

File Edit View Run Kernel Tabs Settings Help

Launcher x lab\_jupyter\_launch\_site\_io +

Python (Pyodide) ⚙

Filter files by name

/ DS0203EN / module\_3 /

Name	Modified
lab_jupyter_laun...	16m ago
● lab_jupyter_laun...	42s ago
spacex_launch_...	9m ago

 Skills Network

## Launch Sites Locations Analysis with Folium

Estimated time needed: 40 minutes

The launch success rate may depend on many factors such as payload mass, orbit type, and so on. It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories. Finding an optimal location for building a launch site certainly involves many factors and hopefully we could discover some of the factors by analyzing the existing launch site locations.

In the previous exploratory data analysis labs, you have visualized the SpaceX launch dataset using `matplotlib` and `seaborn` and discovered some preliminary correlations between the launch site and success rates. In this lab, you will be performing more interactive visual analytics using `Folium`.

### Objectives

This lab contains the following tasks:

- **TASK 1:** Mark all launch sites on a map
- **TASK 2:** Mark the success/failed launches for each site on the map
- **TASK 3:** Calculate the distances between a launch site to its proximities

After completed the above tasks, you should be able to find some geographical patterns about launch sites.

Let's first import required Python packages for this lab:

```
[14]: import pipelite
await pipelite.install(['folium'])
await pipelite.install(['pandas'])
await pipelite.install(['requests'])

[15]: import folium
import pandas as pd
import requests

[16]: # Import folium MarkerCluster plugin
from folium.plugins import MarkerCluster
# Import folium MousePosition plugin
from folium.plugins import MousePosition
# Import folium DivIcon plugin
from folium.features import DivIcon
```

If you need to refresh your memory about folium, you may download and refer to this previous folium lab:

Generating Maps with Python

```
[17]: ## Task 1: Mark all launch sites on a map
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv'
response = requests.get(url)
with open('spacex_launch_geo.csv', 'wb') as file:
    file.write(response.content)
spacex_df = pd.read_csv('spacex_launch_geo.csv')
```

First, let's try to add each site's location on a map using site's latitude and longitude coordinates

The following dataset with the name `spacex_launch_geo.csv` is an augmented dataset with latitude and longitude added for each site.

```
[18]: # Download and read the 'spacex_launch_geo.csv'
from js import fetch
import io

URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv'
resp = await fetch(URL)
spacex_csv_file = io.BytesIO(await resp.arrayBuffer()).to_py()
spacex_df=pd.read_csv(spacex_csv_file)
```

Now, you can take a look at what are the coordinates for each site.

```
[19]: # Select relevant sub-columns: 'Launch Site', 'Lat(Latitude)', 'Long(Longitude)', 'class'
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

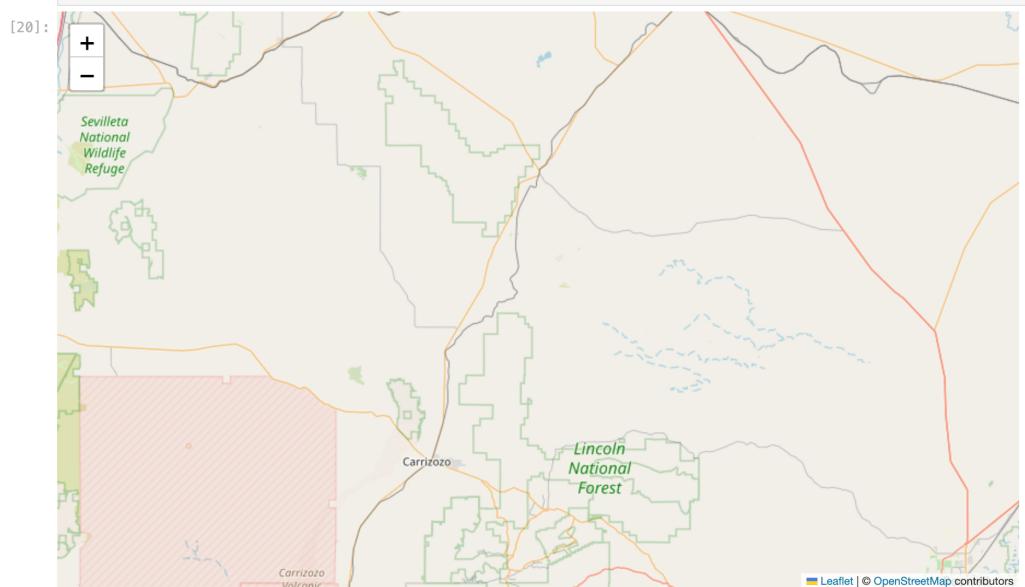
Above coordinates are just plain numbers that can not give you any intuitive insights about where are those launch sites. If you are very good at geography, you can interpret those numbers directly in your mind. If not, that's fine too. Let's visualize those locations by pinning them on a map.

We first need to create a folium `Map` object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.

