IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Junya Otsuka>
<September 24th, 2024>

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
  - Data Collection with Web Scraping
  - Data Wrangling
  - EDA with data visualization
  - EDA with SQL
  - Interactive Map with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

  SpaceX promotes its Falcon 9 rocket launches on its website at a price of $62 million, while other providers charge over $165 million for each launch. A significant portion of the savings comes from SpaceX's ability to reuse the first stage of the rocket. Consequently, by predicting whether the first stage will successfully land, we can estimate the launch cost. This information can be valuable for other companies considering bidding against SpaceX for a rocket launch. The objective of this project is to develop a machine learning pipeline that can accurately predict the successful landing of the first stage.

- Problems you want to find answers
  - What factors influence the successful landing of a rocket?
  - The relationship between various features that contribute to the success rate of a landing.
  - What operational conditions must be established to ensure a successful landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was gathered using the SpaceX API and through web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was utilized for the categorical features.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

The data was collected using various methods

Data collection involved sending a GET request to the SpaceX API.
Next, we decoded the response content as JSON using the .json() function and converted it into a pandas DataFrame with the .json_normalize() method.
After that, we cleaned the data by checking for missing values and filling in any gaps where necessary.
Additionally, we conducted web scraping from Wikipedia for Falcon 9 launch records using BeautifulSoup.
The goal was to extract the launch records as an HTML table, parse the table, and convert it into a pandas DataFrame for subsequent analysis.

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

  We employed a GET request to the SpaceX API to gather data, clean the retrieved information, and perform basic wrangling and formatting.

Link: https://github.com/moatheTP/-IBM-Data-Science-Capstone-SpaceX-/blob/main/1_Data%20Collection%20API.ipynb

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
[7]: spacex_url="https://api.spacexdata.com/v4/launches/past"

[8]: response = requests.get(spacex_url)
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
[13]: # Use json_normalize meethod to convert the json result into a dataframe
      # decode response content as json
      static_json_df = res.json()

[14]: # apply json_normalize
      data = pd.json_normalize(static_json_df)
```

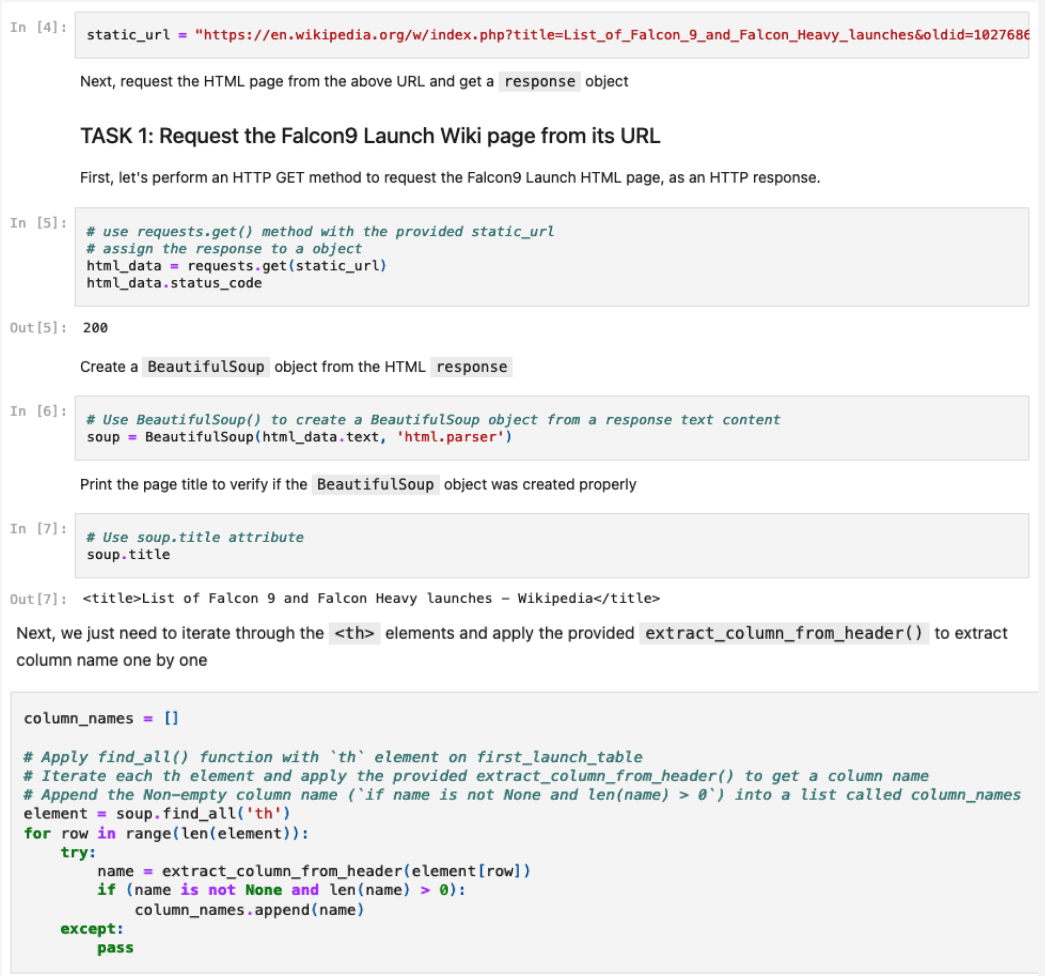
```
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

Data Collection - Scraping

We utilized web scraping to gather Falcon 9 launch records using BeautifulSoup.

We then parsed the table and converted it into a pandas DataFrame.

Link: https://github.com/moatheTP/-IBM-Data-Science-Capstone-SpaceX-/blob/main/2_Web%20Scraping.ipynb



In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

Out[5]:
```
200
```

Create a `BeautifulSoup` object from the HTML `response`

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

Data Wrangling

- We conducted exploratory data analysis and identified the training labels.
- We calculated the number of launches at each site, as well as the frequency and occurrence of each orbit.
- We generated a landing outcome label from the outcome column and exported the results to a CSV file.

EDA with Data Visualization

We examined the data by visualizing the relationships among flight number and launch site, payload and launch site, the success rate of each orbit type, flight number and orbit type, as well as the yearly trend of launch success.

Link: https://github.com/moatheTP/-IBM-Data-Science-Capstone-SpaceX-/blob/main/5_EDA%20Data%20Visualization.ipynb

EDA with SQL

We imported the SpaceX dataset into a PostgreSQL database directly from Jupyter Notebook.

We conducted exploratory data analysis (EDA) using SQL to gain insights from the data. We wrote queries to determine, for example:

- The names of unique launch sites within the specified limitations.

