Python 2

# pipeline\_noLatLon.py

import pandas as pd

import numpy as np

from sklearn.ensemble import GradientBoostingClassifier as GBC

from sklearn.ensemble import RandomForestClassifier as RF

from utils.utilities import get\_all\_data, get\_group

from utils.utilities import \_subsample\_even, \_subsample\_proportional, sample\_by\_vessel

import csv

import json

def threshold\_fishing(df):

    df['is\_fishing'] = df['is\_fishing'].apply(lambda x: 1 if x>0.5 else 0)

    return df

def keep\_columns(df,col\_groups):

    from pipeline\_noLatLon import columns\_models\_dict

    # minimal model

    cols\_to\_keep = ['timestamp','mmsi','course','measure\_daylight','speed','is\_fishing']

    if col\_groups:

        if col\_groups == 'all':

            for key in columns\_models\_dict:

                cols\_to\_keep += [col for col in columns\_models\_dict[key]]

        else:

            for col\_g in col\_groups:

                cols\_to\_keep += columns\_models\_dict[col\_g]

    df =  df[cols\_to\_keep]

    # N\_cols = len(cols\_to\_keep)

    return df

def random\_split(df):

    y = df['is\_fishing'].astype(int).values

    df\_X = df.drop(['mmsi','is\_fishing','timestamp'],axis=1)

    X = df\_X.values

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.2)

    return X\_train, X\_test, y\_train, y\_test

def X\_y\_split(df):

    y = df['is\_fishing'].astype(int).values

    df\_X = df.drop(['mmsi','is\_fishing','timestamp'],axis=1)

    cols = df\_X.columns

    X = df\_X.values

    return X, y, cols

def save\_mmsi(mmsi\_series,file\_name):

    with open(file\_name, 'w') as outfile:

        mmsi\_series.astype(int).to\_csv(outfile,index=False,header=True,encoding='utf-8')

def run\_pipeline(path,gear,col\_groups):

    # path = '../data/labeled'

    # gear = 'longliners'

    data\_dict = get\_all\_data(path)

    df = get\_group(data\_dict,gear)

    df.reindex()

    df = threshold\_fishing(df)

    # decide what group of variables to add to the basic model. It has to be a list!

    df = keep\_columns(df, col\_groups = col\_groups)

    df\_train, df\_cross, df\_test = sample\_by\_vessel(df, size = 20000, even\_split=None, seed=4321)

    mmsi\_tr = set(df\_train.mmsi)

    mmsi\_cr = set(df\_cross.mmsi)

    mmsi\_te = set(df\_test.mmsi)

    # ====DEBUGGER=======

    # import pdb, sys

    # pdb.set\_trace()

    # passing the mmsi as pandas series instead of lists...

    # save\_mmsi(df\_train.mmsi,'used\_mmsi/'+gear+'\_mmsi\_train.txt')

    # save\_mmsi(df\_cross.mmsi,'used\_mmsi/'+gear+'\_mmsi\_cross.txt')

    # save\_mmsi(df\_test.mmsi,'used\_mmsi/'+gear+'\_mmsi\_test.txt')

    X\_train, y\_train, cols = X\_y\_split(df\_train)

    X\_cross, y\_cross, cols = X\_y\_split(df\_cross)

    X\_test, y\_test, cols = X\_y\_split(df\_test)

    return X\_train, y\_train, X\_cross, y\_cross, X\_test, y\_test, cols

models\_dict = {'model\_1': ['course\_norm\_sin\_cos','window\_1800'],

'model\_2': ['course\_norm\_sin\_cos','window\_1800','window\_3600'],

'model\_3': ['course\_norm\_sin\_cos','window\_1800','window\_3600','window\_10800'],

'model\_4': ['course\_norm\_sin\_cos','window\_1800','window\_3600','window\_10800','window\_21600'],

'model\_5': ['course\_norm\_sin\_cos','window\_1800','window\_3600','window\_10800','window\_21600','window\_43200'],

'model\_6': ['course\_norm\_sin\_cos','window\_1800','window\_3600','window\_10800','window\_21600','window\_43200','window\_86400']}

columns\_models\_dict = {

'dist\_to\_land': [

'distance\_from\_port',

'distance\_from\_shore',

'measure\_distance\_from\_port'],

'course\_norm\_sin\_cos' : [

'measure\_course',

'measure\_cos\_course',

'measure\_sin\_course'],

# 'measure\_pos\_1800', took this column out for now.... (feb 2nd) I think it was overfitting to this

'window\_1800':['measure\_coursestddev\_1800\_log',

'measure\_daylightavg\_1800',

'measure\_speedstddev\_1800',

'measure\_count\_1800',

'measure\_courseavg\_1800',

'measure\_coursestddev\_1800',

'measure\_speedavg\_1800',

'measure\_speedstddev\_1800\_log'],

'window\_3600': ['measure\_count\_3600',

'measure\_speedstddev\_3600',

'measure\_speedavg\_3600',

'measure\_courseavg\_3600',

'measure\_daylightavg\_3600',

'measure\_coursestddev\_3600',

'measure\_speedstddev\_3600\_log',

'measure\_coursestddev\_3600\_log'],

'window\_10800': ['measure\_coursestddev\_10800\_log',

'measure\_speedstddev\_10800',

'measure\_speedavg\_10800',

'measure\_daylightavg\_10800',

'measure\_courseavg\_10800',

'measure\_count\_10800',

'measure\_speedstddev\_10800\_log',

'measure\_coursestddev\_10800'],

'window\_21600': ['measure\_coursestddev\_21600',

'measure\_speedavg\_21600',

'measure\_count\_21600',

'measure\_coursestddev\_21600\_log',

'measure\_speedstddev\_21600\_log',

'measure\_speedstddev\_21600',

'measure\_daylightavg\_21600',

'measure\_courseavg\_21600'],

'window\_43200': ['measure\_coursestddev\_43200',

'measure\_courseavg\_43200',

'measure\_daylightavg\_43200',

'measure\_coursestddev\_43200\_log',

'measure\_speedavg\_43200',

'measure\_count\_43200',

'measure\_speedstddev\_43200\_log',

'measure\_speedstddev\_43200'],

'window\_86400': ['measure\_speedavg\_86400',

'measure\_count\_86400',

'measure\_speedstddev\_86400\_log',

'measure\_speedstddev\_86400',

'measure\_coursestddev\_86400\_log',

'measure\_coursestddev\_86400',

'measure\_daylightavg\_86400',

'measure\_courseavg\_86400'],

}

# if \_\_name\_\_ == '\_\_main\_\_':

#   path = '../data/labeled'

#   gear = 'longliners'

#   mmsi\_tr, mmsi\_cr, mmsi\_te, X\_train, y\_train, X\_cross, y\_cross, X\_test, y\_test, cols = run\_pipeline(path,gear)