master-dataset.ipynb

Scope: Construct the working dataset based on three files: FanGraphs-Pitching-Standard.csv, FanGraphs-Pitching-Advanced.csv, and FanGraphcs-Pitching-Value.csv. Export the created dataset to a .csv file for us in WAR-Predictor.ipynb.

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

In [3]:
```
#Reading FanGraphs data into Pandas dataframes
pitching_standard = pd.read_csv('FanGraphs-Pitching-Standard.csv')
pitching_advanced = pd.read_csv('FanGraphs-Pitching-Advanced.csv')
pitching_value = pd.read_csv('FanGraphs-Pitching-Value.csv')
```

In [4]:
```
#Removing irrelevant statistics from pitching_value
pitching_value = pitching_value.drop(columns=["RA9-WAR", "BIP-Wins", "LOB-Wins", "FDP-Wins", "RAR", "Dollars"])

#Joining datasets to create master dataset
pitching_mds = pd.merge(pitching_standard, pitching_advanced, on=["playerid", "Season", "Name", "Team"])
pitching_mds = pd.merge(pitching_mds, pitching_value, on=["playerid", "Season", "Name", "Team"])

#Removing duplicate columns
pitching_mds = pitching_mds.drop(columns=["ERA_y"])
pitching_mds = pitching_mds.rename(columns={"ERA_x" : "ERA"})

#Export master dataset to csv file
pitching_mds.to_csv('pitching-masterdataset.csv', index=False)
```

In [ ]:
```

```

WAR-Predictor.ipynb

Scope: Create a multiple linear regression model and random forest regression model to predict a pitcher's WAR based on other statistics.

In [1]:
```
#Package Imports
import pandas as pd
import numpy as np
import sklearn.metrics as metrics
import os
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor
```

In [2]:
```
#Load master dataset
df_pitching = pd.read_csv('pitching-masterdataset.csv')

print(df_pitching.describe())
```

```
            Season            W            L          ERA            G  \
count  3422.000000  3422.000000  3422.000000  3422.000000  3422.000000
mean   2015.751023     6.230567     5.939801     3.900009    40.379018
std       3.072060     4.267957     3.654775     1.172525    20.029921
min    2011.000000     0.000000     0.000000     0.540000     8.000000
25%    2013.000000     3.000000     3.000000     3.100000    25.000000
50%    2016.000000     5.000000     5.000000     3.810000    33.000000
75%    2018.000000     9.000000     8.000000     4.607500    61.000000
max    2021.000000    24.000000    19.000000     9.580000    85.000000

                GS           CG          ShO           SV          HLD  ...  \
count  3422.000000  3422.000000  3422.000000  3422.000000  3422.000000  ...
mean     13.345120     0.274693     0.134424     3.141146     4.746055  ...
std      12.577168     0.759965     0.443922     8.742923     7.737213  ...
min       0.000000     0.000000     0.000000     0.000000     0.000000  ...
25%       0.000000     0.000000     0.000000     0.000000     0.000000  ...
50%      12.000000     0.000000     0.000000     0.000000     0.000000  ...
75%      26.000000     0.000000     0.000000     1.000000     8.000000  ...
max      35.000000    11.000000     6.000000    57.000000    41.000000  ...

              WHIP        BABIP         ERA-         FIP-        xFIP-  \
count  3422.000000  3422.000000  3422.000000  3422.000000  3422.000000
mean      1.277022     0.289800    94.783168    96.512858    97.558153
std       0.200099     0.032317    27.547303    21.260601    16.754184
min       0.550000     0.170000    13.000000    21.000000    23.000000
25%       1.140000     0.269000    76.000000    83.000000    87.000000
50%       1.270000     0.291000    94.000000    97.000000    98.000000
75%       1.410000     0.311000   111.000000   110.000000   109.000000
max       2.090000     0.388000   220.000000   178.000000   158.000000

               FIP          E-F         xFIP        SIERA          WAR
count  3422.000000  3422.000000  3422.000000  3422.000000  3422.000000
mean      3.967896    -0.067987     4.009936     3.896634     1.287084
std       0.911166     0.747558     0.722406     0.771167     1.404701
min       0.780000    -2.400000     0.880000     0.760000    -1.600000
25%       3.360000    -0.550000     3.550000     3.400000     0.300000
50%       3.950000    -0.110000     4.030000     3.930000     1.000000
75%       4.540000     0.400000     4.480000     4.430000     1.900000
max       7.650000     2.880000     6.530000     6.280000     9.000000

[8 rows x 39 columns]
```