- The total payload mass carried by boosters launched by NASA (CRS).

- The average payload mass carried by the booster version F9 v1.1.

- The total number of successful and failed mission outcomes.

- The failed landing outcomes on drone ships, along with their corresponding booster versions and launch site names.

Build an Interactive Map with Folium

- We marked all launch sites and added map objects such as markers, circles, and lines to indicate the success or failure of launches for each site on the Folium map.
- We categorized the launch outcomes (failures or successes) into classes 0 and 1, with 0 representing failure and 1 representing success.
- Using color-labeled marker clusters, we identified launch sites with relatively high success rates.
- We calculated the distances from each launch site to nearby features and addressed questions such as:
    - Are launch sites located near railways, highways, and coastlines?
    - Do launch sites maintain a certain distance from cities?

Build a Dashboard with Plotly Dash

- We created an interactive dashboard using Plotly Dash.
- We displayed pie charts illustrating the total launches at specific sites.
- We also plotted a scatter graph to show the relationship between launch outcomes and payload mass (in kilograms) for the different booster versions.

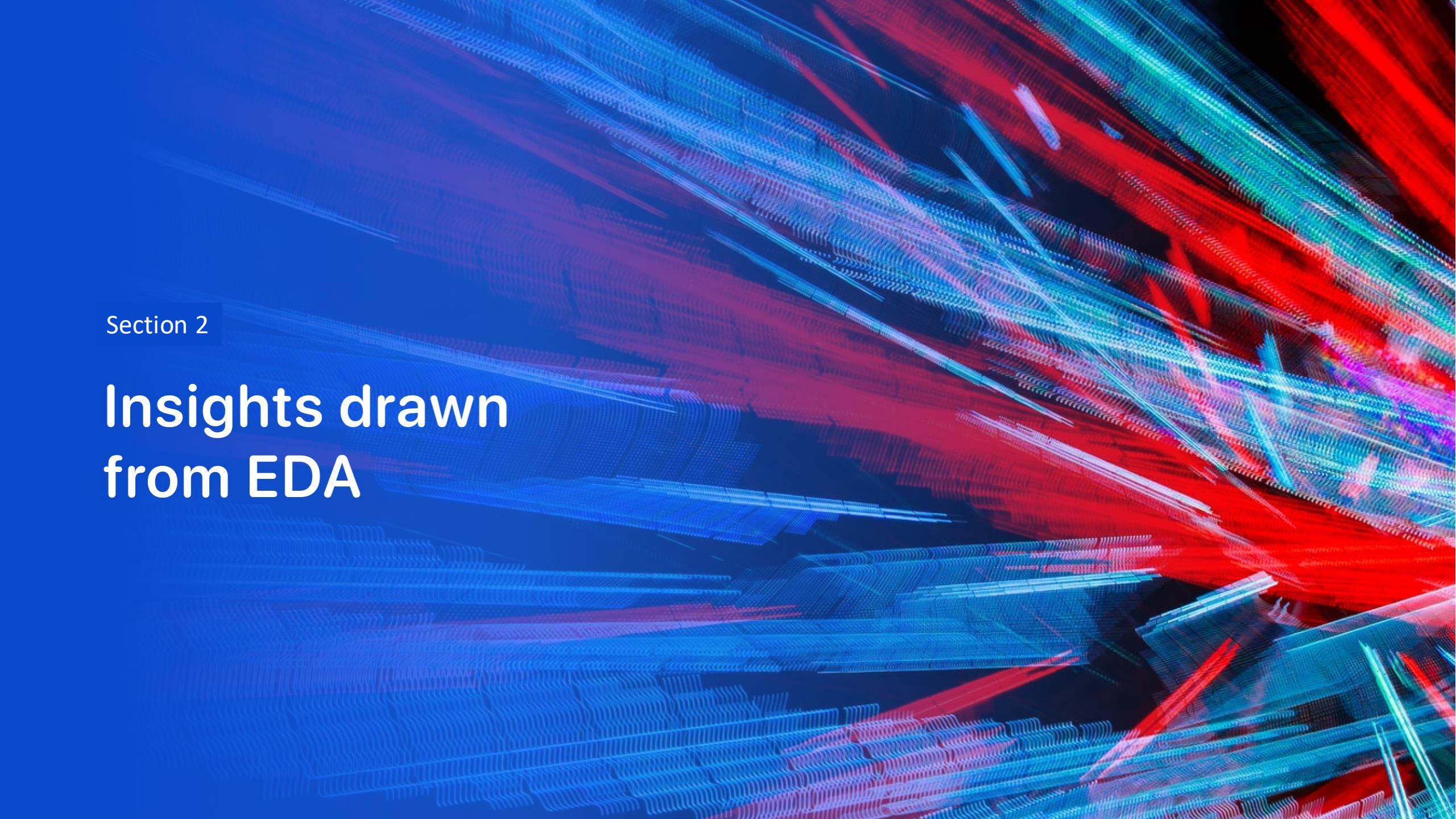
Link: https://github.com/moatheTP/-IBM-Data-Science-Capstone-SpaceX-/blob/main/7_Interactive%20Dashboard%20with%20Ploty%20Dash.py

Predictive Analysis (Classification)

- We loaded the data using NumPy and pandas, transformed it, and split it into training and testing sets.
- We developed various machine learning models and fine-tuned different hyperparameters using GridSearchCV.
- We utilized accuracy as the evaluation metric for our models and enhanced them through feature engineering and algorithm tuning.
- Ultimately, we identified the best-performing classification model.
- Link: https://github.com/moatheTP/-IBM-Data-Science-Capstone-SpaceX-/blob/main/8_Machine%20Learning%20Prediction.ipynb

