Introduction:

See how stats like PA, AB, H, 2B, 3B, HR, RBI, SB, BA, OBP, SLG, etc influence postseason birth

Methods:

Data was from Baseball Reference

Using SQL Server and Python(Spyder)

Gathering MLB regular season team stats from 2012-2019

Using logistic regression

In [1]:
```
import pandas as pd
import pyodbc

#import regular season stats from MLB teams who got into postseason dur
ing 2012-2019
#items include Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, S
O, BA, OBP, SLG, OPS, TB, GDP
#total rows are 8(years)*10(teams each year)=80

sql_conn = pyodbc.connect('''DRIVER={ODBC Driver 13 for SQL Server};
                            SERVER=ALLENHO\MSSQLSERVER002;
                            DATABASE=Playoffbound;
                            Trusted_Connection=yes''')
query = '''
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['19B$']
where Tm in ('WSN','LAD','MIL','ATL','STL','HOU','NYY','MIN','TBR','OA
K')
UNION ALL
```

```
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['18B$']
where Tm in ('BOS','LAD','MIL','ATL','CHC','HOU','NYY','CLE','COL','OA
K')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['17B$']
where Tm in ('BOS','LAD','COL','WSN','CHC','HOU','NYY','CLE','ARI','MI
N')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['16B$']
where Tm in ('TOR','CLE','BOS','BAL','TEX','NYM','CHC','LAD','WSN','SF
G')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['15B$']
where Tm in ('TOR','KCR','HOU','NYY','TEX','NYM','CHC','LAD','STL','PI
T')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['14B$']
where Tm in ('BAL','KCR','OAK','LAA','DET','WSN','STL','LAD','PIT','SF
G')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['13B$']
where Tm in ('BOS','TBR','OAK','CLE','DET','ATL','STL','LAD','PIT','CI
N')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['12B$']
```

```
where Tm in ('TEX','BAL','OAK','NYY','DET','ATL','STL','SFG','WSN','CI
N')
'''
df = pd.read_sql(query, sql_conn)

#stored as df_post
df_post = df

#import regular season stats from MLB teams who DIDN'T get into postsea
son during 2012-2019
#items are the same as above
#total rows are 8(years)*20(teams each year)=160
sql_conn = pyodbc.connect('''DRIVER={ODBC Driver 13 for SQL Server};
                            SERVER=ALLENHO\MSSQLSERVER002;
                            DATABASE=Playoffbound;
                            Trusted_Connection=yes''')
query = '''
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['19B$']
where Tm is not null and Tm not in ('WSN','LAD','MIL','ATL','STL','HO
U','NYY','MIN','TBR','OAK', 'LgAvg')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['18B$']
where Tm is not null and Tm not in ('BOS','LAD','MIL','ATL','CHC','HO
U','NYY','CLE','COL','OAK', 'LgAvg')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['17B$']
where Tm is not null and Tm not in ('BOS','LAD','COL','WSN','CHC','HO
U','NYY','CLE','ARI','MIN', 'LgAvg')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['16B$']
where Tm is not null and Tm not in ('TOR','CLE','BOS','BAL','TEX','NY
```

```
M','CHC','LAD','WSN','SFG', 'LgAvg')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['15B$']
where Tm is not null and Tm not in ('TOR','KCR','HOU','NYY','TEX','NY
M','CHC','LAD','STL','PIT', 'LgAvg')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['14B$']
where Tm is not null and Tm not in ('BAL','KCR','OAK','LAA','DET','WS
N','STL','LAD','PIT','SFG', 'LgAvg')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['13B$']
where Tm is not null and Tm not in ('BOS','TBR','OAK','CLE','DET','AT
L','STL','LAD','PIT','CIN', 'LgAvg')
UNION ALL
select Tm, BatAge, PA, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, BA, O
BP, SLG, OPS, TB, GDP
from [dbo].['12B$']
where Tm is not null and Tm not in ('TEX','BAL','OAK','NYY','DET','AT
L','STL','SFG','WSN','CIN', 'LgAvg')
'''
df = pd.read_sql(query, sql_conn)

#stored as df_npost
df_npost = df

#add each dataframe a new column named POST, which imply whether the te
am made the postseason
df_post['POST']= 1
df_npost['POST']= 0

#append two dataframes together
df_com=df_post.append(df_npost)
```

