IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ahmed Zayed
4/2/2025

Outline

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

Executive Summary

- **Summary of Methodologies**

Data retrieval via API

Data extraction using web scraping

Data preprocessing and wrangling

Exploratory data analysis using SQL

Data exploration through visualization

Interactive visual analytics with Folium

Predictive modeling with machine learning

- **Summary of All Results**

Findings from exploratory data analysis

Interactive analytics presented in screenshots

Results from predictive analytics

Introduction

- **Project Background**

SpaceX offers Falcon 9 rocket launches at a cost of $62 million, significantly lower than other providers, who charge upwards of $165 million per launch. A major contributor to this cost reduction is SpaceX's capability to reuse the rocket's first stage. Accurately predicting whether the first stage will successfully land allows for better estimation of overall launch costs. This insight can be valuable for potential competitors looking to bid against SpaceX for future launches. The objective of this project is to develop a machine learning pipeline to predict the success of first-stage landings.

- **Business Problems**

What are the primary factors that determine the success of a rocket landing?

How do different variables interact to influence the probability of a successful landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The data was collected using the SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - The dataset was cleaned and structured to maintain consistency.
- Perform exploratory data analysis (EDA) using visualization and SQL
- SQL queries and visualizations were utilized to examine trends and patterns.
- Perform interactive visual analytics using Folium and Plotly Dash
- Dynamic data exploration was conducted using Folium and Plotly Dash.
- Perform predictive analysis using classification models
  - Predictive analysis was conducted using four different machine learning classification models.

