Hands-on Lab: Interactive Visual Analytics 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:

[2]:
```
import piplite
await piplite.install(['folium'])
#await piplite.install(['pandas'])
```

[3]:
```
import folium
import pandas as pd
```

[4]:
```
# 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

Task 1: Mark all launch sites on a map

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.

[5]:
```
spacex_df = pd.read_csv('spacex_launch_geo.csv')
spacex_df.head()
```

[5]:

	Flight Number	Date	Time (UTC)	Booster Version	Launch Site	Payload	Payload Mass (kg)	Orbit	Customer	Landing Outcome	class	Lat	Long
0	1	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Failure (parachute)	0	28.562302	-80.577356
1	2	2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel o...	0.0	LEO (ISS)	NASA (COTS) NRO	Failure (parachute)	0	28.562302	-80.577356
2	3	2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2+	525.0	LEO (ISS)	NASA (COTS)	No attempt	0	28.562302	-80.577356
3	4	2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	No attempt	0	28.562302	-80.577356
4	5	2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	No attempt	0	28.562302	-80.577356

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

[6]:
```
# 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
```

[6]:

	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.

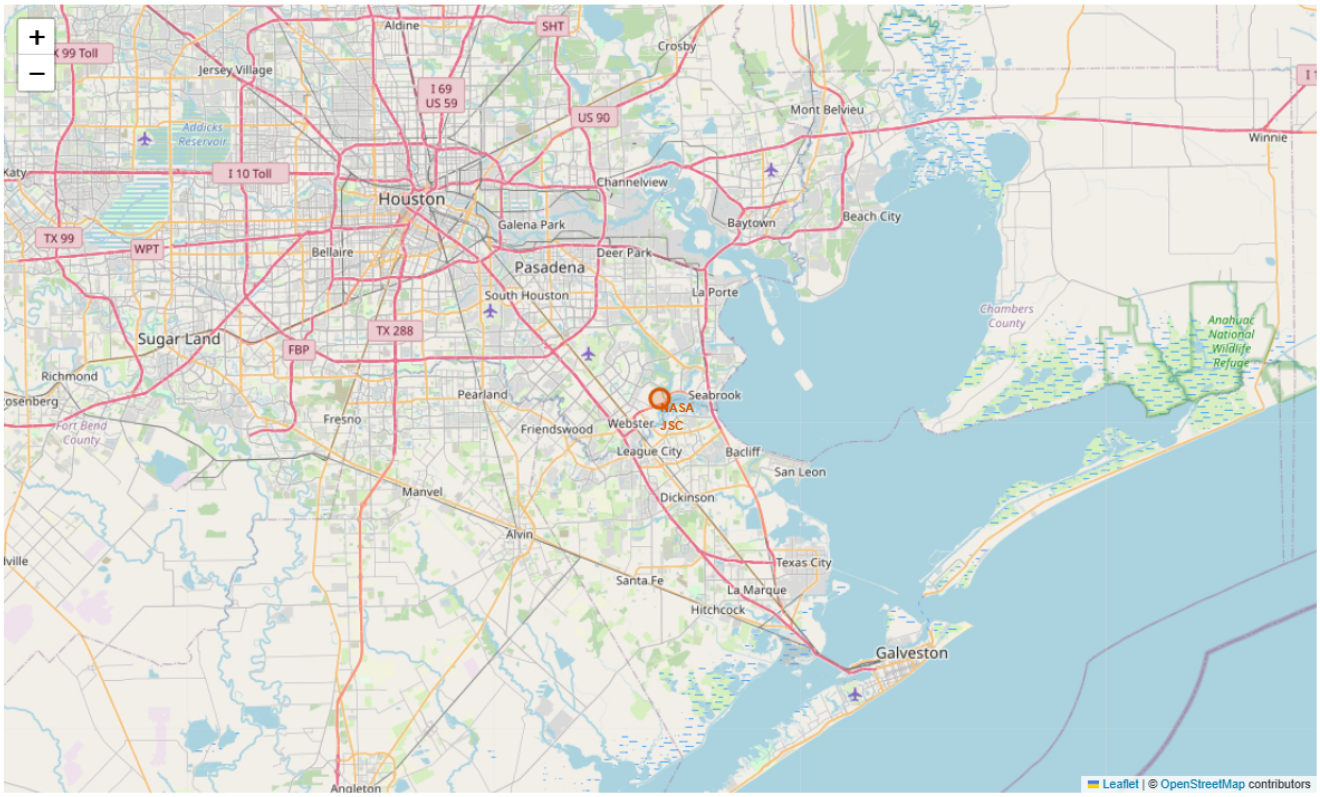
We first need to create a folium `Map` object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.

[7]:
```
# Start location is NASA Johnson Space Center
nasa_coordinate = [29.559684888503615, -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,

[8]:
```
# 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 Johnson Space Center'))
# 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)
```

[8]:



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(...))`

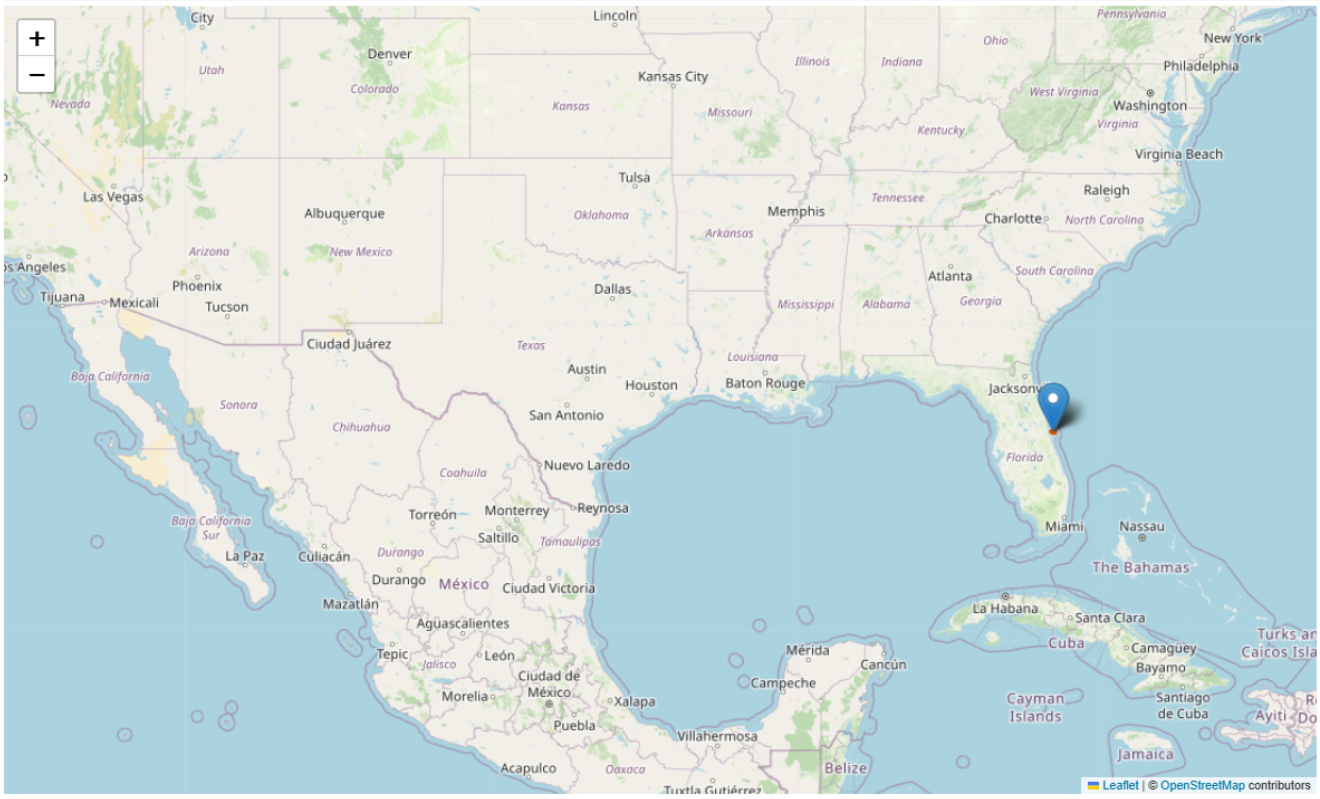
An example of folium.Marker:

`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', ))
```

[9]:
```
# Define a map centered at a specific location
nasa_coordinate = [29.559684888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)

# List of locations and their details
locations = [
    {"coords": [28.562302, -80.577356], "radius": 2200, "color": "#d35400", "name": "CCAFS LC-40"},
    {"coords": [28.563197, -80.576820], "radius": 2200, "color": "#d35400", "name": "CCAFS SLC-40"},
    {"coords": [28.573255, -80.646895], "radius": 2200, "color": "#d35400", "name": "KSC LC-39A"},
    {"coords": [34.632834, -120.610745], "radius": 2200, "color": "#d35400", "name": "VAFB SLC-4E"}
]

# Add circles and tooltips for each location
for loc in locations:
    # Add circle
    folium.Circle(
        location=loc["coords"],       # Circle's center
        radius=loc["radius"],         # Radius in meters
        color=loc["color"],           # Circle outline color
        fill=True,                    # Fill the circle
        fill_color=loc["color"],      # Fill color matches the outline
    ).add_to(site_map)

    # Add tooltip for the location name
    folium.Marker(
        location=loc["coords"],
        tooltip=loc["name"]           # Tooltip that shows location name
    ).add_to(site_map)