In [3]:
```
#Removing % signs from fields that contain them
for x in range(len(df_pitching)):
    df_pitching['K%'][x] = df_pitching['K%'][x][:-1]
    df_pitching['BB%'][x] = df_pitching['BB%'][x][:-1]
    df_pitching['K-BB%'][x] = df_pitching['K-BB%'][x][:-1]
    df_pitching['LOB%'][x] = df_pitching['LOB%'][x][:-1]

#Split the master dataset into training set and testing set
df_training, df_testing = train_test_split(df_pitching, test_size=0.25, random_state=42, shuffle=True)

print("Number of records in training set: " + str(len(df_training)))
print("Number of records in testing set: " + str(len(df_testing)))
print("Total number of records in master dataset: " + str(len(df_pitching)))
```

```
/var/folders/g0/41rv_ph94r71jngnydc3c1840000gn/T/ipykernel_13369/3217800170.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_pitching['K%'][x] = df_pitching['K%'][x][:-1]
/var/folders/g0/41rv_ph94r71jngnydc3c1840000gn/T/ipykernel_13369/3217800170.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_pitching['BB%'][x] = df_pitching['BB%'][x][:-1]
/var/folders/g0/41rv_ph94r71jngnydc3c1840000gn/T/ipykernel_13369/3217800170.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_pitching['K-BB%'][x] = df_pitching['K-BB%'][x][:-1]
/var/folders/g0/41rv_ph94r71jngnydc3c1840000gn/T/ipykernel_13369/3217800170.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_pitching['LOB%'][x] = df_pitching['LOB%'][x][:-1]
Number of records in training set: 2566
Number of records in testing set: 856
Total number of records in master dataset: 3422
```

In [4]:
```
#Determine x and y variables. Let x represent the independent variables and y represent the dependent variable.
y_train = df_training['WAR']
X_train = df_training.drop(columns=['WAR', 'Season', 'Name', 'Team', 'playerid'])

y_test = df_testing['WAR']
X_test = df_testing.drop(columns=['WAR', 'Season', 'Name', 'Team', 'playerid'])
```

In [5]:
```
#Construct multiple regression model
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

#Apply multiple linear regression model prediction
y_pred_linear = regr.predict(X_test)

#R-squared results to determine the success of the multiple linear regression model
r2_linear = r2_score(y_test, y_pred_linear)
print("Multiple Linear Regression R-Squared Score: " + str(r2_linear))
```

```
Multiple Linear Regression R-Squared Score: 0.9747554731866983
```

In [6]:
```
#Construct random forest regression model
forest_regr = RandomForestRegressor(n_estimators=100, random_state=0)
forest_regr.fit(X_train, y_train)

#Apply random forest model prediction
y_pred_forest = forest_regr.predict(X_test)
```

In [7]:
```
def regression_metrics(y_test, y_pred):
    #Regression metrics
    explained_variance = metrics.explained_variance_score(y_test, y_pred)
    mean_absolute_error = metrics.mean_absolute_error(y_test, y_pred)
    mean_squared_error = metrics.mean_squared_error(y_test, y_pred)
    median_absolute_error = metrics.median_absolute_error(y_test, y_pred)

    #Output regression metrics
    print("Explained Variance: " + str(explained_variance))
    print("Mean Absolute Error: " + str(mean_absolute_error))
    print("Mean Squared Error: " + str(mean_squared_error))
    print("Median Absolute Error: " + str(median_absolute_error))
```

In [9]:
```
#Regression metrics for Linear Regression model
print("Multiple Linear Regression Statistics:")
regression_metrics(y_test, y_pred_linear)

print("----------------------------------------")

#Regression metrics for Random Forest Model
print("Random Forest Model Statistics:")
regression_metrics(y_test, y_pred_forest)
```

```
Multiple Linear Regression Statistics:
Explained Variance: 0.9747686033211165
Mean Absolute Error: 0.16817630745819503
Mean Squared Error: 0.048794760547809166
Median Absolute Error: 0.13078011257812044
----------------------------------------
Random Forest Model Statistics:
Explained Variance: 0.9701821411621511
Mean Absolute Error: 0.17526985981308413
Mean Squared Error: 0.05766230490654204
Median Absolute Error: 0.131
```

In [ ]:
```

```