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

Sricharan Nallan Chakravarthula

Space X Falcon 9 First Stage Landing Prediction

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

- Mark all launch sites on a map
- Mark the success/failed launches for each site on the map
- Calculate the distances between a launch site to its proximities

Import Libraries

In [1]:
```
!pip3 install folium
!pip3 install wget
print("Installation complete.")
```

```
Requirement already satisfied: folium in /home/jupyterlab/conda/envs/pyth
on/lib/python3.7/site-packages (0.11.0)
Requirement already satisfied: numpy in /home/jupyterlab/conda/envs/pytho
n/lib/python3.7/site-packages (from folium) (1.21.6)
Requirement already satisfied: jinja2>=2.9 in /home/jupyterlab/conda/env
s/python/lib/python3.7/site-packages (from folium) (3.1.2)
Requirement already satisfied: requests in /home/jupyterlab/conda/envs/py
thon/lib/python3.7/site-packages (from folium) (2.28.1)
Requirement already satisfied: branca>=0.3.0 in /home/jupyterlab/conda/en
vs/python/lib/python3.7/site-packages (from folium) (0.6.0)
Requirement already satisfied: MarkupSafe>=2.0 in /home/jupyterlab/conda/
envs/python/lib/python3.7/site-packages (from jinja2>=2.9->folium) (2.1.
1)
Requirement already satisfied: charset-normalizer<3,>=2 in /home/jupyterl
ab/conda/envs/python/lib/python3.7/site-packages (from requests->folium)
(2.1.1)
Requirement already satisfied: certifi>=2017.4.17 in /home/jupyterlab/con
da/envs/python/lib/python3.7/site-packages (from requests->folium) (2022.
9.24)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/jupyterlab/
conda/envs/python/lib/python3.7/site-packages (from requests->folium) (1.
26.13)
Requirement already satisfied: idna<4,>=2.5 in /home/jupyterlab/conda/env
s/python/lib/python3.7/site-packages (from requests->folium) (3.4)
Requirement already satisfied: wget in /home/jupyterlab/conda/envs/pytho
n/lib/python3.7/site-packages (3.2)
Installation complete.
```

In [2]:
```
import folium
import wget
import pandas as pd
from folium.plugins import MarkerCluster
from folium.plugins import MousePosition
from folium.features import DivIcon
print("All libraries have been imported.")
```

```
All libraries have been imported.
```

Start Here

Task 1

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.

DataFrame

In [3]:
```
# Download and read the `spacex_launch_geo.csv`
spacex_csv_file = wget.download('https://cf-courses-data.s3.us.cloud-objec
spacex_df = pd.read_csv(spacex_csv_file)
spacex_df.head(2)
```

Out[3]:

	Flight Number	Date	Time (UTC)	Booster Version	Launch Site	Payload	Payload Mass (kg)	Orbit	Customer	Lau Outc
0	1	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	F (parac
1	2	2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel o...	0.0	LEO (ISS)	NASA (COTS) NRO	F (parac

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

In [4]:
```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longi
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[4]:

	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.

Simple Map

> **Note**
> Plot launch sites on map using CIRCLES (with POPUPS) and MARKERS

In [5]:
```
# Create lists of coordinates, names and colors
launch_site_map_coordinates = launch_sites_df[['Lat', 'Long']].values.toli
launch_site_map_location_names = list(launch_sites_df['Launch Site'])
launch_site_map_circle_colors = ['#ff8800', '#007799', '#55dd00', '#aa00aa
```

In [6]:

```
# Start location is NASA Johnson Space Center
nasa_coordinates = [29.559684888503615, -95.0830971930759]
launch_site_map = folium.Map(location=nasa_coordinates, zoom_start=11, hei
launch_site_map
```

Out[6]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [7]:

```
# Create a blue circle at NASA Johnson Space Center's coordinate
circle = folium.Circle(nasa_coordinates, radius=1000, color='#0055dd', fil
launch_site_map.add_child(circle)

# MARKER - Create an icon showing an abbreviation of the name 'NASA Johnso
marker = folium.map.Marker(nasa_coordinates, icon=DivIcon(icon_size=(10,10
                                                          icon_anchor=(-20,
                                                          html='<div style=
                                                         )
    )
launch_site_map.add_child(marker)
```

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

Mark Launch Sites with Circles

In [8]:
```
# Create circles at launch site locations with popup labels
for index,i in enumerate(launch_site_map_coordinates):
    # Set Variables
    circle_coordinates = i
    circle_popup_name = launch_site_map_location_names[index]
    circle_color = launch_site_map_circle_colors[index]