# Display the map
site_map
```

[9]:



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.

Explanation

The sites are in close proximities to the equator line and the coast as well.

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

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

[10]:

	Launch Site	Lat	Long	class
46	KSC LC-39A	28.573255	-80.646895	1
47	KSC LC-39A	28.573255	-80.646895	1
48	KSC LC-39A	28.573255	-80.646895	1
49	CCAFS SLC-40	28.563197	-80.576820	1
50	CCAFS SLC-40	28.563197	-80.576820	1
51	CCAFS SLC-40	28.563197	-80.576820	0
52	CCAFS SLC-40	28.563197	-80.576820	0
53	CCAFS SLC-40	28.563197	-80.576820	0
54	CCAFS SLC-40	28.563197	-80.576820	1
55	CCAFS SLC-40	28.563197	-80.576820	0

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

[11]:
```
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

[12]:
```
# 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

def assign_marker_color(launch_class):
    return "Green" if launch_class == 1 else "red"

# Apply the function to create the marker_color column
spacex_df["marker_color"] = spacex_df["class"].apply(assign_marker_color)

# Display the updated dataframe
print(spacex_df)
```

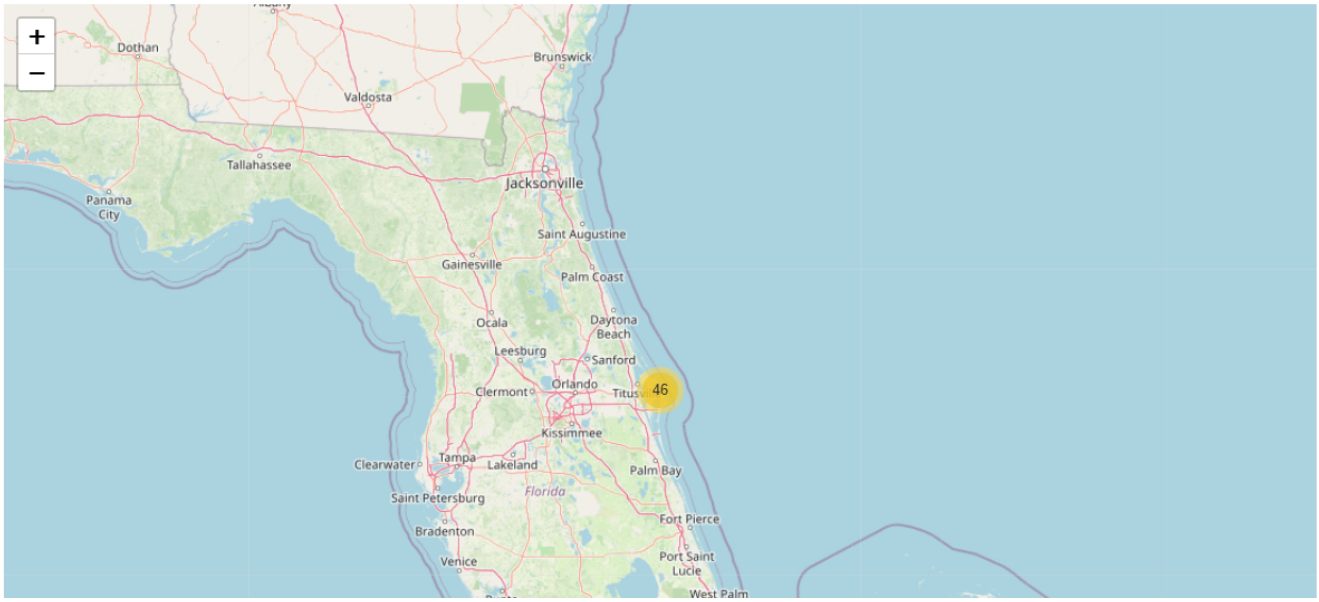
```
     Launch Site        Lat        Long  class marker_color
0    CCAFS LC-40  28.562302  -80.577356      0          red
1    CCAFS LC-40  28.562302  -80.577356      0          red
2    CCAFS LC-40  28.562302  -80.577356      0          red
3    CCAFS LC-40  28.562302  -80.577356      0          red
4    CCAFS LC-40  28.562302  -80.577356      0          red
5    CCAFS LC-40  28.562302  -80.577356      0          red
6    CCAFS LC-40  28.562302  -80.577356      0          red
7    CCAFS LC-40  28.562302  -80.577356      0          red
8    CCAFS LC-40  28.562302  -80.577356      0          red
9    CCAFS LC-40  28.562302  -80.577356      0          red
10   CCAFS LC-40  28.562302  -80.577356      0          red
11   CCAFS LC-40  28.562302  -80.577356      0          red
12   CCAFS LC-40  28.562302  -80.577356      0          red
13   CCAFS LC-40  28.562302  -80.577356      0          red
14   CCAFS LC-40  28.562302  -80.577356      0          red
```