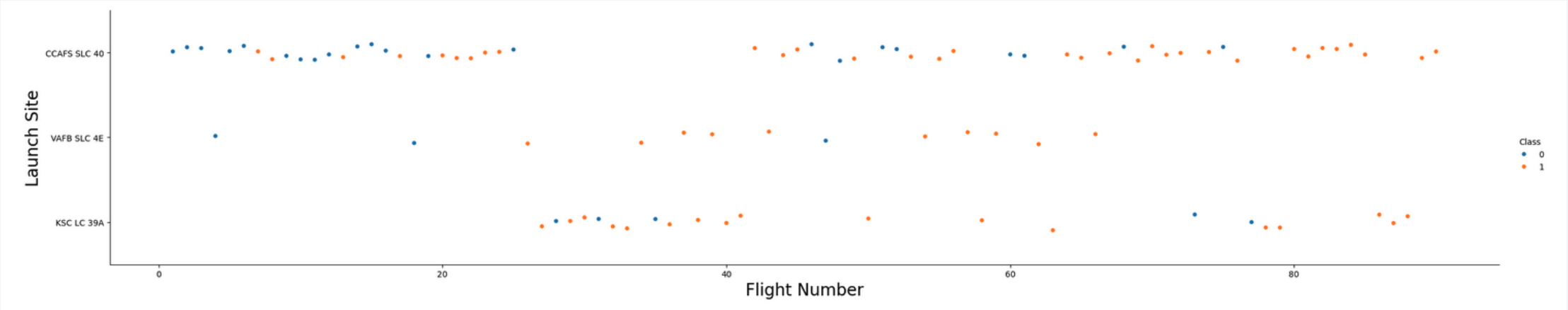
Results

- Results from exploratory data analysis
- Screenshots of the interactive analytics demo
- Outcomes from predictive analysis


Section 2
Insights drawn from EDA

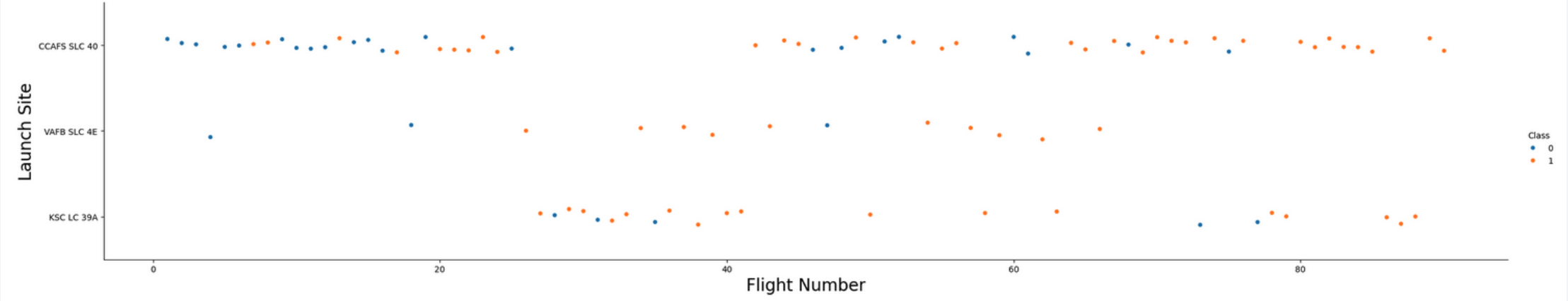
Flight Number vs. Launch Site

The more flights there are at a launch site, the higher the success rate becomes.

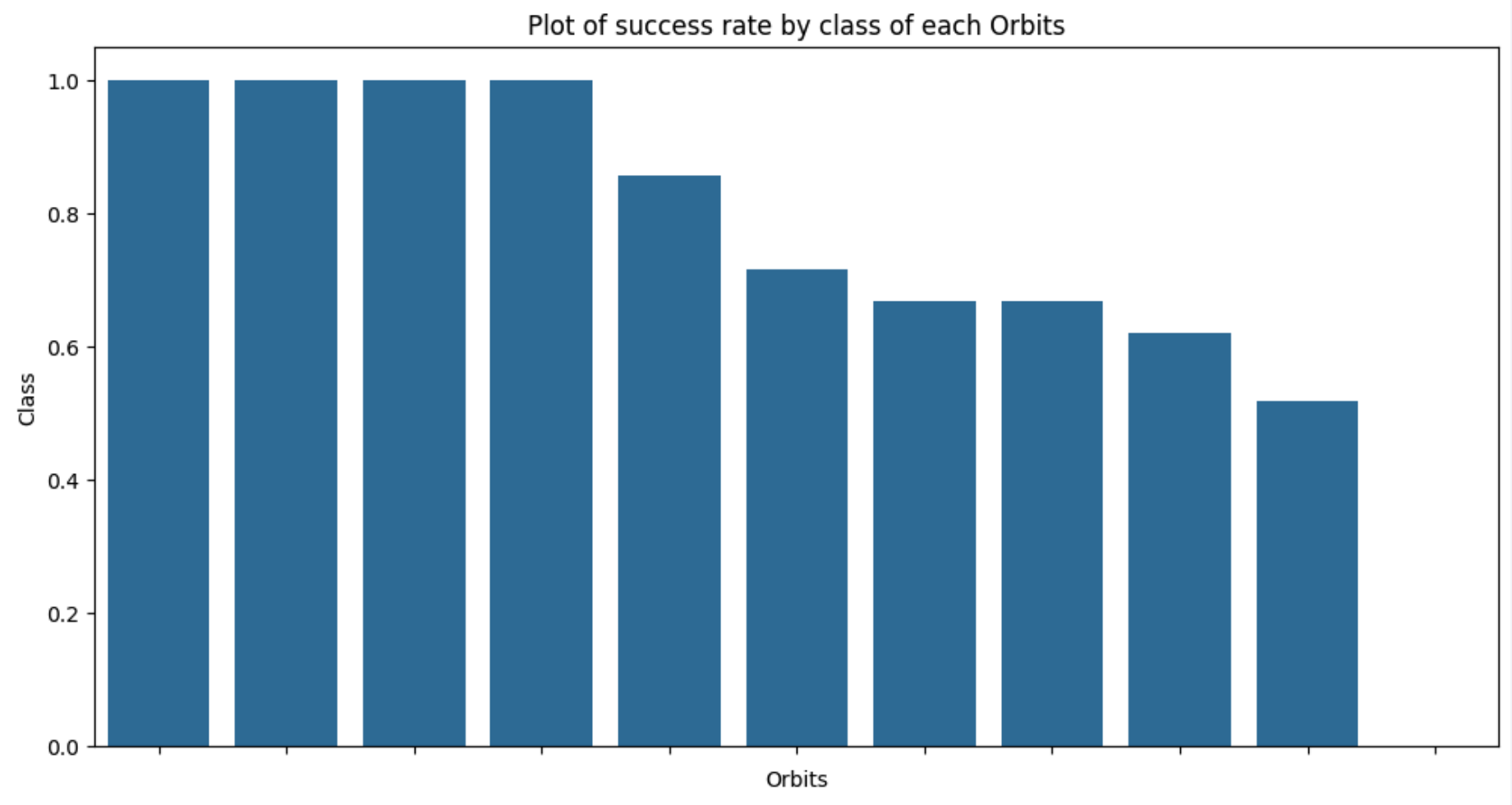


Payload vs. Launch Site

Upon analyzing the Payload Mass vs. Launch Site scatter plot, I observed that for the VAFB-SLC launch site, there were no rockets launched with a heavy payload mass (greater than 10,000 kg).

