

cognitiveclass.ai logo

✓ 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:

```
1 !pip3 install folium
2 !pip3 install wget
```

```
⇒ Requirement already satisfied: folium in /usr/local/lib/python3.10/dist-packages (0.1
Requirement already satisfied: branca>=0.6.0 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from
```

```

Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-p
Collecting wget
  Downloading wget-3.2.zip (10 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: wget
  Building wheel for wget (setup.py) ... done
  Created wheel for wget: filename=wget-3.2-py3-none-any.whl size=9656 sha256=d8ca5af
  Stored in directory: /root/.cache/pip/wheels/8b/f1/7f/5c94f0a7a505ca1c81cd1d9208ae2
Successfully built wget
Installing collected packages: wget
Successfully installed wget-3.2

```

```

1 import folium
2 import wget
3 import pandas as pd

```

```

1 # Import folium MarkerCluster plugin
2 from folium.plugins import MarkerCluster
3 # Import folium MousePosition plugin
4 from folium.plugins import MousePosition
5 # Import folium DivIcon plugin
6 from folium.features import DivIcon

```

✓ 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.

```

1 # Download and read the `spacex_launch_geo.csv`
2 spacex_csv_file = wget.download('https://cf-courses-data.s3.us.cloud-object-storage.ap
3 spacex_df=pd.read_csv(spacex_csv_file)

```

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

```

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

```

5 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	

Next steps:

[Generate code with launch_sites_df](#)

[View recommended plots](#)

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.

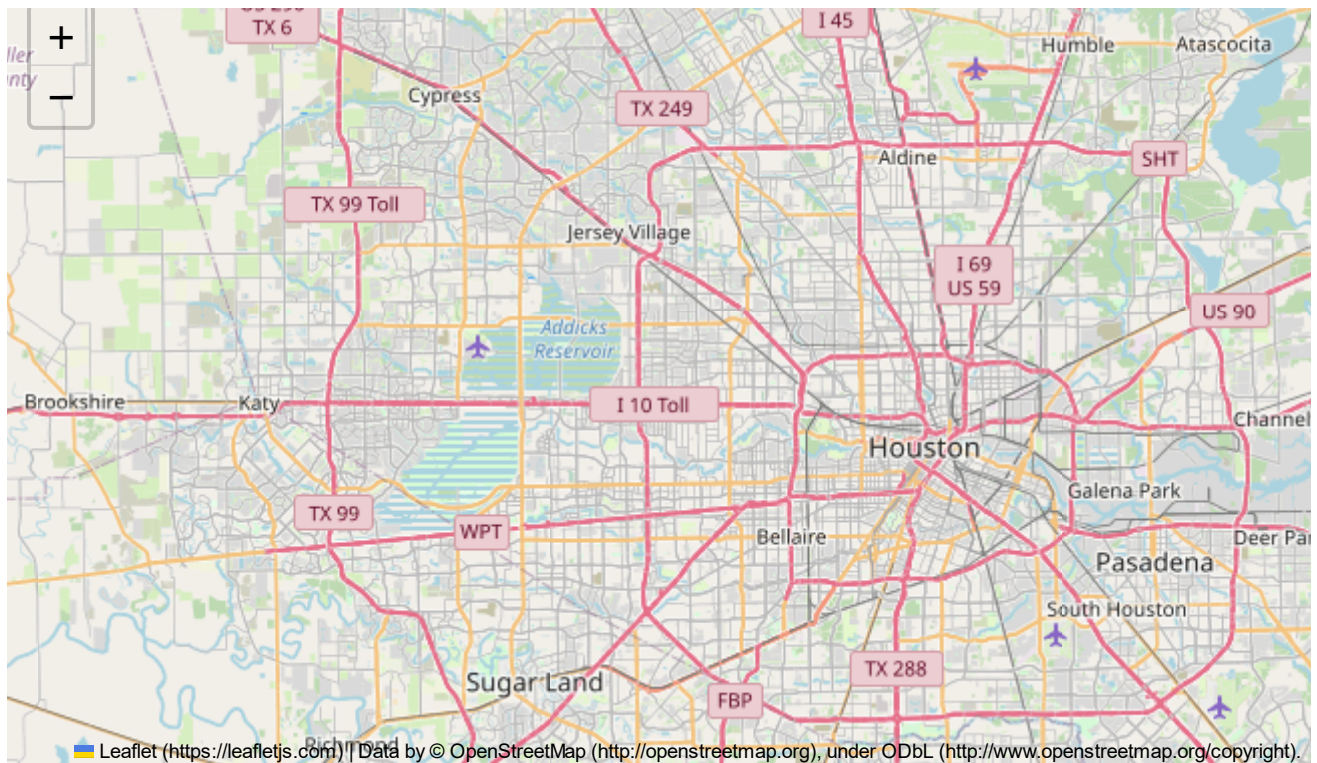
We first need to create a folium `Map` object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.

```
1 # Start location is NASA Johnson Space Center
2 nasa_coordinate = [29.559684888503615, -95.0830971930759]
3 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,

