Alright, so let's build a hacky model

```
import pymc as pm
import pandas as pd
import numpy as np
import arviz as az
from sklearn.metrics import mean_absolute_error
```

```
You are running the v4 development version of Py
MC which currently still lacks key features. You
probably want to use the stable v3 instead which
you can either install via conda or find on the
v3 GitHub branch: https://github.com/pymc-devs/p
ymc/tree/v3
```

```
data = pd.read_csv('data.csv')
train = data[:2985]
test = data[2986:]

ranks=4
team_number = 20
player_names = set(train.name)
opp_defense_rank_no = train.opp_defense_rank.max()
opp_attack_rank_no = train.opp_attack_rank.max()
team_cluster_rank_no = train.team_cluster_rank.max()
opp_cluster_rank_no = train.opp_cluster_rank.max()
num_positions = 4
N = len(train)
```

```
with pm.Model() as model:
    nu = pm.Exponential('nu minus one', 1/29,shape=2) + 1
    err = pm.Uniform('std dev based on rank', 0, 100, shape=ranks)
    err_b = pm.Uniform('std dev based on rank b', 0, 100, shape=ranks)
```

```
with model:
    team_cluster_rank = pm.Data('team_cluster_rank',np.asarray((train['team_cluster_rank']).values, dtype = int))
    opp_cluster_rank = pm.Data('opp_cluster_rank',np.asarray((train['opp_cluster_rank']).values, dtype = int))
    opp_defense_rank = pm.Data('opp_defense_rank',np.asarray((train['opp_defense_rank']).values, dtype = int))
    opp_attack_rank = pm.Data('opp_attack_rank',np.asarray((train['opp_attack_rank']).values, dtype = int))
    initval = pm.Data('initval',np.asarray((train['initval']).values, dtype = int))
    player_home = pm.Data('player_home',np.asarray(train['was_home'].values, dtype = int))
    player_avg = pm.Data('player_avg',np.asarray((train['game_avg_7']).values, dtype = float))
    player_opp = pm.Data('player_opp',np.asarray((train['opponent_team']).values, dtype = int))
    player_team = pm.Data('player_team',np.asarray((train['team']).values, dtype = int))
    player_rank = pm.Data('player_rank',np.asarray((train['rank']-1).values, dtype = int))
    position_FWD = pm.Data('position_FWD',np.asarray((train['position_FWD']).values.astype(int),
                                            dtype = int))
    position_MID = pm.Data('position_MID',np.asarray((train['position_MID']).values.astype(int),
                                            dtype = int))
    position_GK = pm.Data('position_GK',np.asarray((train['position_GK']).values.astype(int),
                                           dtype = int))
    position_DEF = pm.Data('position_DEF',np.asarray((train['position_DEF']).values.astype(int),
                                            dtype = int))
    pos_id = pm.Data('pos_id',np.asarray((train['pos_id']).values, dtype = int))
```

```
with model:
    opp_def = pm.Normal('opp team prior',mu=0, sigma=100, shape=num_positions)
    opp_fwd = pm.Normal('defensive differential fwd',mu= opp_def[0], sigma=100, shape=team_number)
    opp_mid = pm.Normal('defensive differential mid',mu= opp_def[1], sigma=100, shape=team_number)
    opp_gk = pm.Normal('defensive differential gk',mu= opp_def[2], sigma=100, shape=team_number)
    opp_defe = pm.Normal('defensive differential defe',mu= opp_def[3], sigma=100, shape=team_number)

    home_adv = pm.Normal('home additivie prior',mu= 0, sigma=100,shape = num_positions)
    away_adv = pm.Normal('away additivie prior',mu= 0, sigma=100,shape = num_positions)
    pos_home_fwd = pm.Normal('home differential fwd',mu=home_adv[0],sigma=100, shape = ranks)
    pos_home_gk = pm.Normal('home differential gk',mu=home_adv[1],sigma=100, shape = ranks)
    pos_home_defe = pm.Normal('home differential defe',mu=home_adv[2],sigma=100, shape = ranks)
