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

Annex 3.2. Preprocess Weather Ocean

> 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 numpy as np
import pandas as pd

# File management
import os
import tarfile

# Visualization
import plotly.graph_objects as go
import plotly.express as px
```

General Variables

In [2]:

```
# Main data folders
import_data_folder= 'RawDataWeatherOcean'
export_data_folder= 'DataWeatherOcean'

# Toggle for export data to external file
file_export_enabled = False
# Toggle for calculations that takes a long time
protracted_calculation_enabled = False
```

1. Data Acquisition

1.1. Compile monthly maritime meteorology data from the NOAA website

In [3]:

```
if protracted_calculation_enabled :
    # Initiate list to store DataFrames from each .csv file
    filtered_dataframes = []

    # Columns to select from each CSV file
    selected_columns = {
        'STATION': str,
        'DATE': str,
        'LATITUDE': 'float32',
        'LONGITUDE': 'float32',
        'PAST_WX': 'float32',
        'WIND_SPEED': 'float32',
        'VISIBILITY': 'float32',
        'AIR_TEMP': 'float32',
        'WAVE_HGT': 'float32'
    }

    # Iterate over the .tar.gz files in the folder
    for tar_file in os.listdir(import_data_folder):
        if tar_file.endswith('.tar.gz'):
            tar_file_path = os.path.join(import_data_folder, tar_file)

            # Extract the .tar.gz file
            with tarfile.open(tar_file_path, 'r:gz') as tar:
                # Find .csv files within the .tar.gz
                csv_files = [member for member in tar.getmembers() if member.name.endswi

                # Read each .csv file and store in a DataFrame
                for csv_file in csv_files:
                    with tar.extractfile(csv_file) as file:
                        # Read the CSV file and handle missing columns
                        df = pd.read_csv(file, index_col=False, dtype=selected_columns).
                            reindex(columns=selected_columns.keys())

                        # Apply filter for NAs
                        df_filtered = df.dropna(subset=['STATION', 'LONGITUDE', 'LATITUD
                            dropna(thresh=len(df.columns) - 4)

                        # Apply filter for bounding box
                        df_filtered = df_filtered[df_filtered['LONGITUDE'].between(-180,
                              df_filtered['LATITUDE'].between(15, 70)]

                        filtered_dataframes.append(df_filtered)

    # Concatenate all DataFrames into merged one
    marine_stations_comb_1 = pd.concat(filtered_dataframes, ignore_index=True)
    # Column names to lowercase
    marine_stations_comb_1.columns = marine_stations_comb_1.columns.str.lower()
    print(f'marine_stations_comb_1 {marine_stations_comb_1.shape} created')
else:
    marine_stations_comb_1 = pd.read_feather(export_data_folder + '/' + 'marine_stations
    print(f'marine_stations_comb_1 {marine_stations_comb_1.shape} imported from {export_
```

```
marine_stations_comb_1 (97958906, 9) imported from DataWeatherOcean
```

1.2. Export dataframe

In [4]:

```
# Load or export to external file
if file_export_enabled :
    marine_stations_comb_1.to_feather(export_data_folder + '/' + 'marine_stations_comb_1
    print(f'marine_stations_comb_1 {marine_stations_comb_1.shape} exported to {export_da
else:
    marine_stations_comb_1 = pd.read_feather(export_data_folder + '/' + 'marine_stations
    print(f'marine_stations_comb_1 {marine_stations_comb_1.shape} imported to {export_da
```

```
marine_stations_comb_1 (97958906, 9) imported to DataWeatherOcean
```

2. Summarize

2.1. Daily means for Stations' values

In [5]:

```
# Extract only date, leaving hour
marine_stations_comb_1['date'] = pd.to_datetime(marine_stations_comb_1['date']).dt.date

# Select values to summarize
values = list(['latitude', 'longitude', 'past_wx',
               'wind_speed', 'visibility', 'air_temp', 'wave_hgt'])

# Calculate the mean according to STATION and DATE
marine_stations_daily_2 = (marine_stations_comb_1
                           .groupby(['station', 'date'])[values]
                           .mean()
                           .reset_index())