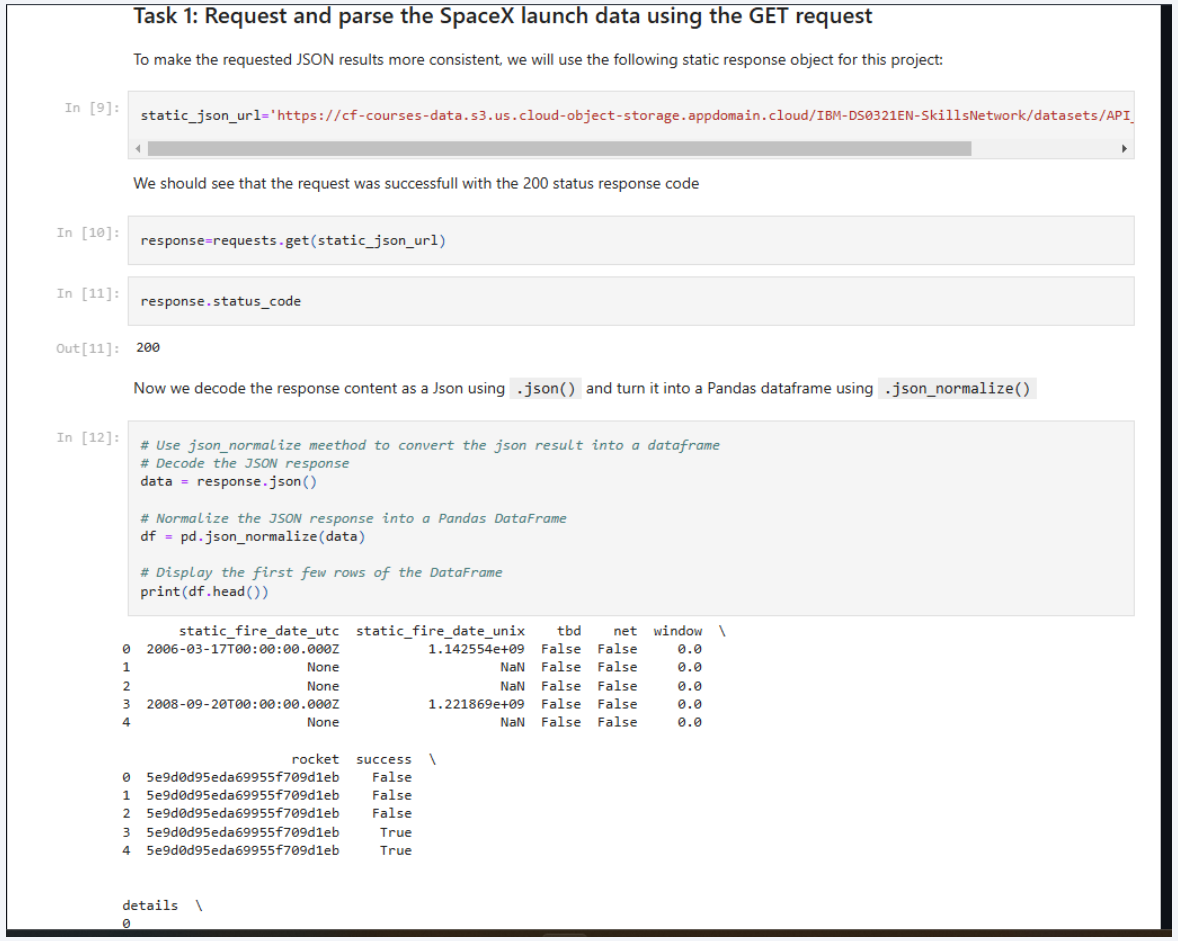
Data Collection

- Data was collected using a GET request to the SpaceX API.
- The response was decoded into JSON format using the .json() function and converted into a Pandas DataFrame with .json_normalize().
- The dataset was cleaned by identifying missing values and performing necessary imputations.
- Web scraping was conducted using BeautifulSoup to extract Falcon 9 launch records from Wikipedia.
- The extracted records, originally in an HTML table format, were parsed and transformed into a Pandas DataFrame for further analysis.

Data Collection – SpaceX API

- Data was gathered via:
- GET request to the SpaceX API.
- Cleaning
- Processing.
- Formatting.
- Add the GitHub URL of the completed SpaceX API calls notebook:

https://github.com/AhmedZayed8/Space-X/blob/main/lab%201%20collecting%20the%20data.ipynb



Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_
```

We should see that the request was successfull with the 200 status response code

In [10]:
```
response=requests.get(static_json_url)
```

In [11]:
```
response.status_code
```

Out[11]: 200

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

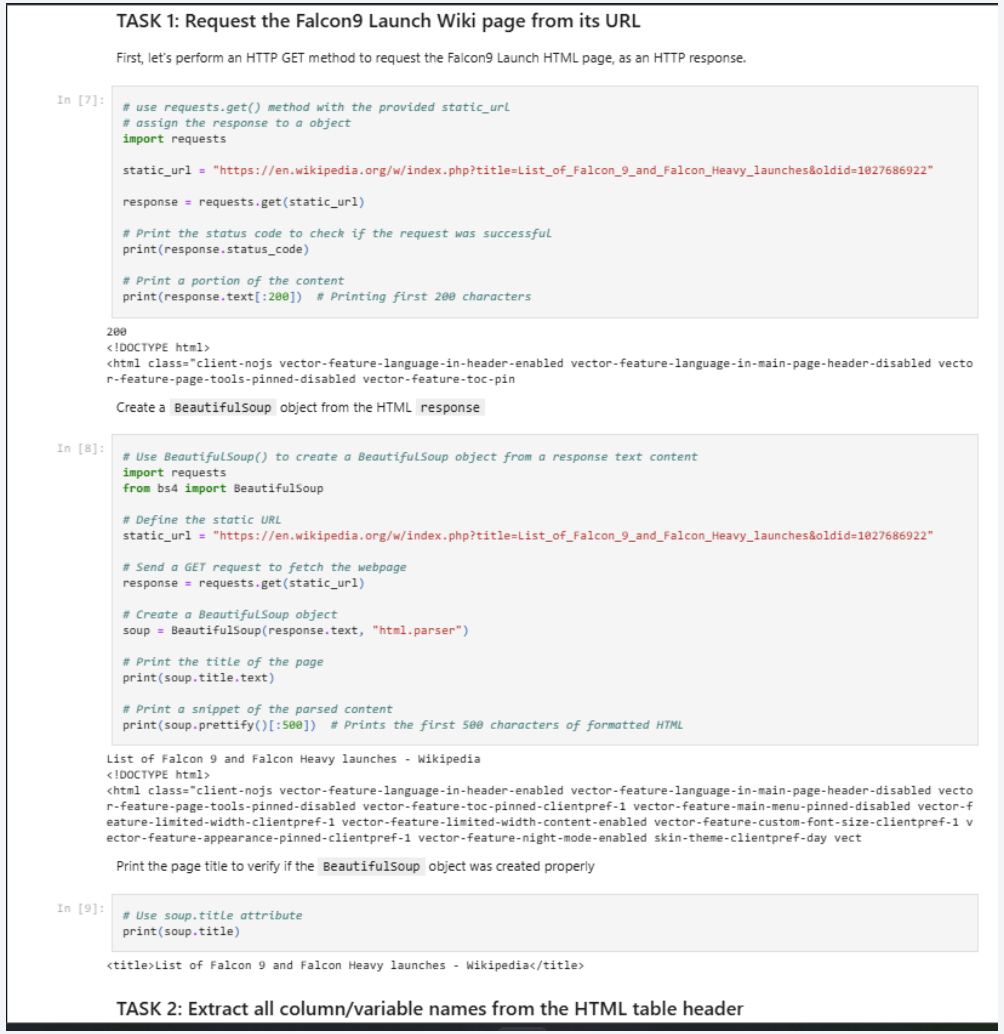
In [12]:
```
# Use json_normalize meethod to convert the json result into a dataframe
# Decode the JSON response
data = response.json()