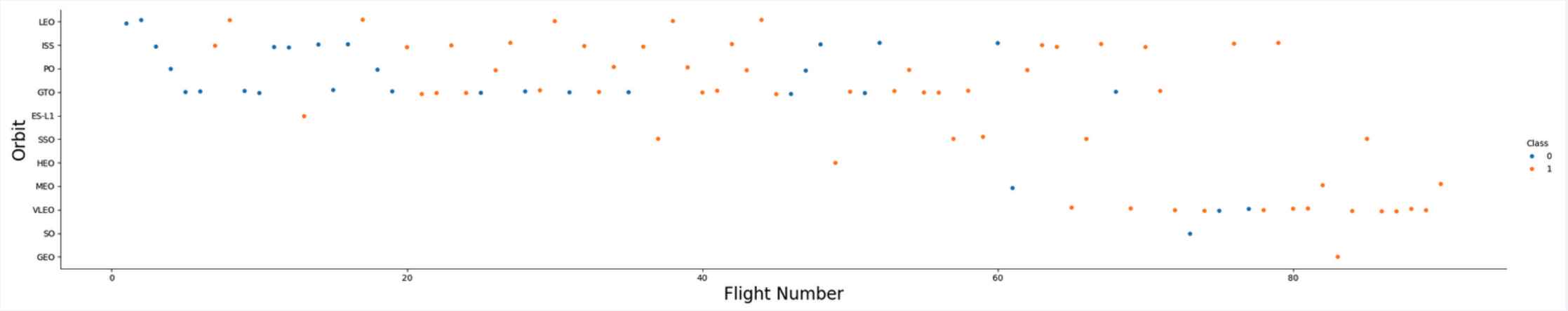


Success Rate vs. Orbit Type



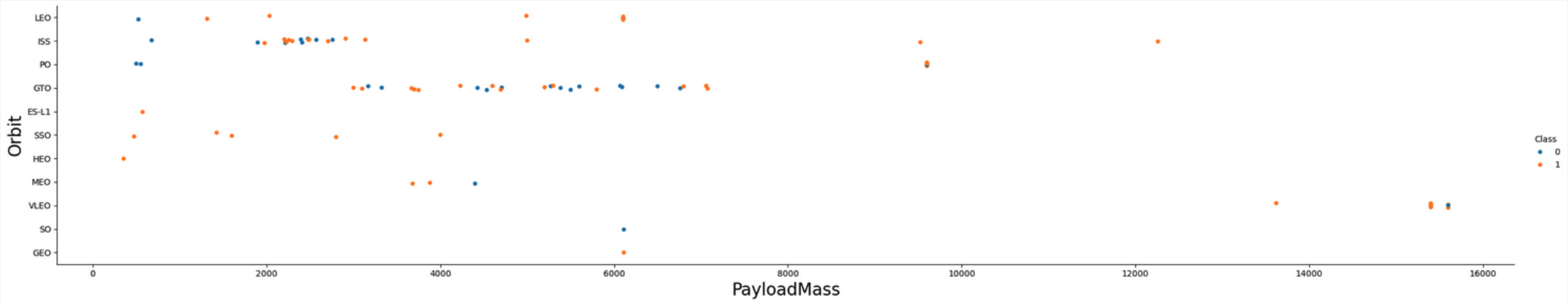
Flight Number vs. Orbit Type

I noticed that in LEO orbit, success seems to correlate with the number of flights. On the other hand, in GTO orbit, there doesn't appear to be any relationship between the number of flights and success.



