----------------Classification tree--

--- Introduction:

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

Methods:

Gathering MLB regular season team stats from 2012-2019, including

Using classification tree

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)
df_com['POST']=df_com['POST'].astype('int')

import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```python
#———constrained tree——
# Import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier

# Import train_test_split
from sklearn.model_selection import train_test_split

# Import accuracy_score
from sklearn.metrics import accuracy_score

# Instantiate dt
dt = DecisionTreeClassifier(max_depth=50, random_state=1)


# Split dataset into 80% train, 20% test
X_train, X_test, y_train, y_test = train_test_split(df_com['OBP'].value
s.reshape(-1, 1),
                 df_com['POST'].values.reshape(-1, 1),
                 test_size=0.1,
                 stratify=df_com['POST'].values.reshape(-1, 1),
                 random_state=1)

# Fit dt to the training setdt.fit(X_train,y_train)
dt.fit(X_train, y_train)

# Predict test set labels
y_pred = dt.predict(X_test)

# Compute test set accuracy
acc = accuracy_score(y_test, y_pred)
print("Test set accuracy: {:.2f}".format(acc))
```

```
Test set accuracy: 0.83
```

In [3]:
```python
#—unconstrained tree(entropy criterion)———-

# Import DecisionTreeClassifier from sklearn.tree
from sklearn.tree import DecisionTreeClassifier
```

```python
# Instantiate dt_entropy, set 'entropy' as the information criterion
dt_entropy = DecisionTreeClassifier(max_depth=100,
                                    criterion='entropy',
                                    random_state=1)

# Fit dt_entropy to the training set
dt_entropy.fit(X_train, y_train)

# Import accuracy_score from sklearn.metrics
from sklearn.metrics import accuracy_score

# Use dt_entropy to predict test set labels
y_pred = dt_entropy.predict(X_test)

# Evaluate accuracy_entropy
accuracy_entropy = accuracy_score(y_test, y_pred)
```

In [4]:

```python
#—unconstrained tree(gini criterion)————-
# Instantiate dt_entropy, set 'gini' as the information criterion
dt_gini = DecisionTreeClassifier(max_depth=100,
                                 criterion='gini',
                                 random_state=1)

# Fit dt_gini to the training set
dt_gini.fit(X_train, y_train)

# Import accuracy_score from sklearn.metrics
from sklearn.metrics import accuracy_score

# Use dt_gini to predict test set labels
y1_pred = dt_gini.predict(X_test)

# Evaluate accuracy_entropy
accuracy_gini = accuracy_score(y_test, y1_pred)

# Print accuracy_entropy
print('Accuracy achieved by using entropy: ', accuracy_entropy)
```

```
# Print accuracy_gini
print('Accuracy achieved by using the gini index: ', accuracy_gini)
```

```
Accuracy achieved by using entropy:  0.8333333333333334
Accuracy achieved by using the gini index:  0.8333333333333334
```

-----------------Regression tree--

Introduction:

see if plate discipline in baseball affected other stats including regular ones like BB%, K%, AVG, OBP, SLG, OPS, ISO, wOBA, wRC+, per game WAR

Methods:

Plate Discipline defined as (ZSwing%-OSwing%)/Swing%

Using Regression tree

In [7]:

```
import pandas as pd
import pyodbc

#connect with sql server, which already contains two tables from FanGra
phs
#1.https://www.fangraphs.com/leaders.aspx?pos=all&stats=bat&lg=all&qual
=1000&type=8&season=2019&month=0&season1=2010&ind=0&team=0&rost=0&age=0
&filter=&players=0&startdate=&enddate=
#2.https://www.fangraphs.com/leaders.aspx?pos=all&stats=bat&lg=all&qual
=1000&type=5&season=2019&month=0&season1=2010&ind=0&team=0&rost=0&age=0
&filter=&players=0&startdate=2010-01-01&enddate=2019-12-31
sql_conn = pyodbc.connect('''DRIVER={ODBC Driver 13 for SQL Server};
                            SERVER=ALLENHO\MSSQLSERVER002;
                            DATABASE=Plate discipline and winning corre
lation;
                            Trusted_Connection=yes''')