    pos_home_mid = pm.Normal('home differential mid',mu=home_adv[3],sigma=100, shape = ranks)
    pos_away_fwd = pm.Normal('away differential fwd',mu=away_adv[0],sigma=100, shape = ranks)
    pos_away_gk = pm.Normal('away differential gk',mu=away_adv[1],sigma=100, shape = ranks)
    pos_away_mid = pm.Normal('away differential mid',mu=away_adv[2],sigma=100, shape = ranks)
    pos_away_defe = pm.Normal('away differential defe',mu=away_adv[3],sigma=100, shape = ranks)
```

```
with model:

    team_strength = pm.Normal('team_strength',mu=0, sigma=100, shape=2) #home and away
    team_strength_home = pm.Normal('team_strength_home',mu=team_strength[0],
                                   sigma=100, shape=num_positions)
    team_strength_away = pm.Normal('team_strength_away',mu=team_strength[1],
                                   sigma=100, shape=num_positions)

    team_strength_home_FWD = pm.Normal('team_strength_home_FWD',mu=team_strength_home[0],
                                       sigma=100, shape=team_cluster_rank_no)
    team_strength_home_MID = pm.Normal('team_strength_home_MID',mu=team_strength_home[1],
                                       sigma=100, shape=team_cluster_rank_no)
    team_strength_home_DEF = pm.Normal('team_strength_home_DEF',mu=team_strength_home[2],
                                       sigma=100, shape=team_cluster_rank_no)
    team_strength_home_GK = pm.Normal('team_strength_home_GK',mu=team_strength_home[3],
                                      sigma=100, shape=team_cluster_rank_no)

    team_strength_away_FWD = pm.Normal('team_strength_away_FWD',mu=team_strength_away[0],
                                       sigma=100, shape=team_cluster_rank_no)
    team_strength_away_MID = pm.Normal('team_strength_away_MID',mu=team_strength_away[1],
                                       sigma=100, shape=team_cluster_rank_no)
    team_strength_away_DEF = pm.Normal('team_strength_away_DEF',mu=team_strength_away[2],
                                       sigma=100, shape=team_cluster_rank_no)
    team_strength_away_GK = pm.Normal('team_strength_away_GK',mu=team_strength_away[3],
                                      sigma=100, shape=team_cluster_rank_no)


    team_strength_effects = (
        (player_home)*position_FWD*team_strength_home_FWD[team_cluster_rank-1] +
        (player_home)*position_MID*team_strength_home_MID[team_cluster_rank-1] +
        (player_home)*position_DEF*team_strength_home_DEF[team_cluster_rank-1] +
        (player_home)*position_GK*team_strength_home_GK[team_cluster_rank-1] +

        (1-player_home)*position_FWD*team_strength_away_FWD[team_cluster_rank-1] +
        (1-player_home)*position_MID*team_strength_away_MID[team_cluster_rank-1] +
        (1-player_home)*position_DEF*team_strength_away_DEF[team_cluster_rank-1] +
        (1-player_home)*position_GK*team_strength_away_GK[team_cluster_rank-1])
```

```
with model:
    opp_strength = pm.Normal('opp_strength',mu=0, sigma=100, shape=3)
    opp_strength_team = pm.Normal('opp_strength_team',mu=opp_strength[0], sigma=100, shape=num_positions)
    opp_strength_defense_team = pm.Normal('opp_strength_defense_team',mu=opp_strength[1], sigma=100, shape=num_positions
    opp_strength_attack_team = pm.Normal('opp_strength_attack_team',mu=opp_strength[2],
                                         sigma=100, shape=num_positions)

    opp_strength_team_FWD = pm.Normal('opp_strength_team_FWD',mu=opp_strength_team[0],
                                      sigma=100, shape=opp_cluster_rank_no)
    opp_strength_team_MID = pm.Normal('opp_strength_team_MID',mu=opp_strength_team[1],
