# English Premier League (EPL) Pythagorean Predictor

In [1]:

```
# Importing Packages

import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
import seaborn as sns

# Custom
import warnings
warnings.filterwarnings('ignore')
%config InlineBackend.figure_formats = ['svg'] # makes everything svg by default
%matplotlib inline
```

In [3]:

```
# Read Data

dataset = pd.read_excel('ds/EPL2017-18.xlsx')
print(dataset.columns.tolist())

display( dataset.head() )
```

```
['Date', 'HomeTeam', 'AwayTeam', 'FTHG', 'FTAG', 'FTR']
```

| | Date | HomeTeam | AwayTeam | FTHG | FTAG | FTR |
|---|---|---|---|---|---|---|
| **0** | 20170811 | Arsenal | Leicester | 4 | 3 | H |
| **1** | 20170812 | Brighton | Man City | 0 | 2 | A |
| **2** | 20170812 | Chelsea | Burnley | 2 | 3 | A |
| **3** | 20170812 | Crystal Palace | Huddersfield | 0 | 3 | A |
| **4** | 20170812 | Everton | Stoke | 1 | 0 | H |

In [4]:

```
# Cleanup
dataset['count'] = 1

dataset['hwinvalue'] = np.where( dataset['FTR']=='H',1, np.where(dataset['FTR']=='D',.5,0) )
dataset['awinvalue'] = np.where( dataset['FTR']=='A',1, np.where(dataset['FTR']=='D',.5,0) )

home1 = dataset[dataset.Date < 20180000].groupby(['HomeTeam'])['count','hwinvalue', 'FTHG','FTAG'].sum().reset_
home1 = home1.rename(columns={'HomeTeam':'Team','count':'MPh','FTHG':'GFh', 'FTAG':'GAh'})
away1 = dataset[dataset.Date < 20180000].groupby(['AwayTeam'])['count','awinvalue', 'FTHG','FTAG'].sum().reset_
away1 = away1.rename(columns={'AwayTeam':'Team','count':'MPa','FTHG':'GAa','FTAG':'GFa'}) # because my goals ir


home2 = dataset[dataset.Date > 20180000].groupby(['HomeTeam'])['count','hwinvalue', 'FTHG','FTAG'].sum().reset_
home2 = home2.rename(columns={'HomeTeam':'Team','count':'MPh','FTHG':'GFh', 'FTAG':'GAh'})
away2 = dataset[dataset.Date > 20180000].groupby(['AwayTeam'])['count','awinvalue', 'FTHG','FTAG'].sum().reset_
away2 = away2.rename(columns={'AwayTeam':'Team','count':'MPa','FTHG':'GAa','FTAG':'GFa'}) # because my goals ir

half1 = pd.merge(home1, away1, on="Team")
half2 = pd.merge(home2, away2, on="Team")
```

In [7]:

```
# Evaluations
halves = [half1, half2]

for half in halves:
    half["MP"] = half["MPh"] + half["MPa"]
    half["wValue"] = half["hwinvalue"] + half["awinvalue"]
    half["GF"] = half["GFh"] + half["GFa"]
    half["GA"] = half["GAh"] + half["GAa"]


half1["pyth1"] = (half1["GF"]**2) / (half1["GF"]**2 + half1["GA"]**2)
half2["wpc2"] = half2["wValue"]/half2["MP"]
```

In [8]:

```
# Cleaned up Dataset
dropCols = ["MPh", "hwinvalue", "GFh", "GAh", "MPa", "awinvalue", "GFa", "GAa"]

for half in halves:
    display(
        half.drop(columns = dropCols).head()
    )
```

| | Team | MP | wValue | GF | GA | pyth | pyth1 |
|---|---|---|---|---|---|---|---|
| **0** | Arsenal | 21 | 13.5 | 38 | 26 | 0.681132 | 0.681132 |
| **1** | Bournemouth | 21 | 7.5 | 20 | 32 | 0.280899 | 0.280899 |
| **2** | Brighton | 21 | 8.5 | 15 | 25 | 0.264706 | 0.264706 |
| **3** | Burnley | 21 | 12.5 | 18 | 17 | 0.528548 | 0.528548 |
| **4** | Chelsea | 21 | 15.5 | 39 | 14 | 0.885847 | 0.885847 |

