Predictive analysis of naval incidents in the USA, 2002 - 2015:

Annex 3.4. Preprocess MergedActivity & VesselBalancedSample

Author: Oscar Anton

Date: 2024

License: CC BY-NC-ND 4.0 DEED

Version: 0.9

0. Loadings

Libraries

In [1]:
```
# General data management

import pandas as pd

# Visualization
import plotly.graph_objects as go

# Ignore warnings
import warnings
warnings.filterwarnings("ignore")
```

General variables

In [2]:
```
# Main data folders
casualty_pollution_folder = 'DataCasualtyAndPollution'
weather_ocean_folder = 'DataWeatherOcean'
weather_river_folder = 'DataWeatherRiver'
merged_activity_folder = 'DataMergedActivity'

# Toggle for export data to external file
file_export_enabled = False
```

1. Data Acquisition

1.1. Load Base Dataframes

In [3]:
```
Events = pd.read_feather(casualty_pollution_folder + '/' + 'Events.feather')
print(f'Events {Events.shape} imported from {casualty_pollution_folder}')
Vessel = pd.read_feather(casualty_pollution_folder + '/' + 'Vessel.feather')
print(f'Vessel {Vessel.shape} imported from {casualty_pollution_folder}')

WeatherOcean = pd.read_feather(weather_ocean_folder + '/' + 'WeatherOcean.feather')
```

```
print(f'WeatherOcean {WeatherOcean.shape} imported from {weather_ocean_folder}')
WeatherRiver = pd.read_feather(weather_river_folder + '/' + 'WeatherRiver.feather')
print(f'WeatherRiver {WeatherRiver.shape} imported from {weather_ocean_folder}')

Injury = pd.read_feather(casualty_pollution_folder + '/' + 'Injury.feather')
print(f'Injury {Injury.shape} imported from {casualty_pollution_folder}')
VslPoll = pd.read_feather(casualty_pollution_folder + '/' + 'VslPoll.feather')
print(f'VslPoll {VslPoll.shape} imported from {casualty_pollution_folder}')

Activity = pd.read_feather(casualty_pollution_folder + '/' + 'Activity.feather')
print(f'Activity {Activity.shape} imported from {casualty_pollution_folder}')
```

```
Events (77674, 18) imported from DataCasualtyAndPollution
Vessel (1346644, 26) imported from DataCasualtyAndPollution
WeatherOcean (32520, 12) imported from DataWeatherOcean
WeatherRiver (11274, 12) imported from DataWeatherOcean
Injury (10367, 14) imported from DataCasualtyAndPollution
VslPoll (21827, 14) imported from DataCasualtyAndPollution
Activity (104476, 4) imported from DataCasualtyAndPollution
```

1.2. Variable Preselection

In [4]:
```
# From the Events dataframe
Events = Events[['activity_id', 'vessel_id', 'vessel_name', 'vessel_class', 'waterway_na
print(f'Events new shape: {Events.shape}')

# From the Vessel dataframe. Only include vessels registered in Events
Vessel = Vessel[['vessel_id', 'gross_ton', 'length', 'flag_abbr', 'classification_societ
Vessel = Vessel[Vessel['vessel_id'].isin(Events['vessel_id'])]
print(f'Vessel new shape: {Vessel.shape}')

# From the WeatherOcean dataframe
WeatherOcean = WeatherOcean[['activity_id', 'wind_speed', 'visibility', 'air_temp', 'wav
print(f'WeatherOcean new shape: {WeatherOcean.shape}')

# From the WeatherRiver dataframe
WeatherRiver = WeatherRiver.assign(wind_speed=WeatherRiver['awnd'], air_temp=(WeatherRiv
WeatherRiver = WeatherRiver[['activity_id', 'wind_speed', 'air_temp']]
print(f'WeatherRiver new shape: {WeatherRiver.shape}')

# From the Injury dataframe
Injury = Injury[['activity_id', 'vessel_id', 'accident_type', 'casualty_type_desc']]
print(f'Injury new shape: {Injury.shape}')

# From the VslPoll dataframe
VslPoll = VslPoll[['activity_id', 'vessel_id', 'chris_cd', 'discharge_amnt_total', 'dama
print(f'VslPoll new shape: {VslPoll.shape}')
```

```
Events new shape: (77674, 13)
Vessel new shape: (30455, 8)
WeatherOcean new shape: (32520, 5)
WeatherRiver new shape: (11274, 3)
Injury new shape: (10367, 4)
VslPoll new shape: (21827, 5)
```

