```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon May 27 13:32:02 2019

@author: torjus

THIS PYTHON FILE CONDUCTS THE INITIAL ANALYSIS


"""

import time
import datetime
from datetime import timedelta
import pandas as pd
from operator import sub
import matplotlib.pyplot as plt
import time
import datetime
from mpl_toolkits.basemap import Basemap
import numpy as np
from scipy import stats
import pandas as pd
import sklearn.cluster as cluster
import scipy.cluster.hierarchy as hcluster
#import loc_check as LC
import networkx as nx
from itertools import cycle , groupby
from pylab import boxplot
import matplotlib.colors as mcolors
import unixTimeConvert as UTC
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
import matplotlib.animation as animation
import calendar



#Initial Analysis
#The initial analysis should be runned with both original data and
#the the preprocessed data

#number of messages total and montly
def messagesMonthly(df1):
    #highdate needs to be first day in first month after intervall
    #both lowdate and highdate needs to be timestamps
    #of the type datetime.datetime(YYYY,M,D)
    lowDate = datetime.datetime(2017,1,1)
    highDate = datetime.datetime(2019,1,1)
    months = list(pd.date_range(lowDate,highDate,freq='MS').floor('d'))

    # To unix : unix = calendar.timegm(lowDate.timetuple())
    nMessages = list()

    for m in range(len(months)-1):
        dfMonth = df1[(df1['DateTime']>=months[m]) & (df1['DateTime']<months[m+1])]
        nMessages.append(dfMonth['mmsi'].count())

    return nMessages


#Check for speed error
def SpeedError(df1):
    dfError = df1[df1['sog']>30]
    totErrorVsNonError = len(dfError)/len(df1)
    speedErrorMonthly = messagesMonthly(dfError)
    totmessagesMonthly = messagesMonthly(df1)
    a = np.array(totmessagesMonthly)
    b = np.array(speedErrorMonthly)
    ratioErrorMonthly = list(b/a)
```

```
    sEs = dfError['mmsi'].value_counts() #speedErrorShips
    mPs = df1['mmsi'].value_counts() #message per ship (all ships)
    mPsE = mPs.loc[sEs.index]#messages per ship with speed error

    errorVsNonError = sEs/mPsE


    return dfError, speedErrorMonthly, errorVsNonError, totErrorVsNonError,ratioErrorMonthly

def positionError(df1):
    posError = df1[~((df1['latitude']<90) & (df1['latitude']>-90)) |
                   ~((df1['longitude']<180) & (df1['longitude']>-180))]
    return posError

#Plot of speed error monthly
def speedErrorMonthlyPlot(speedErrorMonthly,ratioErrorMonthly):
    months = pd.date_range('2017-01-01','2018-12-01', freq='MS').strftime("%Y-%b").tolist()

    fig,ax1 = plt.subplots(figsize=(15,5))

    color = 'tab:blue'
    ax1.set_xlabel('Month')
    ax1.set_ylabel(' Error Messages', color=color)
    ax1.plot(months,speedErrorMonthly,color=color,label='Error Messages')
    #ax1.set_ylim(0,300000)
    ax1.tick_params(axis='y', labelcolor=color)

    ax2 = ax1.twinx()

    color = 'tab:red'
    ax2.set_ylabel('Percentage Error Messages', color=color)  # we already handled the x-label with ax1
    ax2.plot(months,ratioErrorMonthly,color=color, label=' Percentage Error Messages')
    ax2.tick_params(axis='y', labelcolor=color)
    vals = ax2.get_yticks()
    ax2.set_yticklabels(['{:,.2%}'.format(x) for x in vals])
    fig.tight_layout()  # otherwise the right y-label is slightly clipped
    ax1.set_xticklabels(months, rotation=90)
    plt.show()
    #plt.xticks(rotation=45)

def finaldataPlot(df_merged,df_noError,df_final):

    initial = messagesMonthly(df_merged)
    after_error = messagesMonthly(df_noError)
    final = messagesMonthly(df_final)

    months = pd.date_range('2017-01-01','2018-12-01', freq='MS').strftime("%Y-%b").tolist()

    fig,ax1 = plt.subplots(figsize=(15,5))

    ax1.set_xlabel('Month')
    ax1.set_ylabel('Messages')
    ax1.plot(months,initial,label='Initial Data')
    ax1.plot(months,after_error,label='Data After Error Removal')
    ax1.plot(months,final,label = 'Data After Frequency Reduction')
    #ax1.set_ylim(0,300000)
    ax1.tick_params(axis='y')

    ax1.set_xticklabels(months, rotation=90)
    ax1.legend()
    plt.show()
    #plt.xticks(rotation=45)

#number of messages per ship
def plotMessages(df1) :
    #input DataFrame: uniqueMMSI,nMessages
    mps = df1['mmsi'].value_counts()
    ########################################################
