Hands-on Lab: Interactive Visual Analytics with Folium

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:

```
import piplite

await piplite.install(['folium'])

await piplite.install(['pandas'])


import folium

import pandas as pd
```

```
# 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.

```
# 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-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv'

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.

```
# 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

	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.

Start location is NASA Johnson Space Center

nasa_coordinate = [29.559684888503615, -95.0830971930759]

site_map = folium.Map(location=nasa_coordinate, zoom_start=10)

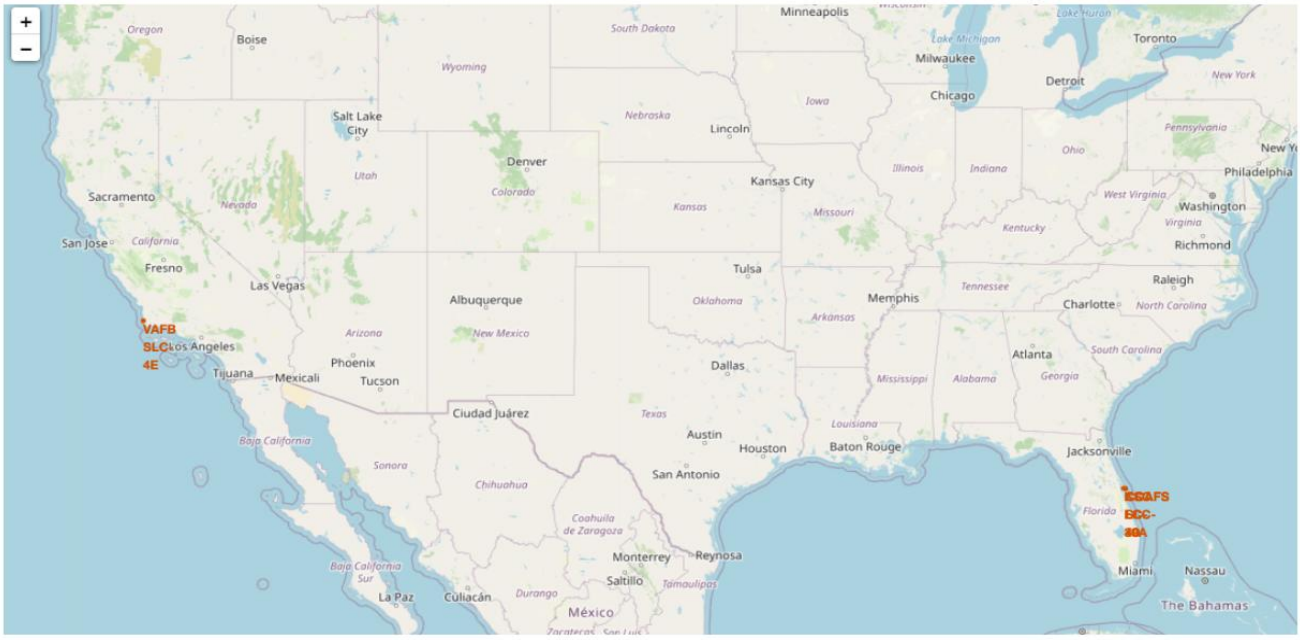
We could use folium.Circle to add a highlighted circle area with a text label on a specific coordinate. For example,

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)
```