    # Add Circle
    circle = folium.Circle(circle_coordinates, radius=50, color=circle_col
    launch_site_map.add_child(circle)

    # Add Marker
    marker_name = launch_site_map_location_names[index]
    marker = folium.map.Marker(circle_coordinates, icon=DivIcon(icon_size=
                                                                icon_ancho
                                                                html=f'<di
                                                               )
    )
    launch_site_map.add_child(marker)

launch_site_map
```

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

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

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

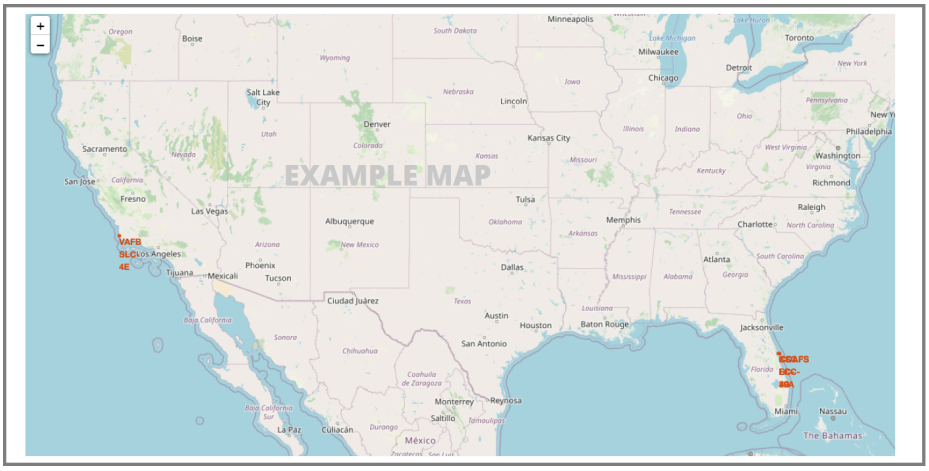
An example of folium.Circle:

```
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', ))
```

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.

Task 2

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

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)`

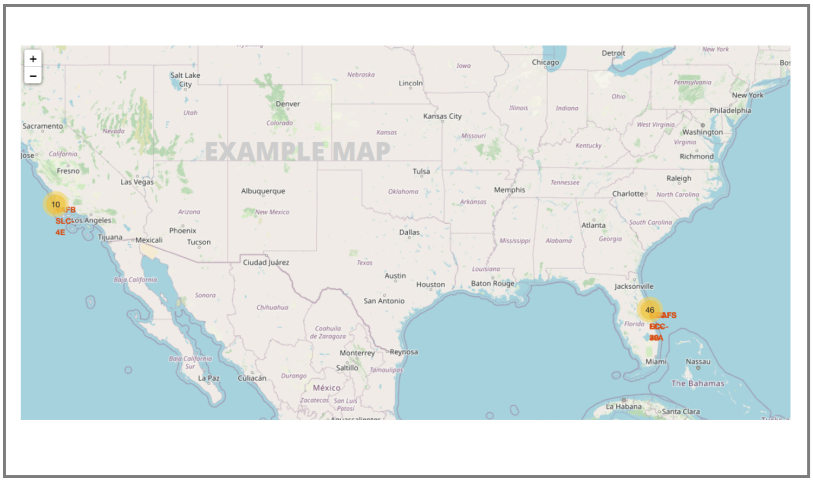
Note that a launch only happens in one of the four launch sites, which means many launch records will have the exact same coordinate. Marker clusters can be a good way to simplify a map containing many markers having the same coordinate.

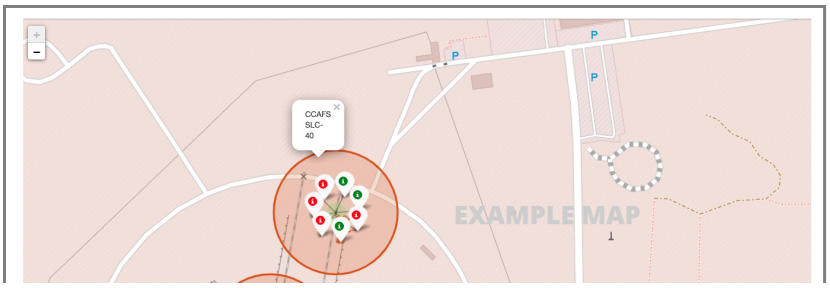
Let's first create a `MarkerCluster` object

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

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

Your updated map may look like the following screenshots:





Mark Successes and Failures

In [9]:
```
# View the dataframe
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