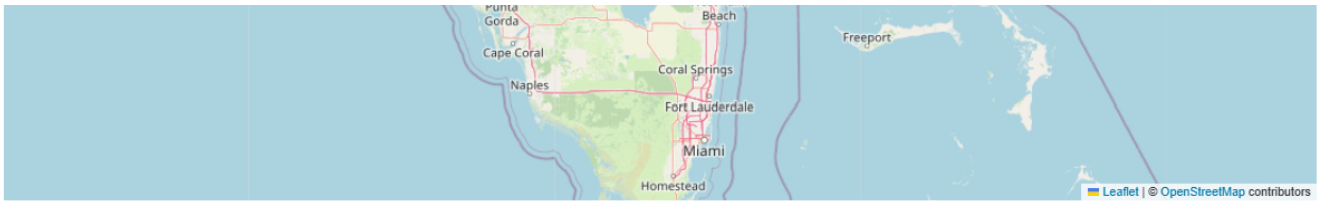
15	CCAFS LC-40	28.562302	-80.577356	0	red
16	CCAFS LC-40	28.562302	-80.577356	0	red
17	CCAFS LC-40	28.562302	-80.577356	1	Green
18	CCAFS LC-40	28.562302	-80.577356	1	Green
19	CCAFS LC-40	28.562302	-80.577356	0	red
20	CCAFS LC-40	28.562302	-80.577356	1	Green
21	CCAFS LC-40	28.562302	-80.577356	1	Green
22	CCAFS LC-40	28.562302	-80.577356	1	Green
23	CCAFS LC-40	28.562302	-80.577356	0	red
24	CCAFS LC-40	28.562302	-80.577356	1	Green
25	CCAFS LC-40	28.562302	-80.577356	1	Green
26	VAFB SLC-4E	34.632834	-120.610745	0	red
27	VAFB SLC-4E	34.632834	-120.610745	0	red
28	VAFB SLC-4E	34.632834	-120.610745	1	Green
29	VAFB SLC-4E	34.632834	-120.610745	1	Green
30	VAFB SLC-4E	34.632834	-120.610745	1	Green
31	VAFB SLC-4E	34.632834	-120.610745	1	Green
32	VAFB SLC-4E	34.632834	-120.610745	0	red
33	VAFB SLC-4E	34.632834	-120.610745	0	red
34	VAFB SLC-4E	34.632834	-120.610745	0	red
35	VAFB SLC-4E	34.632834	-120.610745	0	red
36	KSC LC-39A	28.573255	-80.646895	1	Green
37	KSC LC-39A	28.573255	-80.646895	0	red
38	KSC LC-39A	28.573255	-80.646895	1	Green
39	KSC LC-39A	28.573255	-80.646895	1	Green
40	KSC LC-39A	28.573255	-80.646895	0	red
41	KSC LC-39A	28.573255	-80.646895	1	Green
42	KSC LC-39A	28.573255	-80.646895	1	Green
43	KSC LC-39A	28.573255	-80.646895	0	red
44	KSC LC-39A	28.573255	-80.646895	1	Green
45	KSC LC-39A	28.573255	-80.646895	1	Green
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`

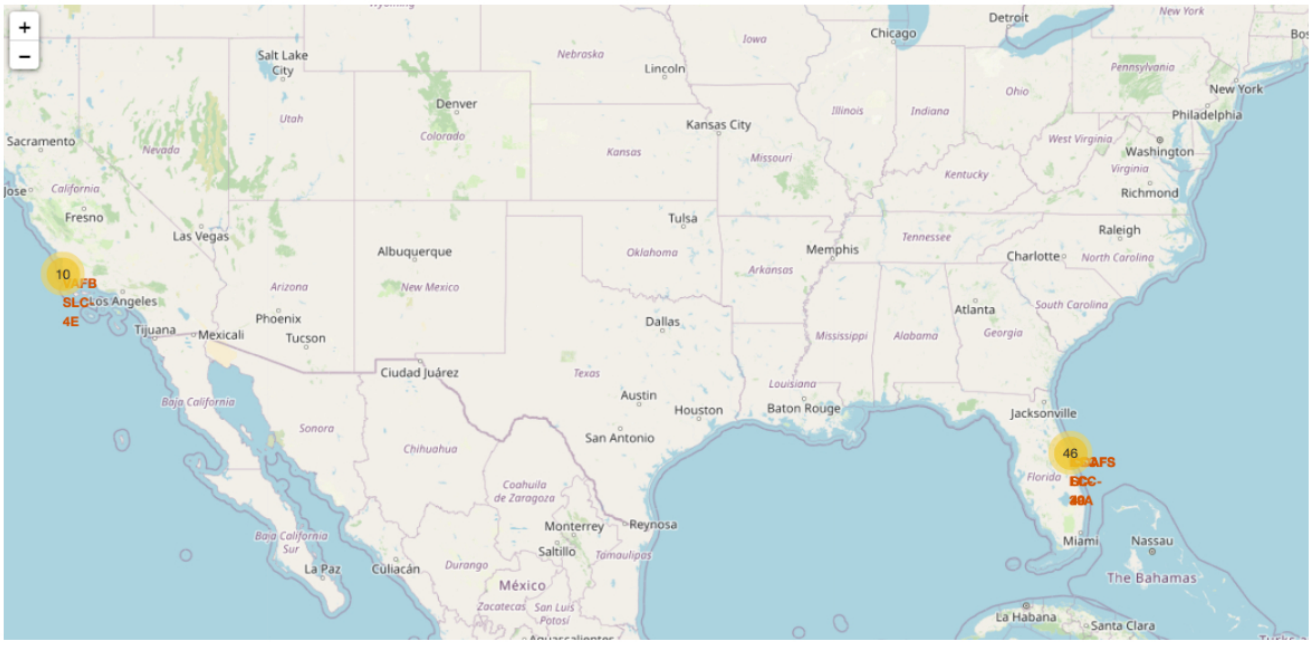
[13]:
```
# Define a function to assign marker colors
def assign_marker_color(launch_class):
    return "green" if launch_class == 1 else "red"  # Folium expects lowercase color names

# Apply the function to create the marker_color column
spacex_df["marker_color"] = spacex_df["class"].apply(assign_marker_color)

# Initialize a folium map centered at an approximate location
site_map = folium.Map(location=[28.5, -80.6], zoom_start=7)

# Create a MarkerCluster object
marker_cluster = MarkerCluster().add_to(site_map)

# Iterate through the dataframe and add markers to the cluster
for _, row in spacex_df.iterrows():
    folium.Marker(
        location=[row["Lat"], row["Long"]],
        icon=folium.Icon(color=row["marker_color"]),  # Use 'color', which is compatible with folium
        tooltip=f"Site: {row['Launch Site']}<br>Class: {row['class']}"
    ).add_to(marker_cluster)

# Display the map
site_map
```

