IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

AKASH S DASS
13-07-2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Data Collected
- Normalized and Analyzed
- EDA and Visualized
- Launch site Payload and many combinations of relations Analyzed
- Launch Sites Analyzed to find relationships between Success and Failure
- Machine Learning Finding the Best Hyperparameter for SVM ,Classification , Trees and Logistic Regression .

Introduction

- The Project is collecting and analyzing data and Predicting , finding the best hyperparameter of the model that is accurate
- Is there a link between the Launch Location and the payload or rocket and payload that defone the success or failure of the launch

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The data was collected by requesting SpaceX API using requests
- Perform data wrangling
  - Data was checked for null values and Missing values
- Perform exploratory data analysis (EDA) using visualization and SQL
- Yes it was done
- Perform interactive visual analytics using Folium and Plotly Dash
- Yes it was done
- Perform predictive analysis using classification models
  - Train test split was used and classification was done

Data Collection

- Describe how data sets were collected.
- Libraries were installed then defined the functions that extracts the launch data and columns rocket , launch pad , payload , cores .
- Then Requested the Rocket launch data from SpaceX using Requests
- Then converted the data to a Pandas Data Frame
- Filtered the dataframe to include Falcon 9 Launches
- Checked for null values and missind values
- Replaced missing values with mean value of payloadmass column

Data Collection – SpaceX API

Link to Git Hub

- IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT-/IBM DATA SCIENCE CAPSTONE PROJECT SPACEX FILE/1 spacex-data-collection-api.ipynb at main · gitAkashDass/IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT- · GitHub

- Libraries were installed then defined the functions that extracts the launch data and columns rocket , launch pad , payload , cores .
- Then Requested the Rocket launch data from SpaceX using Requests
- Then converted the data to a Pandas Data Frame
- Filtered the dataframe to include Falcon 9 Launches
- Checked for null values and missind values
- Replaced missing values with mean value of payloadmass column

Data Collection - Scraping

- Link to github
- IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT-/IBM DATA SCIENCE CAPSTONE PROJECT SPACEX FILE/2 webscraping.ipynb at main · gitAkashDass/IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT- · GitHub

1. Imported Beautifulsoup , requests and all the necessary liabraryes
2. Created a Beautifulsoup object from html response
3. Extracted all the data and then extracted all the columns and headsrs
4. Created a Data Frame by parsing the launch html tables

Data Wrangling

- Imported all the libraries and defined the functions
- Read the data in the data frame and looked at the head using df.head(10)
- Checked for shape and null values checked data types
- Calculated the number id launch sites CCAFS SLC 40 (55) , KSC LC 39A (22) , VAFB SLC 4E (13)
- Calculated the number and occurrences of each orbit
- Calculated the number and occurrence of mission outcome per orbit type
- Created a landing outcome label from outcome column
- Link : IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT-/IBM DATA SCIENCE CAPSTONE PROJECT SPACEX FILE/3 spacex-data_wrangling.ipynb at main · gitAkashDass/IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT- · GitHub

EDA with Data Visualization

- sns.catplot was used to measure the flight number and payload variables to check its effect on launch outcome.
- plt.scatter was used to find relationship between flight number and launch site
- Sns.scatter plot was used for payload vs launch site
- sns.catplot was used to flight number vs launch site
- Sns.boxplot , many more were used refer to the link
- IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT-/IBM DATA SCIENCE CAPSTONE PROJECT SPACEX FILE/5 eda-dataviz.ipynb.ipynb at main · gitAkashDass/IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT- · GitHub

EDA with SQL

- Task 1 was to Display names of unique launch site in space mission solution was %sql SELECT DISTINCT launch_site FROM SPACEXBL
- Task 2 was display 5 records where launch sites begain with “CCA” solution was %sql SELECT * FROM SPACEXBL WHERE launch_site LIKE ‘CCA%’ LIMIT(5)
- Task 3 was display the total payload mass carried by boosters launched by NASA (CRS) solution %sql SELECT SUM(PAYLOAD_MASS_KG_) AS total_payload_mass FROM SPACEXBL WHERE customer = ‘NASA (CSS)’
- Task 4 display average payload mass carried by booster version F9 v1.1 solution %sql SELECT AVG(PAYLOAD_MASS_KG_) AS average_payload_mass FROM SPACEXBL WHERE Booster_version = ‘F9 v1.1’ (CHECK THE LINK BELOW FOR MORE)
- IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT-/IBM DATA SCIENCE CAPSTONE PROJECT SPACEX FILE/4 EDA withsql-coursera_sqllite.ipynb at main · gitAkashDass/IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT- · GitHub

Build an Interactive Map with Folium

- I created a Map object, with an initial center location to be NASA Johnson Space Center at Houston. Then I used folium.Circle to add a highlighted circle area with a text label on a specific coordinate ie NASA Johnson Space Center.
- Then I added folium.Circle and folium.Marker for each launch site on the site map
- I created a launch result in spacex_df data frame, add a folium.Marker to marker_cluster
- Then I created MousePosition on the map to get coordinate for a mouse over a point on the map
- IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT-/IBM DATA SCIENCE CAPSTONE PROJECT SPACEX FILE/6_launch_site_location.jupyterlite.ipynb at main · gitAkashDass/IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT- · GitHub

