# Introduction:

This research aims to estimate the bias and variance for both linear regression model and regression tree model

# Methods:

Data was from FanGraphs

Use linear regression and regression tree

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

mlb players' stats from during 2010-2019

items include BB%, K%, AVG, OBP, ISO, wOBA, wRC+

## Step 1: Import data

In [1]:

```
import pandas as pd
import pyodbc

#connect with sql server, which already contains two tables from FanGraphs
#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=
lter=&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=
lter=&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 correlation;
                            Trusted_Connection=yes''')

#sql query to grab data, including players from 2010-2019 with over 1000 PA.
#Define plate discipline as [Z-Swing%]-[O-Swing%])/[Swing%]
#Also grab data from that corresponding player's BB%, K%, AVG, OBP, ISO, wOBA, wRC+, WAR/PA for th
at 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 discipline) to per_war(represent pe
r game war)
df_new = df.loc[:, 'plated':'per_war']
print(df_new)
```

```
       plated    BB%     K%    AVG    OBP    ISO   wOBA   wRC+   per_war
0    0.908136  0.152  0.212  0.305  0.419  0.276  0.419  172.0  0.013920
1    0.844944  0.094  0.123  0.302  0.371  0.155  0.357  129.0  0.010319
2    1.138686  0.170  0.177  0.306  0.428  0.210  0.403  153.0  0.007821
3    0.997680  0.122  0.184  0.286  0.379  0.195  0.371  136.0  0.007404
4    0.725490  0.077  0.133  0.300  0.359  0.196  0.364  131.0  0.007434
..        ...    ...    ...    ...    ...    ...    ...    ...       ...
549  0.667283  0.034  0.123  0.242  0.267  0.143  0.282   72.0 -0.001271
550  0.649903  0.056  0.176  0.254  0.297  0.102  0.284   75.0 -0.002210
551  0.789588  0.062  0.212  0.200  0.257  0.097  0.247   49.0 -0.002296
```

```
552  0.705376  0.046  0.142  0.272  0.310  0.093  0.297   70.0 -0.003829
553  0.717724  0.073  0.191  0.258  0.318  0.161  0.321  101.0 -0.003015

[554 rows x 9 columns]
```

## Step2: estimate the MSE, bias and variance for linear regression model

In [35]:

```python
# estimate the bias and variance for a regression model
import numpy as np
import matplotlib.pyplot as plt
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from mlxtend.evaluate import bias_variance_decomp

# separate into inputs and outputs
X1= df_new.loc[:,'plated':'wOBA'].values
y1= df_new.loc[:,'wRC+'].values

# split the data
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.25, random_state=1)

# define the model
model1 = LinearRegression()

# estimate bias and variance
mse1, bias1, var1 = bias_variance_decomp(model1, X1_train, y1_train, X1_test, y1_test, loss='mse',
num_rounds=200, random_seed=1)

# summarize results
print('MSE1: %.2f' % mse1)
print('Bias1: %.2f' % bias1)
print('Variance1: %.2f' % var1)
```

```
MSE1: 13.85
Bias1: 13.54
Variance1: 0.30
```

## Step3: estimate the MSE, bias and variance for regression tree model

In [36]:

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

# separate into inputs and outputs
X2= df_new.loc[:,'plated':'wOBA'].values
y2= df_new.loc[:,'wRC+'].values

# split the data
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.25, random_state=1)

# define the model
model2 = DecisionTreeRegressor(max_depth=4,
                               min_samples_leaf=0.1,
                               random_state=3)

# estimate bias and variance
mse2, bias2, var2 = bias_variance_decomp(model2, X2_train, y2_train, X2_test, y2_test, loss='mse',
num_rounds=200, random_seed=1)

# summarize results
print('MSE2: %.2f' % mse2)
print('Bias2: %.2f' % bias2)
print('Variance2: %.2f' % var2)
```

```
MSE2: 52.64
Bias2: 44.95
```

```
Variance2: 7.69
```

**Result: As it turned out, MSE in simpler model like linear regression is expectedly largely from bias. On the other hand, MSE in more complex model like regression tree expectedly has a bigger portion of variance.**