# OroraTech Challenge

Importing Packages

In [18]:
```
import netCDF4 as nc
import os

import numpy as np
from matplotlib import pyplot as plt
from mpl_toolkits.basemap import Basemap
import numpy as np
import cv2 as cv
import geopandas as gpd
import pandas as pd
import numpy.ma as ma
from shapely.geometry import Point
from keplergl import KeplerGl
```

Fetching the Data Set

In [19]:
```
folder_path = 'input_data'
radiance_filename = 'VNP02MOD_NRT.A2020233.1000.001.nc'
geolocation_filename = 'VNP03MOD_NRT.A2020233.1000.001.nc'

# Creating the File Path
radiance_filepath = os.path.join(folder_path, radiance_filename)
geolocation_filepath = os.path.join(folder_path, geolocation_filename)

# Fetching the data file and storing in a variable.
radiance_netcdf = nc.Dataset(radiance_filepath)
geolocation_netcdf = nc.Dataset(geolocation_filepath)
```

In [20]:
```
# Fetching the Variables from the respective group of the nc file.
m13_fire_radiance_variable = radiance_netcdf['/observation_data/M13']
latitude_variable = geolocation_netcdf['/geolocation_data/latitude']
longitude_variable = geolocation_netcdf['/geolocation_data/longitude']

# Fetching the units and the array of values.
m13_units = m13_fire_radiance_variable.units
m13_fire_radiance_array = m13_fire_radiance_variable[:]

# Fetching the array of values.
latitude_array = latitude_variable[:]
longitude_array = longitude_variable[:]
```

In [21]:
```
# Rescaling the fire_radiance Array to display on the screen
def rescaleFrame(frame, scale):
    width = int(frame.shape[1] * scale)
    height = int(frame.shape[0] * scale)
    dimensions = (width, height)

    return cv.resize(frame, dimensions, interpolation=cv.INTER_AREA)

m13_fire_radiance_array_rescaled = rescaleFrame(m13_fire_radiance_array, 0.25)

# Applying a binary threshold value.
ret, thresh1 = cv.threshold(m13_fire_radiance_array_rescaled, 0.85,1, cv.THRESH_BINARY)

# Plotting the Threshold image on the screen.
#cv.imshow('Threshold-fire_radiance',thresh1)

# Saving the image in output directory.
frame_normed = 255 * (thresh1 - thresh1.min()) / (thresh1.max() - thresh1.min())
frame_normed = np.array(frame_normed, np.int)
output_folder = 'output'
threshold_filename = 'binarythreshold_image.png'
threshold_filepath = os.path.join(output_folder, threshold_filename)
cv.imwrite(filename=threshold_filepath, img=frame_normed)
#cv.waitKey(0)
```

```
C:\TheFourthReich\anaconda3\envs\gisenv\lib\site-packages\ipykernel_launcher.py:19: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modif
y any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
```

Out[21]:
```
True
```

Steps for moving the data into DataFrame (& GeoDataFrame)

In [22]:
```
# Flattening the array and converting them to a list to insert them in a dataframe.
longitude_list = list(longitude_array.flatten())
latitude_list = list(latitude_array.flatten())
```

In [23]:
```
# Replacing the Masked value into NaN and converting the array into list for the dataframe.
m13_fire_radiance_list = list(ma.masked_values(m13_fire_radiance_array, np.nan).flatten())
```

In [24]:
```
# Creating a new dataframe, with the fire radiance and coordinates of the spots.
fire_radiance_geolocation_df = pd.DataFrame({'fire_radiance' : m13_fire_radiance_list, 'Longitude' : longitude_list, 'Latitude' : latitude_list})
fire_radiance_geolocation_df.head()
```

Out[24]:

| | fire_radiance | Longitude | Latitude |
|---|---|---|---|
| **0** | NaN | -139.746368 | 47.533764 |
| **1** | NaN | -139.724014 | 47.534237 |
| **2** | NaN | -139.701721 | 47.534702 |
| **3** | NaN | -139.679504 | 47.535152 |
| **4** | NaN | -139.657318 | 47.535599 |

In [25]:
```
# Dropping the entries which have no fire radiance values
fire_radiance_geolocation_df.dropna(inplace=True)
fire_radiance_geolocation_df.head()
```

Out[25]:

| | fire_radiance | Longitude | Latitude |
|---|---|---|---|
| **1008** | 0.365053 | -126.349487 | 47.032814 |
| **1009** | 0.356964 | -126.334282 | 47.031361 |
| **1010** | 0.365053 | -126.319077 | 47.029892 |
| **1011** | 0.381229 | -126.303902 | 47.028431 |
| **1012** | 0.365053 | -126.288765 | 47.026970 |

In [26]:
```
# Filtering the entries which have more than 0.85 Fire Radiance.
fire_radiance_geolocation_gt0_85_df = fire_radiance_geolocation_df.loc[fire_radiance_geolocation_df['fire_radiance'] >= 0.85]
fire_radiance_geolocation_gt0_85_df.head()
```

Out[26]:

| | fire_radiance | Longitude | Latitude |
|---|---|---|---|
| **612846** | 1.238586 | -119.583046 | 44.871090 |
| **616046** | 1.562117 | -119.591080 | 44.866467 |
| **616047** | 1.004026 | -119.581451 | 44.864914 |
| **882414** | 1.036379 | -111.593269 | 42.685574 |
| **891440** | 1.141526 | -117.921265 | 43.992130 |