    #Make Histogram of number of messages per Vessel #
    x = list(mps)

    fig = plt.figure()
    ax1 = fig.add_axes([0.1,0.1,0.9,0.9])
```

```
    ax1.set_xlabel('Messages per Vessel')
    ax1.set_ylabel('Vessels')
    ax1.hist(x,bins=20)

    ax2 = fig.add_axes([0.5,0.5,.4,.4])
    ax2.set_xlabel('Messages per Vessel')
    ax2.set_ylabel('Vessels')
    ax2.hist(x,range=(0,np.percentile(x,75)),bins=20)
    ##################################################

#largest message gap per ship
#number of message gaps over 24 hours, 48 hours, 72 hours, one week, one month per ship
def MessageGapsShip(df1,uniqueMMSI):
    df_out = pd.DataFrame(columns = ['mmsi','maxgapsec','maxgapdays','gap24','gap48','gap72','gapWeek','ga

    '''maxgapList = list([0]*len(uniqueMMSI))
    maxgapdaysList = list([0]*len(uniqueMMSI))
    gap24List = list([0]*len(uniqueMMSI))
    gap48List = list([0]*len(uniqueMMSI))
    gap72List = list([0]*len(uniqueMMSI))
    gapWeekList = list([0]*len(uniqueMMSI))
    gapMonthList = list([0]*len(uniqueMMSI))'''

    for i,mmsi in enumerate(uniqueMMSI):
        df_temp = df1[df1['mmsi']==mmsi]
        df_temp['diff'] = df_temp['unixtime'].diff()
        maxgap = df_temp['diff'].max()
        maxgapdays = maxgap/86400
        gap24 = df_temp['mmsi'][df_temp['diff']>86400].count()  #gaps > 24 hours
        gap48 = df_temp['mmsi'][df_temp['diff']>172800].count() #gaps > 48 hours
        gap72 = df_temp['mmsi'][df_temp['diff']>259200].count() #gaps > 72 hours
        gapWeek = df_temp['mmsi'][df_temp['diff']>604800].count() #gaps > 168 hours (one week)
        gapMonth = df_temp['mmsi'][df_temp['diff']>2592000].count() #gap of one month (30 days)

        '''maxgapList[i] = maxgap
        maxgapdaysList[i] = maxgapdays
        gap24List[i] = gap24
        gap48List[i] = gap48
        gap72List[i] = gap72
        gapWeekList[i] = gapWeek
        gapMonthList[i] = gapMonth'''

        temp_serie = pd.Series([mmsi,maxgap,maxgapdays,gap24,gap48,gap72,gapWeek,gapMonth],index=df_out.co
        df_out = df_out.append(temp_serie,ignore_index = True)

    #df_out = pd.DataFrame(data = [uniqueMMSI,maxgapList,maxgapdaysList,gap24List,gap48List,gap72List,gapW

    n_day1 = df_out.gap24.sum()
    n_day2 = df_out.gap48.sum()
    n_day3 = df_out.gap72.sum()
    n_dayWeek = df_out.gapWeek.sum()
    n_dayMonth= df_out.gapMonth.sum()

    gaps = pd.Series([n_day1,n_day2,n_day3,n_dayWeek,n_dayMonth],index=['1 day','2 days','3 days','7 days

    return df_out,gaps


def plotMessageGaps(gaps17,gaps18):
    fig, ax1 = plt.subplots() #Create matplotlib figure
    width = 0.4
    x_axis1 = [0.8,1.8,2.8,3.8,4.8]
    x_axis2 = [1.2,2.2,3.2,4.2,5.2]
    color = 'tab:blue'
    ax1.set_xlabel('Message Gap Size')
    ax1.set_ylabel('Number of Gaps 2017', color=color)
    ax1.bar(x_axis1,height=list(gaps17),width=width,color=color)
    ax1.tick_params(axis='y', labelcolor=color)

    ax2 = ax1.twinx()
    color = 'tab:red'
```

```
    ax2.set_ylabel('Number of Gaps 2018', color=color)  # we already handled the x-label with ax1
    ax2.bar(x_axis2, height=list(gaps18),width=width, color=color)
    ax2.tick_params(axis='y', labelcolor=color)
    ax2.set_xticklabels([0,'1 day','2 days','3 days','7 days','30 days'])
    fig.tight_layout()  # otherwise the right y-label is slightly clipped
    plt.show()
```

```
def weeklyMMSI(df):
    weeks = pd.date_range(datetime.datetime(2017,1,1),datetime.datetime(2019,1,1))
    df_out = pd.DataFrame(columns = ['week','n_mmsi'])
    for i in range(len(weeks)-1):
        mmsi = df['mmsi'][(df['DateTime']>=weeks[i]) & (df['DateTime']<weeks[i+1])].nunique()
        weeklyData = pd.Series([weeks[i],mmsi],index=df_out.columns)
        df_out = df_out.append(weeklyData,ignore_index=True)

    fig,ax1 = plt.subplots(figsize=(15,5))

    color = 'tab:blue'
    ax1.set_xlabel('Date')
    ax1.set_ylabel(' Number of Weekly Observed MMSI')
    ax1.plot(df_out['week'],df_out['n_mmsi'],color=color)
    ax1.tick_params(axis='y')

    plt.show()
    return df_out
```

```
#Function that returns all IMO numbers and corresponding mmsi
def mmsiIMO(df5):
    uniqueIMO = df5['imo_num'].unique()
    out = pd.DataFrame()
    for imo in uniqueIMO:
        temp = df5[['imo_num','mmsi']][df5['imo_num']==imo]
        out = out.append(temp.iloc[0],ignore_index=True)

    return out
```