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

 # 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

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

```
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

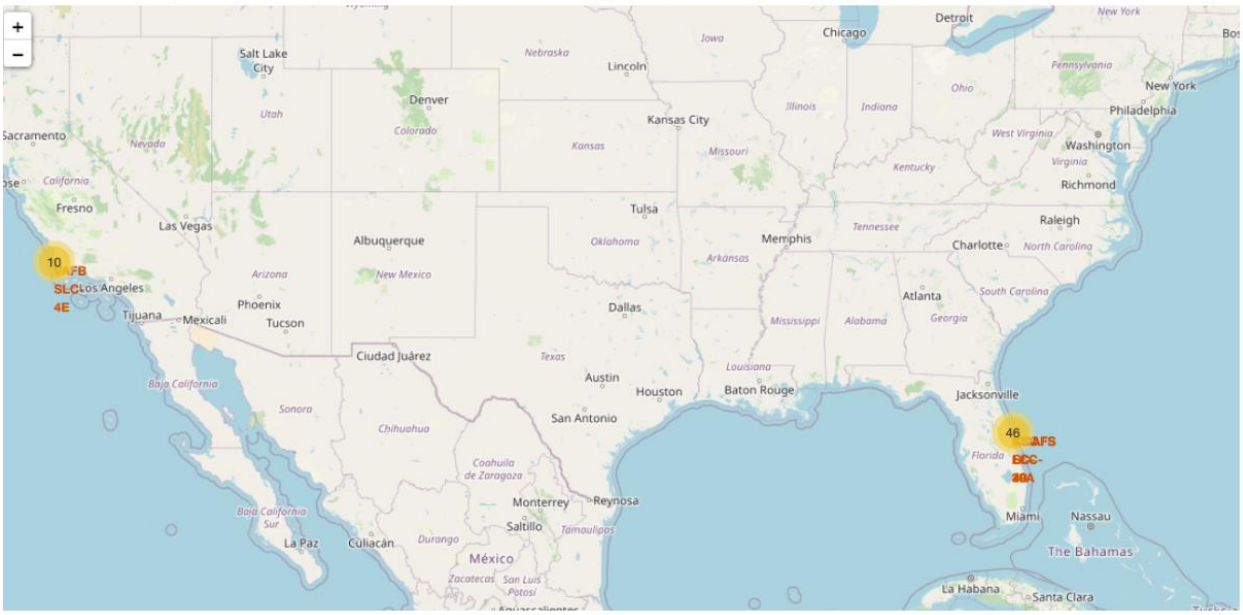
```
# Apply a function to check the value of `class` column
# If class=1, marker_color value will be green
# If class=0, marker_color value will be red
```

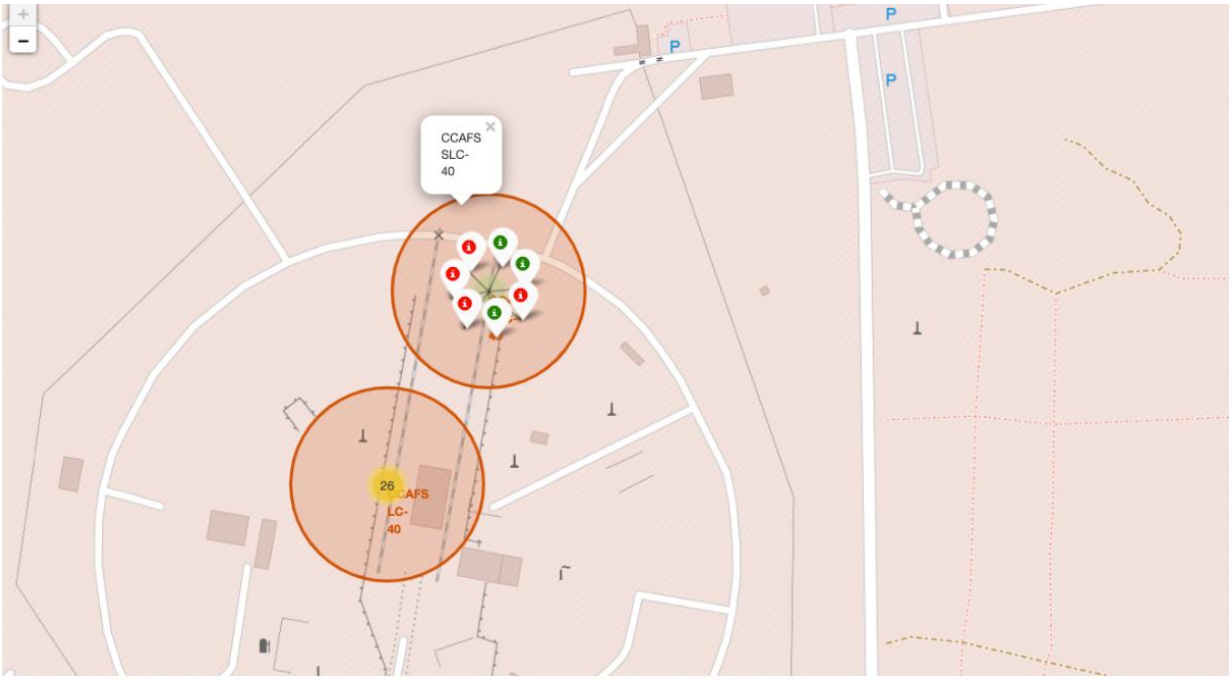
TODO: For each launch result in spacex_df data frame, add a folium.Marker to marker_cluster