```
1 # Create a blue circle at NASA Johnson Space Center's coordinate with a popup label show
2 circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_chi
3 # Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its
4 marker = folium.map.Marker(
5     nasa_coordinate,
6     # Create an icon as a text label
7     icon=DivIcon(
8         icon_size=(20,20),
9         icon_anchor=(0,0),
10        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
11        )
12    )
13 site_map.add_child(circle)
14 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

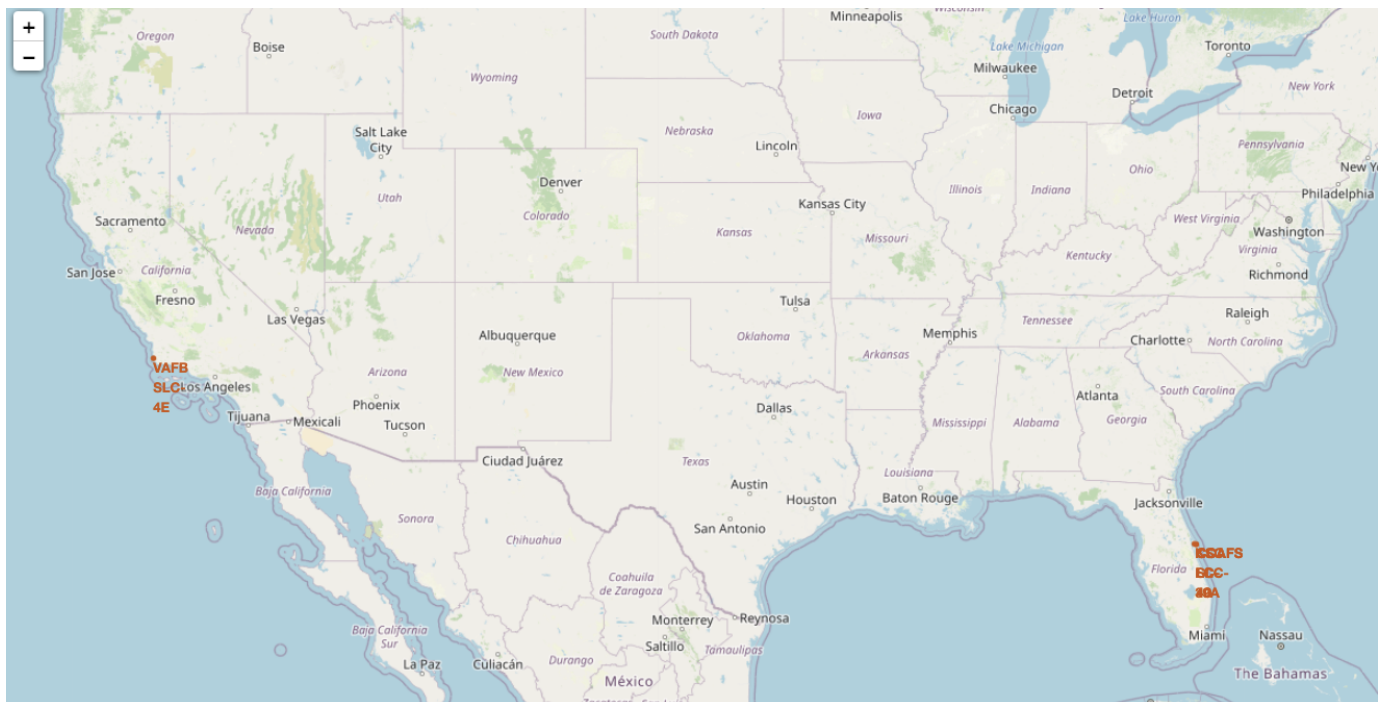
```

1 # Initial the map
2 site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
3 # For each launch site, add a Circle object based on its coordinate (Lat, Long) values.
4 for launch_site, site_lat, site_long in zip(launch_sites_df['Launch Site'], launch_sites['lat'], launch_sites['lon']):
5     site_coordinate = [site_lat, site_long]
6
7     circle = folium.Circle(site_coordinate, radius=1000, color='#d35400', fill=True).add_child(
8
9     marker = folium.Marker(
10         site_coordinate,
11         # Create an icon as a text label
12         icon=DivIcon(
13             icon_size=(20, 20),
14             icon_anchor=(0, 0),
15             html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % launch_site,
16             )
17         )
18     site_map.add_child(circle)
19     site_map.add_child(marker)
20 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.



Explanation:

- Most of Launch sites considered in this project are in proximity to the Equator line. Launch sites are made at the closest point possible to Equator line, because anything on the surface of the Earth at the equator is already moving at the maximum speed (1670 kilometers per hour). For example launching from the equator makes the spacecraft move almost 500 km/hour faster once it is launched compared half way to north pole.
- All launch sites considered in this project are in very close proximity to the coast While starting rockets towards the ocean we minimise the risk of having any debris dropping or exploding near people.

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