```
[7]: # Start location is NASA Johnson Space Center
nasa_coordinate = [29.55964888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=10)
```

We could use `folium.Circle` to add a highlighted circle area with a text label on a specific coordinate. For example,

```
[20]: # Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color="#d35400", fill=True).add_child(folium.Popup('NASA John
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
    )
)
site_map.add_child(circle)
site_map.add_child(marker)
```



and you should find a small yellow circle near the city of Houston and you can zoom-in to see a larger circle.

Now, let's add a circle for each launch site in data frame `launch_sites`

*TODO: Create and add `folium.Circle` and `folium.Marker` for each launch site on the site map*

An example of `folium.Circle`:

```
folium.Circle(coordinate, radius=1000, color='#000000', fill=True).add_child(folium.Popup(...))
```

An example of `folium.Marker`:

```
folium.map.Marker(coordinate, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), html='<div
style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'label', ))
```

```
[21]: # Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch si
for launch_site, site_lat, site_long in zip(launch_sites_df['Launch Site'], launch_sites_df['Lat'], launch_sites_d
    site_coordinate = [site_lat, site_long]

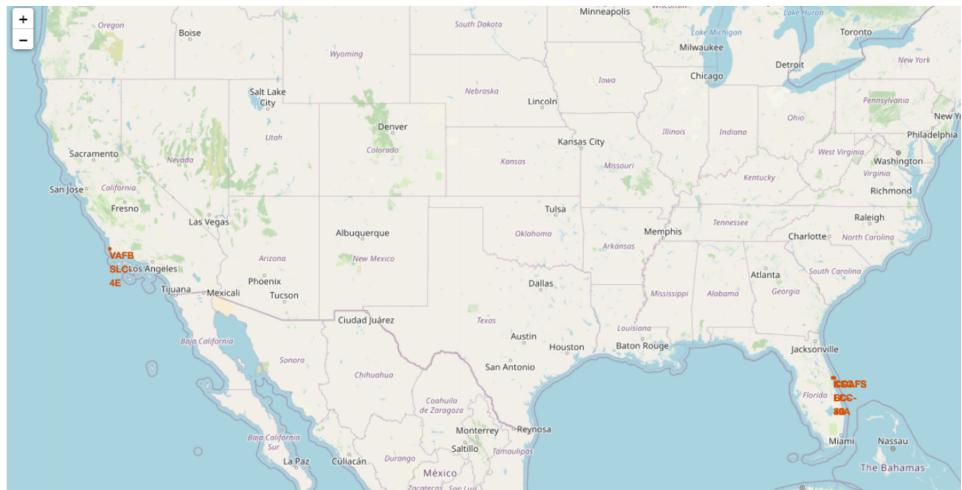
    circle = folium.Circle(site_coordinate, radius=1000, color="#d35400", fill=True).add_child(folium.Popup(launch

    marker = folium.map.Marker(
        site_coordinate,
        # Create an icon as a text label
        icon=DivIcon(
            icon_size=(20, 20),
            icon_anchor=(0, 0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % launch_site,
        )
    )
    site_map.add_child(circle)
    site_map.add_child(marker)
site_map
```





The generated map with marked launch sites should look similar to the following:



Now, you can explore the map by zoom-in/out the marked areas , and try to answer the following questions:

- Are all launch sites in proximity to the Equator line?
- Are all launch sites in very close proximity to the coast?

Also please try to explain your findings.