#sql query to grab data, including players from 2010-2019 with over 100
```

```
0 PA.
#Define plate discipline as [Z-Swing%]-[O-Swing%])/[Swing%]
#Also grab data from that corresponding player's BB%, K%, AVG, OBP, IS
O, wOBA, wRC+, WAR/PA for that period
query = '''
SELECT p.name, ([Z-Swing%]-[O-Swing%])/[Swing%] as plated, [BB%], [K%],
 AVG, OBP, ISO, wOBA, [wRC+], (d.WAR/d.PA) as per_war
FROM ['plate discipline 2010-2019 1000$'] p
JOIN ['dashboard stats 2010-2019 1000P$'] d
on p.name = d.name
order by WAR desc;
'''

#convert the data into dataframe
df = pd.read_sql(query, sql_conn)

#convert columns' type into string
df.columns = df.columns.astype(str)

#slice the data into only columns from plated(respresent plate discipli
ne) to per_war(represent per game war)
df_new = df.loc[:, 'plated':'per_war']
```

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt

# Import train_test_split
from sklearn.model_selection import train_test_split

# Import accuracy_score
from sklearn.metrics import accuracy_score

# Split dataset into 80% train, 20% test
X_train, X_test, y_train, y_test= train_test_split(df_new['plated'].val
ues.reshape(-1, 1),
                                                    df_new['per_war'].va
lues.reshape(-1, 1),
                                                    test_size=0.2,
                                                    random_state=3)
```

```python
# Import DecisionTreeRegressor from sklearn.tree
from sklearn.tree import DecisionTreeRegressor

# Instantiate dt
dt = DecisionTreeRegressor(max_depth=4,
                           min_samples_leaf=0.1,
                           random_state=3)

# Fit dt to the training set
dt.fit(X_train, y_train)

# Import mean_squared_error from sklearn.metrics as MSE
from sklearn.metrics import mean_squared_error as MSE

# Compute y_pred
y_pred = dt.predict(X_test)

# Compute mse_dt
mse_dt = MSE(y_test, y_pred)

# Compute rmse_dt
rmse_dt = mse_dt**(1/2)

# Print rmse_dt
print("Test set RMSE of dt: {:.2f}".format(rmse_dt))
```

```
Test set RMSE of dt: 0.00
```

In [9]:

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predict test set labels
y_pred_lr = regressor.predict(X_test)

# Compute mse_lr
```

```python
mse_lr = MSE(y_test, y_pred_lr)

# Compute rmse_lr
rmse_lr = mse_lr**(1/2)

# Print rmse_lr
print('Linear Regression test set RMSE: {:.2f}'.format(rmse_lr))

# Print rmse_dt
print('Regression Tree test set RMSE: {:.2f}'.format(rmse_dt))
```

```
Linear Regression test set RMSE: 0.00
Regression Tree test set RMSE: 0.00
```

1.Plate Discipline vs BB%

Test set RMSE of dt: 0.01 Linear Regression test set RMSE: 0.01 Regression Tree test set RMSE: 0.01

2.Plate Discipline vs K% Test set RMSE of dt: 0.05 Linear Regression test set RMSE: 0.05 Regression Tree test set RMSE: 0.05

3.Plate Discipline vs AVG Test set RMSE of dt: 0.02 Linear Regression test set RMSE: 0.02 Regression Tree test set RMSE: 0.02

4.Plate Discipline vs OBP Test set RMSE of dt: 0.02 Linear Regression test set RMSE: 0.02 Regression Tree test set RMSE: 0.02

5.Plate Discipline vs ISO Test set RMSE of dt: 0.04 Linear Regression test set RMSE: 0.04 Regression Tree test set RMSE: 0.04

6.Plate Discipline vs wOBA Test set RMSE of dt: 0.02 Linear Regression test set RMSE: 0.02 Regression Tree test set RMSE: 0.02

7.Plate Discipline vs wRC+ Test set RMSE of dt: 15.68 Linear Regression test set RMSE: 15.32 Regression Tree test set RMSE: 15.68

8.7.Plate Discipline vs per game WAR Test set RMSE of dt: 0.00 Linear Regression test set RMSE: 0.00 Regression Tree test set RMSE: 0.00

Conclusion:Right now as the result shows, there is not much difference betweeen using general linear regression and regression tree based on the same RMSE.