Simple NFL Strength of Performance Analysis

This notebook performs a simple strength of performance analysis. The objective is to quantify how well a team has performed with respect to the strength of their opponent. This is done through estimating a team's margin of victory/loss against an average team at a neutral site - we call this metric the team's SOP.

Consider a game between Team A and Team B where team A is at home.

- Team A's SOP = 4.1
- Team B's SOP = -1.5
- Assume the average NFL homefield advantage is $+1.3$

We expect Team A to win by a margin of $4.1 - (-1.5) + 1.3 = 6.9$ points on average. Suppose Team A wins by 3 points, our error is $6.9 - 3 = 3.9$ points.

To determine each team's SOP, we simply minimize this error accross every game.

In [49]:
```python
# load in the necessary packages
import nfl_data_py
import numpy as np
import pandas as pd
from scipy.optimize import minimize
from IPython.display import display, HTML
```

In [2]:
```python
def compute_home_advan(scores: pd.DataFrame) -> float:
    """
    Compute the average home field advantage for the given scores data
    """
    return np.mean(scores["home_score"]) - np.mean(scores["away_score"])
```

In [8]:
```python
def get_scores(season: int) -> pd.DataFrame:
    """
    Get the scores data for the given season
    """
    scores = nfl_data_py.import_schedules([season])
    scores = scores[["home_team", "away_team", "home_score", "away_score"]].dropna()
    return scores
```

In [5]:
```python
def init_sop(X: tuple[float], scores: pd.DataFrame):
    """
    Initialize the SOP dataframe
    """
    sop = pd.DataFrame(
        {"team": sorted(list(scores["home_team"].unique())), "sop": [x for x in X]}
    )
    return sop
```

In [6]:
```python
def sop_error(sop: pd.DataFrame, scores: pd.DataFrame, home_advan: float) -> float:
    """
    Compute the mean error for the give, SOP, scores, and home field advantage
    """

    # create a table with the scores of each game and each teams SOP
    table = pd.merge(left=scores, right=sop, left_on="home_team", right_on="team")
    table = pd.merge(left=table, right=sop, left_on="away_team", right_on="team")
    table = table.rename({"sop_x": "home_sop", "sop_y": "away_sop"}, axis="columns")
    table = table[["home_score", "away_score", "home_sop", "away_sop"]]

    # compute the expected score differential
    exp_diff = table["home_sop"] - table["away_sop"] + home_advan

    # compute the real score differential
    real_diff = table["home_score"] - table["away_score"]

    # compute and return the error in expected vs. real scores
    return np.sqrt(np.mean(np.square(exp_diff - real_diff)))
```

In [35]:
```python
def objective(X: tuple[float], scores: pd.DataFrame, home_advan: float) -> float:
    """
    Define the objective function for minimization
    """
    sop = init_sop(X, scores)
    return sop_error(sop, scores, home_advan)
```

In [36]:
```python
def compute_sop(season: int) -> pd.DataFrame:
    # get the scores data
    scores = get_scores(season)

    # compute the home field advantage
    home_advan = compute_home_advan(scores)

    # minimize the error
    solution = minimize(
        objective, tuple(0 for _ in range(32)), args=(scores, home_advan), tol=1e-6
    )

    # return the SOP table
    sop_table = init_sop(solution.x, scores)
    return sop_table
```

Now let's run the algorithm a few seasons.

In [ ]:
```python
# 2024
sop_table = compute_sop(2024)
sop2024 = sop_table.sort_values(by="sop", ascending=False).style.hide(axis="index")

# 2023
sop_table = compute_sop(2023)
sop2023 = sop_table.sort_values(by="sop", ascending=False).style.hide(axis="index")

# 2022
sop_table = compute_sop(2022)
sop2022 = sop_table.sort_values(by="sop", ascending=False).style.hide(axis="index")

# 2008
sop_table = compute_sop(2008)
sop2008 = sop_table.sort_values(by="sop", ascending=False).style.hide(axis="index")

# 2007
sop_table = compute_sop(2007)
sop2007 = sop_table.sort_values(by="sop", ascending=False).style.hide(axis="index")

# convert dataframes to HTML
html = f"""
<div style="display: flex; justify-content: space-between;">
    <div style="width: 48%; display: flex; flex-direction: column; align-items: center;">
        <h3>2024</h3>
        {sop2024.to_html(index=False)}
    </div>
    <div style="width: 48%; display: flex; flex-direction: column; align-items: center;">
        <h3>2023</h3>
        {sop2023.to_html(index=False)}
    </div>
    <div style="width: 48%; display: flex; flex-direction: column; align-items: center;">
        <h3>2022</h3>
        {sop2022.to_html(index=False)}
    </div>
    <div style="width: 48%; display: flex; flex-direction: column; align-items: center;">
        <h3>2008</h3>
        {sop2008.to_html(index=False)}
    </div>
    <div style="width: 48%; display: flex; flex-direction: column; align-items: center;">
        <h3>2007</h3>
        {sop2007.to_html(index=False)}
    </div>
</div>
"""

# display all dataframes side by side
display(HTML(html))
```