```
[ ]: # Task 2: Mark the success/failed launches for each site on the map
```

Next, let's try to enhance the map by adding the launch outcomes for each site, and see which sites have high success rates. Recall that data frame `spacex_df` has detailed launch records, and the `class` column indicates if this launch was successful or not

```
[ ]: spacex_df.tail(10)
```

Next, let's create markers for all launch records. If a launch was successful (`class=1`) , then we use a green marker and if a launch was failed, we use a red marker (`class=0`)

Note that a launch only happens in one of the four launch sites, which means many launch records will have the exact same coordinate. Marker clusters can be a good way to simplify a map containing many markers having the same coordinate.

Let's first create a `MarkerCluster` object

```
[22]: marker_cluster = MarkerCluster()
```

*TODO: Create a new column in `spacex_df` dataframe called `marker_color` to store the marker colors based on the `class` value*

```
[23]: # Apply a function to check the value of 'class' column
# If class=1, marker_color value will be green
# If class=0, marker_color value will be red
spacex_df['marker_color'] = list(map(lambda x: 'green' if x==1 else 'red', spacex_df['class']))
spacex_df.tail(10)
```

	Launch Site	Lat	Long	class	marker_color
46	KSC LC-39A	28.573255	-80.646895	1	green
47	KSC LC-39A	28.573255	-80.646895	1	green
48	KSC LC-39A	28.573255	-80.646895	1	green
49	CCAFS SLC-40	28.563197	-80.576820	1	green
50	CCAFS SLC-40	28.563197	-80.576820	1	green

```

51 CCAFS SLC-40 28.563197 -80.576820 0 red
52 CCAFS SLC-40 28.563197 -80.576820 0 red
53 CCAFS SLC-40 28.563197 -80.576820 0 red
54 CCAFS SLC-40 28.563197 -80.576820 1 green
55 CCAFS SLC-40 28.563197 -80.576820 0 red