Build a Dashboard with Plotly Dash

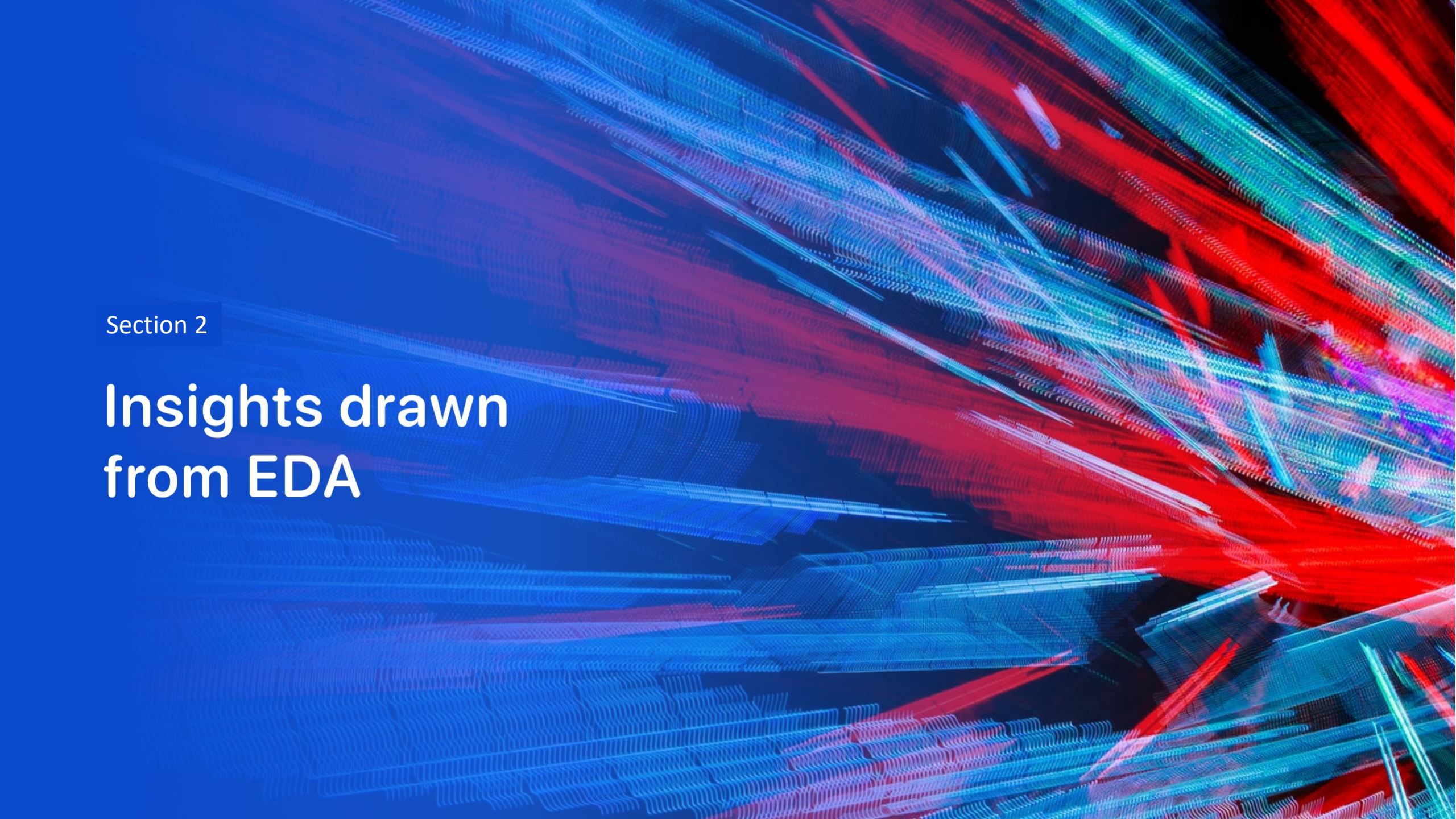
- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

- All the details are mentioned in the link below
- IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT-/IBM DATA SCIENCE CAPSTONE PROJECT SPACEX FILE/7 Machine_Learning_Prediction.ipynb at main · gitAkashDass/IBM-DATA-SCIENCE-CAPSTONE-SPACEX-PROJECT- · GitHub

Results

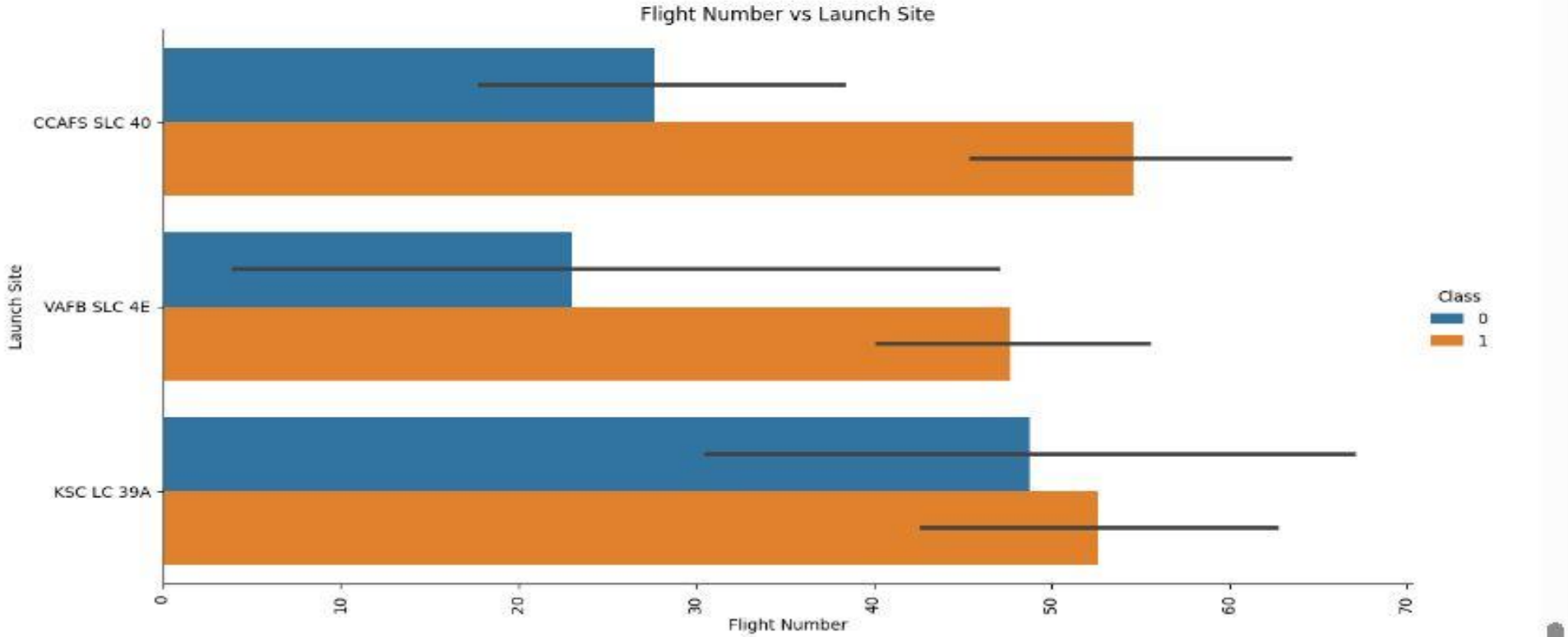
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