                                      sigma=100, shape=opp_cluster_rank_no)
    opp_strength_team_DEF = pm.Normal('opp_strength_team_DEF',mu=opp_strength_team[2],
                                      sigma=100, shape=opp_cluster_rank_no)
    opp_strength_team_GK = pm.Normal('opp_strength_team_GK',mu=opp_strength_team[3],
                                     sigma=100, shape=opp_cluster_rank_no)

    opp_strength_team_effects = (
        position_FWD*opp_strength_team_FWD[opp_cluster_rank-1] +
        position_MID*opp_strength_team_MID[opp_cluster_rank-1] +
        position_DEF*opp_strength_team_DEF[opp_cluster_rank-1] +
        position_GK*opp_strength_team_GK[opp_cluster_rank-1])
```

```
with model:

    opp_strength_defense_team_FWD = pm.Normal('opp_strength_defense_team_FWD',mu=opp_strength_defense_team[0],
                                              sigma=100, shape=opp_defense_rank_no)
    opp_strength_defense_team_MID = pm.Normal('opp_strength_defense_team_MID',mu=opp_strength_defense_team[1],
                                              sigma=100, shape=opp_defense_rank_no)
    opp_strength_defense_team_DEF = pm.Normal('opp_strength_defense_team_DEF',mu=opp_strength_defense_team[2],
                                              sigma=100, shape=opp_defense_rank_no)
    opp_strength_defense_team_GK = pm.Normal('opp_strength_defense_team_GK',mu=opp_strength_defense_team[3],
                                             sigma=100, shape=opp_defense_rank_no)

    opp_strength_defense_team_effects = (
        position_FWD*opp_strength_defense_team_FWD[opp_cluster_rank-1] +
        position_MID*opp_strength_defense_team_MID[opp_cluster_rank-1] +
        position_DEF*opp_strength_defense_team_DEF[opp_cluster_rank-1] +
        position_GK*opp_strength_defense_team_GK[opp_cluster_rank-1])
```

```
with model:

    opp_strength_attack_team_FWD = pm.Normal('opp_strength_attack_team_FWD',mu=opp_strength_attack_team[0],
                                             sigma=100, shape=opp_attack_rank_no)
    opp_strength_attack_team_MID = pm.Normal('opp_strength_attack_team_MID',mu=opp_strength_attack_team[1],
                                             sigma=100, shape=opp_attack_rank_no)
    opp_strength_attack_team_DEF = pm.Normal('opp_strength_attack_team_DEF',mu=opp_strength_attack_team[2],
                                             sigma=100, shape=opp_attack_rank_no)
    opp_strength_attack_team_GK = pm.Normal('opp_strength_attack_team_GK',mu=opp_strength_attack_team[3],
                                            sigma=100, shape=opp_attack_rank_no)

    opp_strength_attack_team_effects = (
        position_FWD*opp_strength_attack_team_FWD[opp_cluster_rank-1] +
        position_MID*opp_strength_attack_team_MID[opp_cluster_rank-1] +
        position_DEF*opp_strength_attack_team_DEF[opp_cluster_rank-1] +
        position_GK*opp_strength_attack_team_GK[opp_cluster_rank-1])
```

```
with model:
    def_effect = (position_FWD*opp_fwd[player_opp-1]+
    position_MID*opp_mid[player_opp-1]+ position_GK*opp_gk[player_opp-1]+
    position_DEF*opp_defe[player_opp-1])

    like1 = pm.StudentT('Diff From Avg', mu=def_effect,
                        sd=err_b[player_rank],nu=nu[1], observed = train['diff_from_avg'])

    initval_coeff = pm.Normal('initval_coeff',0,100, shape = N)
```

```
with model:

    mu = (position_GK*pos_home_gk[player_rank]*(player_home) +
            position_MID*pos_home_mid[player_rank]*(player_home) +
            position_FWD*pos_home_fwd[player_rank]*(player_home) +
            position_DEF*pos_home_defe[player_rank]*(player_home) +