```
# Add marker_cluster to current site_map
site_map.add_child(marker_cluster)


# Function to determine marker color based on class
def get_marker_color(class_value):
    if class_value == 1:
        return 'green'
    else:
        return 'red'


# for each row in spacex_df data frame
# create a marker for each launch site, and add it to the marker_cluster
# on mouse click, on each marker, show the Launch Site label
for index, record in spacex_df.iterrows():
```

```
    # Create a marker with the launch site name as a label
    marker = folium.Marker(
        location=[record['Lat'], record['Long']],
        # Create an icon with a color based on the class
        icon=folium.Icon(color='white', icon_color=get_marker_color(record['class'])),
        popup=record['Launch Site'] # Display the launch site name on click
    )
    marker_cluster.add_child(marker)
```

site_map





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)

Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on the map

formatter = "function(num) {return L.Util.formatNum(num, 5);};"

mouse_position = MousePosition(

position='topright',

separator=' Long: ',

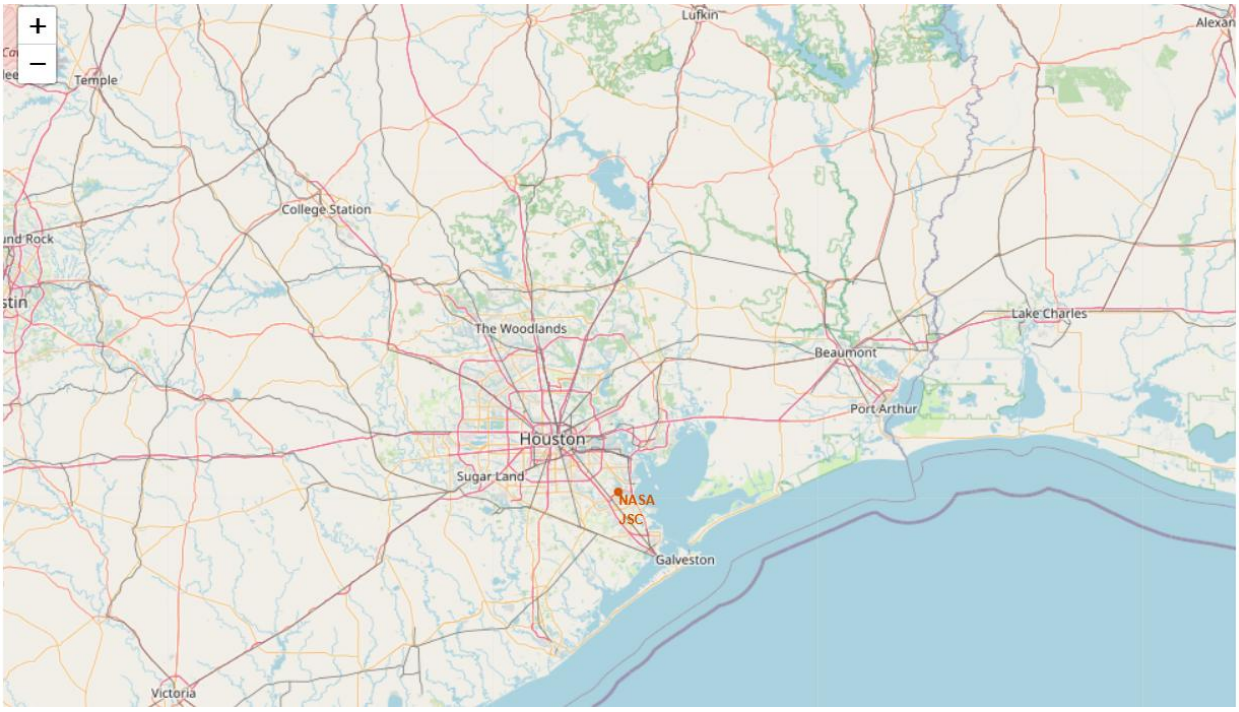
empty_string='NaN',

lng_first=False,

num_digits=20,

prefix='Lat:',

