AIS Anomaly Detection

- Fetch Broadcast Data per Year
- Iterate thru each unique MMSI
- - Identify anomal coordinate for a given vessel
- - Flag any Anomaly and Aggregate all Vessel Data into dataframe along with various stats: Mean, Median, etc
- Output to Statistic_{Year}.csv

In [1]:
```
# from IPython.display import Image, HTML
import os
import numpy as np
import math
import pandas as pd
import datetime
from glob import glob
import geopy.distance
import folium
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import seaborn as sns; sns.set()

import warnings
warnings.filterwarnings("ignore")           # Suppress Warning
```

Global Variables

In [2]:
```
# s3://vault-data-corpus/vessel data/ConsolidatedAIS/
WorkingFolder = "data/vessel data/ConsolidatedAIS/"
OutputDir = WorkingFolder

PROC_YEAR = '2009'
MAX_CLUSTER = 5
MIN_PROCESS_ROW = 500          # Require min number of rows available to calc cluster
```

Load Broadcast Data

In [3]:
```
Broadcast = pd.read_csv(WorkingFolder + "Broadcast_{}.csv".format(PROC_YEAR), sep=",", parse_dates=['date_time'])
Broadcast.head()
```

Out[3]:

	mmsi_id	date_time	lat	lon	speed_over_ground	course_over_ground	voyage_id	heading	status
0	367047170	2008-12-31 23:58:59	45.633627	-122.715003	0	199	1	511	0
1	366763770	2008-12-31 23:58:59	46.027173	-122.869968	0	325	2	511	0
2	368494000	2008-12-31 23:58:59	47.542502	-122.330622	0	354	3	511	0
3	366116000	2008-12-31 23:58:59	48.512612	-122.645723	0	345	4	55	0
4	316003289	2008-12-31 23:58:59	49.144225	-123.033517	0	172	5	96	15

In [4]:
```
print("Raw Count:", Broadcast.shape[0])
```

```
Raw Count: 17749932
```

In [5]:
```
# Make sure voyage id is not null
Broadcast['voyage_id'] = Broadcast['voyage_id'].fillna(0)
Broadcast = Broadcast.astype({"voyage_id": int})           # cast type to int
```

Clustering via K-means

- https://github.com/JosephMagiya/Clustering-GPS-Co-ordinates--Forming-Regions./blob/master/Clustering-GPS-Co-ordinates--Forming-Regions.ipynb (https://github.com/JosephMagiya/Clustering-GPS-Co-ordinates--Forming-Regions./blob/master/Clustering-GPS-Co-ordinates--Forming-Regions.ipynb)