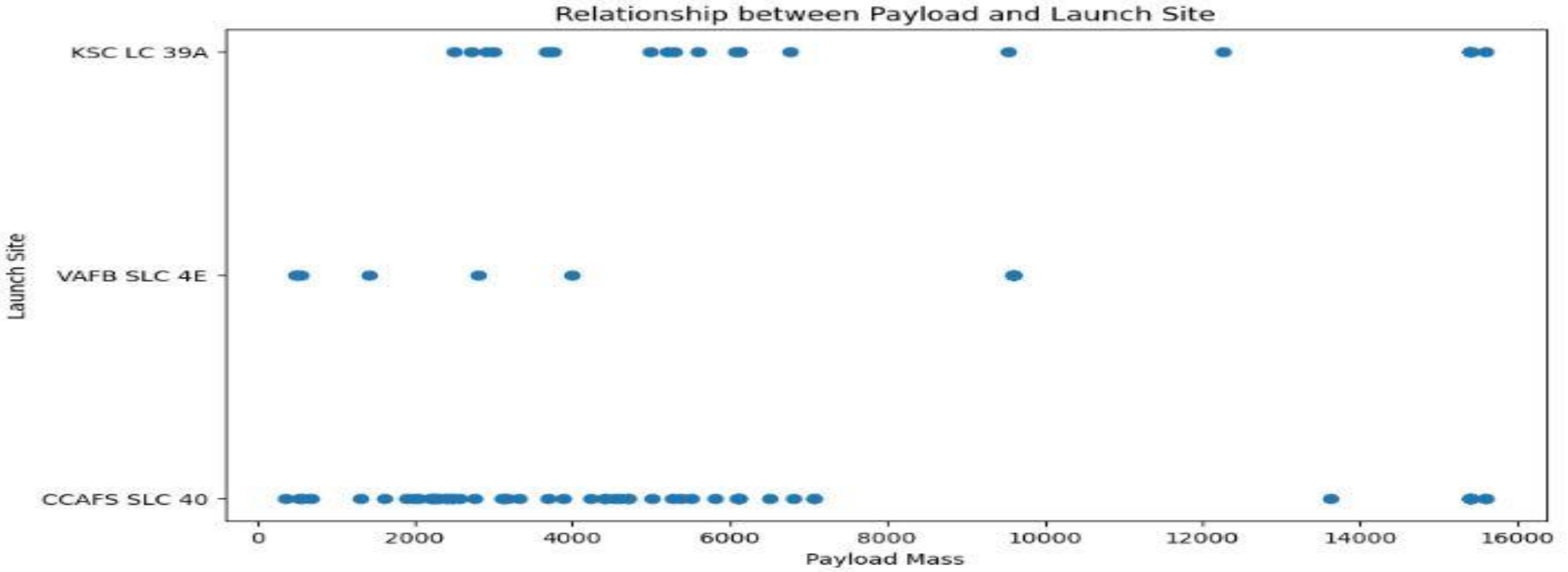
- Catplot

```
# Use catplot to plot FlightNumber vs LaunchSite
sns.catplot(x='FlightNumber', y='LaunchSite', hue='Class', data=df, kind='bar', height=6, aspect=2)
plt.title('Flight Number vs Launch Site')
plt.xlabel('Flight Number')
plt.ylabel('Launch Site')
plt.xticks(rotation=90)
plt.show()
```



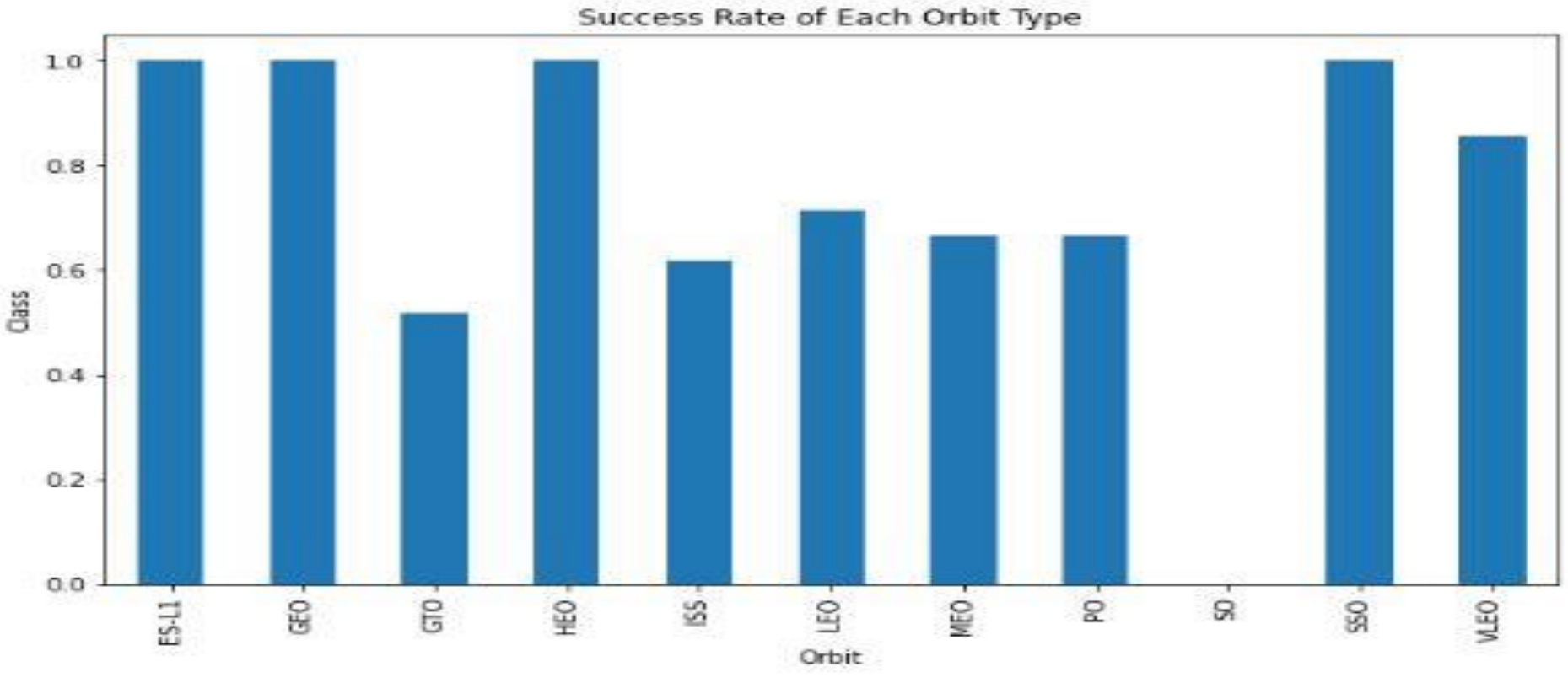
Payload vs. Launch Site

```
# Create a scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(df['PayloadMass'], df['LaunchSite'])
plt.xlabel('Payload Mass')
plt.ylabel('Launch Site')
plt.title('Relationship between Payload and Launch Site')
plt.show()
```



