
```

```
82 -2 0.016 -0.005 0.108 0.042 ... -0.012 0.085
0
   103 425 20
               61
1
       49 25 54 41
                      5 0.020 -0.037 0.110 0.073 ... -0.007 0.105
2
   140 648 32 109 100 32 0.023 0.027 0.179 0.119 ... 0.104 0.167
              77 67 4 0.094 -0.063 0.168 0.017 ... -0.025 0.118
   53 259 29
3
       372 15
               59
                   61 14 0.042 0.011 -0.007 0.074 ... -0.021 0.001
                                                 . . .
                                                ... 0.048
              37 42
                      7 0.015 -0.030 0.057 0.055
   43 279 11
58
                                                          0.032
                                                ... -0.071 0.045
59 107 349 11 53 39 16 0.049 -0.027 -0.036 0.133
60
  33 196 12
               34 27
                      2 -0.003 0.058 0.065 0.042 ... -0.015 0.065
   37 170 16
51 276 23
                  33
61
               52
62
   IFHp Pullp Centp Oppop Softp
                               Medp Hardp WAR
0 0.040 -0.024 0.017 0.007 -0.125 0.042 0.083 8.3
 -0.040 -0.073 0.017 0.057 -0.029 -0.102 0.131
   0.032 0.064 -0.053 -0.010 -0.021 -0.114
                                     0.135
                                           9.7
3 -0.039 0.065 -0.015 -0.050 -0.060 -0.048 0.107
                                           7.7
4 -0.003 -0.003 -0.001 0.004 -0.005 -0.001 0.005 7.2
58 0.007 0.018 0.020 -0.038 0.042 -0.102 0.060 4.2
59 0.046 0.149 0.095 -0.244 -0.166 0.210 -0.044 4.0
60 -0.014 -0.031 0.055 -0.024 -0.082 -0.012 0.095 4.1
62 -0.010 -0.075 0.051 0.026 0.014 -0.094 0.081 5.0
[63 rows x 30 columns]
```

ח מאווו

Step4: Find out the MAE, RMSE, R2

```
In [4]:
```

 $_{L}$ 1117 1/ 1/17 T

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import r2 score
import numpy as np
# split data into X and y
X = df diff.loc[:,'G':'Hardp']
Y = df_diff.loc[:,'WAR']
# split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
# define the model
model = DecisionTreeRegressor(criterion = 'mse', max depth=5)
model.fit(X train, y train)
# make predictions for test data
y pred = model.predict(X test)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print('R Squared Score is:', r2_score(y_test, y_pred))
Mean Absolute Error: 1.9675824175824173
Root Mean Squared Error: 2.408288767583546
R Squared Score is: -3.7123820153061216
```

In [5]:

```
for importance, name in sorted(zip(model.feature_importances_, X_train.columns),reverse=True):
print(name, importance)
wRCp 0.32513125441679036
IFFBp 0.15473553499154155
```

Softp 0.1516055283339745 OBP 0.13360014021037142 HR 0.1039482933629447 RBI 0.04965037978935612 GBp 0.04246566588345434

```
Hardp 0.031040032814516436
FBp 0.00532641375118345
BABIP 0.0022470808012804815
Pullp 0.0002496756445864935
wOBA 0.0
SIG 0.0
SB 0.0
R 0.0
PA 0.0
Oppop 0.0
Medp 0.0
LDp 0.0
Kp 0.0
ISO 0.0
IFHp 0.0
HRFB 0.0
GBFB 0.0
G 0.0
Centp 0.0
BsR 0.0
BBp 0.0
AVG 0.0
```

The result was not a good one judging from negative R2 and so-so correlation coefficient, probably because I tried the correlation with WAR but taking into account of too much batting stats without enough fielding and running.

We turn to a new topic focusing on the batting stats, which include HR, R, BBp, Kp, ISO, BABIP, AVG, OBP, SLG, GBFB, LDp, GBp, FBp, IFFBp, HRFB, IFHp, Pullp, Centp, Oppop, Softp, Medp, Hardp and see if the differential of these categories between jump year and previous year are significantly correlated with the the differential of wRC+.