In [6]:
```
Header = ['mmsi_id', 'PingRecStart', 'PingRecEnd', 'TotalPing', 'LatStd', 'LonStd',
          'MaxSOG', 'MinSOG', 'MeanSOG', 'MedianSOG', 'StdSOG',
          'MaxCOG', 'MinCOG', 'MeanCOG', 'MedianCOG', 'StdCOG', 'AnoThreshold', 'AnoClusterCount']
arrData = list()

# Identify Anomaly
for mmsi in Broadcast.mmsi_id.unique():
#     mmsi = 366985110           #**************** Test data

    df = Broadcast.loc[Broadcast.mmsi_id==mmsi, ['date_time', 'lat', 'lon', 'speed_over_ground', 'course_over_grou

    if df.shape[0] < MIN_PROCESS_ROW: continue              # Skip, not enough ping records for clustering

    Anomaly_THRESHOLD = math.ceil(df.shape[0]/15000)        # dynamically determine the anomaly threshold based on

#     # **** for testing - Fudge some anomaly
#     df.loc[df.date_time=='2017-01-01 00:06:50', 'lat'] = 70.25
#     # ***********************************

    # Extract Date and Hour
    df['PingDate'] = df['date_time'].dt.date
    # df['PingHour'] = df['date_time'].dt.hour

    # Extract Stats
    Stat = df.agg(['count', 'max', 'min', 'mean', 'median', 'std'])
    X = df[['lat', 'lon']]

    kmeans = KMeans(n_clusters=MAX_CLUSTER, init='k-means++')
    kmeans.fit(X)                                       # Compute k-means clustering.
    df['cluster_label'] = kmeans.predict(X)             # Labels of each point

    # Cluster, ping counts
    PingCluster = df.groupby('cluster_label')['date_time'].count().reset_index()

    # Identify Anomaly in coordinate, if any
    AnomalyCluster = PingCluster.loc[PingCluster.date_time <= Anomaly_THRESHOLD, 'cluster_label'].values

    # Construct Vessel Stat Record
    arrData.append([
        mmsi, Stat.PingDate[2], Stat.PingDate[1], Stat.PingDate[0], Stat.lat[5], Stat.lon[5],
        Stat.speed_over_ground[1], Stat.speed_over_ground[2], Stat.speed_over_ground[3], Stat.speed_over_ground[4],
        Stat.course_over_ground[1], Stat.course_over_ground[2], Stat.course_over_ground[3], Stat.course_over_ground
        Anomaly_THRESHOLD, len(AnomalyCluster)
    ])

#     # ********************* Alert Code for identify 1st Anomaly case ******************************
#     # Locate anomaly data points
#     AnaCoordinate = df.loc[df.cluster_label.isin(AnomalyCluster)]

#     if AnaCoordinate.shape[0] > 0:
#         print("MMSI: {} \tPings: {} \tThreshold: {}".format(mmsi, df.shape[0], Anomaly_THRESHOLD))
#         for index, row in AnaCoordinate.sort_values('cluster_label').iterrows():
#             print("\tCluster:{} ({}, {}) {}".format(row.cluster_label, row.lat, row.lon, row.date_time))

#         break   # For Testing - get out early
#     # ***************************************************************************************

#     break   # For Testing - get out early

dfStat = pd.DataFrame(arrData, columns=Header)

dfStat.head()
```

Out[6]:

	mmsi_id	PingRecStart	PingRecEnd	TotalPing	LatStd	LonStd	MaxSOG	MinSOG	MeanSOG	MedianSOG	StdSOG	MaxCOG	MinCOG	Mea
0	367047170	2008-12-31	2009-01-31	38078	0.184945	0.240594	102.0	0.0	3.384710	1.0	4.041475	360.0	0.0	179.5
1	366763770	2008-12-31	2009-01-31	27019	0.107984	0.110426	15.0	0.0	0.980051	0.0	2.397335	359.0	0.0	187.4
2	368494000	2008-12-31	2009-01-31	43422	0.000037	0.000035	0.0	0.0	0.000000	0.0	0.000000	360.0	0.0	180.0
3	366116000	2008-12-31	2009-01-09	11907	0.163482	0.227486	10.0	0.0	0.538339	0.0	1.993043	360.0	0.0	249.2
4	316003289	2008-12-31	2009-01-31	43378	0.195541	0.275708	18.0	0.0	6.057564	0.0	6.823938	360.0	0.0	175.0

In [7]:
```
# Output Stat Data
dfStat.to_csv(OutputDir + "Statistic_{}.csv".format(PROC_YEAR), index=None, header = True)
```

In [8]:
```
Stat
```

Out[8]:

	date_time	lat	lon	speed_over_ground	course_over_ground	PingDate
count	791	791.000000	791.000000	791.000000	791.000000	791
max	2009-01-31 23:58:59	32.798445	-122.471355	19.000000	127.000000	2009-01-31
min	2009-01-31 10:34:00	30.910263	-125.996375	8.000000	118.000000	2009-01-31
mean	2009-01-31 17:16:06.317319936	31.835202	-124.190095	15.906448	122.116308	NaN
median	NaN	31.761025	-124.049887	18.000000	122.000000	NaN
std	NaN	0.519859	0.970443	3.685256	1.470081	NaN

In [9]:
```
df.head()
```

Out[9]:

	date_time	lat	lon	speed_over_ground	course_over_ground
17749646	2009-01-31 23:58:59	34.242787	-120.000647	15	284