In [10]:

```
# Create MarkerCluster using plugins method for adding success/failure mar
from folium import plugins

# Start  with a clean copy of the map
launch_site_map_for_marker_cluster = folium.Map(location=nasa_coordinates,

# Create a marker cluster object for the marker_color category in the data
marker_cluster = plugins.MarkerCluster().add_to(launch_site_map_for_marker

# Create dictionaries and lists for use in the for loop
class_list = list(spacex_df['class'])
outcome_dict = {0:'Failure',1:'Success'}
colors_dict = {0:'#990000',1:'#00aa00'}

# loop through the dataframe and add each data point to the mark cluster
for lat, lng, label in zip(spacex_df.Lat, spacex_df.Long, class_list):
    html_label = f'<div style="padding:0 4px; border-radius:4px; font-size
    folium.Marker(
        location=[lat, lng],
        icon=None,
        popup=html_label,
    ).add_to(marker_cluster)

# Show map
launch_site_map_for_marker_cluster
```

Out[10]: Make this Notebook Trusted to load map: File -> Trust Notebook

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

TASK 3: Calculate the distances from 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

In [11]:
```
# Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on
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,
)

launch_site_map.add_child(mouse_position)
launch_site_map
```

Out[11]: 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:

Mark Distances

In [12]:
```
# Function for calculating distance
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 [13]:
```
# CCAFS LC-40 Launch Site Coordinates
launch_site_lat = 28.562302
launch_site_lon = -80.577356
```

In [14]:
```
# Coordinates of the closet COASTLINE
coastline_lat = 28.56321
coastline_lon = -80.56801
distance_coastline = calculate_distance(launch_site_lat, launch_site_lon,
print(f'Distance to Coastline: {round(distance_coastline,2)} km')

# Create and add a folium.Marker on the selected point on the map
distance_marker = folium.Marker(
    [coastline_lat, coastline_lon],
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(100,-10),
        html=f'<div style="font-size:12px; color:#222; font-weight:bold; w
        )
    ).add_to(launch_site_map)
```

```
Distance to Coastline: 0.92 km
```

In [15]:

```
# Coordinates of the closet RAIL LINE
rail_line_lat = 28.57209
rail_line_lon = -80.58520
distance_rail_line = calculate_distance(launch_site_lat, launch_site_lon,
print(f'Distance to Rail Line: {round(distance_rail_line,2)} km')

# Create and add a folium.Marker on the selected point on the map
distance_marker = folium.Marker(
    [rail_line_lat, rail_line_lon],
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(-40,-20),
        html=f'<div style="font-size:12px; color:#222; font-weight:bold; w
        )
    ).add_to(launch_site_map)
```

```
Distance to Rail Line: 1.33 km
```

In [16]:

```
# Coordinates of a point on the ROAD circling the launch site
road_lat = 28.561695
road_lon = -80.575553
distance_road = calculate_distance(launch_site_lat, launch_site_lon, road_
print(f'Distance to Road: {round(distance_road,2)} km')

# Create and add a folium.Marker on the selected point on the map
distance_marker = folium.Marker(
    [road_lat, road_lon],
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(10,0),
        html=f'<div style="font-size:12px; color:#222; font-weight:bold; w
        )
    ).add_to(launch_site_map)
```

```
Distance to Road: 0.19 km
```

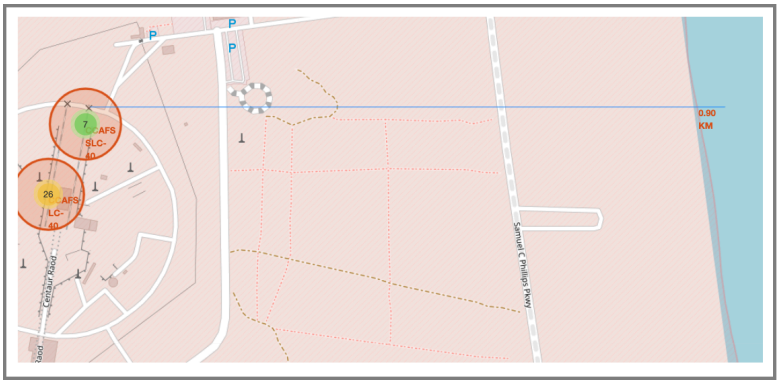
Add Lines to Map

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

In [17]:
```
# Create a `folium.PolyLine` object using the coordinates
coastline_coordinates = [(launch_site_lat, launch_site_lon), (coastline_la
rail_line_coordinates = [(launch_site_lat, launch_site_lon), (rail_line_la
road_coordinates = [(launch_site_lat, launch_site_lon), (road_lat, road_lo