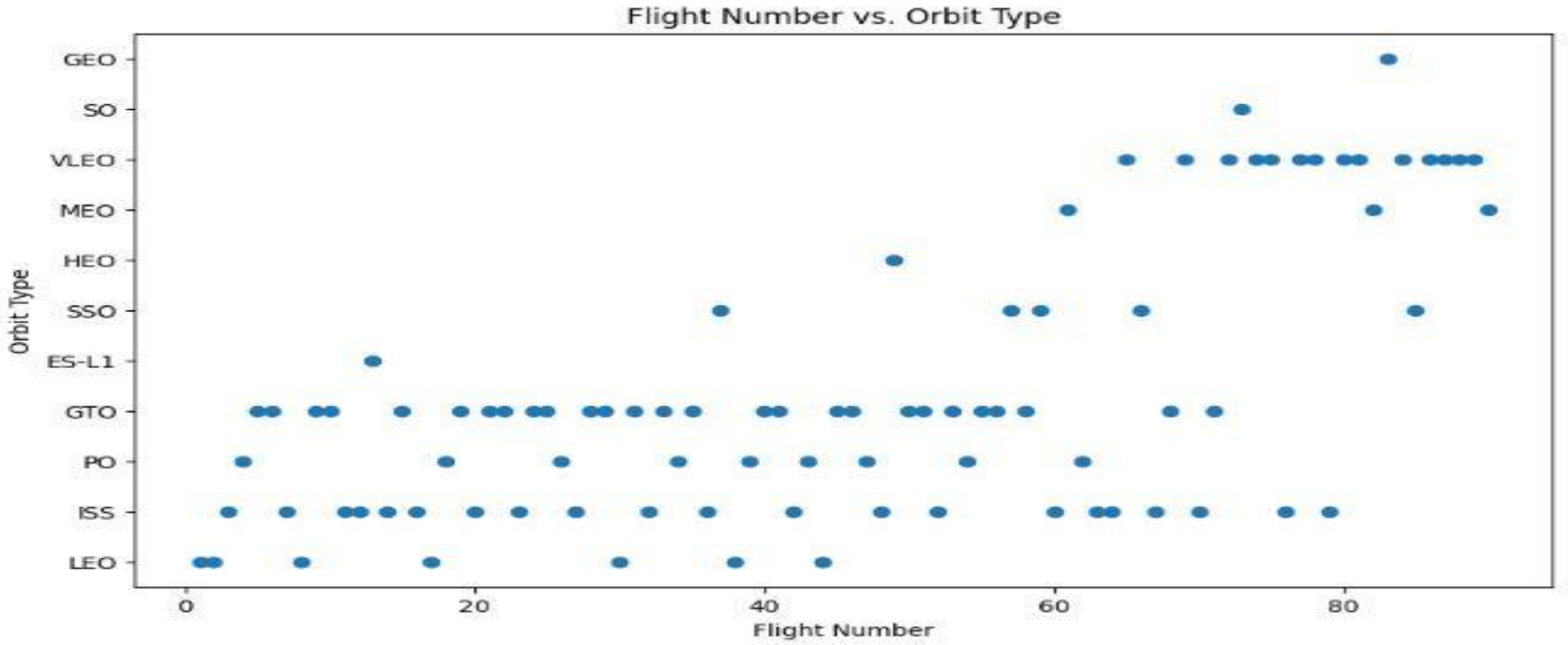
Success Rate vs. Orbit Type

```
# HINT use groupby method on Orbit column and get the mean of Class column
df_bar = df.groupby(['Orbit'])['Class'].mean()
df_bar.plot(kind='bar', figsize=(10, 6))

plt.xlabel('Orbit') # add to x-label to the plot
plt.ylabel('Class') # add y-label to the plot
plt.title('Success Rate of Each Orbit Type') # add title to the plot

plt.show()
```



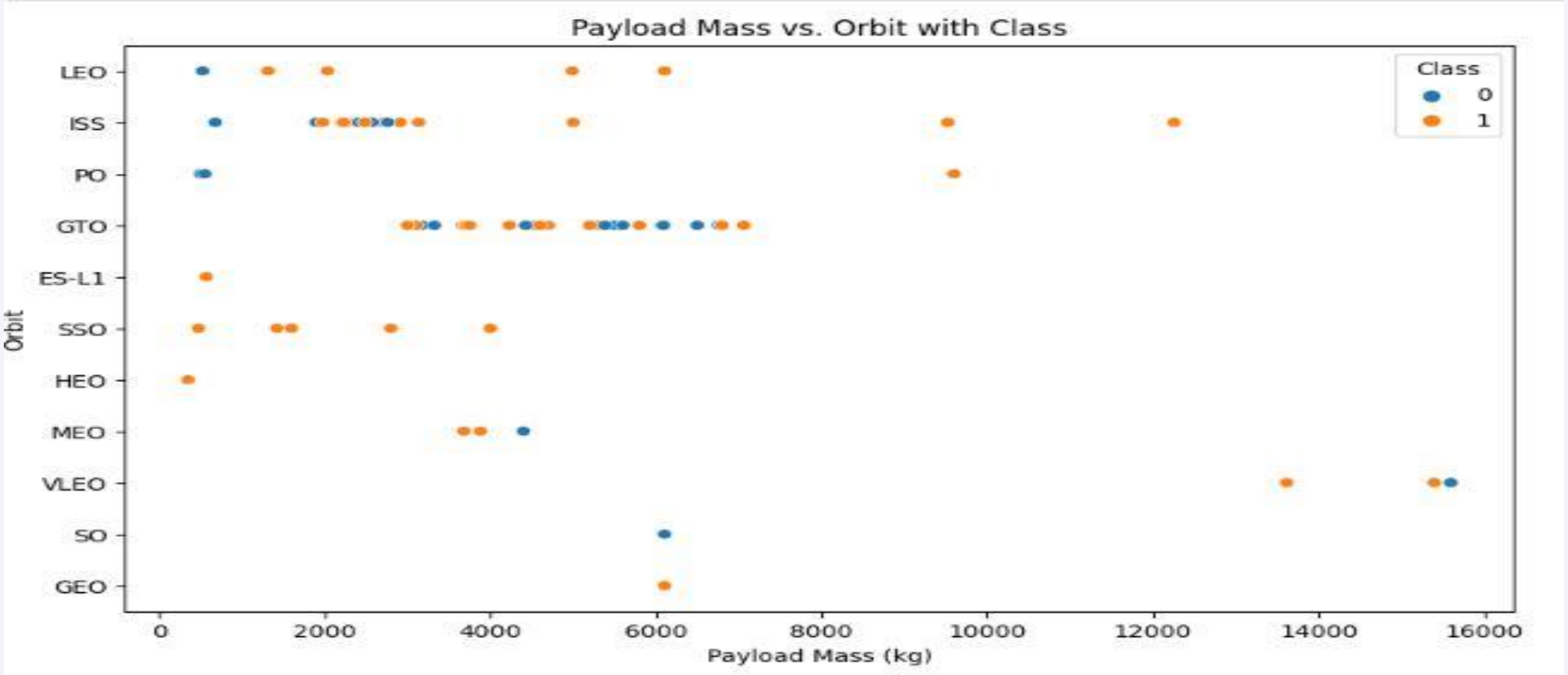
Flight Number vs. Orbit Type

```
### TASK  4: Visualize the relationship between FlightNumber and Orbit type
flight_orbit_data = df[['FlightNumber', 'Orbit']]
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.scatter(flight_orbit_data['FlightNumber'], flight_orbit_data['Orbit'])
plt.xlabel('Flight Number')
plt.ylabel('Orbit Type')
plt.title('Flight Number vs. Orbit Type')
plt.show()
```



