```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed May 29 15:36:30 2019

@author: torjus



THIS IS THE MAIN PYTHON FILE THAT RUN THE MAJORITY OF ANALYSIS



"""

#masterfinal
import dataImportPreprocess as dIP
import initialAnalysis as iA
import datetime
import pandas as pd
import tradeFlowAnalysis as tFA
import AIS_Plotting as PAIS
import time
import densityPlots as dplot
import speedAnalysis as sA



################################################################################
#                          START OF DATAPREPERATION                            #
################################################################################
#Extract Original database from SQLite to dataframe
dataPrep = 0
if dataPrep ==1:
    extractOrig = 0
    if extractOrig ==1:
        filepath = ('/Volumes/Samsung_T5/torjusDB2017.db')
        df1,uniqueMMSI1 = dIP.dataExtraction1(filepath)
        df2,uniqueMMSI2 = dIP.dataExtraction2(filepath)
        df3,uniqueMMSI3 = dIP.dataExtraction3(filepath)
        df5,uniqueMMSI5,imoNum = dIP.dataExtraction5(filepath)

    extractFinal = 0 #This database is made from the steps under
    if extractFinal ==1:
        filepath2 = ('/Volumes/Samsung_T5/test.db')
        df_final,uniqueMMSI = dIP.dataExtraction1(filepath2)

    #Merge Dynamic Messages 1,2,3
    mergeDyn =0
    if mergeDyn == 1:
        df_merged,uniqueMMSIdyn = dIP.mergeDynamic(df1,df2,df3)
        #Conduct the error analysis on df_merged!!

    #Remove Error data
    errorRemove =0
    if errorRemove ==1:
        df_noError,uniqueMMSI = dIP.removeErrorData(df_merged)

    #Conduct frequency reduction
    freqRed =0
    if freqRed == 1:
        df_final,tot = dIP.reduceMessageFreq(df_noError,uniqueMMSI)

    #Extract vesseldata from seaweb
    vesselData = 0
    if vesselData ==1:
        df_V = dIP.importVesselData()

    #Save the final data to SQL
```

```
    toSQL1 = 0
    if toSQL1 ==1:
        df_to_sql = df_final.copy()
        df_to_sql.reset_index(drop=True,inplace=True)
        df_to_sql.drop(columns='DateTime',inplace=True)
        engine_name = 'sqlite:///test.db'
        dIP.makeNewSqlite(df_to_sql,engine_name)

    #Add additional information to the data
    addInfo = 0
    if addInfo ==1:
        df_info, df_no_info = dIP.addVesselinformation(df_final,df_V,uniqueMMSI)
```

```
    #Save the final data to SQL
    toSQL2 = 0
    if toSQL2 ==1:
        df_to_sql2 = df_info.copy()
        df_to_sql2.reset_index(drop=True,inplace=True)
        df_to_sql2.drop(columns='DateTime',inplace=True)
        engine_name2 = 'sqlite:///test2.db'
        dIP.makeNewSqlite(df_to_sql2,engine_name2)
```

```
###############################################################################
#                          END OF DATAPREPERATION                             #
###############################################################################

###############################################################################
#                          START OF INITIAL DATA ANALYSIS                     #
###############################################################################
initAnal =0
if initAnal ==1:
    #Conduct Error analysis (NB: Original data needed (df_merged))
    errorAnalysis =0
    if errorAnalysis == 1:
        dfError, speedErrorMonthly, errorVsNonError,totErrorVsNonError,\
        ratioErrorMonthly = iA.SpeedError(df_merged)
        iA.speedErrorMonthlyPlot(speedErrorMonthly,ratioErrorMonthly)

    finalDataAnalysis=1
    if finalDataAnalysis==1:
        #Messages init vs after error vs after freq red