# Save to external file
if file_export_enabled :
    marine_stations_daily_2.to_feather(export_data_folder + '/' + 'marine_stations_daily
    print(f'marine_stations_daily_2 {marine_stations_daily_2.shape} exported to {export_
else:
    marine_stations_daily_2 = pd.read_feather(export_data_folder + '/' + 'marine_station
    print(f'marine_stations_daily_2 {marine_stations_daily_2.shape} imported to {export_
```

```
marine_stations_daily_2 (3718069, 9) imported to DataWeatherOcean
```

3. Join activity_id

3.1. Load ocean events data

In [6]:

```
# Load dataframe
Events = pd.read_feather('DataCasualtyAndPollution' + '/' + 'Events.feather')

# Variable selection
EventsOcean = Events[(Events.watertype == 'ocean')][['activity_id', 'date', 'longitude',

# Extract only date, leaving hour
EventsOcean['date'] = pd.to_datetime(EventsOcean['date']).dt.date

# Drop duplicates
EventsOcean = EventsOcean.drop_duplicates()

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

```
EventsOcean (32520, 4) created
```

3.2. Nearest weather observation to each ocean incident

In [7]:

```
# Function to calculate nearest weather observation
def near_observation(incident):
    # Select data corresponding to this Activity_id
    coord_incident = EventsOcean[EventsOcean['activity_id'] == incident].iloc[0]
    # Select all weather observations for this day
    coord_station = marine_stations_daily_2[(marine_stations_daily_2['date'] == coord_in

    # Approximate distances
    coord_station['station_dist'] = np.sqrt((coord_station['latitude'] - coord_incident[
                                            (coord_station['longitude'] - coord_incident

    # Return the recorded weather observation located at minimum distance
    min_distance_row = coord_station[coord_station['station_dist'] == coord_station['sta
    # Add activity_id to weather data
    min_distance_row['activity_id'] = incident

    #if coord_station.empty:
        #return pd.Series(dtype='float64')
    return min_distance_row.drop_duplicates(subset=['activity_id'], keep='first')

# Concatenate function returns to create a dataframe
if protracted_calculation_enabled :
    WeatherOcean = pd.concat([near_observation(incident) for incident in EventsOcean['ac
    print(f'WeatherOcean {WeatherOcean.shape} created')
else:
    WeatherOcean = pd.read_feather(export_data_folder + '/' + 'WeatherOcean.feather')
    print(f'WeatherOcean {WeatherOcean.shape} imported from {export_data_folder}')
```

```
WeatherOcean (32520, 12) imported from DataWeatherOcean
```

In [8]:

```
# Export to external file
if file_export_enabled :
    WeatherOcean.reset_index().to_feather(export_data_folder + '/' + 'WeatherOcean.feath
    print(f'WeatherOcean {WeatherOcean.shape} exported to {export_data_folder}')
else:
    WeatherOcean = pd.read_feather(export_data_folder + '/' + 'WeatherOcean.feather')
    print(f'WeatherOcean {WeatherOcean.shape} imported from {export_data_folder}')
```

```
WeatherOcean (32520, 12) imported from DataWeatherOcean
```

4. Data check

4.1. Dataframe structure

In [9]:

```
# Check values printing first observations
WeatherOcean.head()
```

Out[9]:

	index	station	date	latitude	longitude	past_wx	wind_speed	visibility	air_temp
0	3492779	WKWB	2013-04-22	32.700001	-117.199997	NaN	17.772728	NaN	148.181824
1	2335232	MRSL1	2013-06-10	29.440001	-92.059998	NaN	26.545454	NaN	285.863647
2	2736287	PTWW1	2013-06-14	48.110001	-122.760002	NaN	35.391304	NaN	124.041664
3	2425572	NTKM3	2013-06-08	41.290001	-70.099998	NaN	59.478260	NaN	171.875000
4	1409210	CECC1	2013-04-24	41.750000	-124.180000	NaN	46.208332	NaN	133.105270