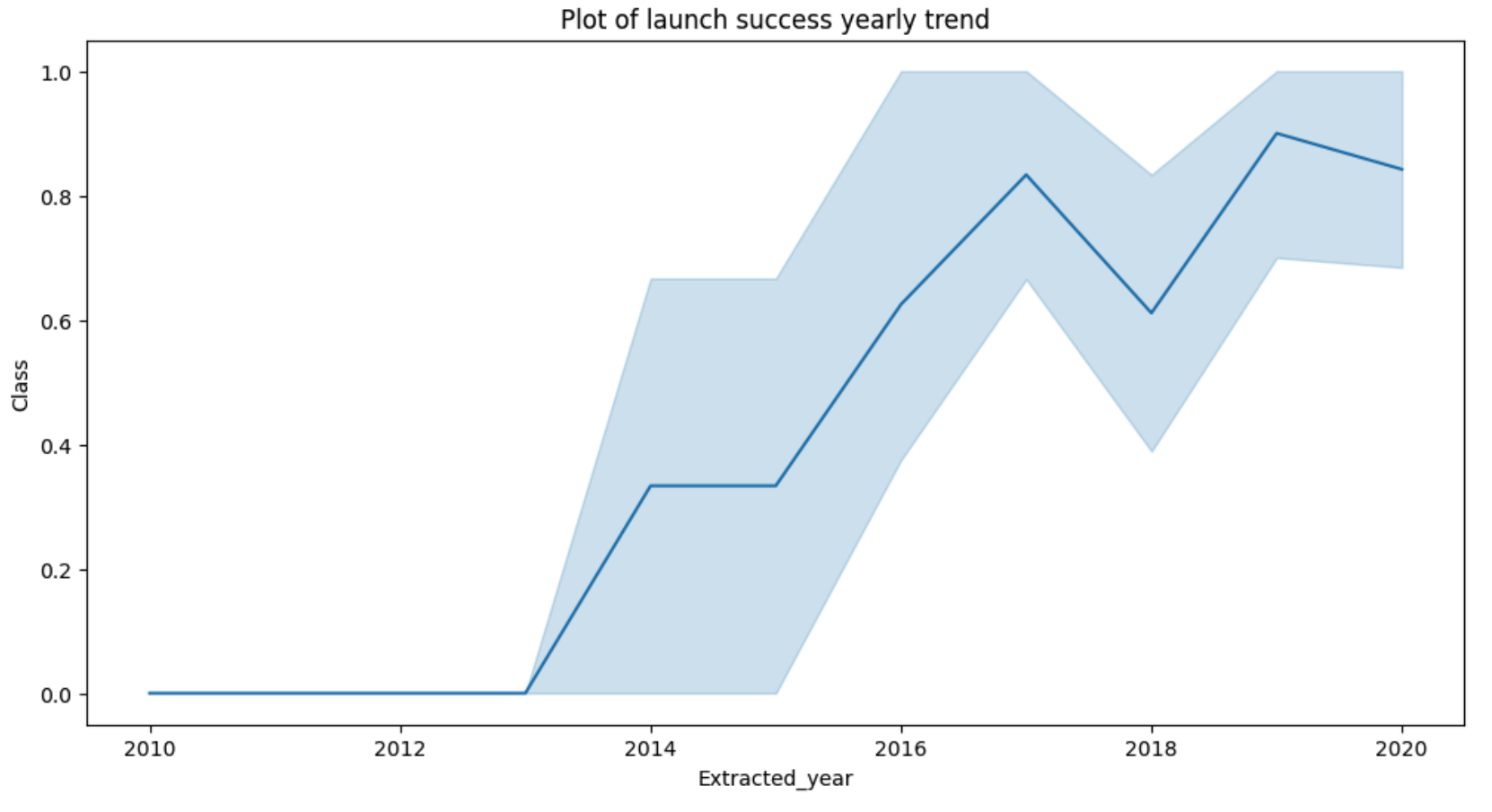
Payload vs. Orbit Type

In my analysis, I found that with heavy payloads, the successful or positive landing rate is higher for Polar, LEO, and ISS orbits. However, for GTO, it's harder to differentiate between successful and unsuccessful landings as both outcomes are observed.



Launch Success Yearly Trend

I observed that the success rate has steadily increased from 2013 until 2020.



All Launch Site Names

Used the key word UNIQUE to show only unique launch sites from the SpaceX data.

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'

Used query above below to display five records where launch sites begin with ‘CCA’.

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

[9]:
```
%sql SELECT * \
    FROM SPACEXTBL \
    WHERE LAUNCH_SITE LIKE'CCA%' LIMIT 5;
```

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

[9]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Calculated the total payload carried by boosters from NASA as 45596 using the query as:

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

[10]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```

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

[10]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

Calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

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

[11]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

[11]:

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

Observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

In [14]:

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

I used the WHERE clause to filter for boosters that successfully landed on a drone ship and applied the AND condition to specify successful landings with a payload mass between 4,000 and 6,000.

In [15]:
```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number \
FROM SPACEXTBL \
GROUP BY MISSION_OUTCOME;
```

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

Mission_Outcome	total_number
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

Identified the booster that carried the maximum payload by using a subquery in the WHERE clause along with the MAX() function.

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

In [17]:

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 Launch Records

Used a combination of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes on drone ships, along with their booster versions and launch site names, for the year 2015.

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

Selected the landing outcomes and the COUNT of those outcomes from the data, using the WHERE clause to filter for landing outcomes between 2010-06-04 and 2010-03-20.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

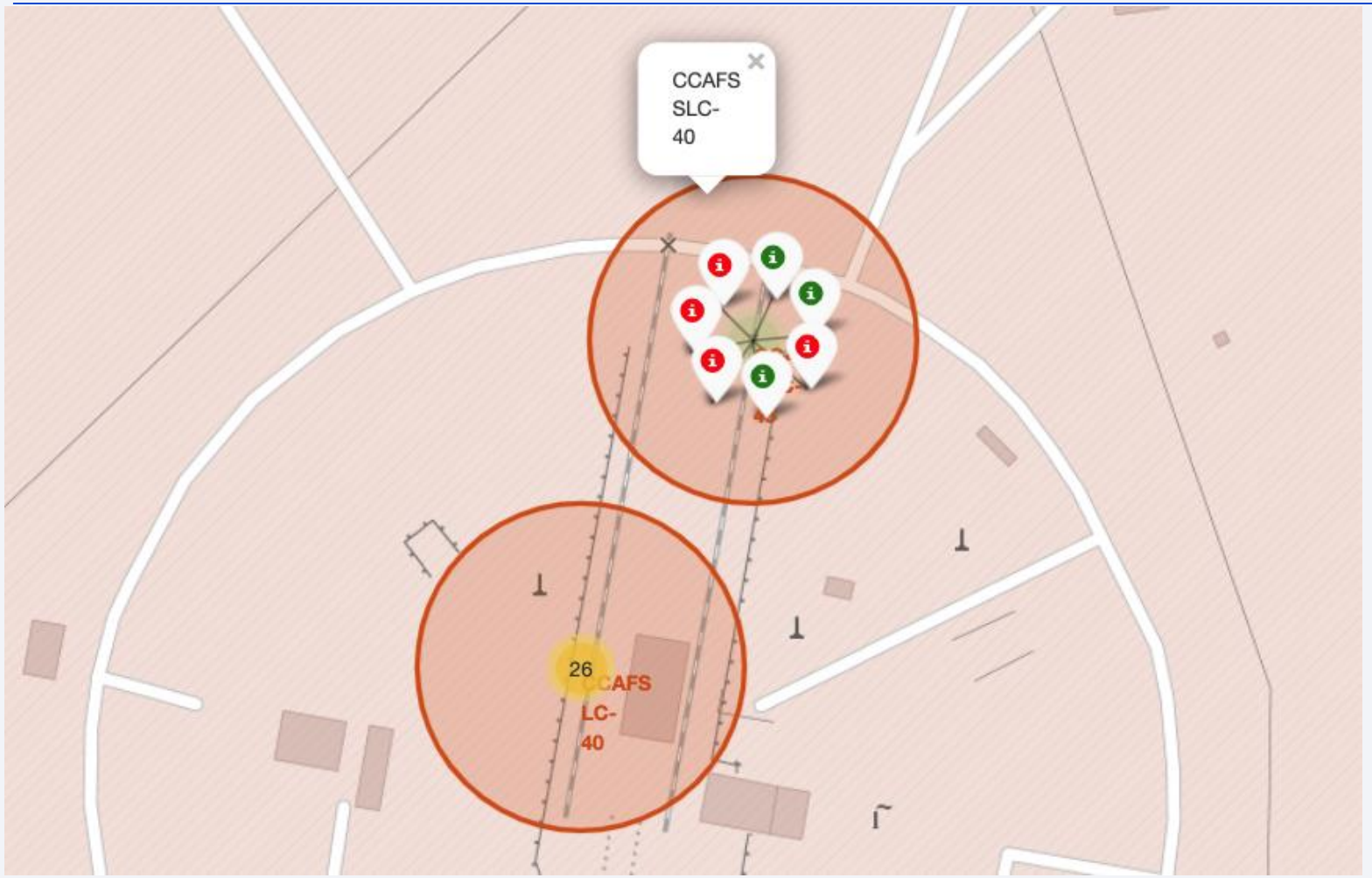
Section 3

Launch Sites Proximities Analysis

Global map displaying markers for all launch sites.