2. Dataframe build: merged_activity

2.1. Data join

In [5]:
```
# Events and vessel data
events_and_vessels = pd.merge(Events, Vessel, how='left', on='vessel_id').drop_duplicate

# Variable adaptation
events_and_vessels['build_year'] = pd.to_numeric(events_and_vessels['build_year'], error
events_and_vessels['gross_ton'] = pd.to_numeric(events_and_vessels['gross_ton'], errors=
events_and_vessels['length'] = pd.to_numeric(events_and_vessels['length'], errors='coerc
events_and_vessels['date'] = pd.to_datetime(events_and_vessels['date']).dt.date

# Delete fake 'nan' values
events_and_vessels = events_and_vessels.replace('nan', '', regex=True)

# Land weather
events_river = events_and_vessels[events_and_vessels['watertype'] == 'river']
events_river_weather = pd.merge(events_river, WeatherRiver, how='inner', on='activity_id

events_river_weather['visibility'] = None
events_river_weather['wave_hgt'] = None

# Maritime weather
events_ocean = events_and_vessels[events_and_vessels['watertype'] == 'ocean']
events_ocean_weather = pd.merge(events_ocean, WeatherOcean, how='left', on='activity_id'

# Vertical union of River + Ocean. Records sorted by date and id
merged_activity = pd.concat([events_ocean_weather, events_river_weather]).loc[:, [
    'activity_id', 'date', 'hour',
    'region', 'latitude', 'longitude',
    'watertype', 'event_type', 'damage_status',
    'vessel_id', 'imo_number', 'vessel_name', 'vessel_class',
    'build_year', 'gross_ton', 'length',
    'flag_abbr', 'classification_society', 'solas_desc',
    'air_temp', 'wind_speed', 'wave_hgt', 'visibility'
]].sort_values(by=['date', 'activity_id']).reset_index(drop=True)

# Check dataframe shape
print(f'merged_activity {merged_activity.shape} created')
```

```
merged_activity (77674, 23) created
```

2.2. Add new variables from previous tables

In [6]:
```
# Damage assessment
merged_activity['damage_assessment'] = Events.merge(Activity, on='activity_id', how='lef

# Personal injuries
merged_activity['casualty'] = Events.merge(Injury, on='activity_id', how='left')['casual

# Pollution
merged_activity['pollution'] = Events.merge(VslPoll, on='activity_id', how='left')['chri

# Age
merged_activity['age'] = pd.to_datetime(merged_activity['date']).dt.year - pd.to_datetim

# Check dataframe shape
print(f'merged_activity {merged_activity.shape} updated')
```

```
merged_activity (77674, 27) updated
```

2.3. Data quality filters

In [7]:
```python
# Filter NAs
merged_activity = merged_activity.dropna(thresh=merged_activity.shape[1]-5)

# Filter unlikely values
merged_activity  = merged_activity [
    (merged_activity ['gross_ton'] >= 1) & (merged_activity ['gross_ton'] <= 250000) &
    (merged_activity ['build_year'] >= 1800) & (merged_activity ['build_year'] <= 2015)
    (merged_activity ['length'] >= 1) & (merged_activity ['length'] <= 1250)
].drop_duplicates(subset=['activity_id', 'vessel_id', 'event_type'], keep='first')

# Check dataframe shape
print(f'merged_activity {merged_activity.shape} updated')
```

```
merged_activity (68565, 27) updated
```

