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]:
```
!pip3 install folium
!pip3 install wget
```

```
Requirement already satisfied: folium in /opt/conda/envs/Python-3.8-main/lib/py
thon3.8/site-packages (0.12.1)
Requirement already satisfied: numpy in /opt/conda/envs/Python-3.8-main/lib/pyt
hon3.8/site-packages (from folium) (1.19.2)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.8-main/lib/
python3.8/site-packages (from folium) (2.25.1)
Requirement already satisfied: branca>=0.3.0 in /opt/conda/envs/Python-3.8-mai
n/lib/python3.8/site-packages (from folium) (0.4.2)
Requirement already satisfied: jinja2>=2.9 in /opt/conda/envs/Python-3.8-main/l
ib/python3.8/site-packages (from folium) (3.0.0)
Requirement already satisfied: MarkupSafe>=2.0.0rc2 in /opt/conda/envs/Python-
3.8-main/lib/python3.8/site-packages (from jinja2>=2.9->folium) (2.0.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/envs/Python-
3.8-main/lib/python3.8/site-packages (from requests->folium) (1.26.6)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/envs/Python-3.8-main/
lib/python3.8/site-packages (from requests->folium) (2.8)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python-3.8
```

```
-main/lib/python3.8/site-packages (from requests->folium) (2021.5.30)
Requirement already satisfied: chardet<5,>=3.0.2 in /opt/conda/envs/Python-3.8-
main/lib/python3.8/site-packages (from requests->folium) (3.0.4)
Requirement already satisfied: wget in /opt/conda/envs/Python-3.8-main/lib/pyth
on3.8/site-packages (3.2)
```

In [2]:
```
import folium
import wget
import pandas as pd
```

In [3]:
```
# 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
```

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 [4]:
```
# Download and read the `spacex_launch_geo.csv`
spacex_csv_file = wget.download('https://cf-courses-data.s3.us.cloud-object-s
spacex_df=pd.read_csv(spacex_csv_file)
```

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

In [5]:
```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitud
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[5]:

	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	-120.610746

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 [6]:
```
# 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 [7]:
```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=Tr
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon
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>' % '
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```

Out[7]: Make this Notebook Trusted to load map: File -> Trust Notebook

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`

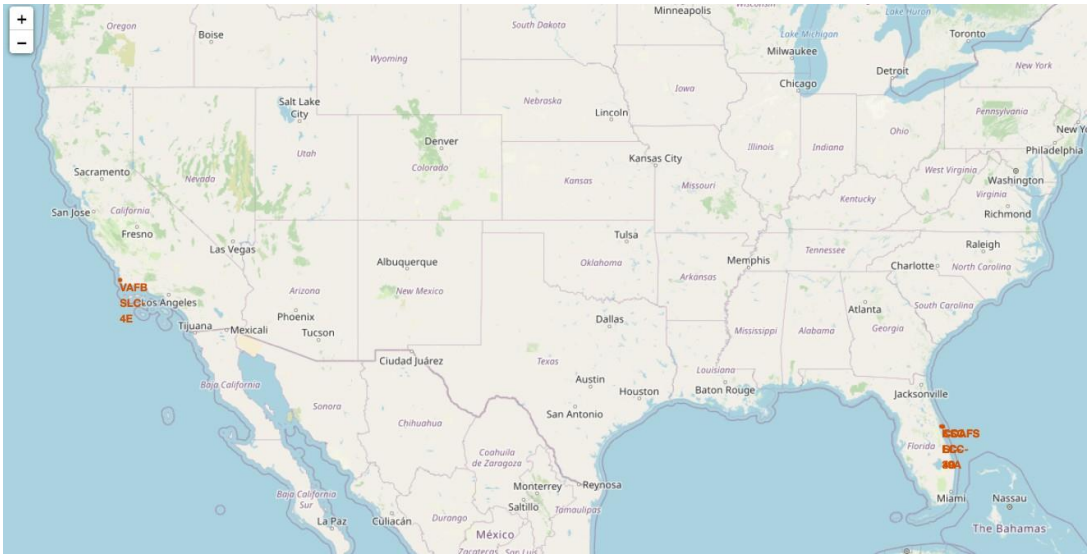
TODO: Create and add `folium.Circle` and `folium.Marker` for each launch site on the site map

In [8]:

```
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Lon
for launch_site, site_lat, site_long in zip(launch_sites_df['Launch Site'], l
    site_coordinate = [site_lat, site_long]

    circle = folium.Circle(site_coordinate, radius=1000, color='#d35400', fil

    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>'
            )
        )
    site_map.add_child(circle)
    site_map.add_child(marker)
site_map
```

Out[8]: Make this Notebook Trusted to load map: File -> Trust Notebook

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

In [9]:
```
spacex_df.tail(10)
```

Out[9]:

	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 [10]:
```
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 [11]:
```
# 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', sp
spacex_df.tail(10)
```

Out[11]:

	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

In [12]:
```
# Function to assign color to launch outcome
def assign_marker_color(launch_outcome):
    if launch_outcome == 1:
        return 'green'
    else:
        return 'red'

spacex_df['marker_color'] = spacex_df['class'].apply(assign_marker_color)
spacex_df.tail(10)
```

Out[12]:

	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

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`

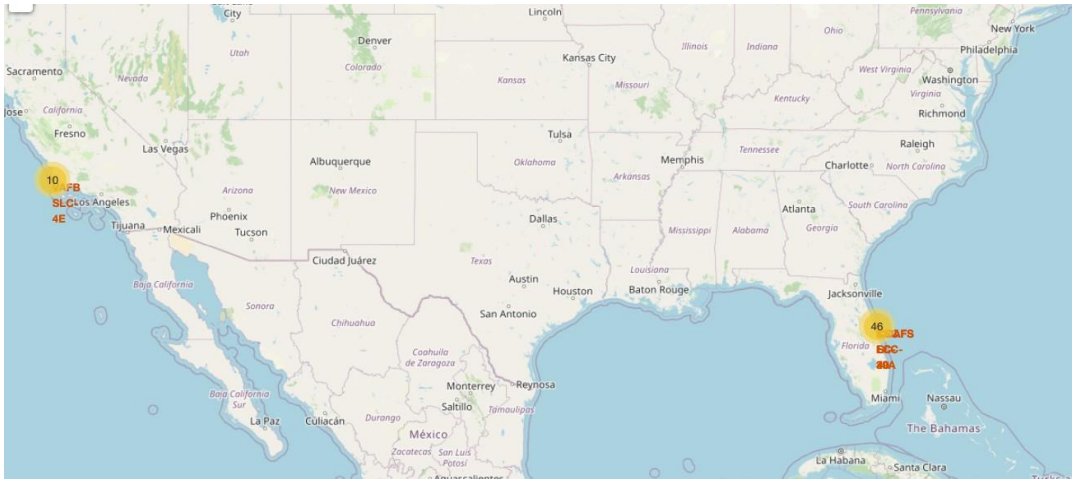
In [13]:

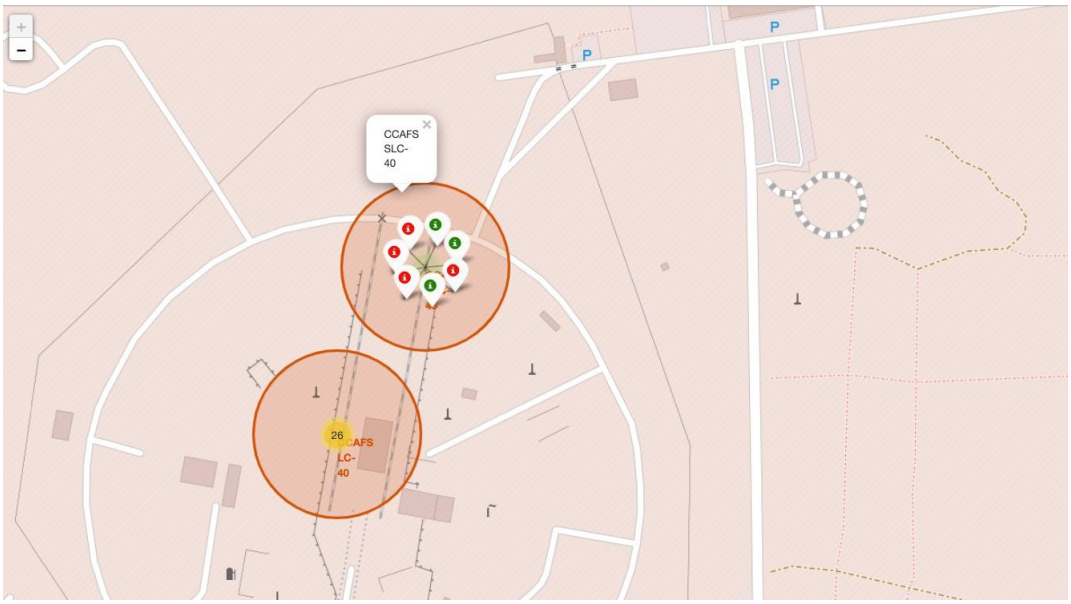
```
# Add the Marker cluster to the site map
site_map.add_child(marker_cluster)

# 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 suc
for site_lat, site_long, marker_color in zip(spacex_df['Lat'], spacex_df['Lon
    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
```

Out[13]: Make this Notebook Trusted to load map: File -> Trust Notebook

Your updated map may look like the following screenshots:


Detroit
Chicago
New York
Iowa
Nebraska
Salt Lake





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)

In [14]:

```
# Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on th
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[14]: Make this Notebook Trusted to load map: File -> Trust Notebook

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:

In [15]:
```
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 railway using MousePosition and calculate the distance between the railway point to the launch site.

In [16]:
```
# distance_railway = calculate_distance(lat1, lon1, lat2, lon2)
railway marker = [28 55752  -80 80155]
```

```
launch_coordinate = [28.57337, -80.64669]
distance_railway = calculate_distance(railway_marker[0], railway_marker[1], l
distance_railway  # distance in km
```

Out[16]: 15.230651245141603

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

In [17]:

```
# create and add a folium.Marker on your selected closest raiwaly point on th
# show the distance to the 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>'
            )
    )
marker.add_to(site_map)

site_map
```

Out[17]: Make this Notebook Trusted to load map: File -> Trust Notebook

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

In [18]:

```
# Create a `folium.PolyLine` object using the railway point coordinate and la
folium.PolyLine([railway_marker, launch_coordinate], color='blue').add_to(sit
site_map
```

Out[18]: Make this Notebook Trusted to load map: File -> Trust Notebook

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, coastline, highway, etc.

In [19]:

```
# Create a marker with distance to a closest city, coastline, highway, etc.
# Draw a line between the marker to the launch site
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], la
coastline_distance = calculate_distance(coastline[0], coastline[1], launch_co
highway_distance = calculate_distance(highway[0], highway[1], launch_coordina

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

for coordinate ,distance, color, html_color in zip([city, coastline, highway]
    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
                )
            )
    marker.add_to(site_map)
    folium.PolyLine([coordinate, launch_coordinate], color=color).add_to(site
site_map
```

Out[19]: Make this Notebook Trusted to load map: File -> Trust Notebook

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 Ploty 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.