In [2]:
```python
#-----using logistic model to see team season OBP vs making the postsea
son that year or not-------------------------------

# Load libraries and functions
import statsmodels.api as sm
from statsmodels.formula.api import glm
import numpy as np

# Define the formula of the logistic model to see how well team regular
 season OBP predict postseason birth
model_formula = 'POST ~ OBP'
# Define the correct probability distribution and the link function of
 the response variable
link_function = sm.families.links.logit
model_family = sm.families.Binomial(link = link_function)

# Fit the model
OBP_fit = glm(formula = model_formula,
                data = df_com,
                family = model_family).fit()
# View the results of the OBP_fit model
print(OBP_fit.summary())
```

```
C:\Users\allen\anaconda3\lib\site-packages\ipykernel_launcher.py:12: De
precationWarning: Calling Family(..) with a link class as argument is d
eprecated.
Use an instance of a link class instead.
  if sys.path[0] == '':
```

```
                 Generalized Linear Model Regression Results

=======================================================================
=======
Dep. Variable:                   POST   No. Observations:
    240
Model:                            GLM   Df Residuals:
    238
Model Family:                Binomial   Df Model:
      1
Link Function:                  logit   Scale:
```

```
1.0000
Method:                          IRLS   Log-Likelihood:
-110.50
Date:                Tue, 04 Aug 2020   Deviance:
221.01
Time:                        20:55:09   Pearson chi2:
   212.
No. Iterations:                     5

Covariance Type:            nonrobust

==============================================================================
=======
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-------
Intercept    -46.1098      6.456     -7.142      0.000     -58.764
-33.456
OBP          141.2667     19.986      7.068      0.000     102.095
180.439
==============================================================================
=======
```

Intercept = -46.10975255466771 Slope = 141.2666813157292 ===> beta coefficient is where the likelihood takes on maximum value

In [3]:

```
# Extract coefficients from the fitted model OBP_fit
intercept, slope = OBP_fit.params

# Print coefficients
print('Intercept =', intercept)
print('Slope =', slope)

# Extract and print confidence intervals
print(OBP_fit.conf_int())
```

```
# Compute the multiplicative effect on the odds
print('Odds: \n', np.exp(OBP_fit.params))
```

```
Intercept = -46.10975255466771
Slope = 141.2666813157292
                    0           1
Intercept  -58.763962  -33.455543
OBP        102.094524  180.438839
Odds:
 Intercept    9.436021e-21
OBP          2.245640e+61
dtype: float64
```

In [4]:
```
# Define x at 0.333
x = 0.333

# Compute and print the estimated probability
est_prob = np.exp(intercept + slope*x)/(1+np.exp(intercept + slope*x))
print('Estimated probability at x = 0.333: ', round(est_prob, 4))

# Compute the slope of the tangent line for parameter beta at x
slope_tan = slope * est_prob * (1 - est_prob)
print('The rate of change in probability: ', round(slope_tan,4))
```

```
Estimated probability at x = 0.333:  0.7175
The rate of change in probability:  28.6344
```

imply that if your team has a season avg obp of 0.333, you have over 70% chance of breaking into postseason

In [5]:
```
# Estimated covariance matrix: OBP_cov
OBP_cov = OBP_fit.cov_params()
print(OBP_cov)

# Compute standard error (SE): std_error
std_error = np.sqrt(OBP_cov.loc['OBP', 'OBP'])
print('SE: ', round(std_error, 4))
```

```
# Compute Wald statistic
wald_stat = slope/std_error
print('Wald statistic: ', round(wald_stat,4))

# Extract and print confidence intervals
print(OBP_fit.conf_int())

# Compute confidence intervals for the odds
print(np.exp(OBP_fit.conf_int()))
```