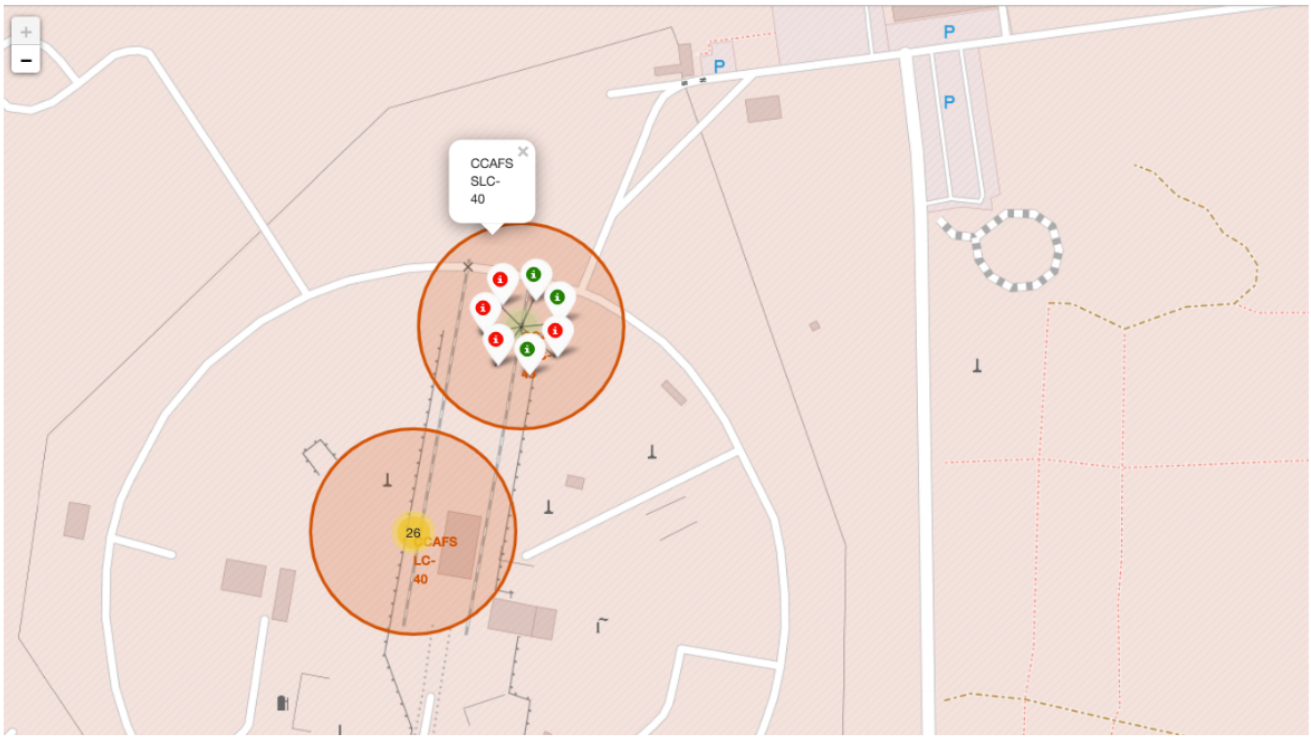
[13]:





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.