4.2. Map visualization

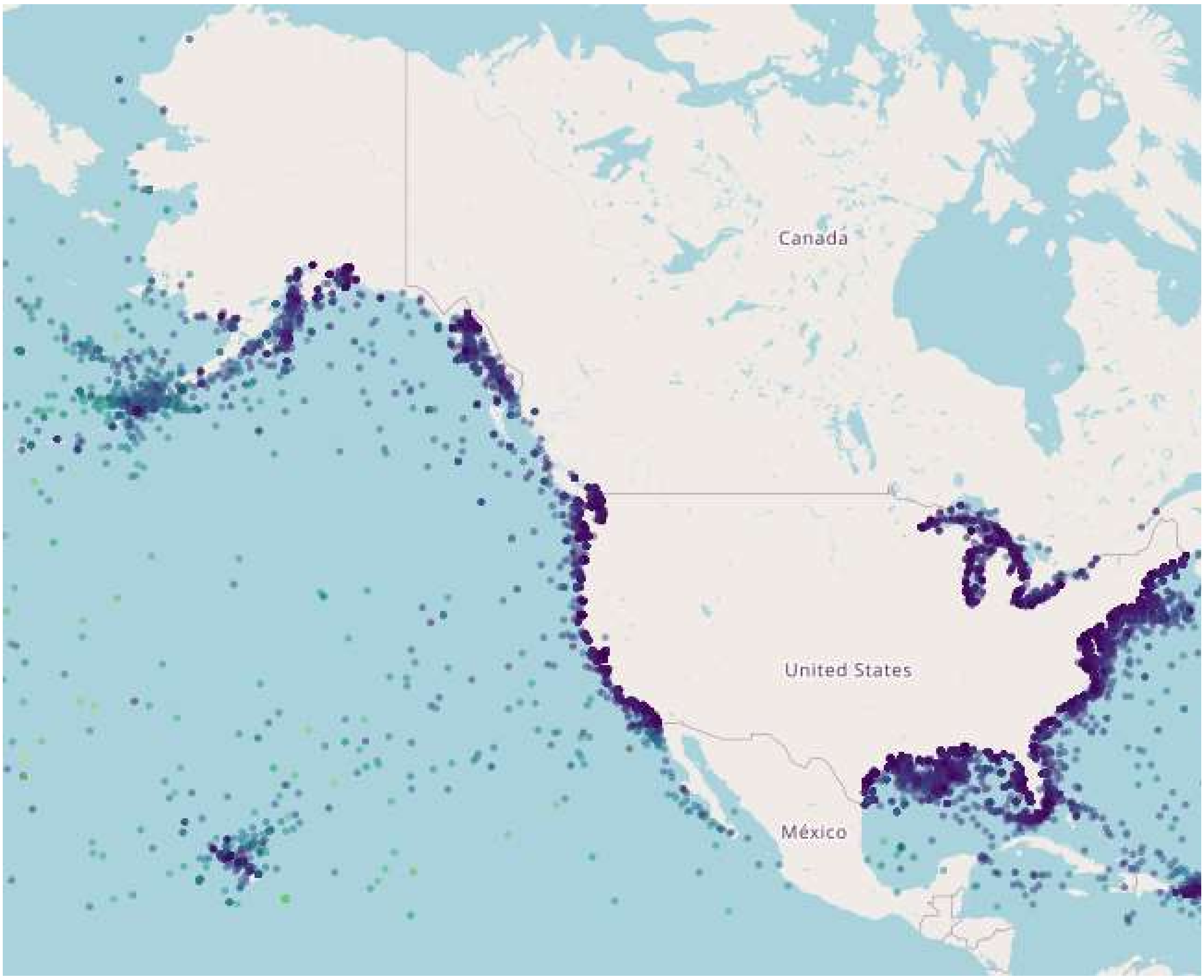
In [10]:

```
# Create figure object
fig = go.Figure()

# Aggregate WeatherOcean points
fig.add_trace(go.Scattermapbox(
    lat=WeatherOcean['latitude'],
    lon=WeatherOcean['longitude'],
    mode='markers',
    marker=dict(
        size=5,
        color=np.log1p(WeatherOcean['station_dist']),   # logarithmic scale
        colorscale=px.colors.sequential.Viridis,
        opacity=0.5,
    ),
    text=WeatherOcean.apply(lambda row:f"station:{row['station']}<br>station_dist: {row[
))

# 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