2.4. Classification model target variable: event_class

In [8]:
```python
# Function from event_type to event_class
def classify_event(event_type):
    if event_type in ["Sinking", "Implosion", "Capsize", "Loss of Stability", "Vessel Ma
        return "Critical Events"
    elif event_type in ["Loss of Electrical Power", "Fire", "Emergency Response", "Explo
        return "Onboard Emergencies"
    elif event_type in ["Grounding", "Allision", "Collision"]:
        return "Maritime Accidents"
    elif event_type in ["Material Failure (Vessels)", "Material Failure (Non-vessels)",
        return "Material Issues"
    elif event_type in ["Damage to the Environment", "Damage to Cargo", "Fouling", "Evas
        return "Third-party Damages"
    else:
        return None

# Apply function
merged_activity['event_class'] = merged_activity['event_type'].apply(classify_event)

# Check new variable counts
merged_activity['event_class'].value_counts()
```

Out[8]:
```
event_class
Maritime Accidents     18518
Material Issues        17343
Critical Events        17100
Third-party Damages     8882
Onboard Emergencies     6722
Name: count, dtype: int64
```

2.5. Export merged_activity dataframe to external file

In [9]:
```python
# R Data synchronization
import pyreadr
merged_activity = pd.DataFrame(pyreadr.read_r(merged_activity_folder + '/' + 'MergedActi
merged_activity['build_year'] = pd.to_numeric(merged_activity['build_year'], errors='coe
merged_activity['date'] = pd.to_datetime(merged_activity['date'], errors='coerce')

# Export to external file
if file_export_enabled :
    merged_activity.reset_index().to_feather(merged_activity_folder + '/' + 'merged_acti
    print(f'merged_activity {merged_activity.shape} exported to {merged_activity_folder}
else:
    merged_activity = pd.read_feather(merged_activity_folder + '/' + 'merged_activity.fe
    print(f'merged_activity {merged_activity.shape} imported from {merged_activity_folde
```

```
merged_activity (68000, 29) imported from DataMergedActivity
```

3. Dataframe build: vessel_balanced_sample

In [10]:
```
# Read all vessel data
Vessel = pd.read_feather(casualty_pollution_folder + '/' + 'Vessel.feather')
print(f'Vessel {Vessel.shape} imported from {casualty_pollution_folder}')
```

```
Vessel (1346644, 26) imported from DataCasualtyAndPollution
```

3.1. Vessels involved in incidents

In [11]:
```
# Variable selection from merged_activity
VesselActivity = merged_activity[['vessel_id', 'imo_number', 'vessel_name', 'vessel_clas
                                  'gross_ton', 'length', 'flag_abbr', 'classification_soc
                                  'event_type', 'damage_status']].drop_duplicates()

# Check dataframe shape
print(f'VesselActivity {VesselActivity.shape} created')
```

```
VesselActivity (54918, 12) created
```

3.2. Vessels not involved in incidents

In [12]:
```
# Find vessels not included in merged_activity
VesselNoActivity = Vessel[~Vessel['vessel_id'].isin(merged_activity['vessel_id'])]

# Variable adaptation
VesselNoActivity['build_year'] = pd.to_numeric(VesselNoActivity['build_year'], errors='c
VesselNoActivity['gross_ton'] = pd.to_numeric(VesselNoActivity['gross_ton'], errors='coe
VesselNoActivity['length'] = pd.to_numeric(VesselNoActivity['length'], errors='coerce')

# Filter unlikely values
VesselNoActivity = VesselNoActivity [
    (VesselNoActivity['gross_ton'] >= 1) & (VesselNoActivity['gross_ton'] <= 250000) &
    (VesselNoActivity['build_year'] >= 1800) & (VesselNoActivity['build_year'] <= 2015)
    (VesselNoActivity['length'] >= 1) & (VesselNoActivity['length'] <= 1250)
].drop_duplicates(keep='first')

# Variable selection
VesselNoActivity = VesselNoActivity[['vessel_id', 'imo_number', 'vessel_name', 'vessel_c
                                     'gross_ton', 'length',
                                     'flag_abbr', 'classification_society', 'solas_desc'
VesselNoActivity['event_type'] = 'No event'
VesselNoActivity['damage_status'] = 'Undamaged'

# Balanced Sample: same length
VesselNoActivitySample = VesselNoActivity.sample(n=len(VesselActivity))

# Check dataframe shape
print(f'VesselNoActivitySample {VesselNoActivitySample.shape} created')
```

```
VesselNoActivitySample (54918, 12) created
```