coastline_line = folium.PolyLine(locations=coastline_coordinates, weight=2
rail_line_line = folium.PolyLine(locations=rail_line_coordinates, weight=2
road_line = folium.PolyLine(locations=road_coordinates, weight=2, color=la

launch_site_map.add_child(coastline_line)
launch_site_map.add_child(rail_line_line)
launch_site_map.add_child(road_line)
```

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

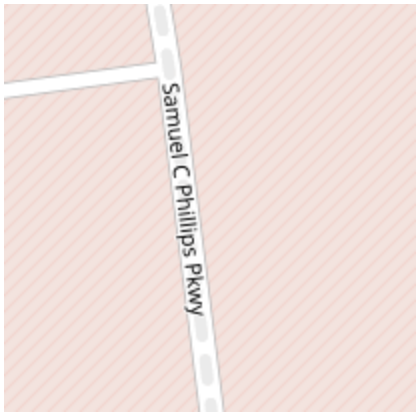
Your updated map with distance line should look like the following screenshot:



TODO: Similarly, you can draw a line betwee a launch site to its closest city, 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:

A highway map symbol may look like this:



A city map symbol may look like this:

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

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.

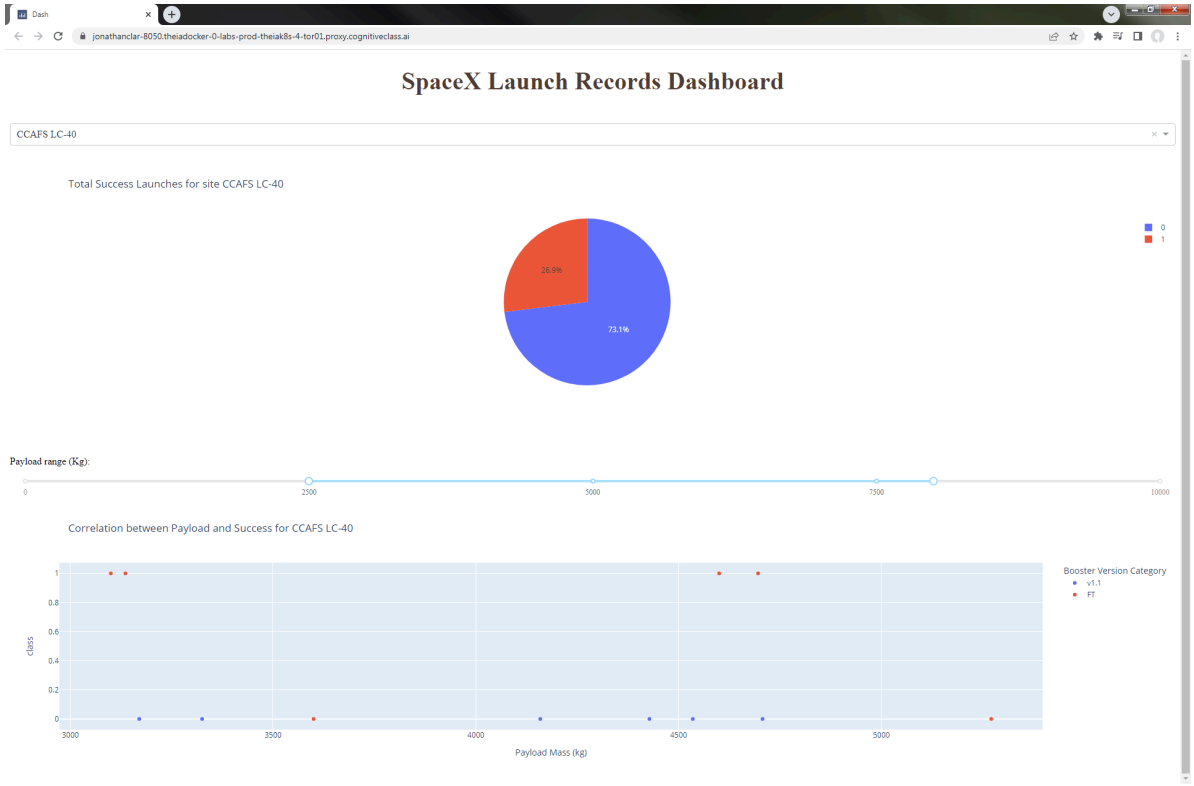
Create Dashboard with Plotly Dash

Python File Code:
2022-12-17
JonathanMClark_spacex_dash_app.py

In [ ]:

```
# Import required libraries
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output
import plotly.express as px

# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("spacex_launch_dash.csv")
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()

def assign_launch_outcome(launch_outcome):
    if launch_outcome == 1:
        return 'success'
    else:
        return 'failure'

# Create a dash application
app = dash.Dash(__name__)

# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'col
                                               'font-size': 40}),
                                html.Br(),

                                # _____ DROPDOWN SELECTOR _____
                                # TASK 1: Add a dropdown list to enable La
                                # The default select value is for ALL site
                                dcc.Dropdown(
                                    id='site-dropdown',
                                    options=[
                                        {'label': 'All Sites', 'value': 'A
                                        {'label': 'CCAFS LC-40', 'value':
                                        {'label': 'CCAFS SLC-40', 'value':
                                        {'label': 'KSC LC-39A', 'value': '
                                        {'label': 'VAFB SLC-4E', 'value':
                                    ],
                                    value='ALL',
                                    placeholder="Select a Launch Site here
                                    searchable=True
                                ),
                                html.Br(),

                                # _____ PIE CHART _____
                                # TASK 2: Add a pie chart to show the tota
                                # If a specific launch site was selected,
                                html.Div(dcc.Graph(id='success-pie-chart')
                                html.Br(),

                                # _____ RANGE SLIDER _____
                                html.P("Payload range (Kg):"),
                                # TASK 3: Add a slider to select payload r
                                dcc.RangeSlider(
                                    id='payload-slider',
```

```
                                    min=0,
                                    max=10000,
                                    step=1000,
                                    marks={0:'0', 2500:'2500', 5000:'5000'
                                    value=[min_payload, max_payload]
                                ),

                                # _____ SCATTER CHART _____
                                # TASK 4: Add a scatter chart to show the
                                html.Div(dcc.Graph(id='success-payload-sca
            ])

# _____ PIE CHART - CALLBACK FUNCTION _____
# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart
@app.callback(  Output(component_id='success-pie-chart', component_property
                Input(component_id='site-dropdown', component_property='va

def get_pie_chart(entered_site):
    filtered_df = spacex_df.loc[spacex_df['Launch Site'] == entered_site]
    total_launch_count = filtered_df['class'].count()
    successful_launch_count = filtered_df['class'].sum()
    failed_launch_count = total_launch_count - successful_launch_count
    dict_test = {'names':[0,1], 'values':[failed_launch_count,successful_l
    new_df = pd.DataFrame(dict_test)
    values = list(new_df['values'])
    names = list(new_df['names'])
    title = f'Total Success Launches for site {entered_site} '
    if entered_site == 'ALL':
        fig = px.pie(spacex_df, values='class', names='Launch Site', title
        return fig
    else:
        fig = px.pie(filtered_df, values=values, names=names, title=title)
        return fig

# _____ SCATTERPLOT - CALLBACK FUNCTION _____
# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as input
@app.callback(  Output(component_id='success-payload-scatter-chart', compo
                Input(component_id='site-dropdown', component_property='va
                Input(component_id='payload-slider', component_property='v

def get_scatter_plot(entered_site, slider_range_list):
    min_value = slider_range_list[0]
    max_value = slider_range_list[1]
    filtered_df = spacex_df.loc[(spacex_df['Launch Site'] == entered_site)
    title = f'Correlation between Payload and Success for {entered_site} '
    if entered_site == 'ALL':
        fig = px.scatter(spacex_df, x='Payload Mass (kg)', y='class', colo
        return fig
    else:
        fig = px.scatter(filtered_df, x='Payload Mass (kg)', y='class', co
        return fig

# Run the app
if __name__ == '__main__':
    app.run_server()
```


Dash
jonathanclar-8050.theiadocker-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai
SpaceX Launch Records Dashboard
CCAFS LC-40
Total Success Launches for site CCAFS LC-40
26.9%
73.1%
0
1
Payload range (Kg):
0
2500
5000
7500
10000
Correlation between Payload and Success for CCAFS LC-40
class
Payload Mass (kg)
3000
3500
4000
4500
5000
Booster Version Category
v1.1
FT

End Here