In [27]:
```
# Adding a geometry column by converting the longitude and latitude for creating the geodataframe.
fire_radiance_geolocation_gt0_85_df['geometry'] = [Point(x,y) for x,y in zip(fire_radiance_geolocation_gt0_85_df['Longitude'], fire_radiance_geolocation_gt0_85_df['Latitude'])]
fire_radiance_geolocation_gt0_85_df.head()
```

```
C:\TheFourthReich\anaconda3\envs\gisenv\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[27]:

| | fire_radiance | Longitude | Latitude | geometry |
|---|---|---|---|---|
| **612846** | 1.238586 | -119.583046 | 44.871090 | POINT (-119.58304595594727 44.87108993530273) |
| **616046** | 1.562117 | -119.591080 | 44.866467 | POINT (-119.59107971191414 44.866465222168) |
| **616047** | 1.004026 | -119.581451 | 44.864914 | POINT (-119.58145141160156 44.864913940429969) |
| **882414** | 1.036379 | -111.593269 | 42.685574 | POINT (-111.59326934814445 42.685573577880086) |
| **891440** | 1.141526 | -117.921265 | 43.992130 | POINT (-117.92126464684375 43.99213027954102) |

In [28]:
```
# Creating a geodataframe.
fire_radiance_geolocation_gdf = gpd.GeoDataFrame(data=fire_radiance_geolocation_gt0_85_df, geometry=fire_radiance_geolocation_gt0_85_df['geometry'])
fire_radiance_geolocation_gdf.head()
```

Out[28]:

| | fire_radiance | Longitude | Latitude | geometry |
|---|---|---|---|---|
| **612846** | 1.238586 | -119.583046 | 44.871090 | POINT (-119.58305 44.87109) |
| **616046** | 1.562117 | -119.591080 | 44.866467 | POINT (-119.59108 44.86647) |
| **616047** | 1.004026 | -119.581451 | 44.864914 | POINT (-119.58145 44.86491) |
| **882414** | 1.036379 | -111.593269 | 42.685574 | POINT (-111.59327 42.68557) |
| **891440** | 1.141526 | -117.921265 | 43.992130 | POINT (-117.92126 43.99213) |

In [29]:
```
# Setting an Coordinate reference system of EPSG:4326 to the geodataframe.
fire_radiance_geolocation_gdf.set_crs(epsg=4326, inplace=True)
```

Out[29]:

| | fire_radiance | Longitude | Latitude | geometry |
|---|---|---|---|---|
| **612846** | 1.238586 | -119.583046 | 44.871090 | POINT (-119.58305 44.87109) |
| **616046** | 1.562117 | -119.591080 | 44.866467 | POINT (-119.59108 44.86647) |
| **616047** | 1.004026 | -119.581451 | 44.864914 | POINT (-119.58145 44.86491) |
| **882414** | 1.036379 | -111.593269 | 42.685574 | POINT (-111.59327 42.68557) |
| **891440** | 1.141526 | -117.921265 | 43.992130 | POINT (-117.92126 43.99213) |
| **...** | ... | ... | ... | ... |
| **6066441** | 0.858437 | -115.831406 | 32.429142 | POINT (-115.83141 32.42914) |
| **6114444** | 0.850349 | -115.837440 | 32.329426 | POINT (-115.83744 32.32943) |
| **6152845** | 0.850349 | -115.855011 | 32.261822 | POINT (-115.85501 32.26182) |
| **6152846** | 0.850349 | -115.843140 | 32.259171 | POINT (-115.84314 32.25917) |
| **6152847** | 0.850349 | -115.831268 | 32.256527 | POINT (-115.83127 32.25653) |

3501 rows × 4 columns

## Plotting the Output in different formats.

Plotting the image on the KeplerGL package.

In [30]:
```
fire_radiance_map = KeplerGl(height=600, width=800)
fire_radiance_map.add_data(data=fire_radiance_geolocation_gdf.copy(), name = 'area_of_interest')
fire_radiance_filename = 'hotspots_locations.html'
fire_radiance_filepath = os.path.join(output_folder, fire_radiance_filename)
fire_radiance_map.save_to_html(file_name=fire_radiance_filepath)
```

```
User Guide: https://docs.kepler.gl/docs/keplergl-jupyter
Map saved to output\hotspots_locations.html!
```

Hotspots Points

Hotspots Heatmap based on Radiance Value

Hotspots Heatmap based on Density of the Hotspot Points.

Exporting the hotspot locations into geojson format.

In [31]:
```
geojson_filename = 'hotspots_locations.geojson'
geojson_filepath = os.path.join(output_folder, geojson_filename)
fire_radiance_geolocation_gdf.to_file(geojson_filepath, driver='GeoJSON')
```

Exporting the hotspot locations into the csv format.

In [32]:
```
csv_filename = 'hotspots_locations.csv'
csv_filepath = os.path.join(output_folder, csv_filename)
fire_radiance_geolocation_gdf.to_csv(csv_filepath, index=None)
```

In [ ]:
```
```