```
1 spacex_df.tail(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

```
1 marker_cluster = MarkerCluster()
```

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

```
1 # Apply a function to check the value of `class` column
2 # If class=1, marker_color value will be green
3 # If class=0, marker_color value will be red
4 spacex_df['marker_color'] = list(map(lambda x: 'green' if x==1 else 'red', spacex_df['c.
5 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	



```
1 # Function to assign color to launch outcome
2 def assign_marker_color(launch_outcome):
3     if launch_outcome == 1:
4         return 'green'
```



```

5     else:
6         return 'red'
7
8 spacex_df['marker_color'] = spacex_df['class'].apply(assign_marker_color)
9 spacex_df.tail(10)

```

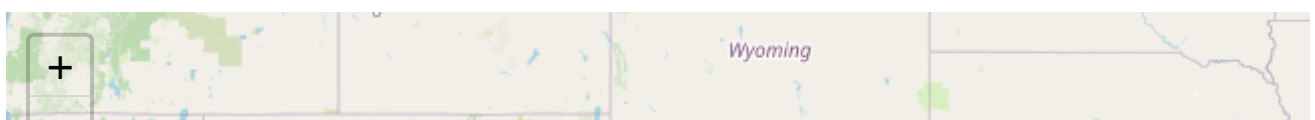
	Launch Site	Lat	Long	class	marker_color	
46	KSC LC-39A	28.573255	-80.646895	1	green	
47	KSC LC-39A	28.573255	-80.646895	1	green	
48	KSC LC-39A	28.573255	-80.646895	1	green	
49	CCAFS SLC-40	28.563197	-80.576820	1	green	
50	CCAFS SLC-40	28.563197	-80.576820	1	green	
51	CCAFS SLC-40	28.563197	-80.576820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