        #iA.finaldataPlot(df_merged,df_noError,df_final)
        start2018 = datetime.datetime(2018,1,1)
        df_2017 = df_final[df_final['DateTime']<start2018].copy()
        df_2018 = df_final[df_final['DateTime']>=start2018].copy()
        df_messagegaps17, gaps17 = iA.MessageGapsShip(df_2017,uniqueMMSI)
        df_messagegaps18,gaps18 = iA.MessageGapsShip(df_2018,uniqueMMSI)
        iA.plotMessageGaps(gaps17,gaps18)
        weeklyMMSI = iA.weeklyMMSI(df_final)
```

```

###############################################################################
#                          END OF INITIAL DATA ANALYSIS                       #
###############################################################################

###############################################################################
#                          START OF Speed ANALYSIS                            #
###############################################################################
speedAnalysis = 0
if speedAnalysis ==1:
   df_q=df_info[df_info.category==4].copy() #Qmax/Qflex
   unique_q = df_q.mmsi.unique()

   df_p = df_info[df_info.category==3].copy()#NewPanamax
   unique_p = df_p.mmsi.unique()

   df_c = df_info[df_info.category==2].copy()#Conventional
   unique_c = df_c.mmsi.unique()

   df_s = df_info[df_info.category==1].copy()#Small-Scale
   unique_s = df_s.mmsi.unique()
```

```
    #Density Plots
    density =0
    if density ==1:
        dplot.densityqmax(df_q,unique_q)
        dplot.densityconv(df_c,unique_c)
        dplot.densitypanam(df_p,unique_p)
        dplot.densitysmall(df_s,unique_s)

    #monthly speed
    dfs_q = sA.speedAnalysisMonthly(df_q)
    dfs_p = sA.speedAnalysisMonthly(df_p)
    dfs_c = sA.speedAnalysisMonthly(df_c)
    dfs_s = sA.speedAnalysisMonthly(df_s)
    sA.speedPlot2(dfs_q,dfs_p,dfs_c,dfs_s,rates)

    #Speed histogram
    sA.histogramSpeedPlot(df_q,df_c,df_s,df_p)
```

```
################################################################################
#                          End OF Speed ANALYSIS                               #
################################################################################
```

```
################################################################################
#                          START OF Trade Flow ANALYSIS                        #
################################################################################
tradeFlowAnalysis = 0
if tradeFlowAnalysis ==1:
    a = time.time()
    #Import Predefined Nodes from Excel
    nodes1 = pd.read_excel('nodes.xlsx',sheet_name='diameter1')
    nodes2 = pd.read_excel('nodes.xlsx',sheet_name='diameter2')
    nodes3 = pd.read_excel('nodes.xlsx',sheet_name='diameter3')
    nodes4  = nodes1.copy()
    #Generate all polygons
    polygonsD1 = tFA.genAllTerminalAreas(nodes1)
    polygonsD2 = tFA.genAllTerminalAreas(nodes2)
    polygonsD3 = tFA.genAllTerminalAreas(nodes3)
    m1 = PAIS.ocean_polygon(polygonsD1)
    #m2 = PAIS.ocean_polygon(polygonsD2)
    #m3 = PAIS.ocean_polygon(polygonsD3)

    #Split 2017 and 2018 data
    start2018 = datetime.datetime(2018,1,1)
    df_2017 = df_final[df_final['DateTime']<start2018].copy()
    df_2018 = df_final[df_final['DateTime']>=start2018].copy()

    #Create a low speed and moored/anchored dataframe
    df_2017_LS = df_2017[(df_2017['sog']<2) & ((df_2017['nav_status']) == 1 | \
                         (df_2017['nav_status']==5))].copy()
    df_2018_LS = df_2018[(df_2018['sog']<2) & ((df_2018['nav_status']) == 1 | \
                         (df_2018['nav_status']==5))].copy()


    #Extract all data inside the nodes/polygons
    #2017
