```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 15 09:34:41 2019

@author: torjus


THIS PYTHON CODE IS USED TO GENERATE POLYGONS 

"""

import numpy as np
import json
import geog
import shapely.geometry
import time
import sqlite3
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
from datetime import timedelta
from shapely.geometry import MultiPoint
from math import radians, cos, sin, asin, sqrt


#This script defines areas of interest



#Large Areas of Interest

def generate_polygons():
    global polygons
    polygons = list()
    #East Pacific 
    EstP =[[-102,90],
           [-102,20],
           [-75,5],
           [-65,-18],
           [-75,-90],
           [-180,-90],
           [-180,90]]
    EstP.append(EstP[0]) #Adding the first point 
    polygons.append(EstP)

    #Atlantic Ocean
    Atl= [[-102,90],
          [-12,90],
          [-5.5,36],
          [25,0],
          [23,-90],
          [-75,-90],
          [-65,-18],
          [-75,5],
```

```
            [-102,20]]
    Atl.append(Atl[0]) #Adding the first point
    polygons.append(Atl)

    #Mediterranean Ocean
    Med = [[-5.5,36],
           [5,50],
           [77,50],
           [32.25,30.25],
           [25,0]]
    Med.append(Med[0]) #Adding the first point
    polygons.append(Med)

    #Arabian Gulf
    AG = [[25,0],
          [32.25,30.25],
          [77,50],
          [77,9.5]]
    AG.append(AG[0]) #Adding the first point
    polygons.append(AG)

    #North West Europe

    NWE = [[-5.5,36],
           [-12,90],
           [50,90],
           [50,65],
           [77,50],
           [5,50]]

    NWE.append(NWE[0]) #Adding the first point
    polygons.append(NWE)

    #North East Passage
    NEP = [[50,90],
           [180,90],
           [180,60],
           [77,50],
           [50,65]]
    NEP.append(NEP[0]) #Adding the first point
    polygons.append(NEP)

    #Far East
    FE = [[77,9.5],
          [77,50],
          [180,60],
          [180,-90],
          [110,-40]]
    FE.append(FE[0]) #Adding the first point
    polygons.append(FE)

    #Indian Ocean
    Ind = [[77,9.5],
           [110,-40],
           [180,-90],
           [23,-90],
           [25,0]]
    Ind.append(Ind[0]) #Adding the first point
    polygons.append(Ind)

def rayCasting(x,y,poly):
    n = len(poly)
    inside =False

    p1x,p1y = poly[0]
    for i in range(n+1):
        p2x,p2y = poly[i % n]
        if y > min(p1y,p2y):
            if y <= max(p1y,p2y):
                if x <= max(p1x,p2x):
                    if p1y != p2y:
                        xinters = (y-p1y)*(p2x-p1x)/(p2y-p1y)+p1x
                    if p1x == p2x or x <= xinters:
```

```
                        inside = not inside
        p1x,p1y = p2x,p2y

    return inside


def createTerminalArea(x,y):
    p = shapely.geometry.Point([x, y])

    n_points = 10 #number of points in polygon
    d_km = 60 #Diameter in km
    d =  d_km * 1000  #Diameter in meters
    angles = np.linspace(0, 360, n_points)
    polygon = geog.propagate(p, angles, d)
    polygon=polygon.tolist()
    return polygon

def createTerminalArea2(x,y,d_km):
    p = shapely.geometry.Point([x, y])

    n_points = 10 #number of points in polygon
    d =  d_km * 1000  #Diameter in meters
    angles = np.linspace(0, 360, n_points)
    polygon = geog.propagate(p, angles, d)
    polygon=polygon.tolist()
    return polygon
```

```
def genAllTerminalAreas(df):
    polygons = list()
    for i in range(0,len(df['longitude'])):
        p = createTerminalArea(df['longitude'][i],df['latitude'][i])
        p.append(p[0]) #Adding the first point
        polygons.append(p)

    return polygons


'''def inside(row,polygon,poly_number): #row is a series (row in a dataframe)
    x = row.longitude
    y = row.latitude
    if rayCasting(x,y,polygon):
        return poly_number
    else:
        return 0'''

def pointsInsidePolygon(polygons,df,poly_number):
    cluster = []

    for polygon in polygons :
        lon,lat = zip(*polygon)
        xMax = max(lon)
        xMin = min(lon)
        yMax = max(lat)
        yMin = min(lat)
        '''df_temp = df[df['longitude']<xMax]
        df_temp = df_temp[df_temp['longitude']>xMin]
        df_temp = df_temp[df_temp['latitude']>yMin]
        df_temp = df_temp[df_temp['latitude']<yMax]'''
        for i, row in enumerate(df.itertuples(),1):
            x = row.longitude
            y = row.latitude
            if rayCasting(x,y,polygon):
                cluster.append(1)
            else:
                cluster.append(0)
    df_out = df.copy()
    df_out['cluster']=cluster

    '''df_out = df_out[df_out['cluster']!=0]'''

    return df_out


def insidePolygon(row,polygons):
```

```
    x = row.longitude
    y = row.latitude
    for i, polygon in enumerate(polygons):
        lon,lat = zip(*polygon)
        xMax = max(lon)
        xMin = min(lon)
        yMax = max(lat)
        yMin = min(lat)
        if x<xMax:
            if x>xMin:
                if y<yMax:
                    if y>yMin:
                        if rayCasting(x,y,polygon):
                            return i
    return -1 #If not inside any defined polygon
```

```
def haversine(lon1, lat1, lon2, lat2):
    """This function is not created by Torjus Halden
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371 # Radius of earth in kilometers. Use 3956 for miles
    return c * r

def assignPolygon(df,polygons):
    start = time.perf_counter()
    df['cluster']=-2 #-2 means not yet classified
    df['cluster'] = df.apply(insidePolygon,args=[polygons],axis=1)
    end = time.perf_counter()
    tot = end-start
    return df,tot


def inside(row):
    x = row.longitude
    y = row.latitude
    if rayCasting(x,y,polygon):
        return 1
    else:
        return 0
```

```
#Extract data inside Kuwait Terminal
extract = 0
if extract ==1:
    df['cluster']=-1
    df['cluster']=df.apply(inside,axis=1)
    kuwait = df[df['cluster']==1]
```