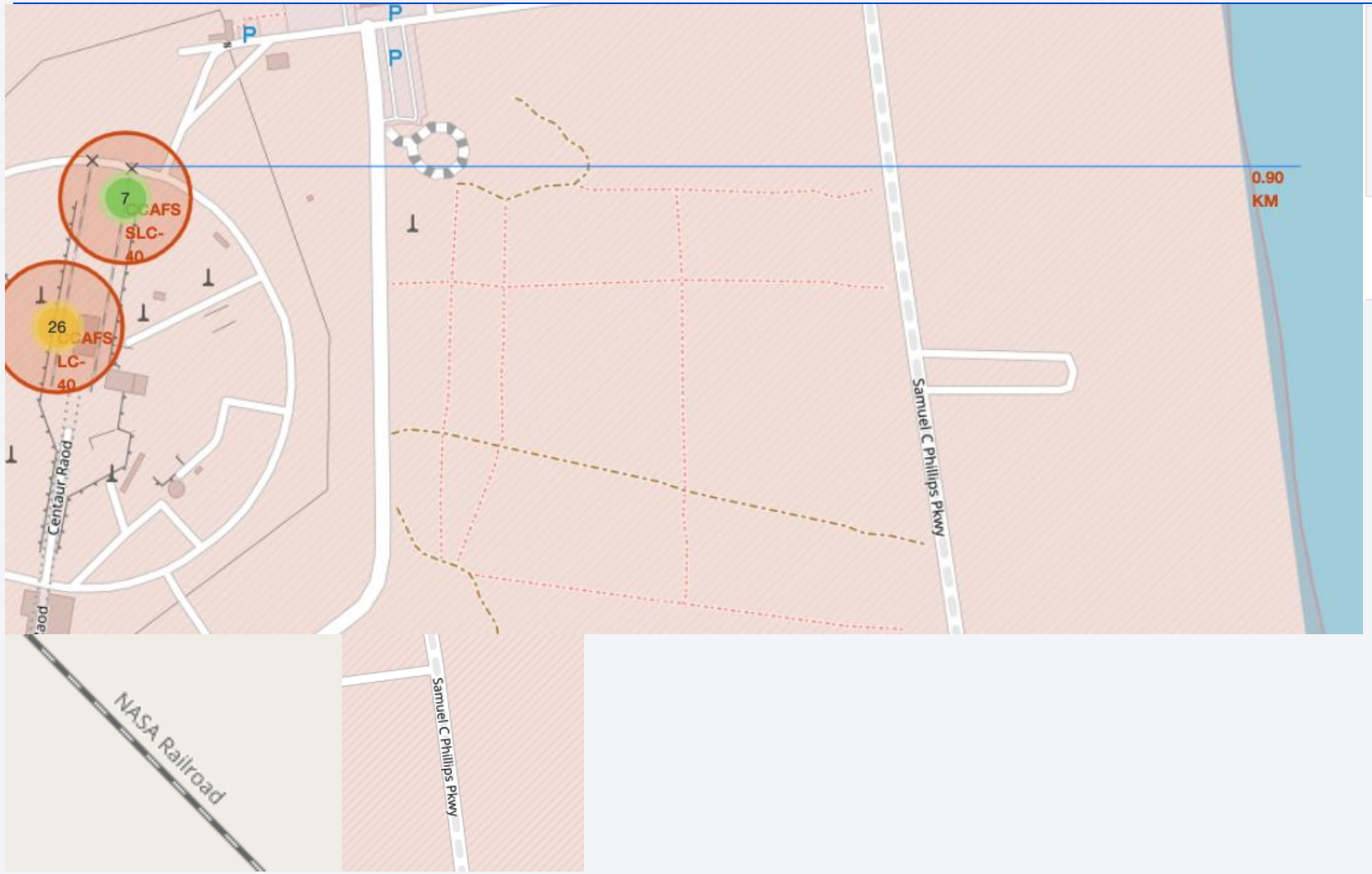
It is clear that SpaceX launch sites are located on the coasts of the United States, specifically in Florida and California.

Markers indicating launch sites with corresponding color labels.



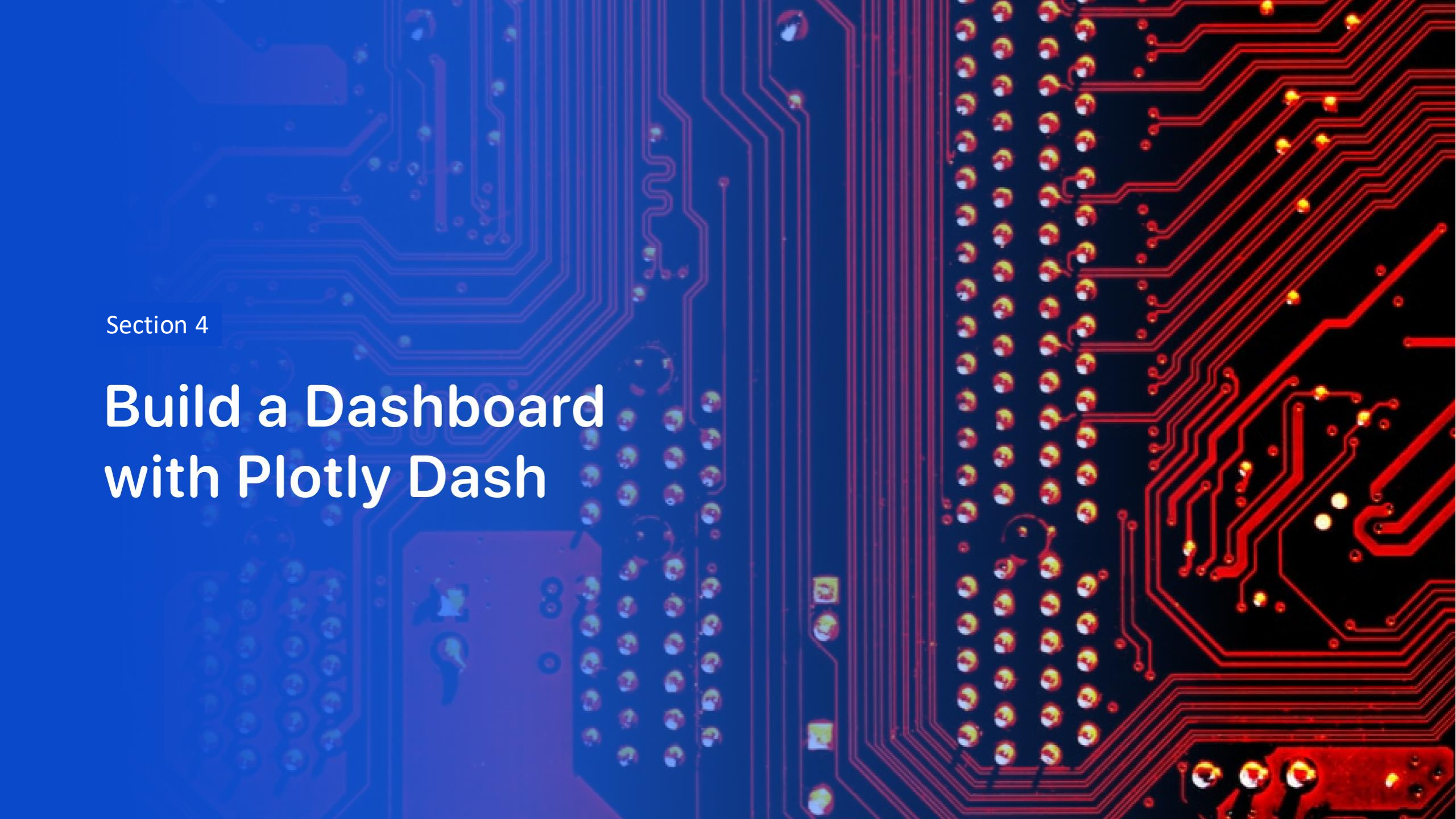
The green markers represent successful launches, while the red markers indicate failures.