```

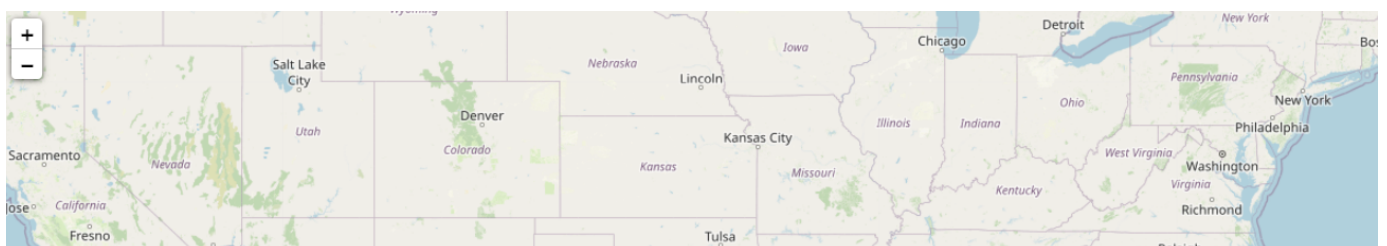
1 # Add the Marker cluster to the site map
2 site_map.add_child(marker_cluster)
3
4 # for each row in spacex_df data frame
5 # create a Marker object with its coordinate
6 # and customize the Marker's icon property to indicate if this launch was succeeded or 1
7 for site_lat, site_long, marker_color in zip(spacex_df['Lat'], spacex_df['Long'], spacex_df['marker_color']):
8     site_coordinate = [site_lat, site_long]
9     marker = folium.map.Marker(
10         site_coordinate,
11         # Create an icon as a text label
12         icon=folium.Icon(color='white',
13                           icon_color=marker_color)
14     )
15     marker.add_to(marker_cluster)
16
17 site_map

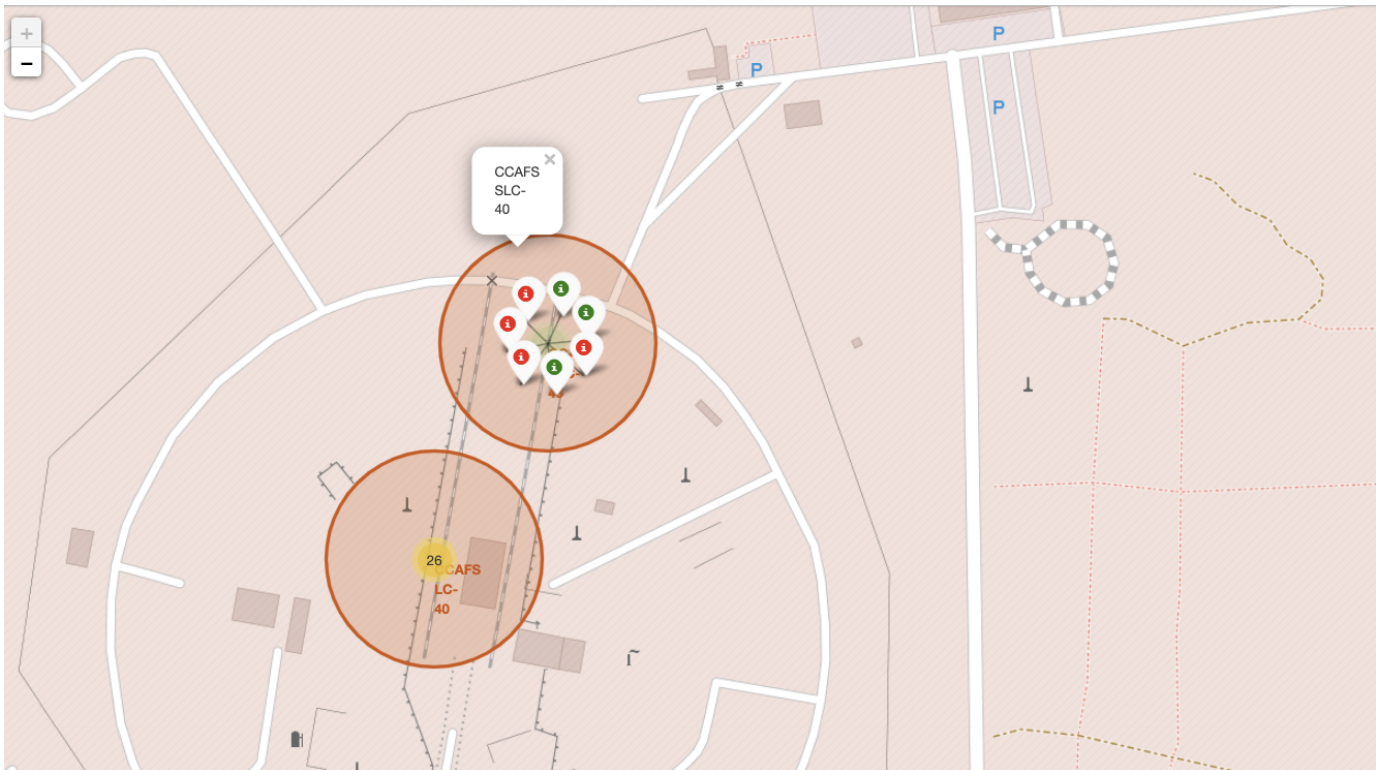
```





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)

