Launch Sites Locations Analysis with Folium

Estimated time needed: **1 hour**

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 proximity to 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 lab, 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 detailed visual analysis using `Folium`.

Objectives

This lab contains the following tasks:

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

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

Let's first import required libraries and data:

```
!pip install folium pandas
```

```
import folium
import pandas as pd
import math, numpy, datetime, time, sys
import warnings
warnings.filterwarnings("ignore")
```

```
# Import folium MarkerCluster plugin
from folium.plugins import MarkerCluster
from folium.plugins import MousePosition
from folium.plugins import DivIcon
from folium.plugins import HeatMap
from folium.plugins import Draw
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 Folium

Task 1: Mark all launch sites on a map

First, let's try to add each site's location on a map using siti'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.

```
# Download and read the `spacex_launch_geo.csv`
!wget -q -O spacex_launch_geo.csv https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv
spacex_csv_file = "./spacex_launch_geo.csv"
spacex_df = pd.read_csv(spacex_csv_file)
spacex_df.head(5)
```

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

```
# Select launch sites and their coordinates
launch_sites = spacex_df[['Launch Site', 'Lat', 'Long']].groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites[['Launch Site', 'Lat', 'Long']]
launch_sites_df.rename(columns={'Lat':'Latitude', 'Long':'Longitude'},inplace=True)
launch_sites_df
```

	Launch Site	Latitude	Longitude
0	CCAFS LC-40	28.562302	-80.577366
1	CCAFS SLC-40	28.563857	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

Across coordinates, one can find out coordinates that can not give you any detailed insights about where exact those launch sites. If you are very good at geography, you can interpret those coordinates directly in your mind. If not, then let's first let's calculate those locations by plotting them on a map.

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

```
# Start location is NASA Johnson Space Center
nasa_coordinate = [29.559684, -95.0830091]
site_map = folium.Map(location=nasa_coordinate, zoom_start=10)
```

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

```
# Create and add a circle marker for the NASA Johnson Space Center coordinate.
circle = folium.Circle(nasa_coordinate, radius=1000, color='#000000', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
marker = folium.Marker(nasa_coordinate, icon=folium.DivIcon(icon_size=(20,20),icon_anchor=(0,0),
                       html='<div style="font-size: 24pt">%d</div>'%1,
                       ).add_child(circle)
                       )
# Add to map
site_map.add_child(circle)
site_map.add_child(marker)
site_map
```

and you should find a small yellow circle near the city of Houston and you can zoom out 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 map

An example of `launch_sites`

```
launch_sites_df[['Launch Site', 'Latitude', 'Longitude']]
```

	Launch Site	Latitude	Longitude
0	CCAFS LC-40	28.562302	-80.577366
1	CCAFS SLC-40	28.563857	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For one launch site, add a circle marker, one additional text label, and a text label
# add circle
circle = folium.Circle(nasa_coordinate, radius=1000, color='#000000', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
# add text
marker = folium.Marker(nasa_coordinate,
                       icon=folium.DivIcon(icon_size=(20,20),icon_anchor=(0,0),
                       html='<div style="font-size: 24pt">%d</div>'%1,
                       )
                       )
# for each site, add a circle
for lat,lon,site in zip(launch_sites_df.Latitude,launch_sites_df.Longitude,launch_sites_df['Launch Site']):
    circle = folium.Circle([lat,lon], radius=1000, color='#000000', fill=True).add_child(folium.Popup(site))
    marker = folium.Marker([lat,lon],
                           icon=folium.DivIcon(icon_size=(20,20),icon_anchor=(0,0),
                           html='<div style="font-size: 24pt">%d</div>'%1,
                           )
                           )
    # add circle and marker to map
    site_map.add_child(circle)
    site_map.add_child(marker)
site_map
```

The expected 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 in the previous exploratory data analysis lab, you have created a column for `Class` indicating whether a launch was successful or not:

```
spacex_df.head(10)
```

	Launch Site	Lat	Long	Class
0	CCAFS LC-40	28.562302	-80.577366	0
1	CCAFS LC-40	28.562302	-80.577366	0
2	CCAFS LC-40	28.562302	-80.577366	0
3	CCAFS SLC-40	28.563857	-80.576820	1
4	CCAFS LC-40	28.562302	-80.577366	0
5	CCAFS SLC-40	28.563857	-80.576820	1
6	CCAFS SLC-40	28.563857	-80.576820	1
7	CCAFS LC-40	28.562302	-80.577366	0
8	CCAFS SLC-40	28.563857	-80.576820	0
9	CCAFS SLC-40	28.563857	-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 site happens to one of the launch sites, which means many launch records will have the exact same coordinate. Marker clusters can be used in order to group a lot of markers in a big map containing many markers having the same coordinate.