```
    lat_formatter=formatter,
    lng_formatter=formatter,
)

site_map.add_child(mouse_position)
site_map
```



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.

```
from math import sin, cos, sqrt, atan2, radians


def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
```

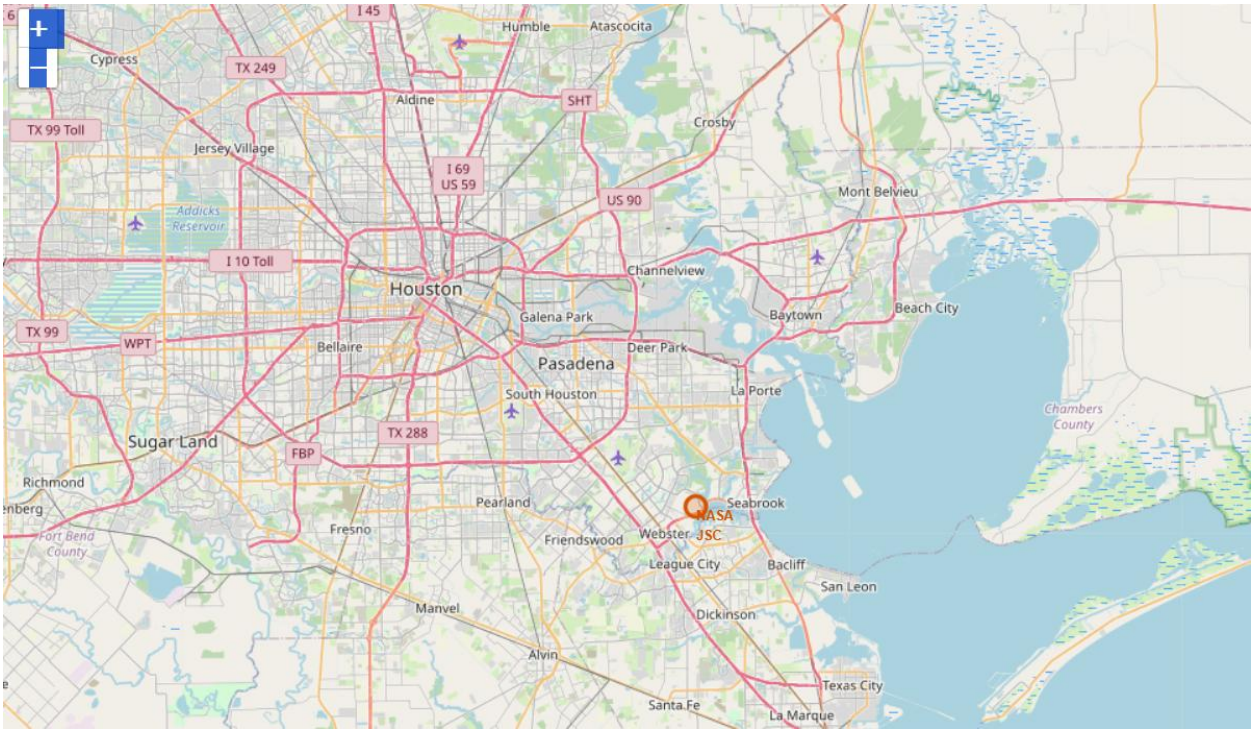
```
    R = 6373.0