3.3. Involved and Not involved join

In [13]:
```python
# Join above dataframes
VesselBalancedSample = pd.concat([VesselActivity, VesselNoActivitySample], axis=0)

# Check dataframe shape
VesselBalancedSample['event_type'].value_counts().head()
```

Out[13]:
```
event_type
No event                        54918
Material Failure (Vessels)      11463
Grounding                        7909
Vessel Maneuverability           7511
Damage to the Environment        6439
Name: count, dtype: int64
```

3.4. Export dataframe to external file

In [14]:
```python
# R Data synchronization
import pyreadr
VesselBalancedSample = pd.DataFrame(pyreadr.read_r(merged_activity_folder + '/' + 'Vesse
VesselBalancedSample['build_year'] = pd.to_numeric(VesselBalancedSample['build_year'], e

# Export joined dataframe to external file
if file_export_enabled :
    VesselBalancedSample.reset_index().to_feather(merged_activity_folder + '/' + 'Vessel
    print(f'VesselBalancedSample {VesselBalancedSample.shape} exported to {merged_activi
else:
    VesselBalancedSample = pd.read_feather(merged_activity_folder + '/' + 'VesselBalance
    print(f'VesselBalancedSample {VesselBalancedSample.shape} imported to {merged_activi
```

```
VesselBalancedSample (109836, 13) imported to DataMergedActivity
```

4. Data verification

4.1. Dataframes structures

In [15]:
```python
# Print first observations
merged_activity.head()
```

Out[15]:

	index	activity_id	date	hour	region	latitude	longitude	watertype	event_type	da
0	0	1475897	2002-01-01	03:45	Mississippi	37.017330	-88.274720	river	Grounding	
1	1	1475897	2002-01-01	03:45	Mississippi	37.017330	-88.274720	river	Grounding	
2	2	1477008	2002-01-01	13:53	East Coast	39.322020	-76.363650	ocean	Damage to the Environment	
3	3	1477373	2002-01-01	18:10	Mississippi	31.525833	-87.971667	river	Material Failure (Vessels)	
4	4	1477402	2002-01-01	10:00	Gulf of Mexico	30.641667	-88.034167	ocean	Grounding	

5 rows × 29 columns

In [16]:
```python
# Print first observations
VesselBalancedSample.head()
```

Out[16]:

	index	vessel_id	imo_number	vessel_name	vessel_class	build_year	gross_ton	length	flag_a
0	0	5820		ISABELLA MARIE	Recreational	1999	14	32.8	
1	1	170582		TERMINATOR	Fishing Vessel	1979	17	38.1	
2	2	257931		SUMMER ISLE	Recreational	1984	8	29.9	
3	3	151752		NORJERNAN	Passenger Ship	1976	18	35.2	
4	4	308953		NONSENSE	Recreational	1987	8	27.0	