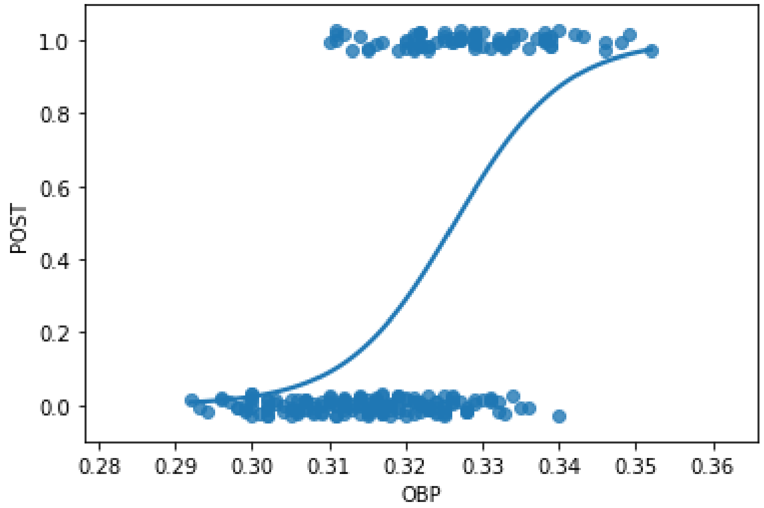
```
             Intercept         OBP
Intercept    41.684431 -128.994834
OBP        -128.994834  399.446666
SE:  19.9862
Wald statistic:  7.0682
                    0           1
Intercept  -58.763962  -33.455543
OBP        102.094524  180.438839
                      0             1
Intercept  3.013946e-26  2.954217e-15
OBP        2.183174e+44  2.309894e+78
```

when Wald statistics>2, imply the variable(obp here) is statistically significant

In [6]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Plot OBP and POST and add overlay with the logistic fit
sns.regplot(x = 'OBP', y = 'POST',
            y_jitter=0.03,
            data = df_com,
            logistic = True,
            ci = None)
# Display the plot
plt.show()
```



In [7]:
```
#-----below are multivariate logistic regression-----------------------
----------------------
#-----using BA, OBP, SLG, OPS as example
# Define model formula
formula = 'POST ~ BA+OBP+SLG+OPS'

# Fit GLM
model = glm(formula, data = df_com, family = sm.families.Binomial()).fi
t()

# Print model summary
print(model.summary())
```

```
                 Generalized Linear Model Regression Results

==============================================================================
=======
Dep. Variable:                   POST   No. Observations:
    240
Model:                            GLM   Df Residuals:
    235
Model Family:                Binomial   Df Model:
      4
```

```
Link Function:                        logit   Scale:
1.0000
Method:                                IRLS   Log-Likelihood:
-107.74
Date:                      Tue, 04 Aug 2020   Deviance:
215.49
Time:                              20:57:09   Pearson chi2:
   213.
No. Iterations:                           5

Covariance Type:                  nonrobust

========================================================================
=======
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------
-------
Intercept    -45.3516      6.671     -6.799      0.000     -58.426
-32.277
BA           -40.6929     24.142     -1.686      0.092     -88.009
  6.624
OBP         -379.9535    337.463     -1.126      0.260   -1041.369
281.462
SLG         -551.9089    340.083     -1.623      0.105   -1218.460
114.642
OPS          551.5402    339.361      1.625      0.104    -113.596    1
216.676
========================================================================
=======
```

In [8]:

```python
# Import functions
from statsmodels.stats.outliers_influence import variance_inflation_fac
tor

# Get variables for which to compute VIF and add intercept term
X = df_com[['BA', 'OBP', 'SLG', 'OPS']]
X['Intercept'] = 1
```

```python
# Compute and view VIF
vif = pd.DataFrame()
vif["variables"] = X.columns
vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.s
hape[1])]

# View results using print
print(vif)
```

```
   variables          VIF
0         BA     2.466888
1        OBP   584.537475
2        SLG  3010.441751
3        OPS  5539.163303
4  Intercept   805.919300
```

```
C:\Users\allen\anaconda3\lib\site-packages\ipykernel_launcher.py:6: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

if the VIF is above 2.5 should consider there is effect of multicollinearity on fitted model, result is quite accurate because OBP, SLG and OPS are highly correlated to each other

In [9]:

```python
# Compare deviance of null and residual model
diff_deviance = model.null_deviance - model.deviance

# Print the computed difference in deviance
print(diff_deviance)
```

```
90.03965842095988
```