```
1 # Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on the map
2 formatter = "function(num) {return L.Util.formatNum(num, 5)};"
3 mouse_position = MousePosition(
4     position='topright',
5     separator=' Long: ',
6     empty_string='NaN',
7     lng_first=False,
8     num_digits=20,
9     prefix='Lat:',
10    lat_formatter=formatter,
11    lng_formatter=formatter,
12 )
13
14 site_map.add_child(mouse_position)
15 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.

You can calculate the distance between two points on the map based on their Lat and Long values using the following method:

```
1 from math import sin, cos, sqrt, atan2, radians
2
3 def calculate_distance(lat1, lon1, lat2, lon2):
4     # approximate radius of earth in km
5     R = 6373.0
6
7     lat1 = radians(lat1)
8     lon1 = radians(lon1)
9     lat2 = radians(lat2)
10    lon2 = radians(lon2)
11
12    dlon = lon2 - lon1
13    dlat = lat2 - lat1
14
15    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
16    c = 2 * atan2(sqrt(a), sqrt(1 - a))
17
18    distance = R * c
19    return distance
```

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

```
1 # distance_railway = calculate_distance(lat1, lon1, lat2, lon2)
2 railway_marker = [28.55752, -80.80155]
3 launch_coordinate = [28.57337, -80.64669]
```

```

4 distance_railway = calculate_distance(railway_marker[0], railway_marker[1], launch_coo
5 distance_railway # distance in km

15.230651245141603

```

TODO: After obtained its coordinate, create a `folium.Marker` to show the distance

```

1 # create and add a folium.Marker on your selected closest railway point on the map
2 # show the distance to the launch site using the icon property
3 marker = folium.map.Marker(
4     railway_marker,
5     # Create an icon as a text label
6     icon=DivIcon(
7         icon_size=(400, 400),
8         icon_anchor=(0, 0),
9         html='<div style="font-size:400; color:#0c10f2;"><b>%s</b></div>' % str(ro
10     )
11 )
12 marker.add_to(site_map)
13
14 site_map