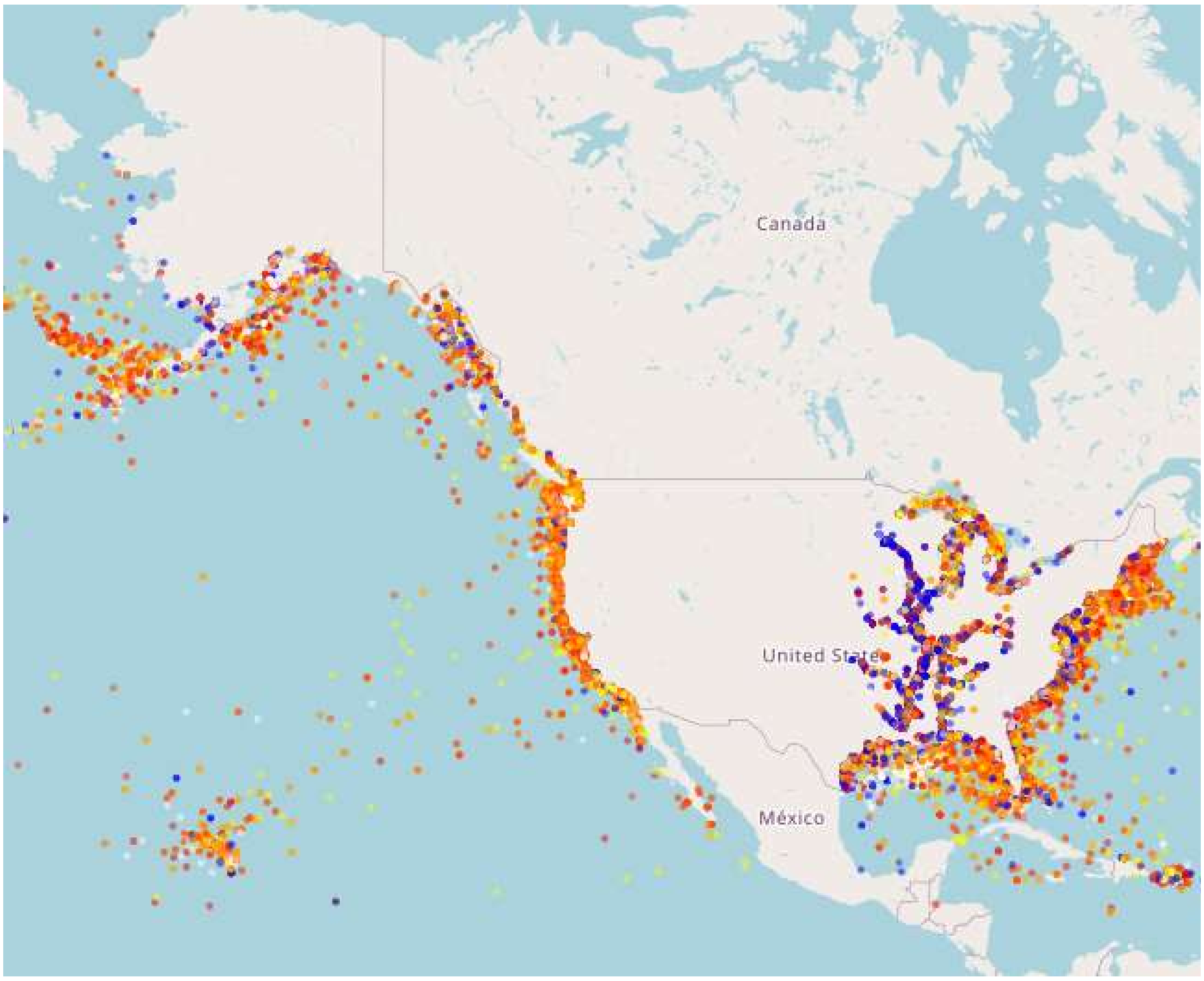
4.2. Map visualization (merged_activity)

In [17]:
```python
# Create figure object
fig = go.Figure()

# Aggregate WeatherRiver points
fig.add_trace(go.Scattermapbox(
    lat=merged_activity['latitude'],
    lon=merged_activity['longitude'],
    mode='markers',
    marker=dict(size=5,
                color=merged_activity['event_class'].map({'Critical Events': 'red',
                                                          'Onboard Emergencies': 'orangered',
                                                          'Maritime Accidents': 'blue',
                                                          'Material Issues': 'yellow',
                                                          'Third-party Damages': 'white'}),
                opacity=0.5),
    text=merged_activity.apply(lambda row:f"event_class:{row['event_class']}<br>event_ty
))

# Set up map design
fig.update_layout(
    margin ={'l':0,'t':0,'b':0,'r':0},
    mapbox = {
        'style': "open-street-map",
        'center': {'lon': -112, 'lat': 48},
        'zoom': 2})

# Show map
fig.show()
```


Canada
United States
México