In [10]:

```python
# define formula_BA
formula_BA = 'POST ~ BA'
```

```
# Fit GLM
model_BA = glm(formula_BA, data = df_com, family = sm.families.Binomial
()).fit()

# Compare deviance of null and residual model
diff_deviance = model_BA.null_deviance - model_BA.deviance

# Print the computed difference in deviance
print('Adding BA to the null model reduces deviance by: ',
      round(diff_deviance,3))
```

```
Adding BA to the null model reduces deviance by:  31.766
```

In [11]:

```
# define formula_OBP
formula_OBP = 'POST ~ OBP'

# Fit GLM
model_OBP = glm(formula_OBP, data = df_com, family = sm.families.Binomi
al()).fit()

# Compute the difference in adding OBP variable
diff_deviance_1 = model_OBP.deviance - model_BA.deviance

# Print the computed difference in deviance
print('Adding OBP to the BA model reduces deviance by: ',
      round(diff_deviance_1,3))
```

```
Adding OBP to the BA model reduces deviance by:  -52.756
```

In [12]:

```
# define formula_SLG
formula_SLG = 'POST ~ SLG'

# Fit GLM
model_SLG = glm(formula_SLG, data = df_com, family = sm.families.Binomi
al()).fit()

# Compute the difference in adding BA variable
diff_deviance_2 = model_SLG.deviance - model_BA.deviance
```

```
# Print the computed difference in deviance
print('Adding SLG to the BA model reduces deviance by: ',
      round(diff_deviance_2,3))
```

```
Adding SLG to the BA model reduces deviance by:  -10.567
```

In [13]:
```
# define formula_OPS
formula_OPS = 'POST ~ OPS'

# Fit GLM
model_OPS = glm(formula_OPS, data = df_com, family = sm.families.Binomi
al()).fit()

# Compute the difference in adding BA variable
diff_deviance_3 = model_OPS.deviance - model_BA.deviance

# Print the computed difference in deviance
print('Adding OPS to the BA model reduces deviance by: ',
      round(diff_deviance_3,3))
```

```
Adding OPS to the BA model reduces deviance by:  -29.213
```

imply that adding BA to the null model, it is valuable, however not valuable for adding OBP, SLG, OPS later

In [14]:
```
# Import function dmatrix()
from patsy import dmatrix
import numpy as np

# Construct model matrix with OBP
model_matrix = dmatrix('OBP', data = df_com, return_type = 'dataframe')
print(model_matrix.head())

# Construct model matrix with OBP and OPS
model_matrix_1 = dmatrix('OBP+OPS', data = df_com, return_type = 'dataf
rame')
print(model_matrix_1.head())
```

```
   Intercept    OBP
0        1.0  0.336
1        1.0  0.352
2        1.0  0.338
3        1.0  0.329
4        1.0  0.338
   Intercept    OBP    OPS
0        1.0  0.336  0.789
1        1.0  0.352  0.848
2        1.0  0.338  0.810
3        1.0  0.329  0.767
4        1.0  0.338  0.832
```

In [15]:

```
# Construct model matrix for OBP with log transformation
dmatrix('np.log(OBP)', data = df_com,
       return_type = 'dataframe').head()


# Define model formula
formula = 'POST ~ np.log(OBP)'
# Fit GLM
model_log_OBP = glm(formula, data = df_com,
                     family = sm.families.Binomial()).fit()
# Print model summary
print(model_log_OBP.summary())
```

```
                 Generalized Linear Model Regression Results

==============================================================================
=======
Dep. Variable:                   POST   No. Observations:
    240
Model:                            GLM   Df Residuals:
    238
Model Family:                Binomial   Df Model:
      1
Link Function:                  logit   Scale:
1.0000
```

```
Method:                          IRLS   Log-Likelihood:
-110.47
Date:                Tue, 04 Aug 2020   Deviance:
220.94
Time:                        21:00:38   Pearson chi2:
   212.
No. Iterations:                     5

Covariance Type:            nonrobust

======================================================================
========
                  coef    std err          z      P>|z|      [0.025
  0.975]
----------------------------------------------------------------------
--------
Intercept      51.0160      7.303      6.985      0.000      36.702
  65.330
np.log(OBP)    45.5547      6.460      7.052      0.000      32.894
  58.215
======================================================================
========
```