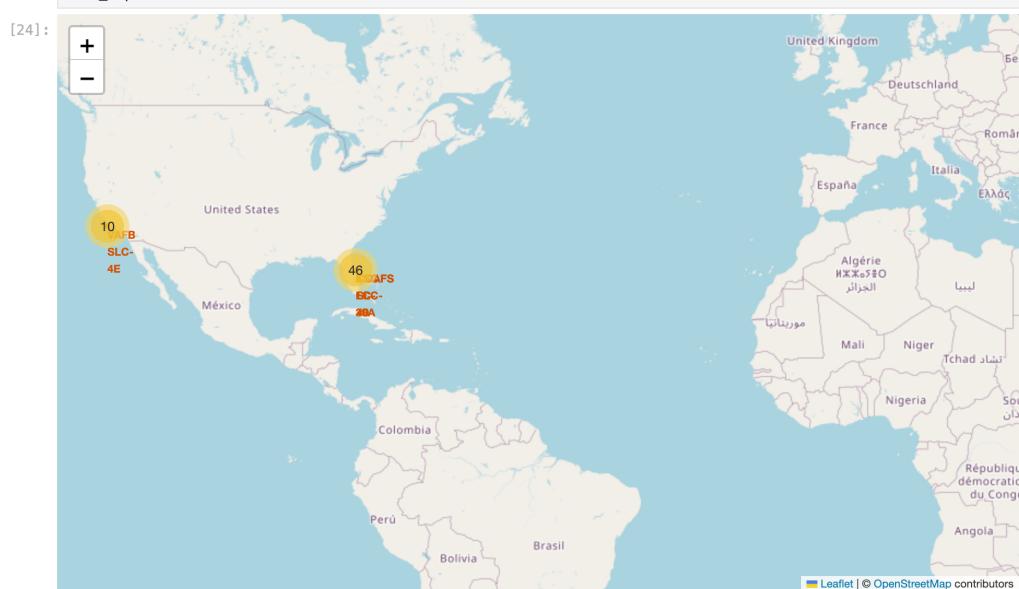
```

*TODO:* For each launch result in `spacex_df` data frame, add a `folium.Marker` to `marker_cluster`

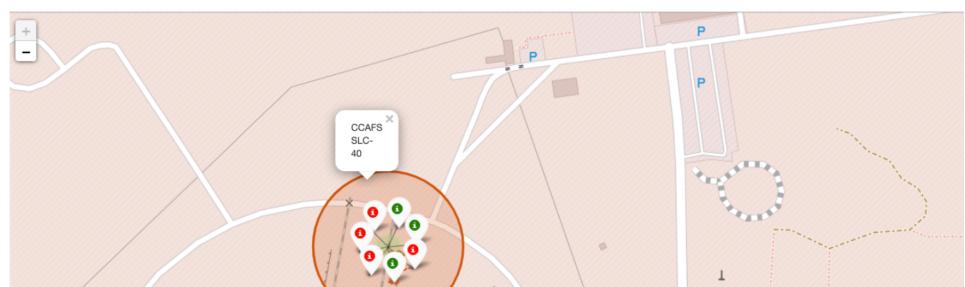
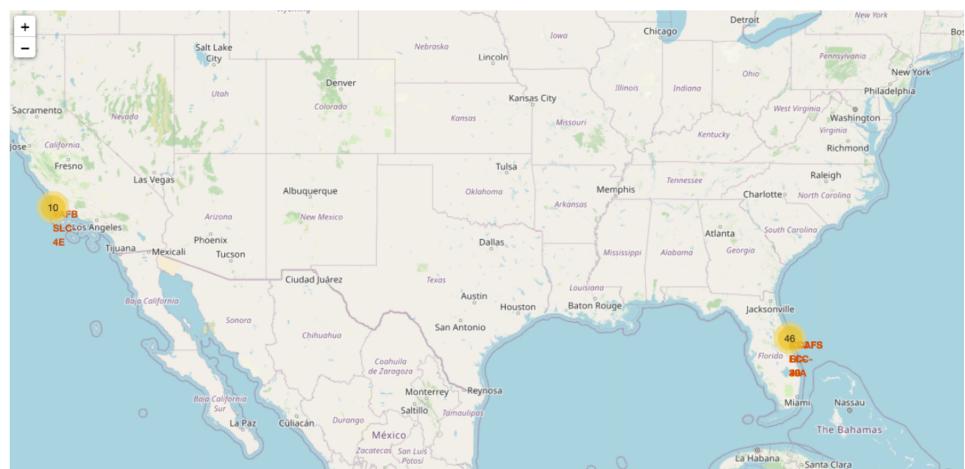
```
[24]: # Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color'])
for site_lat, site_long, marker_color in zip(spacex_df['Lat'], spacex_df['Long'], spacex_df['marker_color']):
    site_coordinate = [site_lat, site_long]
    marker = folium.map.Marker(
        site_coordinate,
        # Create an icon as a text label
        icon=folium.Icon(color='white',
                          icon_color=marker_color)
    )
    marker.add_to(marker_cluster)

