```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Wed May 29 14:35:03 2019

@author: Brice Loose

Script for GAM fit to SWIMS data.

Ref:  Loose et al., (2020), Instrument bias correction with machine learning
algorithms: Application to field-portable mass spectrometry, Frontiers in
Geoscience, DOI: ??

Download the most up to date version of the Backfit Algorithm at
https://github.com/bloose/Python_GAM_Backfit.

Put backfit_algorithm.py in the same directory as this file.

.
#
# DISCLAIMER:
#    This is provided "as is" without warranty of any kind.
#==============================================================================

"""
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
import imp
game = imp.load_source('backfit_algorithm', 'backfit_algorithm.py')



# Oxygen
target = 64

# Penalty value in backfit
lam = 1e-4

# Indicate the columns containing the environmental covariates.
#'Pressure (db)', 'Temperature (C) Temperature (C) ',
#       'uC Temp. (°C)', 'Sample Temperature (°C)', 'MASS( 5 )',
#       'MASS( 18 )']
idx = [3,5,36,28,57,62]


###----------- DEVELOP GAM FIT USING IN-SITU CALIBRATION TOW ---------- #####
### See Section 2.2 of the above-referenced manuscript.

ms = pd.read_csv(r'Insitu_cal.csv',header='infer',index_col=0);
hdrs = ms.columns.values
```

```
X = ms[hdrs[idx]].interpolate(method='linear', order=1, limit_direction='both')
#X = pd.DataFrame(ms[hdr15[idx]])
y15 = ms[hdrs[target]]

# Standardized environmental covariates to ensure equal weighting.
x = (X-X.mean(axis=0))/X.std(axis=0)


# Apply backfit algorithm to produce GAM fit.
ycal15, pp15, b15,a15 = game.GAMcal(x,y15,lam)



#%%     PREDICTION STEP HERE


###------------- NNLS optimization function for calib --------------- #####
from scipy.optimize import minimize
def optocf(x0,ytrain,yp,o2):
    import numpy as np
    den = (np.mean(ytrain-x0))
    CF = 210.68/den
    yrect = (yp + x0)*CF

    return np.median(np.abs(o2 - yrect))


MA = pd.read_csv(r'SWIMS_Lat36_38.csv',header='infer',index_col=0);
hdrs = MA.columns.values


b = b15; pp = pp15; a = a15; y = y15;


# Standardized environmental covariates to ensure equal weighting.
X = MA[hdrs[idx]].interpolate(method='linear', order=1, limit_direction='both')
x = (X-X.mean(axis=0))/X.std(axis=0)


# Apply GAM model to predict bias signal in SWIMS data
yr, alf = game.GAMapply(x,b,pp,hdrs[idx],MA[hdrs[target]])


MA = pd.concat([MA,yr],axis=1).dropna(how='any')


####  Calibration step.  ################
### Non-linear optimization to find best fit to SBE oxygen data #
## See Section 2.3 on calibration after bias removal ###
t = minimize(optocf,1e-15,args=(y,MA[hdrs[target]]-MA[0],MA[hdrs[8]]),
```

```
            method='Nelder-Mead',tol=1e-11)

# This is y-intercept in calib.
ic = t.x

# Equil. solubility at S=35.5 and T =23 C.
cf = 210.68/np.mean(y-ic)

yraw = pd.DataFrame(yr); yraw.rename(columns={0:'yraw'},inplace=True)

# Apply calibration
yrect = pd.DataFrame((MA[hdrs[target]].subtract(yr)+ic)*cf,columns={'yrect'})

Yp = pd.concat([MA,yraw,yrect],axis=1)

#%%  Plotting section

mse = np.round(np.sqrt(np.nansum((Yp['yrect'] - Yp[hdrs[8]])**2)/
 len(Yp[hdrs[8]])),3)



f,ax = plt.subplots(1,1)
#plt.plot(Y17[hdrs[0]],Y17['Ar17'],'k')
ax.plot(Yp['yrect'],'r',label='SWIMS DO')

ax.plot(Yp[hdrs[8]],'g',label='SBE35 DO')
ax.legend()
#plt.plot(ctdt,ctddat[:,10],'r')
ax.set_ylim(100,300)
ax.set_title('RMSE='+mse.astype(str))
axb = ax.twinx()
axb.plot(Yp[hdrs[idx[-1]]])
axb.set_ylim(100,350)
plt.show()
```