In [16]:

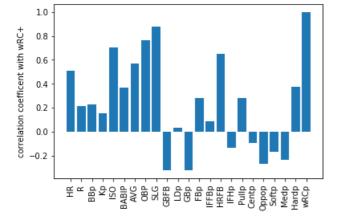
```
df_beforeoff=df_beforeleap[['HR','R','BBp', 'Kp', 'ISO', 'BABIP', 'AVG', 'OBP', 'SLG', 'GBFB', 'LDp
', 'GBp', 'FBp', 'IFFBp', 'HRFB', 'IFHp', 'Pullp', 'Centp', 'Oppop', 'Softp', 'Medp', 'Hardp', 'wRCp
df leapoff=df leapyear[['HR','R','BBp', 'Kp', 'ISO', 'BABIP', 'AVG', 'OBP', 'SLG', 'GBFB', 'LDp', '
GBp', 'FBp', 'IFFBp', 'HRFB', 'IFHp', 'Pullp', 'Centp', 'Oppop', 'Softp', 'Medp', 'Hardp', 'wRCp']]
{\tt df\_diff\_off=} {\tt df\_leapoff-} {\tt df\_beforeoff}
df_corr_diff_off = df_diff_off.corr()
print(df_corr_diff_off.loc[:,'wRCp'])
print(df diff off)
         0.508888
HR
R
         0.213211
BBp
         0.227693
         0.150763
Κр
ISO
         0.703420
BABIP
         0.371309
AVG
         0.571028
         0.761551
OBP
SLG
         0.877834
GBFB
        -0.325276
         0.033460
LDp
GBp
        -0.321055
FBp
        0.282038
        0.083147
IFFBp
         0.648738
HRFB
        -0.132628
IFHp
       0.283420
Pullp
        -0.092512
Centp
        -0.268495
Oppop
Softp
        -0.167955
Medp
        -0.237407
       0.371935
Hardp
wRCp
        1.000000
Name: wRCp, dtype: float64
                                            AVG
    HR R BBp Kp
                             ISO BABIP
                                                   OBP
                                                           SLG GBFB
                                                                         . . .
             0.016 -0.005  0.108  0.042  0.052  0.040  0.160 -0.18
0
    20
         61
                                                                         . . .
         54 0.020 -0.037 0.110 0.073 0.094 0.098 0.205 -0.07
1
    2.5
    32 109 0.023 0.027 0.179 0.119 0.129 0.135 0.308 -0.16
   29 77 0.094 -0.063 0.168 0.017 0.057 0.116 0.226 -0.28
    1.5
        59 0.042 0.011 -0.007 0.074 0.044 0.066 0.038 0.04
4
               . . .
                       . . .
                               . . .
                                      . . .
                                             . . .
                                                     . . .
```

```
58
   11
        37
            0.015 -0.030 0.057
                                0.055
                                       0.059
                                             0.072 0.115 -0.04
        53 0.049 -0.027 -0.036 0.133
59
   11
                                       0.115
                                             0.141
                                                    0.079 0.67
        34 -0.003 0.058 0.065 0.042 0.019
60
   12
                                             0.015 0.084 -0.30
        33 0.031 -0.002 0.110 0.070 0.065 0.086 0.175 -0.20
  16
62 2.3
        52 -0.009 0.012 0.129 0.074 0.073 0.072 0.202 -0.39
   IFFBp
          HRFB
                 IFHp Pullp Centp Oppop Softp
                                                                  wRCp
                                                   Medp Hardp
0
  -0.012 0.085 0.040 -0.024
                              0.017 0.007 -0.125 0.042
1
  -0.007 0.105 -0.040 -0.073 0.017 0.057 -0.029 -0.102 0.131
   0.104 0.167 0.032 0.064 -0.053 -0.010 -0.021 -0.114 0.135 124.0
  -0.025 0.118 -0.039 0.065 -0.015 -0.050 -0.060 -0.048 0.107
3
                                                                 82.0
  -0.021
          0.001 -0.003 -0.003 -0.001 0.004 -0.005 -0.001
  0.048 0.032 0.007
                       0.018
                              0.020 -0.038 0.042 -0.102 0.060
                                                                  47.0
58
59 -0.071 0.045 0.046 0.149 0.095 -0.244 -0.166 0.210 -0.044
60 \ -0.015 \ \ 0.065 \ -0.014 \ \ -0.031 \ \ \ 0.055 \ \ -0.024 \ \ -0.082 \ \ -0.012 \ \ \ 0.095
                                                                  30.0
61 -0.043 0.100 -0.048 0.080 -0.025 -0.056 -0.040 -0.057
                                                         0.097
                                                                  79.0
   0.067 0.110 -0.010 -0.075 0.051 0.026 0.014 -0.094
                                                         0.081
                                                                  71.0
```