site_map
```



Your updated map may look like the following screenshots:





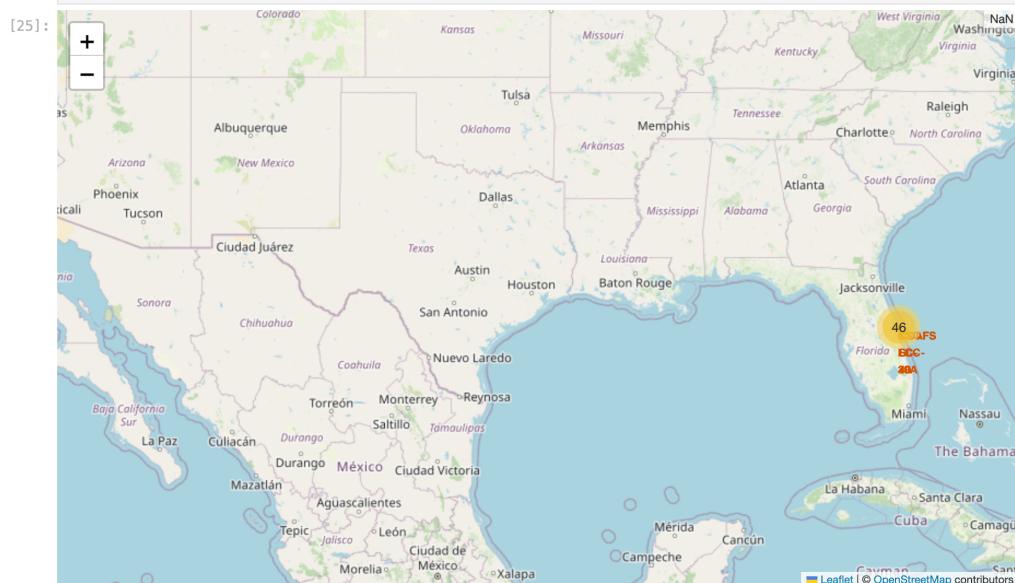
From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.

```
[ ]: # TASK 3: Calculate the distances between a launch site to its proximities
```

Next, we need to explore and analyze the proximities of launch sites.

Let's first add a `MousePosition` on the map to get coordinate for a mouse over a point on the map. As such, while you are exploring the map, you can easily find the coordinates of any points of interests (such as railway)

```
[25]: # Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on the map
formatter = "function(num) {return L.Util.formatNum(num, 5);};""
mouse_position = MousePosition(
    position='topright',
    separator=' Long: ',
    empty_string='NaN',
    lng_first=False,
    num_digits=20,
    prefix='Lat:',
    lat_formatter=formatter,
    lng_formatter=formatter,
)
site_map.add_child(mouse_position)
site_map
```



Now zoom in to a launch site and explore its proximity to see if you can easily find any railway, highway, coastline, etc. Move your mouse to these points and mark down their coordinates (shown on the top-left) in order to the distance to the launch site.

Now zoom in to a launch site and explore its proximity to see if you can easily find any railway, highway, coastline, etc. Move your mouse to these points and mark down their coordinates (shown on the top-left) in order to the distance to the launch site.

```
[26]: from math import sin, cos, sqrt, atan2, radians
def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```

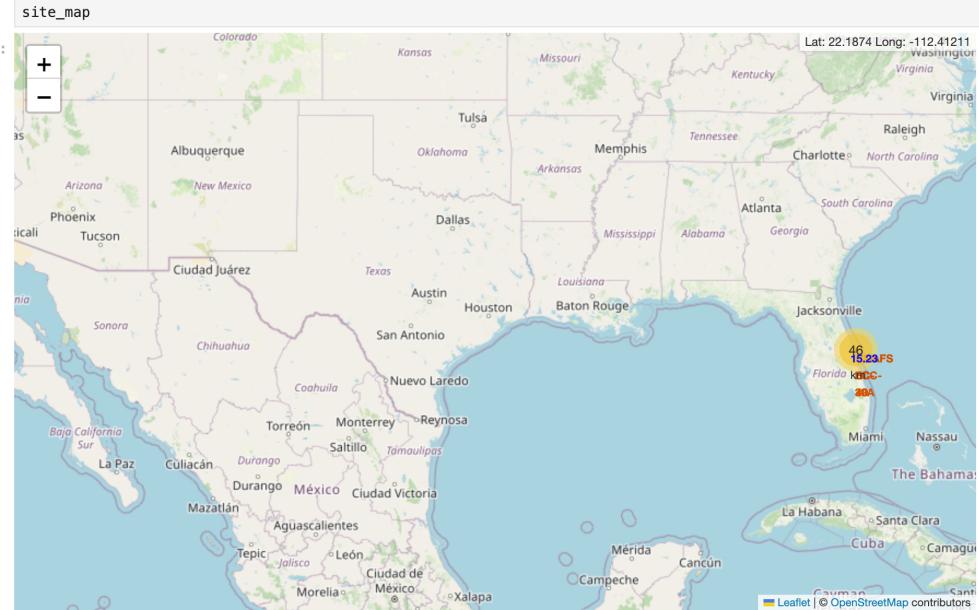
*TODO:* Mark down a point on the closest coastline using `MousePosition` and calculate the distance between the coastline point and the

