SpaceX Falcon 9 First Stage Landing Prediction

Interactive Visual Analytics with Folium

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

Launch Sites Locations Analysis 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.

```
import folium
import pandas as pd
from folium.plugins import MarkerCluster
from folium.plugins import MousePosition
from folium.features import DivIcon
```

TASK 1: Mark all launch sites on a map

```
# 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

# Start location is NASA Johnson Space Center
nasa_coordinate = [29.559684888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=10)
```

```
# Create a red 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 red 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)
```

```
<folium.folium.Map at 0x6f935b8>
```

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

You should find a small red 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='%s' % 'label',))

```
# Initialize map
center_coords = [launch_sites_df[['Lat']].max() * 0.5 +
launch_sites_df[['Lat']].min() * 0.5, launch_sites_df[['Long']].max()
* 0.5 + launch_sites_df[['Long']].min() * 0.5]
site_map = folium.Map(location=center_coords, zoom_start=4)

# For each launch site, add a Circle object based on its coordinate
(Lat, Long) values. In addition, add Launch site name as a popup label
for _, launch_site in launch_sites_df.iterrows():
    coord = [launch_site['Lat'], launch_site['Long']]
    ls_name = launch_site['Launch Site']
    # print(ls_name, coord)
    # Create a blue circle at site's coordinate with a popup label
showing its name
    circle = folium.Circle(coord, radius=1000, color='#0054d3',
fill=True).add_child(folium.Popup(ls_name))
    # Create a blue circle at site's coordinate with a icon showing
its name
    marker = folium.map.Marker(
        coord,
        # Create an icon as a text label
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#0054d3;"><b>
%s</b></div>' % ls_name,
            )
        )
    site_map.add_child(circle)
    site_map.add_child(marker)

site_map  # display map
```

```
/lib/python3.12/site-packages/folium/utilities.py:94: FutureWarning:
Calling float on a single element Series is deprecated and will raise
a TypeError in the future. Use float(ser.iloc[0]) instead
  float(coord)
/lib/python3.12/site-packages/folium/utilities.py:100: FutureWarning:
Calling float on a single element Series is deprecated and will raise
a TypeError in the future. Use float(ser.iloc[0]) instead
  if math.isnan(float(coord)):
/lib/python3.12/site-packages/folium/utilities.py:102: FutureWarning:
Calling float on a single element Series is deprecated and will raise
a TypeError in the future. Use float(ser.iloc[0]) instead
  return [float(x) for x in coords]

<folium.folium.Map at 0x5f1a1f0>
```

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? no (+-30°N) Are all launch sites in very close proximity to the coast? yes

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().add_to(site_map)
```

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

```
# 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)
```

```
      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

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

# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch
was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
for index, record in spacex_df.iterrows():
    # TODO: Create and add a Marker cluster to the site map
    # marker = folium.Marker(...)

    folium.Marker(
        location=[record['Lat'], record['Long']],
        icon=folium.Icon(color=record['marker_color'],
icon_color=record['marker_color'],
        popup=record['class'])
    ).add_to(marker_cluster)

site_map
```

```
<folium.folium.Map at 0x691f220>
```

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

<folium.folium.Map at 0x691f220>
```

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:

```
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
# distance_coastline = calculate_distance(launch_site_lat,
launch_site_lon, coastline_lat, coastline_lon)

launch_site_lat, launch_site_lon = 28.563197, -80.576820
coastline_lat, coastline_lon = 28.56319, -80.56785

distance_coastline = calculate_distance(launch_site_lat,
launch_site_lon, coastline_lat, coastline_lon)
distance_coastline

0.8762983388668404
```

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

```
# 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(
   location=[coastline_lat, 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(distance_coastline),
       )
   )
```

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

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

<folium.folium.Map at 0x691f220>
```

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

```
city_lat, city_lon = 28.53, -81.38  # coords for Orlando

line2=folium.PolyLine(locations=[[launch_site_lat, launch_site_lon],
[city_lat, city_lon]], weight=3)
site_map.add_child(line2)

<folium.folium.Map at 0x691f220>
```

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?

Regarding Kennedy Space Center: Distance to highways, cities and railways is good enough, also regarding the fact that rockets will head towards the east.

Regarding Vandenberg AFB: The only larger place to the East of the AFB is Lompoc, CA.