[63 rows x 23 columns]

In [20]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress
fig, ax = plt.subplots()
#----plot the correlation table as bar plot-----
ax.bar(df corr diff off.index, df corr diff off['wRCp'])
ax.set xticklabels(df corr diff off.index, rotation=90)
ax.set ylabel('correlation coefficent with wRC+')
plt.show()
```



When taking a deep look into the data, we can see that SLG, OBP, ISO, HR, HR/FB are the categories that had more than 0.5 correlation coefficient. These gives us an idea that players making a huge offensive leap were more inclined through power surge

Also if we focus on the batted ball type(LDp, GBp, FBp), we can see that increased FBp is a good indicator of increased wRC+, which probably imply that of these players making a leap, having more Fly ball is largely responsible for it.

If we focus on Batted ball direction(Pullp, Centp, Oppop), Pullp is the only one that has positive correlation with wRC+

If we focus on Quality of contact(Softp, Medp, Hardp), Hardp is expectedly the one.

Although we can not imply the player who has more Fly ball, Pull and Hard Contact percentage can have a leap year from this study, judging from the data of these already proven players, these stats may be largely responsible for their offensive leap that year.

Then I tried to build a model for this offensive part of data and see which categories are more responsible for the model

```
In [17]:
```

```
from sklearn.tree import DecisionTreeRegressor
  om chlearn model celection import train test colit
```

```
TIOM SKIEGIN. MOUEL SELECTION IMPOLE CLAIM CESC SPILE
 from sklearn import metrics
 from sklearn.metrics import r2_score
 import numpy as np
  # split data into X and y
 X2 = df diff off.loc[:,'HR':'Hardp']
 Y2 = df diff off.loc[:,'wRCp']
  # split data into train and test sets
 X2_train, X2_test, y2_train, y2_test = train_test_split(X2, Y2, test_size=0.2)
 # define the model
 model = DecisionTreeRegressor(criterion = 'mse', max depth=5)
 model.fit(X2 train, y2 train)
  # make predictions for test data
 y2_pred = model.predict(X2_test)
 print('Mean Absolute Error:', metrics.mean absolute error(y2 test, y2 pred))
 print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y2_test, y2_pred)))
 print('R Squared Score is:', r2_score(y2_test, y2_pred))
Mean Absolute Error: 11.23076923076923
Root Mean Squared Error: 14.403525209529036
R Squared Score is: 0.6913155253473262
In [18]:
 \textbf{for} \ \texttt{importance, name} \ \textbf{in} \ \texttt{sorted} \\ (\texttt{zip} \\ (\texttt{model.feature\_importances\_,} \ \texttt{X2\_train.columns}) \\ \textbf{,} \\ \texttt{reverse=True}) \\ \textbf{:} \\ \textbf{:
    print(name, importance)
SLG 0.6403694812602536
OBP 0.2051029123341741
AVG 0.10867789810146299
BABIP 0.010977159198084903
LDp 0.010589952263366184
HRFB 0.007661030864485111
GBFB 0.00611975838287273
FBp 0.004564634596278609
Centp 0.0027293366861886096
R 0.0016117882160652484
BBp 0.0015960480967677335
Softp 0.0
Pullp 0.0
Oppop 0.0
Medp 0.0
Kp 0.0
ISO 0.0
IFHp 0.0
IFFBp 0.0
Hardp 0.0
HR 0.0
GBp 0.0
```

As it turned out it is the SLG, OBP that took the large component of responsibility of this model, which is not surprising given the fact that these two categories are popular and largely seen as a standard for players' offensive output.

Conclusion

From this study we found out that the categories pushing players making a leap(four war differential from last year) were power related(SLG, OBP, ISO, HR, HR/FB) and more fly ball, pull and hard contact percentage. Although we should not making more assumption other than this from this study alone, this research may give us an idea of how player can approach in order for a potential offensive leap.