launch site.

```
[27]: # find coordinate of the closest coastline
railway_marker = [28.55752, -80.80155]
launch_coordinate = [28.57337, -80.64669]
distance_railway = calculate_distance(railway_marker[0], railway_marker[1], launch_coordinate[0], launch_coordinate[1])
distance_railway # distance in km
```

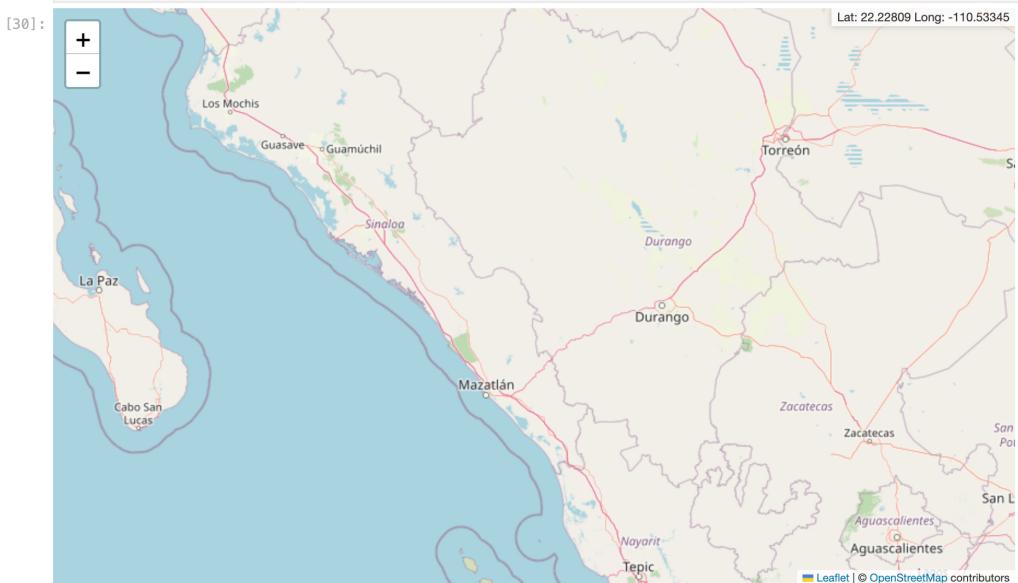
```
[27]: 15.230651245141605
```

```
[28]: # Create and add a folium.Marker on your selected closest coastline point on the map
# Display the distance between coastline point and launch site using the icon property
marker = folium.map.Marker(
    railway_marker,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(400, 400),
        icon_anchor=(0, 0),
        html=<div style="font-size:400; color:#0c10f2;"><b>%s</b></div>' % str(round(distance_railway, 2))+''
    )
)
marker.add_to(site_map)
```