[14]:
```
# 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)

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.

TODO: Mark down a point on the closest coastline using MousePosition and calculate the distance between the coastline point and the launch site.

Distance of Launch Site CCAFS SLC-40 FROM THE COAST

[16]:
```
# find coordinate of the closet coastline
# e.g.,: Lat: 28.56367  Lon: -80.57163
# distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
```

[17]:
```
from math import sin, cos, sqrt, atan2, radians

# Function to calculate the distance using the Haversine formula
def calculate_distance(lat1, lon1, lat2, lon2):
    # Approximate radius of Earth in kilometers
    R = 6373.0

    # Convert degrees to radians
    lat1, lon1, lat2, lon2 = map(radians, [lat1, lon1, lat2, lon2])

    # Differences in coordinates
    dlon = lon2 - lon1
    dlat = lat2 - lat1

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

    # Distance in kilometers
    distance = R * c
    return distance

# Launch site coordinates
lat1, lon1 = 28.563197, -80.576820

# Coastline coordinates
lat2, lon2 = 28.56354, -80.56797

# Calculate the distance
distance_km = calculate_distance(lat1, lon1, lat2, lon2)
distance_km
```

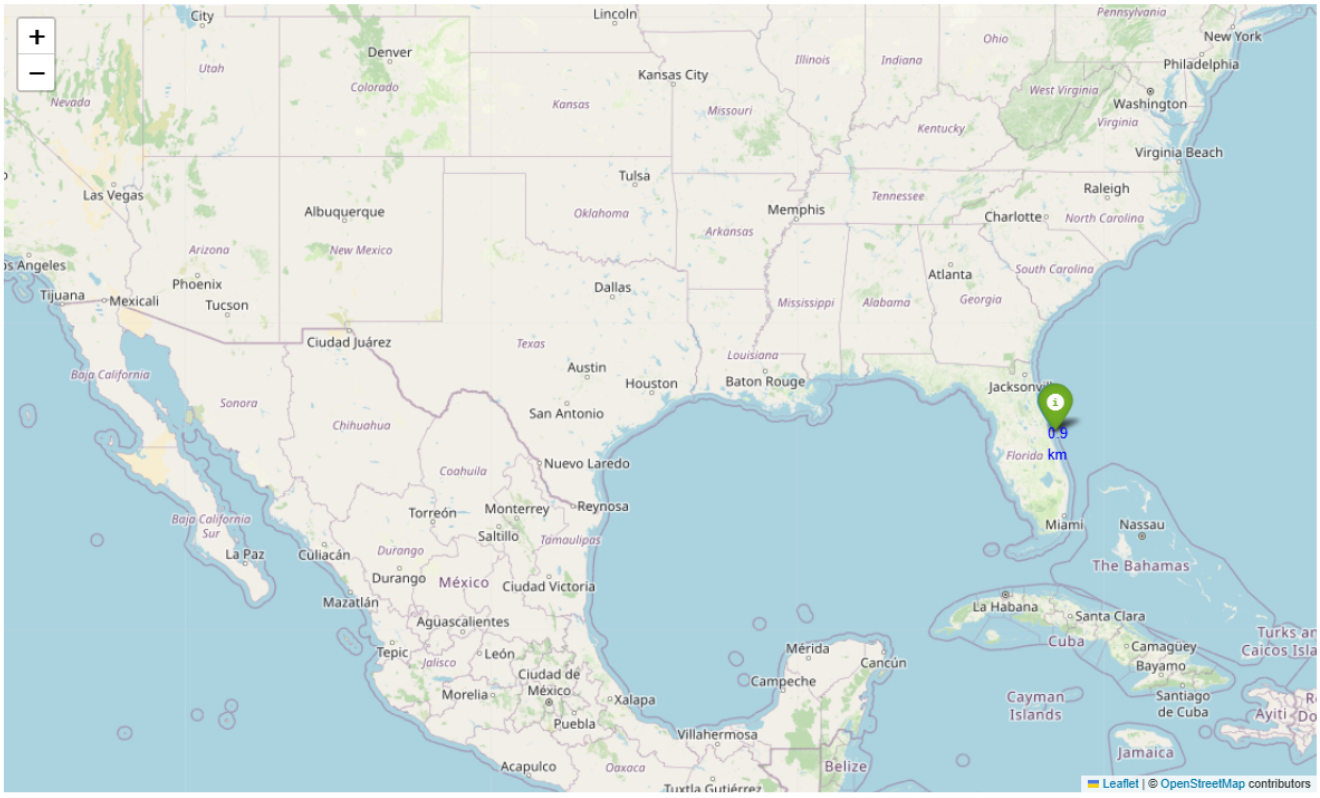
[17]:
```
0.865414876503625
```

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

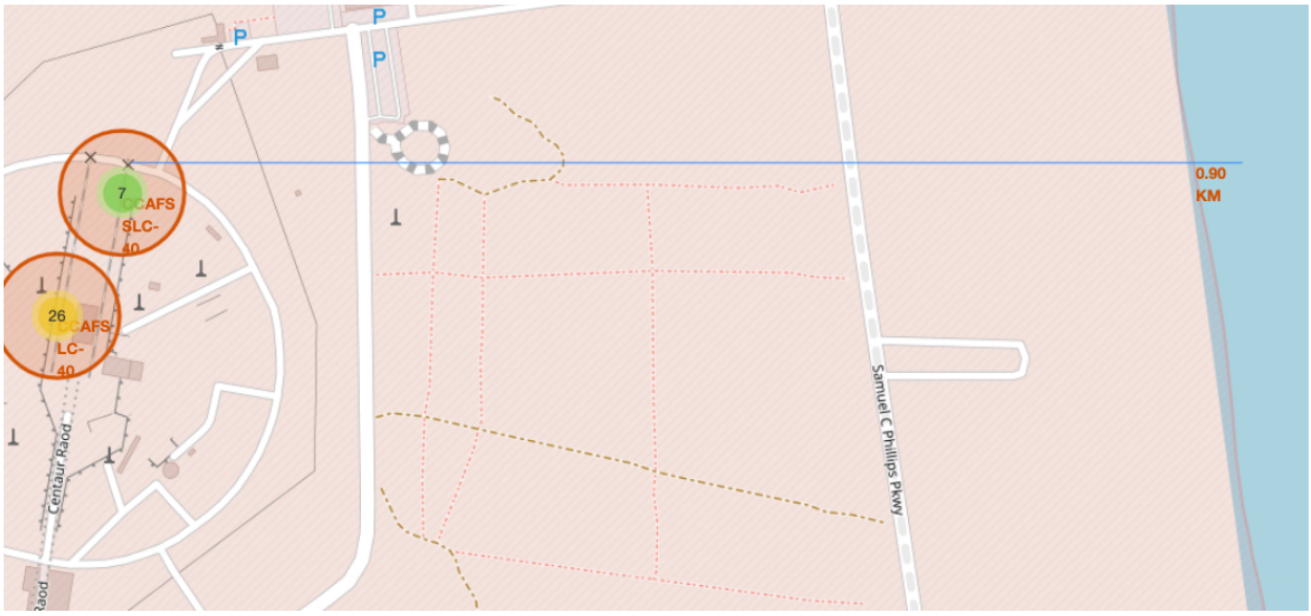
[23]:
```
# Create and add a folium.Marker on your selected closest coastline point on the map
# Display the distance between coastline point and launch site using the icon property
# for example
# distance_marker = folium.Marker(
#    coordinate,
#    icon=DivIcon(
#        icon_size=(20,20),
#        icon_anchor=(0,0),
#        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance),
#        )
#    )