    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.

```
# 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)

# Find the closest coastline point for each launch site and add a marker with the distance

# Approximate coastline coordinates near the launch sites (these are example coordinates, you might need more precise ones)
coastline_coordinates = [
```

```
    [28.56367, -80.57163],  # Near CCAFS
    [34.63343, -120.62503]  # Near VAFB
]

for index, site in launch_sites_df.iterrows():
    launch_site_lat = site['Lat']
    launch_site_lon = site['Long']
    launch_site_name = site['Launch Site']

    # Find the closest coastline point to the current launch site
    min_distance = float('inf')
    closest_coastline_lat = None
    closest_coastline_lon = None

    for coast_lat, coast_lon in coastline_coordinates:
        distance = calculate_distance(launch_site_lat, launch_site_lon, coast_lat, coast_lon)
        if distance < min_distance:
            min_distance = distance
            closest_coastline_lat = coast_lat
            closest_coastline_lon = coast_lon

    # Create a marker for the closest coastline point
    coastline_marker = folium.Marker(
        [closest_coastline_lat, closest_coastline_lon],
        icon=DivIcon(
            icon_size=(20,20),
```

```
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#115E67;"><b>%s</b></div>' % "Coastline",
      )
    )
    site_map.add_child(coastline_marker)

    # 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
    distance_marker = folium.Marker(
      [closest_coastline_lat, closest_coastline_lon],
      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(min_distance),
        )
      )
    site_map.add_child(distance_marker)

site_map
```



ODO: 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

Create a marker with distance to a closest city, railway, highway, etc.

Draw a line between the marker to the launch site

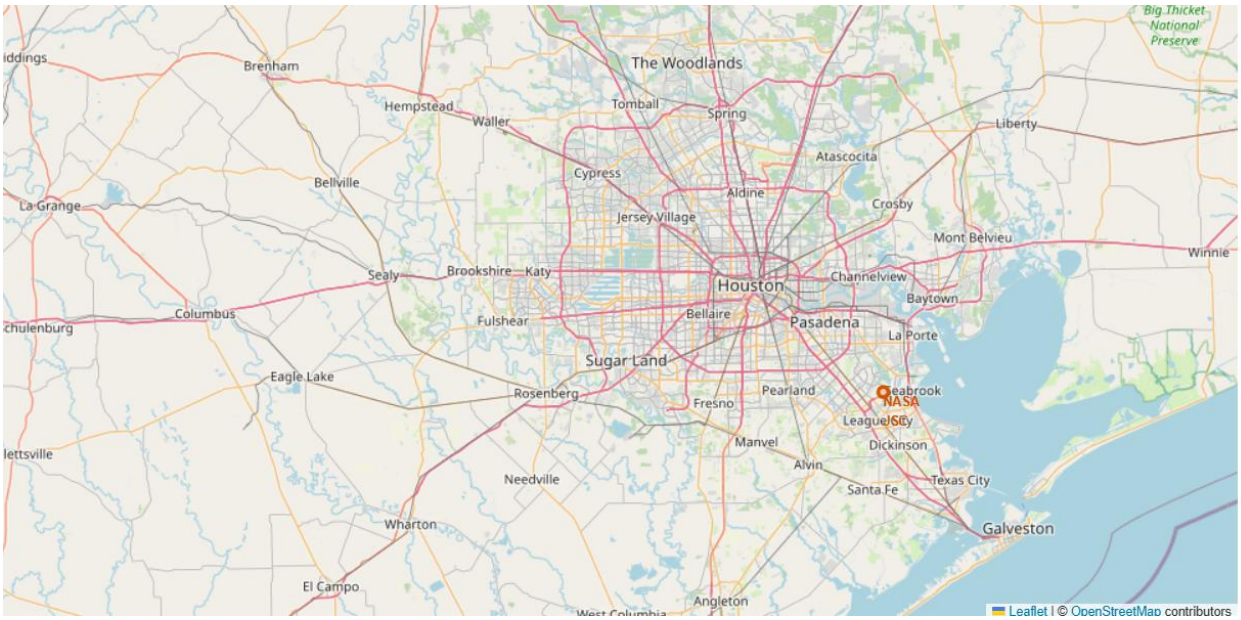
Define coordinates for a few cities, railways, and highways (example coordinates)