            position_GK*pos_away_gk[player_rank]*(1-player_home) +
            position_MID*pos_away_mid[player_rank]*(1-player_home) +
            position_FWD*pos_away_fwd[player_rank]*(1-player_home) +
            position_DEF*pos_away_defe[player_rank]*(1-player_home) +
            team_strength_effects + opp_strength_team_effects +
            opp_strength_attack_team_effects +
            opp_strength_defense_team_effects +
            initval*initval_coeff+player_avg + def_effect)

    like2 = pm.StudentT('Score', mu=mu, sd=err[player_rank],
                        nu=nu[0], observed=train['total_points'])
```

```
with model:
#     trace = pm.sample(10000, pm.sample())
    trace=az.from_netcdf('data')
    assert all(az.rhat(trace) < 1.03)
```

```
with model:
    pm.set_data({'team_cluster_rank': np.asarray((train['team_cluster_rank']).values, dtype = int)})
    pm.set_data({'opp_cluster_rank': np.asarray((test['opp_cluster_rank']).values, dtype = int)})
    pm.set_data({'opp_defense_rank': np.asarray((test['opp_defense_rank']).values, dtype = int)})
    pm.set_data({'opp_attack_rank': np.asarray((test['opp_attack_rank']).values, dtype = int)})
    pm.set_data({'initval': np.asarray((test['initval']).values, dtype = int)})
    pm.set_data({'player_home': np.asarray(test['was_home'].values, dtype = int)})
    pm.set_data({'player_avg': np.asarray((test['game_avg_7']).values, dtype = float)})
    pm.set_data({'player_opp': np.asarray((test['opponent_team']).values, dtype = int)})
    pm.set_data({'player_team': np.asarray((test['team']).values, dtype = int)})
    pm.set_data({'player_rank': np.asarray((test['rank']-1).values, dtype = int)})
    pm.set_data({'position_FWD': np.asarray((test['position_FWD']).values.astype(int),dtype = int)})
    pm.set_data({'position_MID': np.asarray((test['position_MID']).values.astype(int), dtype = int)})
    pm.set_data({'position_GK': np.asarray((test['position_GK']).values.astype(int),dtype = int)})
    pm.set_data({'position_DEF': np.asarray((test['position_DEF']).values.astype(int), dtype = int)})
    pm.set_data({'pos_id': np.asarray((test['pos_id']).values, dtype = int)})

    ppc=pm.sample_posterior_predictive(trace, samples=44000)
```

100.00% [44000/44000 01:36<00:00]

```
test['pred_points'] = ppc['Score'][0].tolist()
pts=test.groupby(['name']).sum()
pts.sort_values(by =['pred_points'], inplace = True,ascending=False)
pred = set(pts[:15].index)
pts=test.groupby(['name']).sum()
pts.sort_values(by =['total_points'], inplace = True,ascending=False)
truth = set(pts[:15].index)
len(pred.intersection(truth))
```

```
7
```

```
mean_absolute_error(test.loc[:,'total_points'].values, ppc['Score'].mean(axis=0))
```

```
2.5331934828587186
```

```
pred
```

```
{'Callum Wilson',
 'Christian Benteke',
 'Dominic Calvert-Lewin',
 'Emiliano Martínez',
 'Harry Kane',
 'Heung-Min Son',
 'Hugo Lloris',
 'Jordan Pickford',
 'Matheus Pereira',
 'Mohamed Salah',
 'Patrick Bamford',
 'Pierre-Emerick Aubameyang',
 'Roberto Firmino',
 'Rodrigo Moreno',
 'Stuart Dallas'}
```

```
truth
```

```
{'Emiliano Martínez',
 'Harry Kane',
 'Illan Meslier',
 'Jordan Pickford',
 'Kelechi Iheanacho',
 'Leandro Trossard',
 'Lewis Dunk',
 'Lucas Digne',
 'Matheus Pereira',
 'Mohamed Salah',
 'Nicolas Pépé',
 'Patrick Bamford',
 'Sam Johnstone',
 'Stuart Dallas',
 'Trent Alexander-Arnold'}
```

References

1. https://srome.github.io/Bayesian-Hierarchical-Modeling-Applied-to-Fantasy-Football-Projections-for-Increased-Insight-and-Confidence/
2. https://www.degruyter.com/document/doi/10.1515/jqas-2017-0066/html
3. https://pymc-examples.readthedocs.io/en/latest/case_studies/multilevel_modeling.html