Payload vs. Orbit Type

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='PayloadMass', y='Orbit', hue='Class')
plt.xlabel('Payload Mass (kg)')
plt.ylabel('Orbit')
plt.title('Payload Mass vs. Orbit with Class')
plt.legend(title='Class')
plt.show()
```



Launch Success Yearly Trend

- Show a line chart of yearly average success rate

- Show the screenshot of the scatter plot with explanations

All Launch Site Names

In [7]:
```
%sql SELECT DISTINCT launch_site FROM SPACEXTBL;
```

```
 * sqlite:///my_data1.db
Done.
```

Out[7]:

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

Launch Site Names Begin with 'CCA'

Display 5 records where launch sites begin with the string 'CCA'

```
[8]: %sql SELECT * FROM SPACEXTBL WHERE launch_site LIKE 'CCA%' LIMIT 5;
```

```
 * sqlite:///my_data1.db
Done.
```

[8]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parac
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parac
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No att
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No att
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No att

Total Payload Mass

Display the total payload mass carried by boosters launched by NASA (CRS)

In [9]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload_mass FROM SPACEXTBL WHERE customer = 'NASA (CRS)';
```

```
 * sqlite:///my_data1.db
Done.
```

Out[9]:

total_payload_mass
45596.0

Average Payload Mass by F9 v1.1

Display average payload mass carried by booster version F9 v1.1

In [10]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS average_payload_mass FROM SPACEXTBL WHERE Booster_Version = 'F9 v1.1';
```

```
 * sqlite:///my_data1.db
Done.
```

Out[10]:

average_payload_mass
2928.4

First Successful Ground Landing Date

In [14]:
```
#%sql SELECT Date FROM SPACEXTBL WHERE Landing_Outcome = 'Success' limit 1 ;
%sql SELECT MIN(Date) AS FirstSuccessfull_landing_date FROM SPACEXTBL WHERE Landing_Outcome LIKE 'Success (ground pad)'
```

```
 * sqlite:///my_data1.db
Done.
```

Out[14]:

FirstSuccessfull_landing_date
01/08/2018

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [26]:
```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_> 4000 AND PA
```

```
* sqlite:///my_data1.db
Done.
```

Out[26]:

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [27]:
```
#%sql SELECT COUNT(Mission_Outcome) FROM SPACEXTBL WHERE Mission_Outcome = 'Success'
%sql SELECT Mission_Outcome, COUNT(*) AS Total_Count FROM SPACEXTBL GROUP BY Mission_Outcome;
```

```
 * sqlite:///my_data1.db
Done.
```

Out[27]:

Mission_Outcome	Total_Count
None	898
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [15]:
```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_= (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

```
 * sqlite:///my_data1.db
Done.
```

Out[15]:

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

In [32]:

```
import sqlite3

# Establish a connection to the database
con = sqlite3.connect("my_data1.db")
cur = con.cursor()

# Execute the SQL query
cur.execute("""
    SELECT substr(Date, 4, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site
    FROM SPACEXTBL
    WHERE substr(Date, 7, 4) = '2015' AND Landing_Outcome LIKE '%Failure (drone ship)%'
""")

# Fetch all the records from the result set
records = cur.fetchall()

# Print the records
for record in records:
    print(record)
```

```
('10', 'Failure (drone ship)', 'F9 v1.1 B1012', 'CCAFS LC-40')
('04', 'Failure (drone ship)', 'F9 v1.1 B1015', 'CCAFS LC-40')
```

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

In [22]:
```
%sql SELECT Landing_Outcome, COUNT(Landing_Outcome) FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY
```

```
* sqlite:///my_data1.db
Done.
```

Out[22]:

Landing_Outcome	COUNT(Landing_Outcome)