# Initialize a folium map centered at the launch site
nasa_coordinate = [29.559684888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)

# Add a marker for the launch site
folium.Marker(
    location=[lat1, lon1],
    popup="Launch Site",
    icon=folium.Icon(color="blue", icon="info-sign")
).add_to(site_map)

# Add a marker for the coastline point, displaying the distance
folium.Marker(
    location=[lat2, lon2],
    popup=f"Coastline Point<br>Distance: {distance_km:.2f} km",
    icon=folium.Icon(color="green", icon="info-sign")
).add_to(site_map)

# Add a straight line (polyline) from the launch site to the coastline
folium.PolyLine(
    locations=[[lat1, lon1], [lat2, lon2]],
    color="red",
    weight= 2.0,
    opacity=1
).add_to(site_map)

# Create a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
# lines=folium.PolyLine(locations=coordinates, weight=1)

# Add a marker with the manually set distance at the midpoint of the line
midpoint_lat = (lat1 + lat2) / 2
midpoint_lon = (lon1 + lon2) / 2
```

```
folium.Marker(
    location=[midpoint_lat, midpoint_lon],
    popup="Approx. Distance: 0.9 km",  # Manually set the displayed distance
    icon=folium.DivIcon(html=f"""<div style="font-size: 12px; color: blue;">0.9 km</div>""")
).add_to(site_map)