Distance from the launch sites to nearby landmarks.

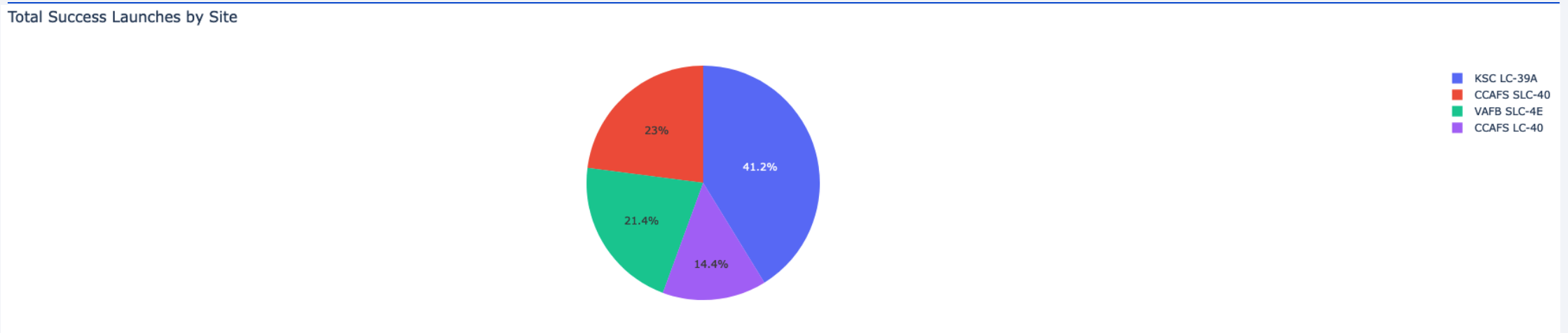



```
print("City Distance", city_distance)

print("Railway Distance", railway_distance)

print("Highway Distance", highway_distance)

print("Coastline Distance", distance_coastline)
```

```
City Distance 23.234752126023245
Railway Distance 21.961465676043673
Highway Distance 26.88038569681492
Coastline Distance 0.8627671182499878
```

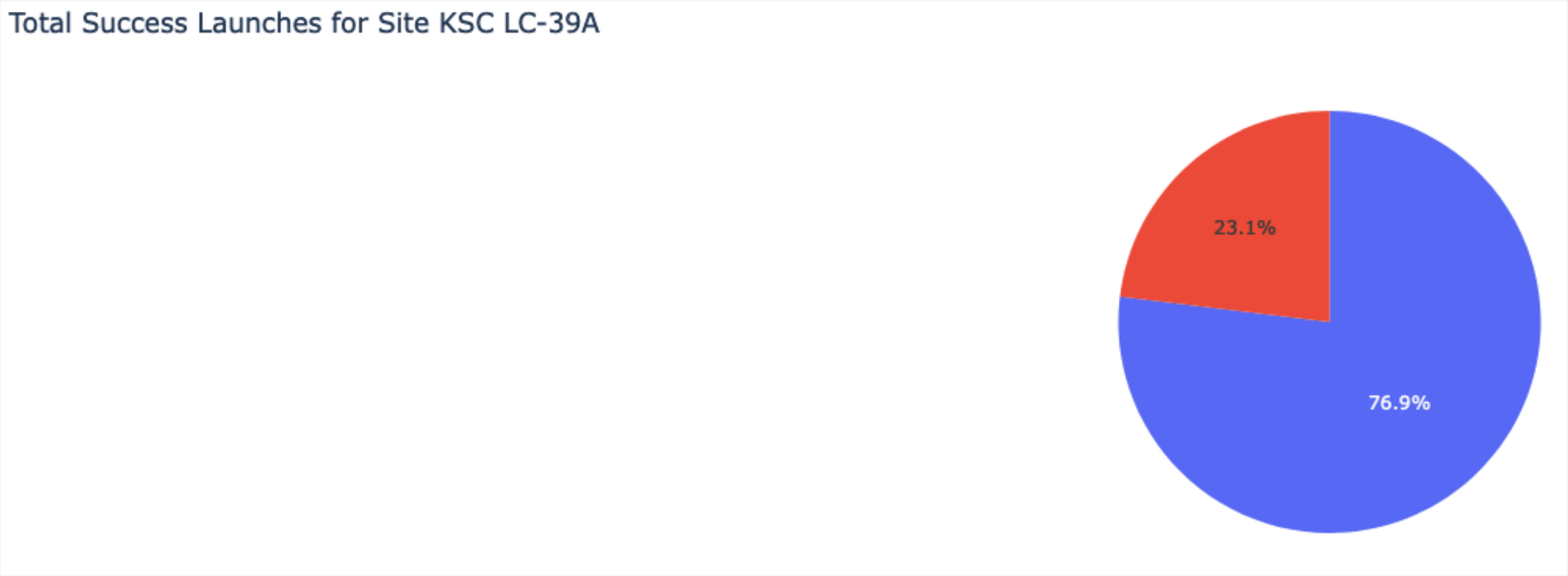

Section 4
Build a Dashboard with Plotly Dash

A pie chart displaying the success rate for each launch site.