    df_polyD1_17,totTime1_17 = tFA.assignPolygon(df_2017,polygonsD1)
    df_polyD2_17,totTime2_17 = tFA.assignPolygon(df_2017,polygonsD2)
    df_polyD3_17,totTime3_17 = tFA.assignPolygon(df_2017,polygonsD3)
    df_polyD1_17_LS,totTime1_17_LS = tFA.assignPolygon(df_2017_LS,polygonsD1)
    portlogList1_17 = tFA.obtainUniqueVisits(df_polyD1_17,nodes1,df_V)
    portlogList2_17 = tFA.obtainUniqueVisits(df_polyD2_17,nodes2,df_V)
    portlogList3_17 = tFA.obtainUniqueVisits(df_polyD3_17,nodes3,df_V)
    portlogList1_17_LS = tFA.obtainUniqueVisits(df_polyD1_17_LS,nodes1,df_V)
    #2018
    df_polyD1_18,totTime1_18 = tFA.assignPolygon(df_2018,polygonsD1)
```

```
    df_polyD2_18,totTime2_18 = tFA.assignPolygon(df_2018,polygonsD2)
    df_polyD3_18,totTime3_18 = tFA.assignPolygon(df_2018,polygonsD3)
    df_polyD1_18_LS,totTime1_18_LS = tFA.assignPolygon(df_2018_LS,polygonsD1)
    portlogList1_18 = tFA.obtainUniqueVisits(df_polyD1_18,nodes1,df_V)
    portlogList2_18 = tFA.obtainUniqueVisits(df_polyD2_18,nodes2,df_V)
    portlogList3_18 = tFA.obtainUniqueVisits(df_polyD3_18,nodes3,df_V)
    portlogList1_18_LS = tFA.obtainUniqueVisits(df_polyD1_18_LS,nodes1,df_V)

    #Add the number of visits and accumulated LNG to the nodes
    for i,portlog in enumerate(portlogList1_17):
        visits = len(portlog)
        LNG1 = portlog.LNG.sum()
        nodes1['visits_2017'].iloc[i]=visits
        nodes1['LNG_2017'].iloc[i]=LNG1
    for i,portlog in enumerate(portlogList2_17):
        visits = len(portlog)
        LNG1 = portlog.LNG.sum()
        nodes2['visits_2017'].iloc[i]=visits
        nodes2['LNG_2017'].iloc[i]=LNG1
    for i,portlog in enumerate(portlogList3_17):
        visits = len(portlog)
        LNG1 = portlog.LNG.sum()
        nodes3['visits_2017'].iloc[i]=visits
        nodes3['LNG_2017'].iloc[i]=LNG1
    for i,portlog in enumerate(portlogList1_17_LS):
        visits = len(portlog)
        LNG1 = portlog.LNG.sum()
        nodes4['visits_2017'].iloc[i]=visits
        nodes4['LNG_2017'].iloc[i]=LNG1


    for i,portlog in enumerate(portlogList1_18):
        visits = len(portlog)
        LNG1 = portlog.LNG.sum()
        nodes1['visits_2018'].iloc[i]=visits
        nodes1['LNG_2018'].iloc[i]=LNG1
    for i,portlog in enumerate(portlogList2_18):
        visits = len(portlog)
        LNG1 = portlog.LNG.sum()
        nodes2['visits_2018'].iloc[i]=visits
        nodes2['LNG_2018'].iloc[i]=LNG1
    for i,portlog in enumerate(portlogList3_18):
        visits = len(portlog)
        LNG1 = portlog.LNG.sum()
        nodes3['visits_2018'].iloc[i]=visits
        nodes3['LNG_2018'].iloc[i]=LNG1
    for i,portlog in enumerate(portlogList1_18_LS):
        visits = len(portlog)
        LNG1 = portlog.LNG.sum()
        nodes4['visits_2018'].iloc[i]=visits
        nodes4['LNG_2018'].iloc[i]=LNG1
    #Write to excel
    path = 'LNG_TradeFLow.xlsx'
    writer = pd.ExcelWriter(path,engine = 'xlsxwriter')
    nodes1.to_excel(writer,sheet_name = 'diameter1')
    nodes2.to_excel(writer,sheet_name = 'diameter2')
    nodes3.to_excel(writer,sheet_name = 'diameter3')
    nodes4.to_excel(writer,sheet_name = 'diameter1_LS')
    writer.save()
    writer.close()
    b = time.time()
    c= b-a
#Tradeflow with low speed and moored/Anchored condition
tradeflowLowspeed =1
if tradeflowLowspeed == 1:
    speedMax = 2 #knots


##############################################################################
#                       End OF Trade Flow ANALYSIS                           #
```

```
##################################################################################################
```