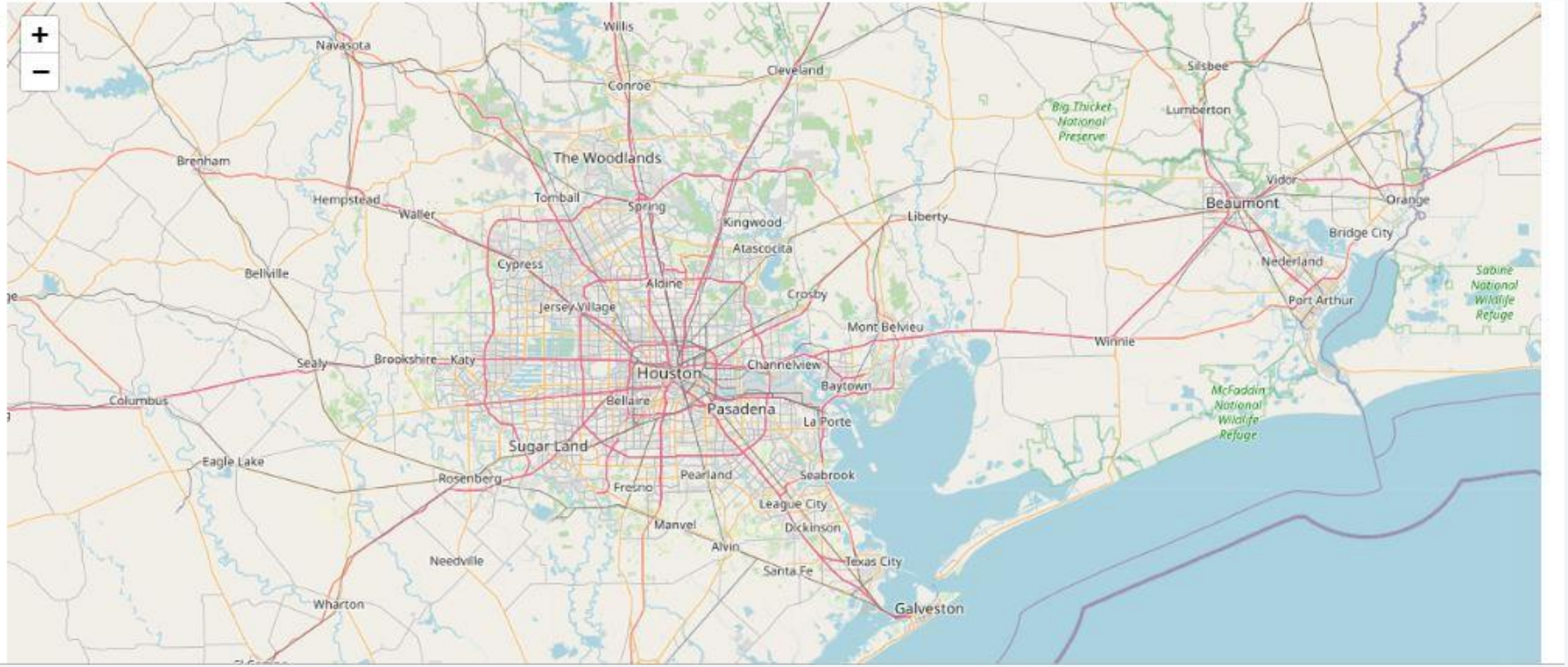
Section 3

Launch Sites Proximities Analysis

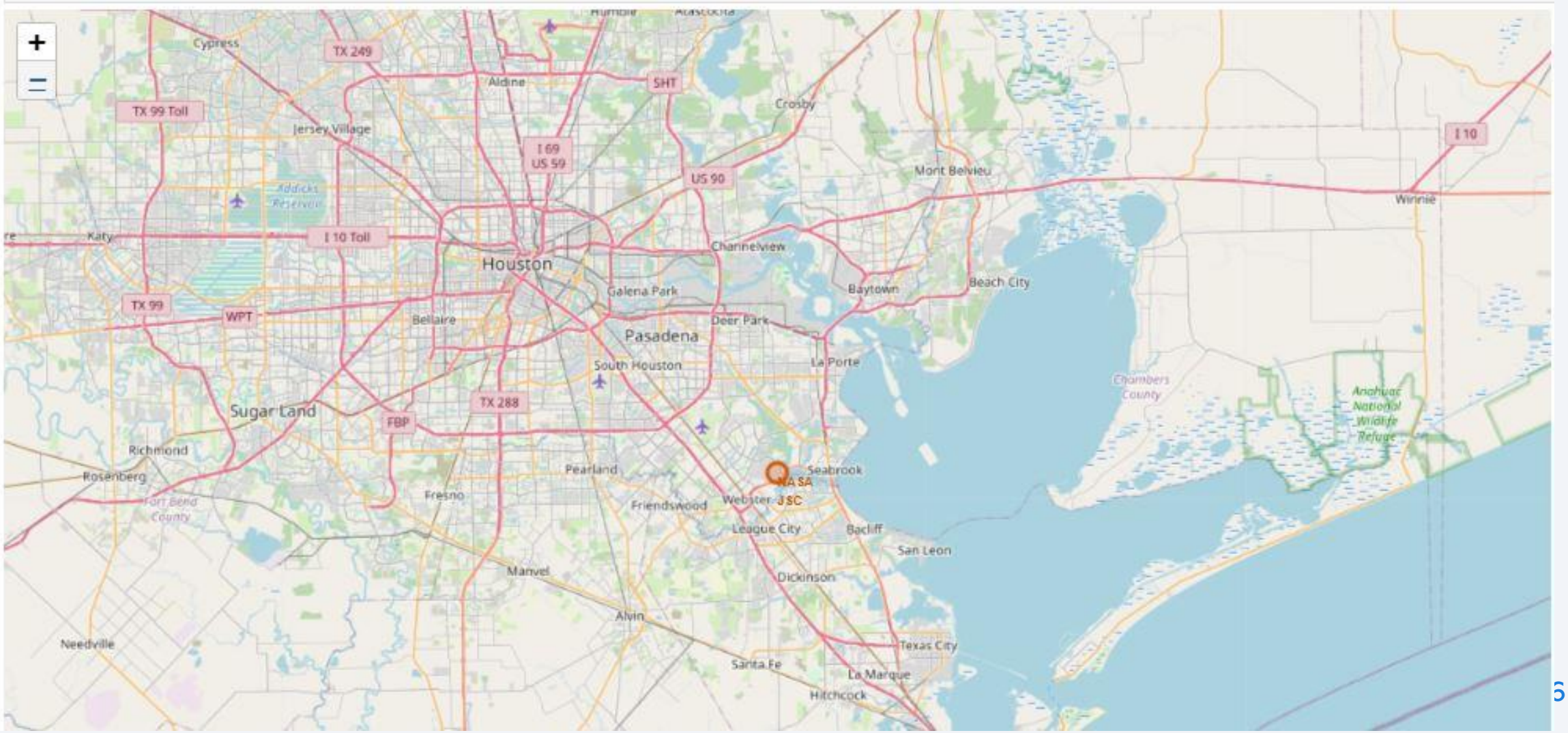
NASA Map Screenshot

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

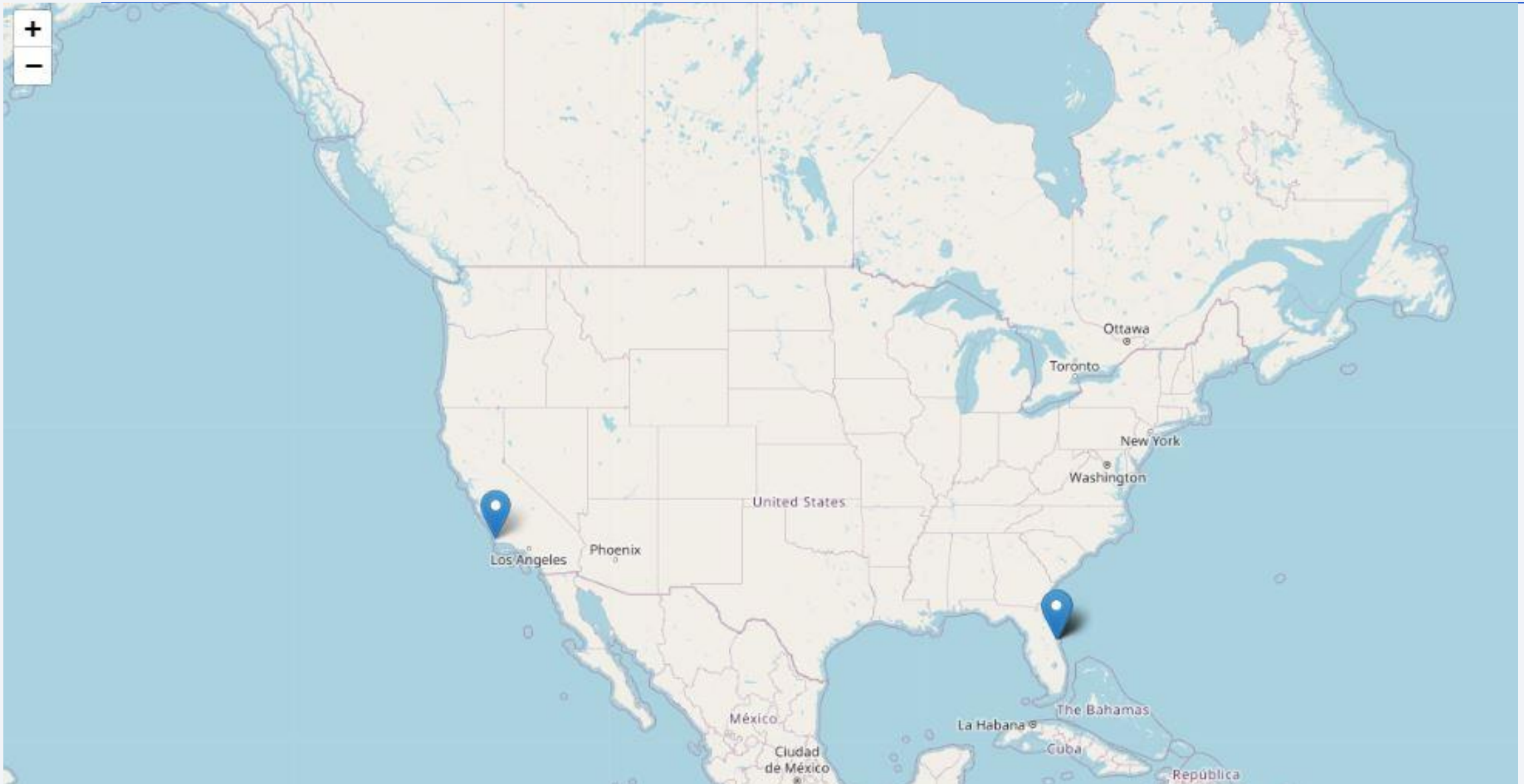
[9]:

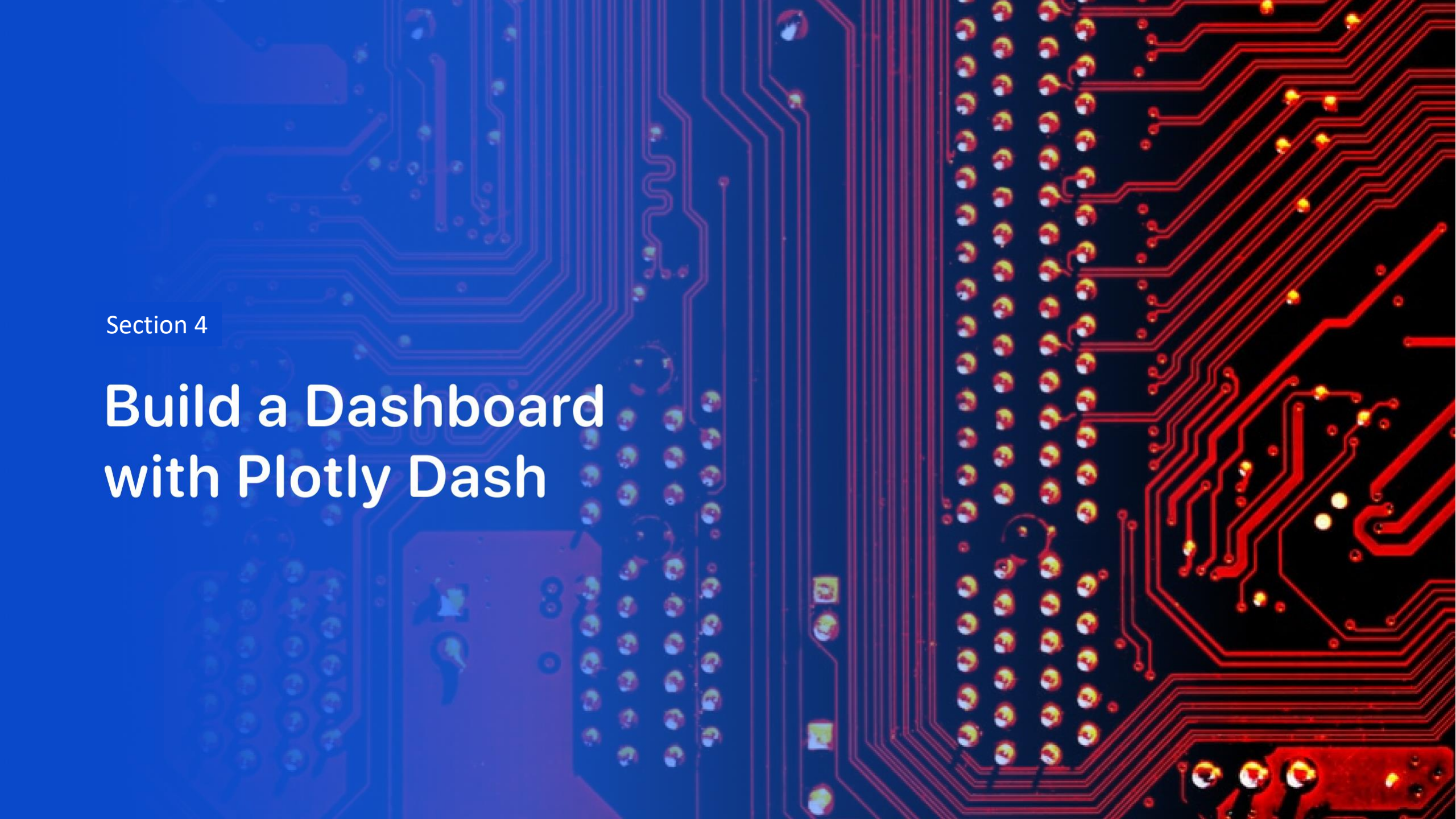


Task 2 Map Screenshot

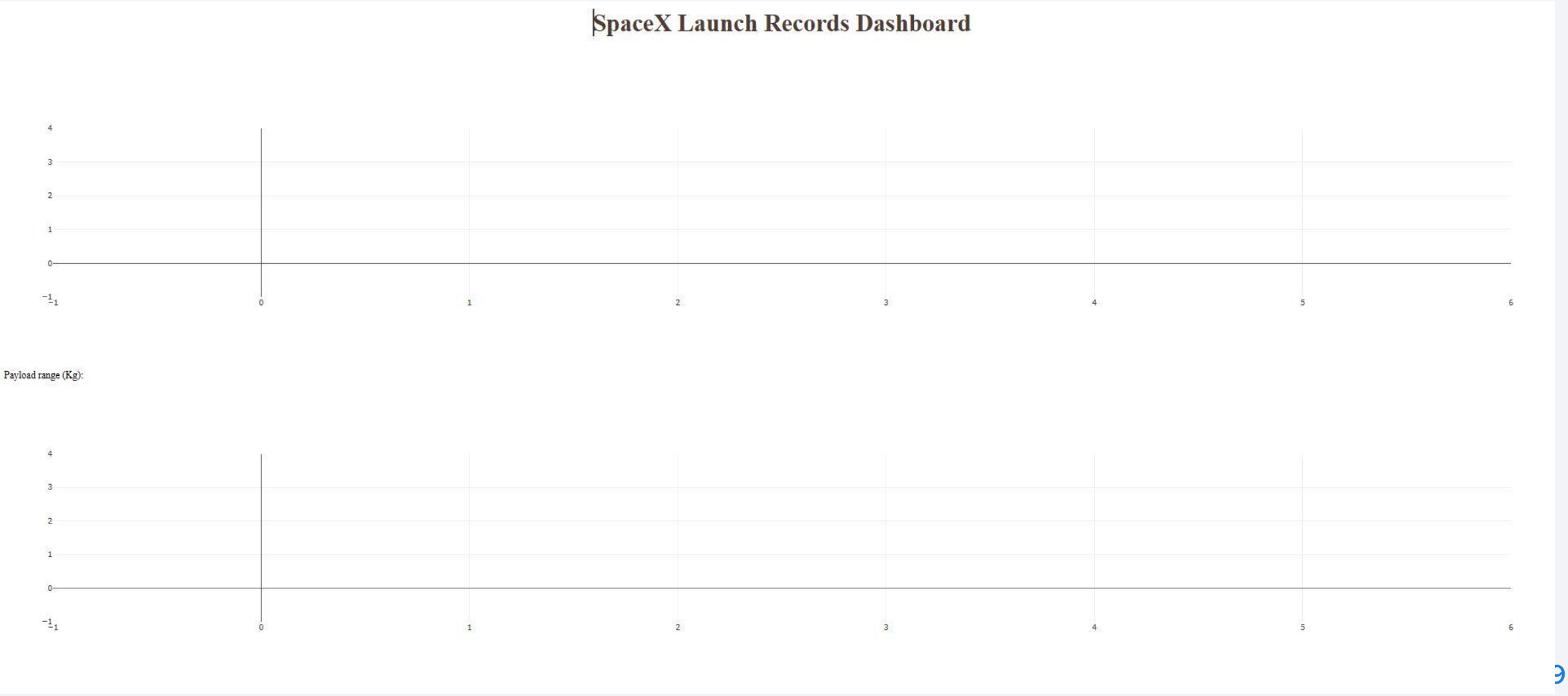


Map Screenshot




Section 4
Build a Dashboard with Plotly Dash

Dashboard using Plotly Screenshot



<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title
- Show the screenshot of the piechart for the launch site with highest launch success ratio
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title
- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

Predictive Analysis (Classification)

Classification Accuracy = Logistic regression performs best

```
# Calculate the accuracy scores for each method
logreg_score = logreg_cv.best_estimator_.score(X_test, Y_test)
svm_score = svm_cv.best_estimator_.score(X_test, Y_test)
tree_score = tree_cv.best_estimator_.score(X_test, Y_test)

# Print the accuracy scores
print("Accuracy score for Logistic Regression:", logreg_score)
print("Accuracy score for Support Vector Machine:", svm_score)
print("Accuracy score for Decision Tree Classifier:", tree_score)

# Identify the best-performing method
best_method = max(logreg_score, svm_score, tree_score)