```

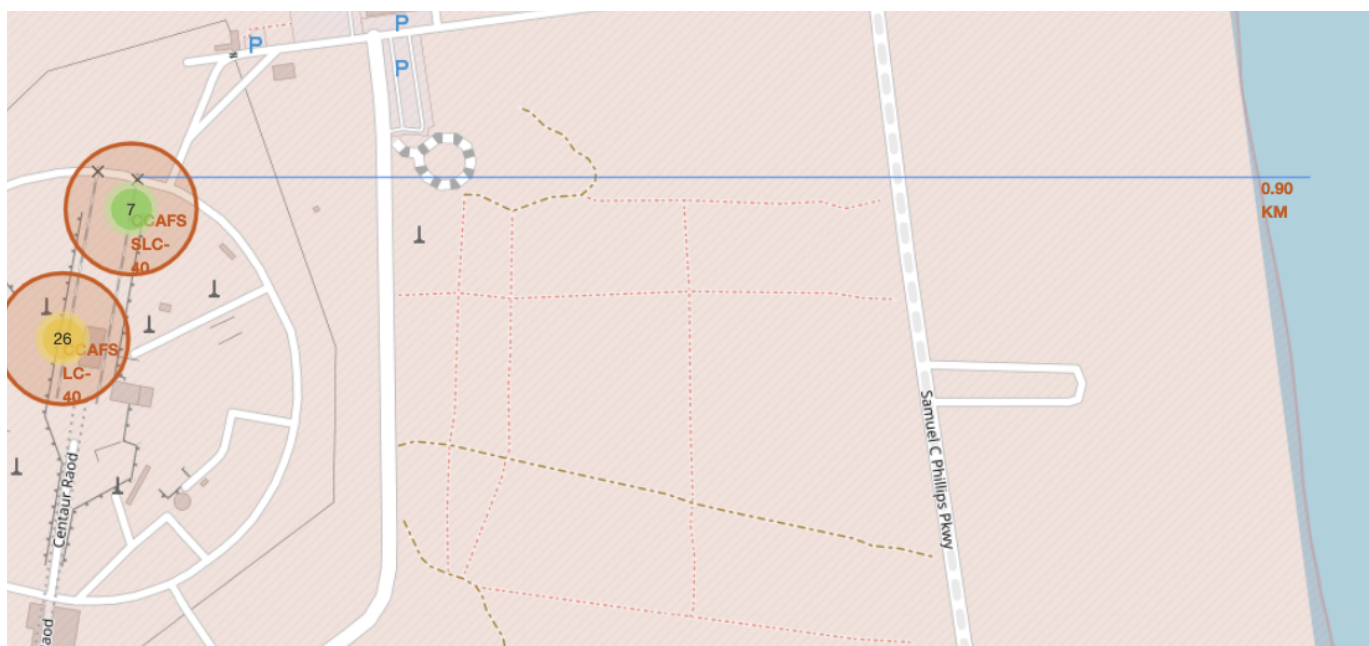


TODO: Draw a `PolyLine` between a launch site to the selected

```
1 # Create a `folium.PolyLine` object using the railway point coordinate and launch site
2 folium.PolyLine([railway_marker, launch_coordinate], color='blue').add_to(site_map)
3 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, coastline, highway, etc.

```

1 # Create a marker with distance to a closest city, coastline, highway, etc.
2 # Draw a line between the marker to the launch site
3 city = [28.61200, -80.80788]
4 coastline = [28.5858, -80.79952]
5 highway = [28.5402, -80.85079]
6
7 city_distance = calculate_distance(city[0], city[1], launch_coordinate[0], launch_coordinate[1])
8 coastline_distance = calculate_distance(coastline[0], coastline[1], launch_coordinate[0], launch_coordinate[1])

```

```

9 highway_distance = calculate_distance(highway[0], highway[1], launch_coordinate[0], laur
10
11 colors = ['red','orange','green']
12 html_colors = ['#dc3545','#fd7e14','#198754']
13
14 for coordinate ,distance, color, html_color in zip([city, coastline, highway], [city_di
15     marker = folium.map.Marker(
16         coordinate,
17         # Create an icon as a text label
18         icon=DivIcon(
19             icon_size=(20,20),
20             icon_anchor=(0,0),
21             html='<div style="font-size: 12; color: '+html_color+';"><b>%s</b></div>'
22         )
23     )
24     marker.add_to(site_map)
25     folium.PolyLine([coordinate, launch_coordinate], color=color).add_to(site_map)
26 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.

Explanation:

- From the visual analysis of the launch site KSC LC-39A we can clearly see that it is:
 - relative close to railway (15.23 km)
 - relative close to highway (20.28 km)
 - relative close to coastline (14.99 km)
- Also the launch site KSC LC-39A is relative close to its closest city Titusville (16.32 km).
- Failed rocket with its high speed can cover distances like 15-20 km in few seconds. It could be potentially dangerous to populated areas.

✓ 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

[Yan Luo](#)

▼ Other Contributors

Joseph Santarcangelo

▼ Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-05-26	1.0	Yan	Created the initial version

Copyright © 2021 IBM Corporation. All rights reserved.