(https://skills.network/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=1000 SkillsNetwork-Channel-SkillsNetworkCoursesIBMDS0321ENSkillsNetwork26802033-2022-01-01)

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:

In [1]:
```
import piplite
await piplite.install(['folium'])
await piplite.install(['pandas'])
```

In [6]:
```
import folium
import pandas as pd
```

In [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 (https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_3/DV0101EN-3-5-1-Generating-Maps-in-Python-py-v2.0.ipynb)](https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_3/DV0101EN-3-5-1-Generating-Maps-in-Python-py-v2.0.ipynb)

In [4]:
```
## 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.

In [7]:
```
# 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-
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.

In [8]:
```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude
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
```

Out[8]:

	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.

In [9]:
```
# 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,

In [10]:
```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=Tru
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon s
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>' % 'N
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```

Out[10]:

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(...))
```

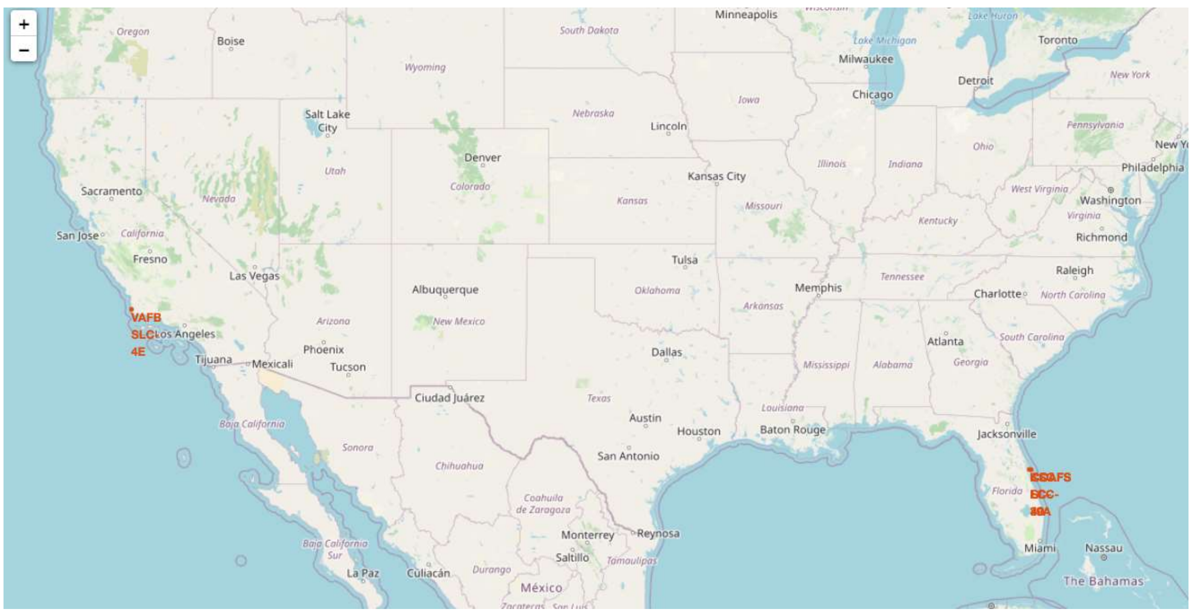
An example of folium.Marker:

```
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', ))
```

In [11]:
```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
launch_sites_dict = launch_sites_df.set_index('Launch Site').T.to_dict('list')
for launch_site_name, coordinates in launch_sites_dict.items():
    latitude, longitude = coordinates

    # Create a Circle object for each launch site with its coordinate
    circle = folium.Circle(
        location=[latitude, longitude],
        radius=1000,  # Adjust the radius as needed
        color='#d35400',
        fill=True,
        fill_color='#d35400'
    )
    cor=latitude, longitude
    marker=folium.map.Marker(
        cor,
        icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>'
        )
    )


    # Create a Popup label for each launch site with its name
    popup_label = folium.Popup(launch_site_name)

    # Add the Circle object and Popup label to the map
    circle.add_child(popup_label)
    site_map.add_child(circle)
    site_map.add_child(marker)

# Display the map with Circle objects and Popup labels
site_map.save("launch_sites_map.html")  # Save the map to an HTML file
site_map
# For each launch site, add a Circle object based on its coordinate (Lat, Long
```

Out[11]:

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.

In [ ]:
```
# 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

In [13]:
```
spacex_df.tail(10)
```

Out[13]:

	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

In [14]:
```
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

In [15]:
```

# 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 get_marker_color(row):
    if row==1:
        return "green"
    else:
        return "red"
spacex_df["marker_color"]=spacex_df["class"].apply(get_marker_color)
spacex_df.tail(10)
```

Out[15]:

	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

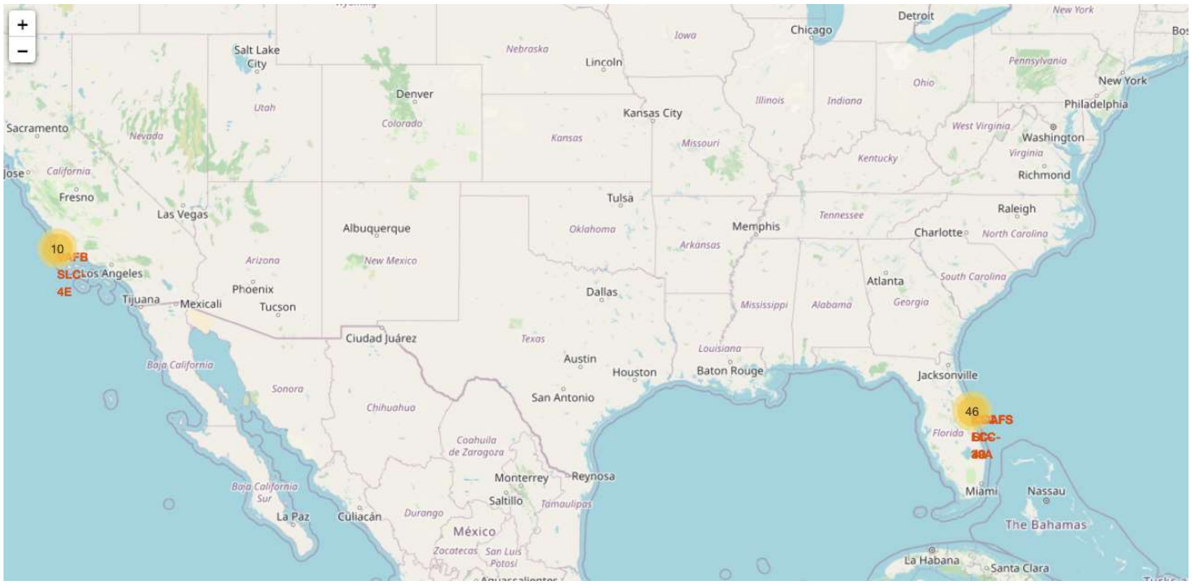
TODO: For each launch result in `spacex_df` data frame, add a `folium.Marker` to `marker_cluster`

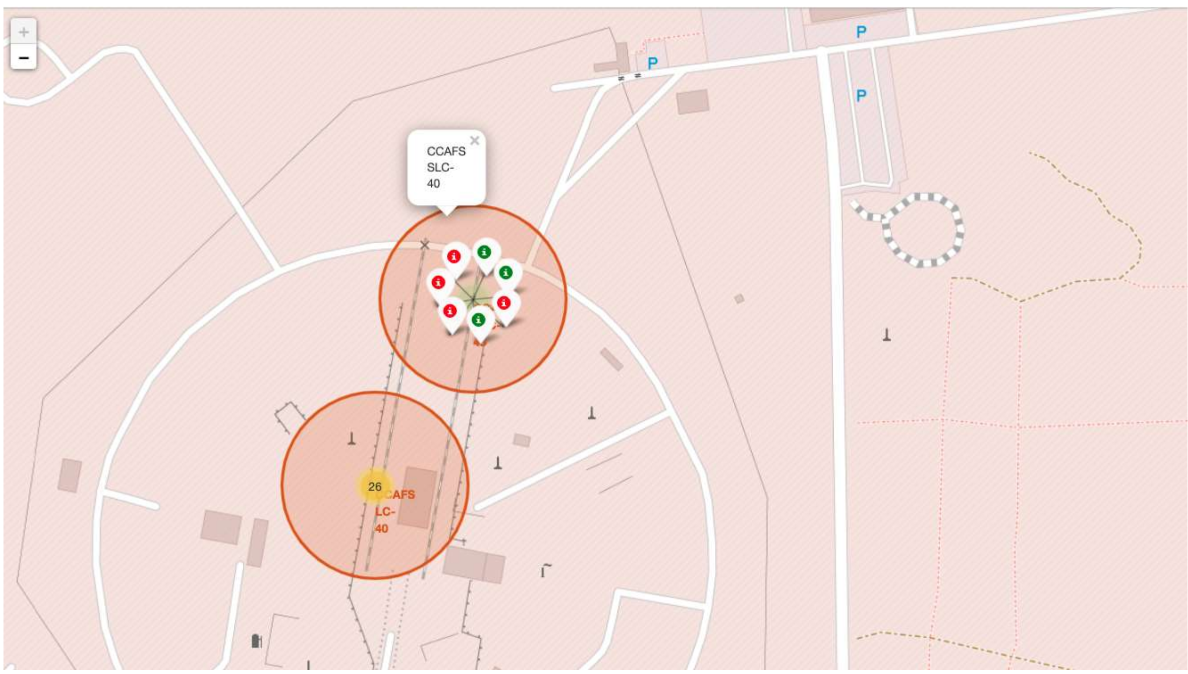
In [16]:

```
# 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 succ
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
for index, record in spacex_df.iterrows():
    launch_site_name = record['Launch Site']
    latitude = record['Lat']
    longitude = record['Long']
    marker_color = 'green' if record['class'] == 1 else 'red'


    # TODO: Create and add a Marker cluster to the site map
    marker = folium.Marker(
        location=[latitude, longitude],
        icon=folium.Icon(color='white', icon_color=marker_color),
        popup=launch_site_name
    )

    # Add the Marker to the MarkerCluster
    marker.add_to(marker_cluster)

# Add the MarkerCluster to the site_map
site_map.add_child(marker_cluster)


# Display the map with Marker objects and MarkerCluster
site_map.save("launch_sites_map.html")  # Save the map to an HTML file
marker_cluster.add_child(marker)
site_map
```

Out[16]:

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.

In [ ]:
```python
# 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)

In [17]:
```
# Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on the
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
```

Out[17]:

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.

In [18]:
```
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.

In [19]:
```
# find coordinate of the closet coastline
# e.g.,: Lat: 28.56367  Lon: -80.57163
launch_site_lat = 28.56321
launch_site_lon = -80.57681

# Replace these values with the latitude and longitude of the closest coastlin
coastline_lat = 28.56391
coastline_lon = -80.56797
distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coas
```

In [20]:
```
# Create and add a folium.Marker on your selected closest coastline point on t
# Display the distance between coastline point and launch site using the icon
# 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>' % "
#        )
#    )
coastline_marker = folium.Marker(
    location=[coastline_lat, coastline_lon],
    popup='Closest Coastline Point',
    icon=folium.DivIcon(
        icon_size=(20, 20),
        icon_anchor=(0, 0),
        html=f'<div style="font-size: 12; color:#d35400;"><b>{distance_coastli
    ),
)
site_map.add_child(coastline_marker)

# Display the map
site_map
```

Out[20]:

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

In [21]:
```
# Create a `folium.PolyLine` object using the coastline coordinates and launch
lines = folium.PolyLine(locations=[[launch_site_lat, launch_site_lon], [coastl
site_map.add_child(lines)
```

Out[21]:

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

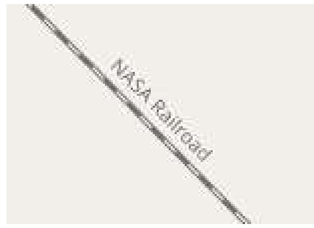
A railway map symbol may look like this:

download.png

download.png

railway.png

download.png



A highway map symbol may look like this:

A city map symbol may look like this:

In [37]:

```
# Create a marker with distance to a closest city, railway, highway, etc.
# Draw a line between the marker to the launch site
# Create a map centered at the launch site
site_map = folium.Map(railway_location, zoom_start=13)

# Create a marker at the launch site with the distance to the closest railway
marker = folium.Marker(
    location=launch_site,
    icon=folium.DivIcon(
        icon_size=(40, 20),
        icon_anchor=(20, 0),
        html=f'<div style="font-size: 5px; color:lite black;"><bkm</b></div>',
    ),
)
lines = folium.PolyLine(locations=[[launch_site_lat, launch_site_lon], [coastl
site_map.add_child(marker)

# Draw a line between the marker (launch site) and the railway location

# Display the map
site_map
```

Out[37]:

In [ ]:

In [ ]:

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 (https://www.linkedin.com/in/pratiksha-verma-6487561b1/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=1000
SkillsNetwork-Channel-SkillsNetworkCoursesIBMDS0321ENSkillsNetwork865-2022-01-01)

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-11-09	1.0	Pratiksha Verma	Converted initial version to Jupyterlite