if best_method == logreg_score:
    print("Logistic Regression performs the best.")
elif best_method == svm_score:
    print("Support Vector Machine performs the best.")
else:
    print("Decision Tree Classifier performs the best.")
```

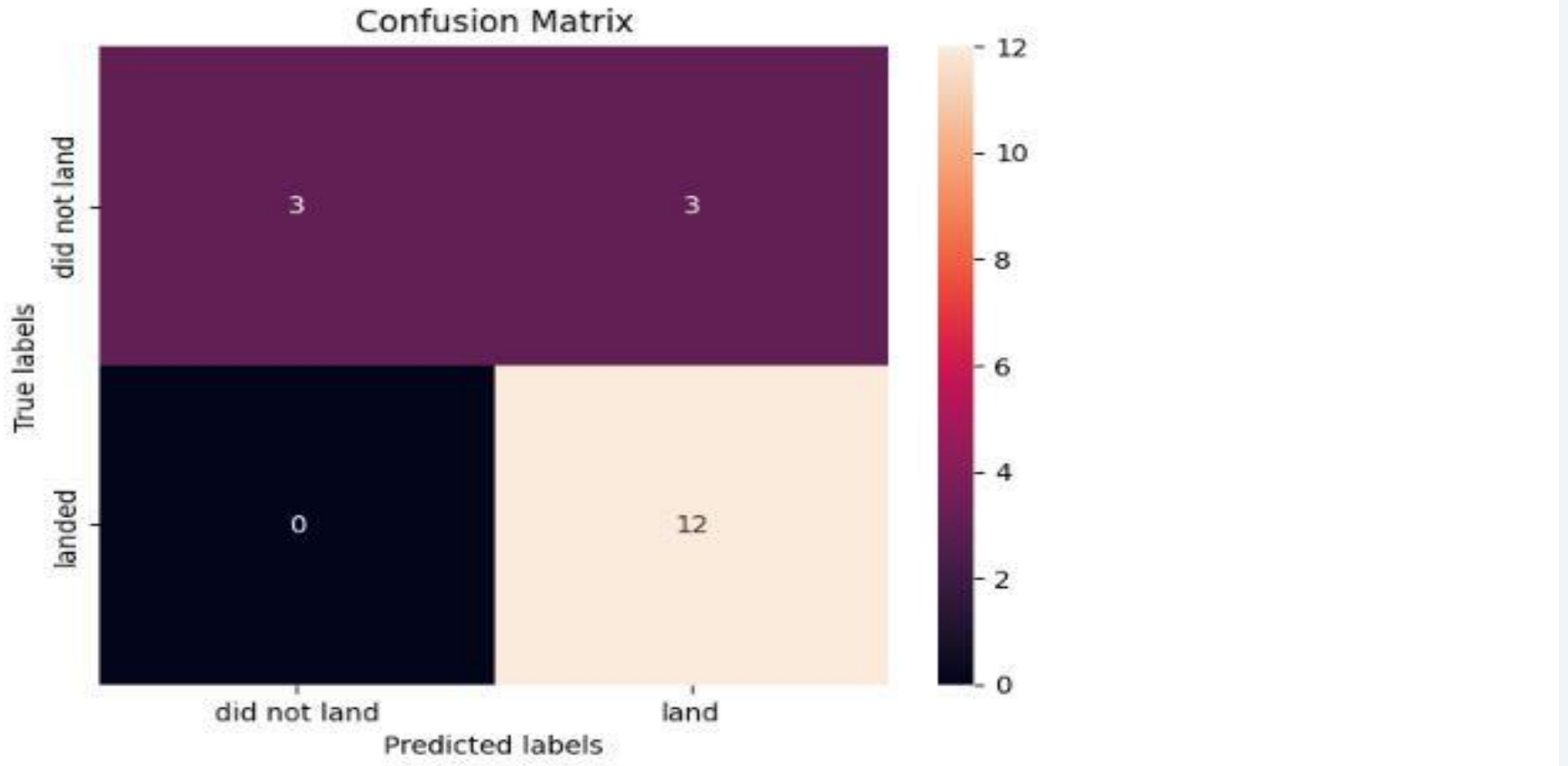
```
Accuracy score for Logistic Regression: 0.8333333333333334
Accuracy score for Support Vector Machine: 0.8333333333333334
Accuracy score for Decision Tree Classifier: 0.7777777777777778
Logistic Regression performs the best.
```

Confusion Matrix of Logistic Regression

```
Accuracy on test data: 0.8333333333333334
```

We can plot the confusion matrix

```
[43]: yhat=svm_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- Accuracy score for logistic regression : 0.8333333333333334
- Accuracy score for Support Vector Machine: 0.83333333333333
- Accuracy score for Decision Tree Classifier : 0.7777777777778
- Logistic Regression Performs the Best.

Appendix

- From Web Scraping to ML Models the entire files are uploaded to the github .
- Thank You Coursera
- Thank You IBM Data Science course Instructors

Thank you!