# Display the updated map with the midpoint marker
site_map
```

[23]:



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



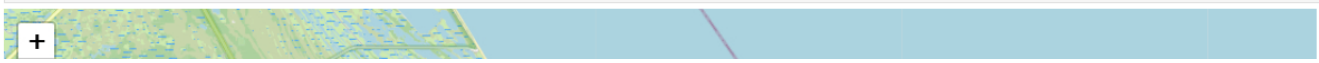
TODO: Similarly, you can draw a line betwee a launch site to its closest city, railway, highway, etc. You need to use `MousePosition` to find the their coordinates on the map first

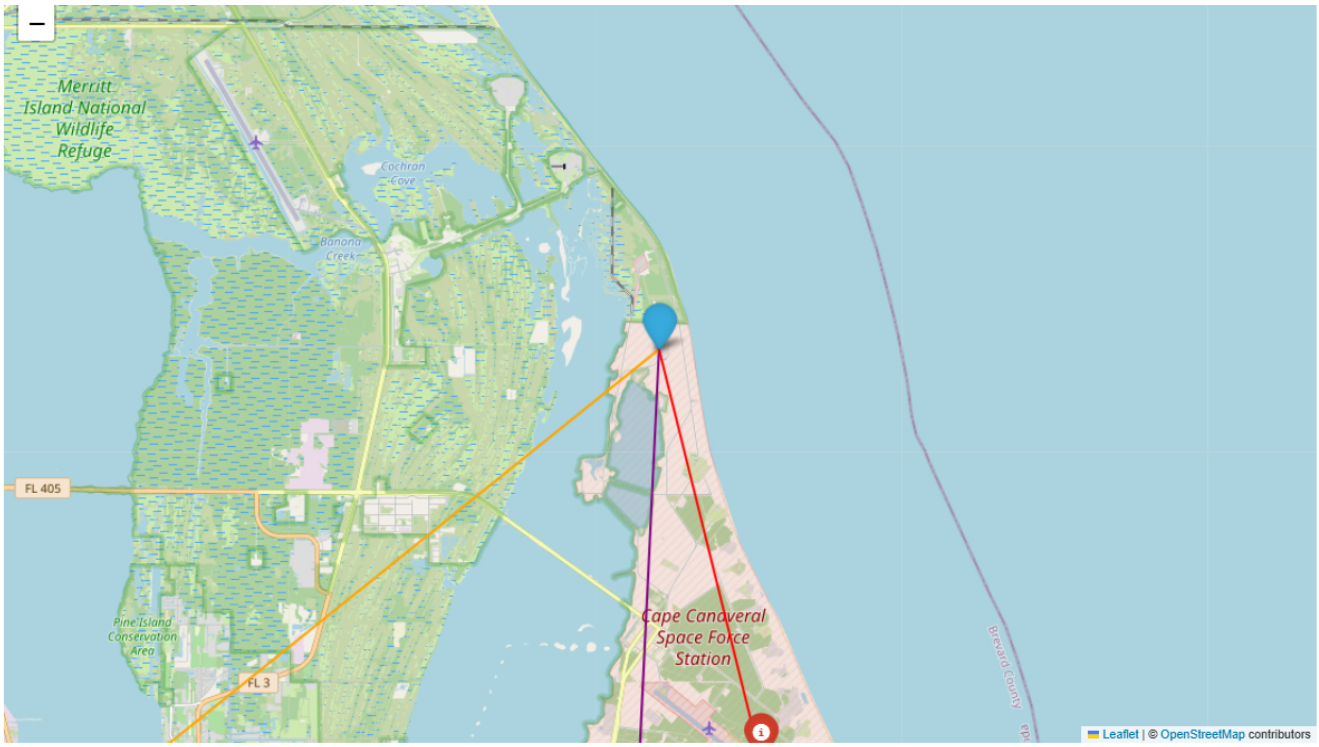
A railway map symbol may look like this:

A highway map symbol may look like this:

A city map symbol may look like this:

[22]:
```
import folium
import math

# Initialize the map centered at the launch site
launch_site_coords = [28.562302, -80.577356]
site_map = folium.Map(location=launch_site_coords, zoom_start=12)

# Haversine function to calculate distances
def haversine(lat1, lon1, lat2, lon2):
    # Radius of the Earth in kilometers
    R = 6371.0

    # Convert degrees to radians
    lat1, lon1, lat2, lon2 = map(math.radians, [lat1, lon1, lat2, lon2])

    # Differences in coordinates
    dlat = lat2 - lat1
    dlon = lon2 - lon1

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

    # Distance in kilometers
    distance = R * c
    return distance

# Define the function to add a marker and a polyline
def add_marker_and_line(map_object, lat, lon, label, color, distance=None):
    # Add a marker for the point of interest
    folium.Marker(
        location=[lat, lon],
        icon=folium.Icon(color=color),
        tooltip=f"{label}: {distance:.2f} km"
    ).add_to(map_object)

    # Add a polyline from the point of interest to the launch site
    folium.PolyLine(
        locations=[launch_site_coords, [lat, lon]],
        color=color,
        weight=2,
        tooltip=f"Distance: {distance:.2f} km"
    ).add_to(map_object)

# Coordinates of points of interest
city_coords = [28.07808, -80.6045]
railway_coords = [28.42936, -80.7653]
highway_coords = [28.45888, -80.54821]

# Calculate distances to the points of interest using the haversine function
city_distance = haversine(launch_site_coords[0], launch_site_coords[1], city_coords[0], city_coords[1])
railway_distance = haversine(launch_site_coords[0], launch_site_coords[1], railway_coords[0], railway_coords[1])
highway_distance = haversine(launch_site_coords[0], launch_site_coords[1], highway_coords[0], highway_coords[1])

# Add markers and lines for each point of interest
add_marker_and_line(site_map, city_coords[0], city_coords[1], "Closest City", "purple", city_distance)
add_marker_and_line(site_map, railway_coords[0], railway_coords[1], "Closest Railway", "orange", railway_distance)
add_marker_and_line(site_map, highway_coords[0], highway_coords[1], "Closest Highway", "red", highway_distance)

# Add a marker for the launch site itself
folium.Marker(
    location=launch_site_coords,
    icon=folium.Icon(color="blue", icon="rocket"),
    tooltip="Launch Site"
).add_to(site_map)

# Display the map
site_map
```

[22]:




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 Ploty Dash on detailed launch records.

Authors

Pratiksha Verma