city_coordinates = {

```
    'Cape Canaveral City': [28.3898, -80.6030],
    'Lompoc': [34.6391, -120.4579]
}

railway_coordinates = {
    'Florida East Coast Railway': [28.5634, -80.5865],
    'Pacific Coast Railway (Historical)': [34.6317, -120.6214]
}

highway_coordinates = {
    'US-1 (Near CCAFS)': [28.5629, -80.5849],
    'I-5 (Near VAFB)': [34.6684, -120.5352]
}

# Function to find the closest point from a set of coordinates to a launch site
def find_closest_point(launch_lat, launch_lon, points):
    min_distance = float('inf')
    closest_point_coord = None
    closest_point_name = None
    for name, coord in points.items():
        distance = calculate_distance(launch_lat, launch_lon, coord[0], coord[1])
        if distance < min_distance:
            min_distance = distance
            closest_point_coord = coord
            closest_point_name = name
    return closest_point_coord, closest_point_name, min_distance
```

```
# Iterate through launch sites and find closest city, railway, and highway
for index, site in launch_sites_df.iterrows():
    launch_site_lat = site['Lat']
    launch_site_lon = site['Long']
    launch_site_name = site['Launch Site']

    # Find closest city
    closest_city_coord, closest_city_name, city_distance = find_closest_point(launch_site_lat, launch_site_lon, city_coordinates)
    # Find closest railway
    closest_railway_coord, closest_railway_name, railway_distance = find_closest_point(launch_site_lat, launch_site_lon, railway_coordinates)
    # Find closest highway
    closest_highway_coord, closest_highway_name, highway_distance = find_closest_point(launch_site_lat, launch_site_lon, highway_coordinates)

    # Add marker and line for closest city
    if closest_city_coord:
        folium.Marker(
            closest_city_coord,
            icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0),html='<div style="font-size: 12; color:#0000FF;"><b>%s</b></div>' % "{:10.2f} KM".format(city_distance),)
        ).add_to(site_map)
        folium.PolyLine(locations=[[launch_site_lat, launch_site_lon], closest_city_coord], weight=1, color='blue').add_to(site_map)
```

```
  # Add marker and line for closest railway
  if closest_railway_coord:
    folium.Marker(
      closest_railway_coord,
      icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0),html='<div style="font-size: 12; color:#FF0000;"><b>%s</b></div>' % "{:10.2f} KM".format(railway_distance),)
    ).add_to(site_map)
    folium.PolyLine(locations=[[launch_site_lat, launch_site_lon], closest_railway_coord], weight=1, color='red').add_to(site_map)

  # Add marker and line for closest highway
  if closest_highway_coord:
    folium.Marker(
      closest_highway_coord,
      icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0),html='<div style="font-size: 12; color:#00FF00;"><b>%s</b></div>' % "{:10.2f} KM".format(highway_distance),)
    ).add_to(site_map)
    folium.PolyLine(locations=[[launch_site_lat, launch_site_lon], closest_highway_coord], weight=1, color='green').add_to(site_map)

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.

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