2024

team	sop
DET	14.634425
BUF	9.464874
GB	7.784117
MIN	7.182150
PHI	6.275717
BAL	6.222153
PIT	4.325490
TB	4.180493
LAC	4.175214
KC	3.966440
DEN	3.929199
WAS	3.742980
ARI	2.818797
SEA	2.024035
HOU	1.342537
SF	1.256442
LA	-0.691942
IND	-1.062668
CIN	-1.083878
CHI	-1.763314
NO	-2.758870
ATL	-3.173219
MIA	-3.255589
NYJ	-3.336904
JAX	-6.678877
TEN	-6.901944
DAL	-7.173419
CLE	-7.913812
NYG	-8.263873
LV	-8.512733
NE	-8.549837
CAR	-12.203129

2023

team	sop
BAL	12.380044
SF	10.299989
DAL	7.418777
KC	6.303398
BUF	6.296673
DET	5.093401
MIA	3.521020
GB	3.499999
NO	2.740861
LA	2.684864
TB	2.270992
JAX	1.531306
HOU	1.412988
CLE	1.398406
CIN	1.365799
MIN	0.094985
PIT	0.078879
LV	-0.347082
IND	-1.290352
CHI	-1.356917
PHI	-1.410255
SEA	-1.625806
LAC	-1.760773
TEN	-3.245778
DEN	-3.309998
ATL	-4.466598
NYJ	-5.954149
ARI	-6.035931
NE	-8.049655
NYG	-8.377508
CAR	-9.466670
WAS	-11.694743

2022

team	sop
BUF	9.208579
PHI	8.462653
CIN	8.044024
SF	7.266307
DAL	7.000276
KC	6.679477
BAL	3.097545
DET	2.099778
MIA	1.954941
NE	1.807962
JAX	1.784926
NYJ	0.541411
GB	0.206864
CLE	-0.107234
MIN	-0.664278
PIT	-0.665882
LAC	-0.709582
NYG	-0.755402
WAS	-0.768505
SEA	-1.090411
NO	-1.210903
ATL	-2.231060
CAR	-2.313986
LV	-2.460645
TB	-2.827250
TEN	-3.461207
LA	-4.296358
DEN	-4.966238
ARI	-6.339002
CHI	-6.566442
HOU	-8.312721
IND	-8.407650

2008

team	sop
PIT	10.349663
BAL	10.293259
PHI	8.821211
TEN	8.554753
NYG	7.983826
IND	6.037208
SD	4.772166
NE	4.186565
CAR	3.794677
NO	3.505897
MIN	3.175284
ATL	2.997559
GB	2.427408
TB	1.729480
CHI	1.667963
ARI	1.152734
DAL	1.022310
NYJ	0.484443
HOU	-0.604479
MIA	-0.757151
WAS	-1.323621
JAX	-2.695309
BUF	-3.066942
CLE	-4.480330
SF	-4.719831
DEN	-6.083666
CIN	-6.807985
SEA	-7.040567
OAK	-7.757769
KC	-9.532278
DET	-13.574002
STL	-14.512261

2007

team	sop
NE	18.457604
IND	11.421139
SD	9.480065
GB	9.390769
DAL	8.982306
JAX	7.134558
NYG	5.887100
PHI	5.343060
PIT	4.879325
MIN	4.065947
WAS	3.438906
SEA	2.163644
CHI	1.505161
TEN	0.728566
TB	0.700866
HOU	0.004038
CLE	-1.258532
NO	-2.550440
CIN	-2.578267
DET	-3.324190
DEN	-3.800445
NYJ	-3.921197
ARI	-4.016165
BUF	-4.315079
KC	-5.303934
OAK	-5.823977
CAR	-5.854636
BAL	-6.940641
MIA	-8.601452
ATL	-10.489072
SF	-11.802916
STL	-13.001811

In [ ]: