In [5]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from xgboost import XGBRegressor

nfl=pd.read_csv('https://raw.githubusercontent.com/steve122192/Unit-2-Project/maste

# Clean NFL Data
nfl = nfl[nfl['Rk'] != 'Rk']
nfl['To'] = pd.to_numeric(nfl['To'])
nfl['From'] = pd.to_numeric(nfl['From'])
nfl['Games GS'] = pd.to_numeric(nfl['Games GS'])

#Engineer Target ('starts_per_season')
nfl['Seasons'] = (nfl['To']-nfl['From'])+1
nfl = nfl[['Player','Seasons','Games GS']]
nfl['starts_per_season'] = nfl['Games GS']/nfl['Seasons']
nfl = nfl[['Player','starts_per_season']]


college = pd.read_csv('https://raw.githubusercontent.com/steve122192/Unit-2-Project

# Merge and Clean College Data
df = pd.merge(nfl, college, on='Player', how='outer')
df.dropna(subset=['Rk'], inplace=True)
cols = df.columns[3:5]
df[cols] = df[cols].apply(pd.to_numeric, errors='coerce', axis=1)
cols = df.columns[6:]
df[cols] = df[cols].apply(pd.to_numeric, errors='coerce', axis=1)
df = df[df['Player'] != 'Player']
df['School'].fillna(value='multiple', inplace=True)

# Engineer 'per_year' features
df['pass_per_year'] = df['Passing Att']/(df['To']-df['From']+1)
df = df[df['pass_per_year']>200]
df['cmp_per_year'] = df['Passing Cmp']/(df['To']-df['From']+1)
df['yds_per_year'] = df['Passing Yds']/(df['To']-df['From']+1)
df['tds_per_year'] = df['Passing TD']/(df['To']-df['From']+1)
df['int_per_year'] = df['Passing Int']/(df['To']-df['From']+1)

#Clean before Merge
df['starts_per_season'].fillna(value=0, inplace=True)
df = df[df['To']<2017]
df.drop(['Rk','From','To'], axis=1, inplace=True)

# Clean & Merge Combine Data
combine = pd.read_csv('https://raw.githubusercontent.com/steve122192/Unit-2-Project
combine.drop(['Year','POS','Wonderlic'], axis=1, inplace=True)
combine.rename(columns = {'Name':'Player'}, inplace = True)
df = pd.merge(df, combine, on='Player', how='left')
```

```
df['no_combine'] = df['College'].isnull()

# Engineer 'power_5' feature
list1 = ['Boston College','Clemson','Duke','Florida State', 'Georgia Tech',
         'Louisville','Miami (FL)','North Carolina','North Carolina State',
         'Pittsburgh','Syracuse','Virginia','Virginia Tech','Wake Forest',
         'Notre Dame','Illinois','Indiana','Iowa','Maryland','Michigan',
         'Michigan State','Minnesota','Nebraska','Northwestern','Ohio State',
         'Penn State','Purdue','Rutgers','Wisconsin','Baylor','Iowa State',
         'Kansas','Kansas State','Oklahoma','Oklahoma State','Texas Christian',
         'Texas','Texas Tech','West Virginia','Arizona','Arizona State',
         'California','UCLA','Colorado','Oregon','Oregon State','Southern Californi
         'Stanford','Utah','Washington','Washington State','Alabama', 'Arkansas',
         'Auburn','Florida','Georgia','Kentucky','Louisiana State','Mississippi',
         'Mississippi State','South Carolina','Tennessee','Texas A&M','Vanderbilt',
         'multiple']
df['power_5'] = df['School'].apply(lambda school: school in list1)

# Clean and Impute combine data
df.drop(['College','Bench Press'], axis=1, inplace=True)
forty_max = df['40 Yard'].max()
vert_min = df['Vert Leap (in)'].min()
broad_min = df['Broad Jump (in)'].min()
shuttle_max = df['Shuttle'].max()
cone_max = df['3Cone'].max()
df['40 Yard'].fillna(value=forty_max, inplace=True)
df['Vert Leap (in)'].fillna(value=vert_min, inplace=True)
df['Broad Jump (in)'].fillna(value=broad_min, inplace=True)
df['Shuttle'].fillna(value=shuttle_max, inplace=True)
df['3Cone'].fillna(value=cone_max, inplace=True)
df.drop(['School'], axis=1, inplace=True)

# Clean & Merge conference championship data
conference = pd.read_csv('https://raw.githubusercontent.com/steve122192/Unit-2-Proj
conference = conference[['Player','G']]
conference = conference[conference['Player'] !='Player']
conference['G'] = pd.to_numeric(conference['G'])
conference.rename(columns = {'G':'Conference_Championships'}, inplace=True)
df = pd.merge(df, conference, on='Player', how='left')
df['Conference_Championships'].fillna(value=0, inplace=True)

# Clean Win Data
wins = pd.read_csv('https://raw.githubusercontent.com/steve122192/Unit-2-Project/ma
wins = wins[wins['Player'] !='Player']
wins['G'] = pd.to_numeric(wins['G'])
wins['To'] = pd.to_numeric(wins['To'])
wins['From'] = pd.to_numeric(wins['From'])

# Engineer 'wins_per_season' feature
wins['Seasons'] = (wins['To']-wins['From']+1)
wins['wins_per_season'] = (wins['G']/wins['Seasons'])
wins = wins[['Player','wins_per_season']]

# Merge & Clean win data
df = pd.merge(df, wins, on='Player', how='left')
df['wins_per_season'].fillna(value=0, inplace=True)
```

```
df.drop('Player', axis=1, inplace=True)

# Rename Columns
df.columns = ['starts_per_season','games_played','passing_completions','passing_att
              'passing_percentage','passing_yards','passing_tds','passing_ints',
              'passer_rating','passes_per_year','completions_per_year','yards_per_y
              'tds_per_year','ints_per_year','height','weight','forty_yard_dash',
              'vert_leap','broad_jump','shuttle_run','three_cone','no_combine_atten
              'power_five_conf','conference_championships','wins_per_year']


train, test =  train_test_split(df, train_size=.80, test_size=.2, random_state=42)

test.reset_index(drop=True, inplace=True)
train.reset_index(drop=True, inplace=True)

target = 'starts_per_season'
X_train = train.drop(target, axis=1)
y_train = train[target]
X_test = test.drop(target, axis=1)
y_test = test[target]


# Use best parameters
pipeline = make_pipeline(
    SimpleImputer(strategy='mean'),
    StandardScaler(),
    XGBRegressor(n_estimators=10, learning_rate=0.1, max_depth=10, n_jobs=-1)
)

pipeline.fit(X_train, y_train)
```

```
[16:04:04] WARNING: src/objective/regression_obj.cu:152: reg:linear is now deprecate
d in favor of reg:squarederror.
```

Out[5]:

```
Pipeline(memory=None,
         steps=[('simpleimputer',
                 SimpleImputer(add_indicator=False, copy=True, fill_value=None,
                               missing_values=nan, strategy='mean',
                               verbose=0)),
                ('standardscaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('xgbregressor',
                 XGBRegressor(base_score=0.5, booster='gbtree',
                              colsample_bylevel=1, colsample_bynode=1,
                              colsample_bytree=1, gamma=0,
                              importance_type='gain', learning_rate=0.1,
                              max_delta_step=0, max_depth=10,
                              min_child_weight=1, missing=None, n_estimators=10,
                              n_jobs=-1, nthread=None, objective='reg:linear',
                              random_state=0, reg_alpha=0, reg_lambda=1,
                              scale_pos_weight=1, seed=None, silent=None,
                              subsample=1, verbosity=1))],
         verbose=False)
```

In [6]:

```
y_pred = pipeline.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
```

```
r2 = r2_score(y_test, y_pred)
print('Test MAE: ',mae)
print('Test R2: ',r2)
```

```
Test MAE:  1.4292250435490166
Test R2:  0.21357401958355315
```

In [3]:
```
import joblib
import sklearn
import xgboost
print(f'joblib=={joblib.__version__}')
print(f'scikit-learn=={sklearn.__version__}')
print(f'xgboost=={xgboost.__version__}')
```

```
joblib==0.14.1
scikit-learn==0.22.1
xgboost==0.90
```

In [7]:
```
from joblib import dump
dump(pipeline, 'pipeline.joblib', compress=True)
```

Out[7]:
```
['pipeline.joblib']
```

In [8]:
```
df.columns.tolist()
```

Out[8]:
```
['starts_per_season',
 'games_played',
 'passing_completions',
 'passing_attempts',
 'passing_percentage',
 'passing_yards',
 'passing_tds',
 'passing_ints',
 'passer_rating',
 'passes_per_year',
 'completions_per_year',
 'yards_per_year',
 'tds_per_year',
 'ints_per_year',
 'height',
 'weight',
 'forty_yard_dash',
 'vert_leap',
 'broad_jump',
 'shuttle_run',
 'three_cone',
 'no_combine_attendance',
 'power_five_conf',
 'conference_championships',
 'wins_per_year']
```

In [ ]:
```

```