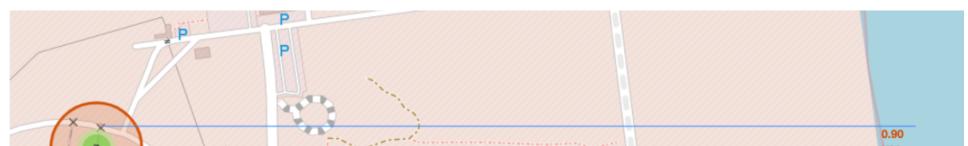


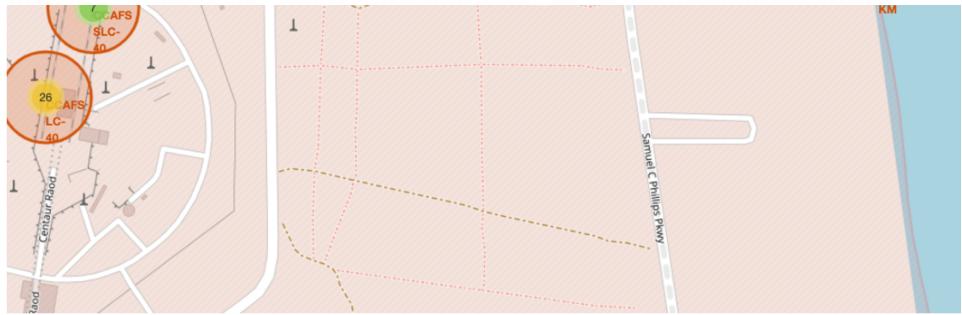
TODO: Draw a PolyLine between a launch site to the selected coastline point

```
[30]: # Create a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
folium.PolyLine([railway_marker, launch_coordinate], color='blue').add_to(site_map)
```



Your updated map with distance line should look like the following screenshot:



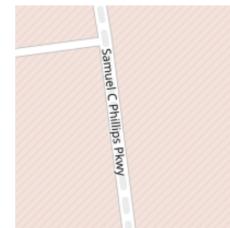


*TODO:* Similarly, you can draw a line between a launch site to its closest city, railway, highway, etc. You need to use `MousePosition` to find their coordinates on the map first

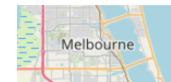
A railway map symbol may look like this:



A highway map symbol may look like this:



A city map symbol may look like this:



```
[ ]: # Create a marker with distance to a closest city, railway, highway, etc.
# Draw a line between the marker to the launch site
```

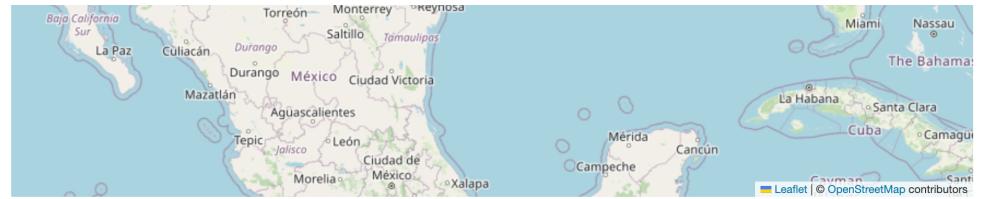
```
[31]: city      = [28.61200, -80.80788]
coastline  = [28.5858, -80.79952]
highway    = [28.5402, -80.85079]

city_distance = calculate_distance(city[0], city[1], launch_coordinate[0], launch_coordinate[1])
coastline_distance = calculate_distance(coastline[0], coastline[1], launch_coordinate[0], launch_coordinate[1])
highway_distance = calculate_distance(highway[0], highway[1], launch_coordinate[0], launch_coordinate[1])

colors = ['red','orange','green']
html_colors = ['#dc3545','#fd7e14','#198754']

for coordinate ,distance, color, html_color in zip([city, coastline, highway], [city_distance, coastline_distance, highway_distance], colors, html_colors):
    marker = folium.map.Marker(
        coordinate,
        # Create an icon as a text label
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html=<div style="font-size: 12; color:' +html_color+';"><b>%s</b></div>' % str(round(distance, 2))
        )
    )
    marker.add_to(site_map)
    folium.PolyLine([coordinate, launch_coordinate], color=color).add_to(site_map)
site_map
```





[ ]:



After you plot distance lines to the proximities, you can answer the following questions easily:

- Are launch sites in close proximity to railways?
- Are launch sites in close proximity to highways?
- Are launch sites in close proximity to coastline?
- Do launch sites keep certain distance away from cities?

Also please try to explain your findings.

## Next Steps:

Now you have discovered many interesting insights related to the launch sites' location using folium, in a very interactive way. Next, you will need to build a dashboard using Plotly Dash on detailed launch records.

## Authors

[Pratiksha Verma](#)

IBM Corporation 2022. All rights reserved.