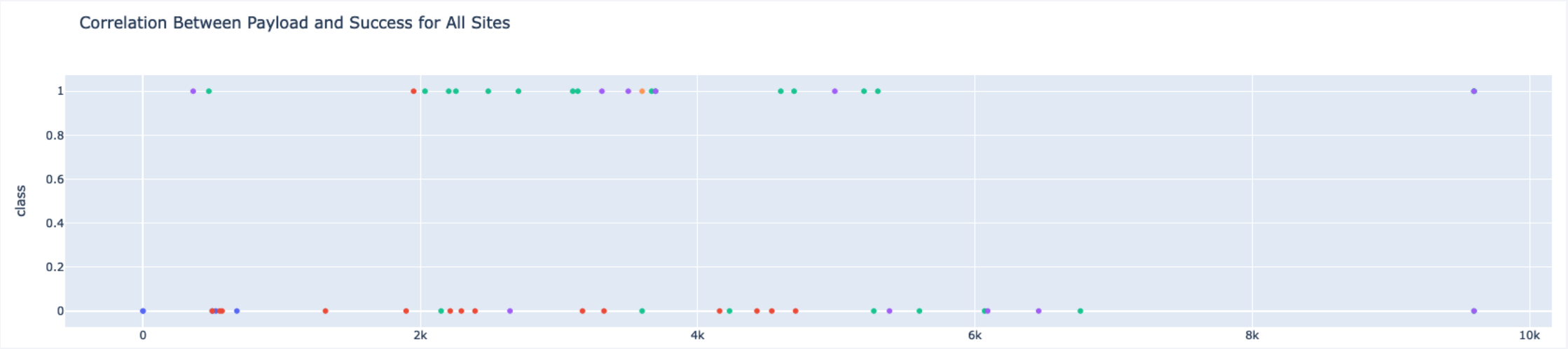
It is evident that KSC LC-39A had the highest number of successful launches among all the sites.

Pie chart displaying the launch site with the highest success ratio for launches.



KSC LC-39A had a success rate of 76.9% and a failure rate of 23.1%.

Scatter plot of payload versus launch outcome for all sites, featuring a range slider to select different payload values.



The success rate for low-weight payloads is higher than that for heavy-weight payloads.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

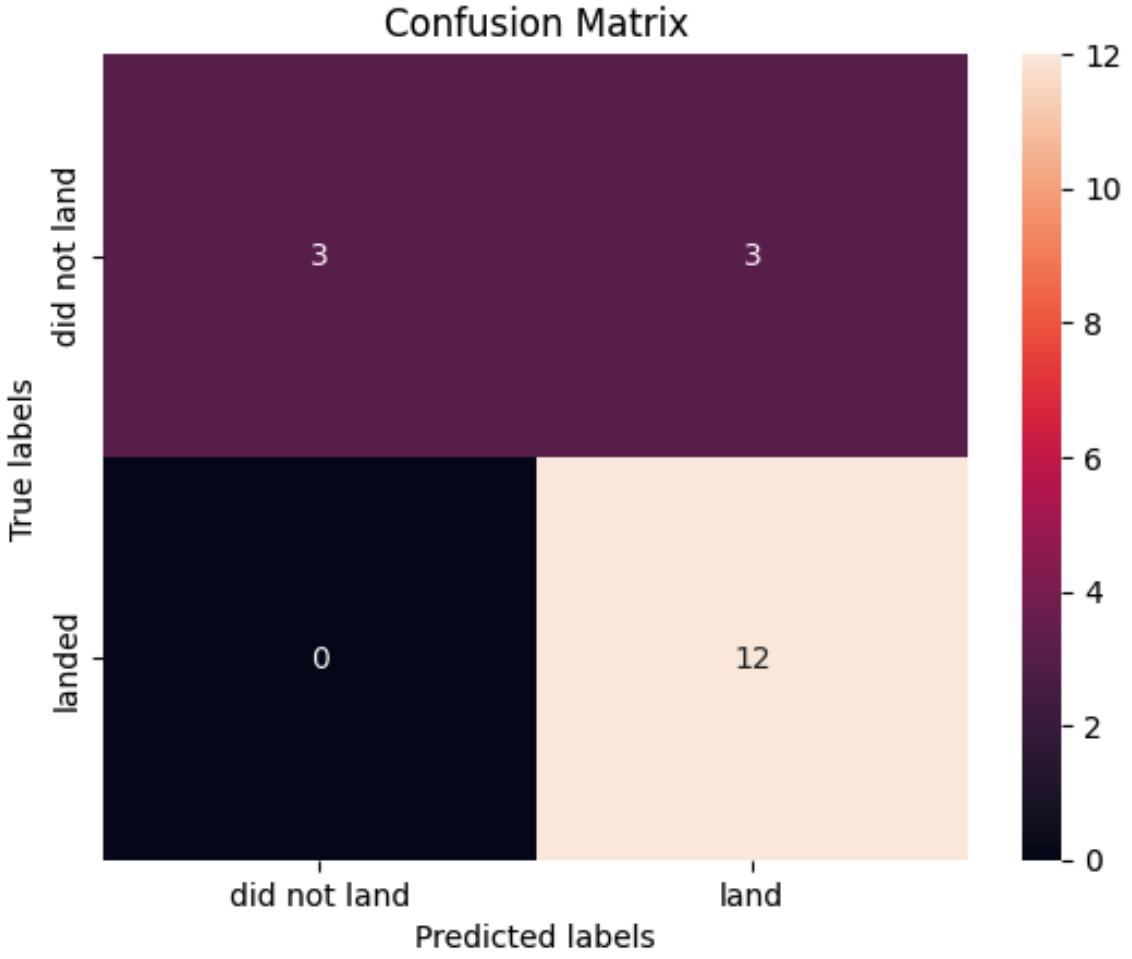
Find the method performs best:

```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

The decision tree classifier exhibits the highest classification accuracy among the models evaluated.

Confusion Matrix



The confusion matrix for the decision tree classifier indicates that it can differentiate between the various classes. However, the primary issue is the occurrence of false positives, where unsuccessful landings are incorrectly classified as successful landings by the model.

Conclusions

- The success rate at a launch site increases with the volume of flights conducted there.
- From 2013 to 2020, the launch success rate experienced consistent growth.
- The orbits ES-L1, GEO, HEO, SSO, and VLEO achieved the highest success rates.
- Among all launch sites, KSC LC-39A recorded the most successful launches.
- The decision tree classifier is the most effective machine learning algorithm for this task.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!