Let's first create a `MarkerCluster` object

```
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

```
# Apply a function to check the value of `class` column
def assign_marker_color(class_value):
# If class value is 0, return 'red', otherwise return 'green'
    return 'red' if class_value == 0 else 'green'
```

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

```
# For each row in spacex_df data frame
# lat = Latitude, long = Longitude, and class = Class
# Add a marker to the marker_cluster
# Assign a color to the marker based on the Class value
# Hint: icon=folium.Icon(color=marker_color)
# Add marker_cluster to the map
for lat, long, class in zip(spacex_df.Lat, spacex_df.Long, spacex_df.Class):
    marker = folium.Marker([lat, long],
                           icon=folium.Icon(color='white',
                           )
                           )
    marker_cluster.add_child(marker)
site_map
```

Your updated map may look like the following screenshots:

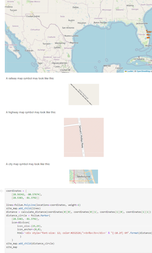
From the zoom-in/out marked markers in market clusters, you should be able to easily identify which launch sites have relatively high success rates.

TASK 3: Calculate the distances between a launch site to its proximities

Next, let's try to explore and analyze the proximities of launch sites.

Let's first add a `MousePosition` on the map to get coordinates 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)

```
# Add Mouse Position to get the coordinates (Lat, Long) of a mouse over on the map
formatter = "function(num) {return L.Util.formatNum(num, 5);}"
mouse_position = MousePosition(
    position='bottomright',
    separator=' Long: ',
    empty_string="Not",
    lng_first=False,
    num_digits=20,
    prefix='Lat:',
    lat_formatter=formatter,
    lng_formatter=formatter,
)
site_map.add_child(mouse_position)
site_map
```



Now a point is in a second site and wanted to proximity to see if you can easily find any railway, highway, coastline, etc. Then you can move to these points and mark down their coordinates (shown on the top-left) in order to the distance to the launch site.

You're now in a position to calculate the proximity to see if you find any railway, highway, coastline, etc. Then you can draw a line to the closest railway, highway, coastline, etc. For example, you can draw a line to the closest city to the launch site.

```
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
```

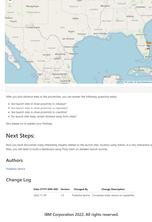
TODO: Calculate and add the closest coastline using MousePosition and calculate the distance between the coastline point and the launch site.

```
# Find coordinates of the closest coastline
# e.g., lat, lon, coastline_lat, coastline_lon
distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
coastline_text = folium.Marker(
    [coastline_lat, coastline_lon],
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12pt">%.2f</div>',
        )
    )
site_map.add_child(coastline_text)
```

TODO: Calculate and add a `PolyLine` between a launch site to the selected coastline point

```
# Create a PolyLine object using the coordinates and distance
coordinates = [[launch_site_lat, launch_site_lon], [coastline_lat, coastline_lon]]
lines = folium.PolyLine(locations=coordinates, weight=1)
site_map.add_child(lines)
```

Now, you can explore the map and calculate the distance from the following coordinates:



A railway map symbol may look like this:

A highway map symbol may look like this:

A city map symbol may look like this:


```
# calculate distance
distance_highway = calculate_distance(launch_site_lat, launch_site_lon, highway_lat, highway_lon)
print('distance_highway', distance_highway)
highway_text = folium.Marker(
    [highway_lat, highway_lon],
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12pt">%.2f</div>',
        )
    )
site_map.add_child(highway_text)
```

```
# calculate distance
distance_city = calculate_distance(launch_site_lat, launch_site_lon, city_lat, city_lon)
print('distance_city', distance_city)
city_text = folium.Marker(
    [city_lat, city_lon],
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12pt">%.2f</div>',
        )
    )
site_map.add_child(city_text)
```

After you pick distance lines to the proximities, you can wonder the following questions ideally:

- 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 enough 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

Rodolfo Ibeles

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-11-09	1.0	Rodolfo Ibeles	Initial Version Created with extensive suggestions

© IBM Corporation 2022. All rights reserved.