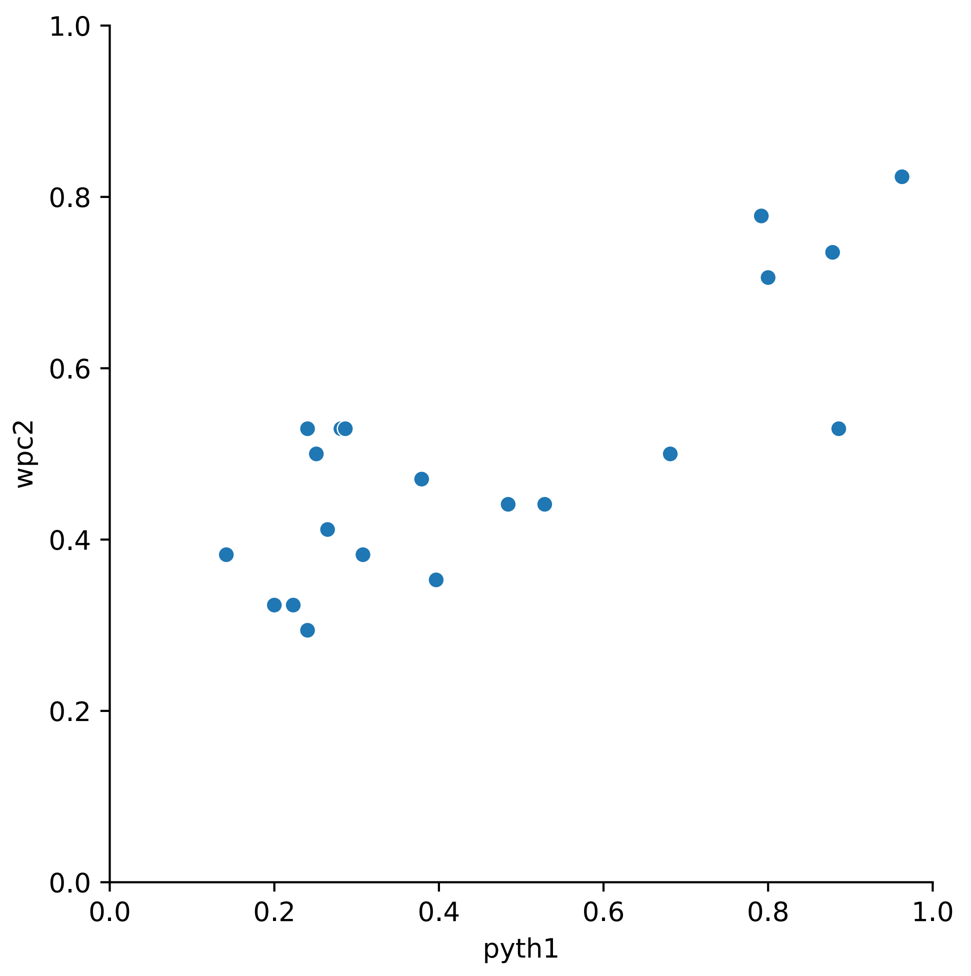
| | Team | MP | wValue | GF | GA | wpc | wpc2 |
|---|---|---|---|---|---|---|---|
| **0** | Arsenal | 17 | 8.5 | 36 | 25 | 0.500000 | 0.500000 |
| **1** | Bournemouth | 17 | 9.0 | 25 | 29 | 0.529412 | 0.529412 |
| **2** | Brighton | 17 | 7.0 | 19 | 29 | 0.411765 | 0.411765 |
| **3** | Burnley | 17 | 7.5 | 18 | 22 | 0.441176 | 0.441176 |
| **4** | Chelsea | 17 | 9.0 | 23 | 24 | 0.529412 | 0.529412 |

In [15]:

```
# using half 1 pyth as predictor for half 2 wpc
predictor = pd.merge(half1, half2, on = "Team")
```

In [16]:

```
# Plotting
sns.relplot(x="pyth1", y="wpc2", data = predictor)
plt.xlim(0, 1), plt.ylim(0, 1)
plt.show()
```




In [19]:

```
# Regression

regression = smf.ols(formula = 'wpc2 ~ pyth1', data=predictor).fit()
regression.summary()
```

Out[19]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | wpc2 | **R-squared:** | 0.633 |
| **Model:** | OLS | **Adj. R-squared:** | 0.613 |
| **Method:** | Least Squares | **F-statistic:** | 31.06 |
| **Date:** | Tue, 25 Jan 2022 | **Prob (F-statistic):** | 2.73e-05 |
| **Time:** | 21:57:27 | **Log-Likelihood:** | 19.534 |
| **No. Observations:** | 20 | **AIC:** | -35.07 |
| **Df Residuals:** | 18 | **BIC:** | -33.08 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 0.2897 | 0.043 | 6.690 | 0.000 | 0.199 | 0.381 |
| **pyth1** | 0.4543 | 0.082 | 5.573 | 0.000 | 0.283 | 0.626 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 4.877 | **Durbin-Watson:** | 2.048 |
| **Prob(Omnibus):** | 0.087 | **Jarque-Bera (JB):** | 1.521 |
| **Skew:** | -0.033 | **Prob(JB):** | 0.467 |
| **Kurtosis:** | 1.650 | **Cond. No.** | 4.65 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.