# Normalize the JSON response into a Pandas DataFrame
df = pd.json_normalize(data)

# Display the first few rows of the DataFrame
print(df.head())
```

```
      static_fire_date_utc  static_fire_date_unix    tbd    net  window  \
0  2006-03-17T00:00:00.000Z           1.142554e+09  False  False     0.0
1                     None                    NaN  False  False     0.0
2                     None                    NaN  False  False     0.0
3  2008-09-20T00:00:00.000Z           1.221869e+09  False  False     0.0
4                     None                    NaN  False  False     0.0

                     rocket  success  \
0  5e9d0d95eda69955f709d1eb    False
1  5e9d0d95eda69955f709d1eb    False
2  5e9d0d95eda69955f709d1eb    False
3  5e9d0d95eda69955f709d1eb     True
4  5e9d0d95eda69955f709d1eb     True


details  \
0
```

Data Collection - Scraping

- Request rocket launch data from its Wikipedia page.
- Extract all column/variable names from the HTML table header.
- Create a data frame by parsing the launch HTML tables.
- Add the GitHub URL of the completed web scraping notebook:

https://github.com/AhmedZayed8/Space-X/blob/main/jupyter-labs-webscraping.ipynb



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 [7]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
import requests

static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

response = requests.get(static_url)

# Print the status code to check if the request was successful
print(response.status_code)

# Print a portion of the content
print(response.text[:200])  # Printing first 200 characters
```

```
200
<!DOCTYPE html>
<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-disabled vecto
r-feature-page-tools-pinned-disabled vector-feature-toc-pin
```

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

In [8]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
import requests
from bs4 import BeautifulSoup

# Define the static URL
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

# Send a GET request to fetch the webpage
response = requests.get(static_url)

# Create a BeautifulSoup object
soup = BeautifulSoup(response.text, "html.parser")

# Print the title of the page
print(soup.title.text)

# Print a snippet of the parsed content
print(soup.prettify()[:500])  # Prints the first 500 characters of formatted HTML
```

```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
<!DOCTYPE html>
<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-disabled vecto
r-feature-page-tools-pinned-disabled vector-feature-toc-pinned-clientpref-1 vector-feature-main-menu-pinned-disabled vector-f
eature-limited-width-clientpref-1 vector-feature-limited-width-content-enabled vector-feature-custom-font-size-clientpref-1 v
ector-feature-appearance-pinned-clientpref-1 vector-feature-night-mode-enabled skin-theme-clientpref-day vect
```

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

In [9]:
```
# Use soup.title attribute
print(soup.title)
```

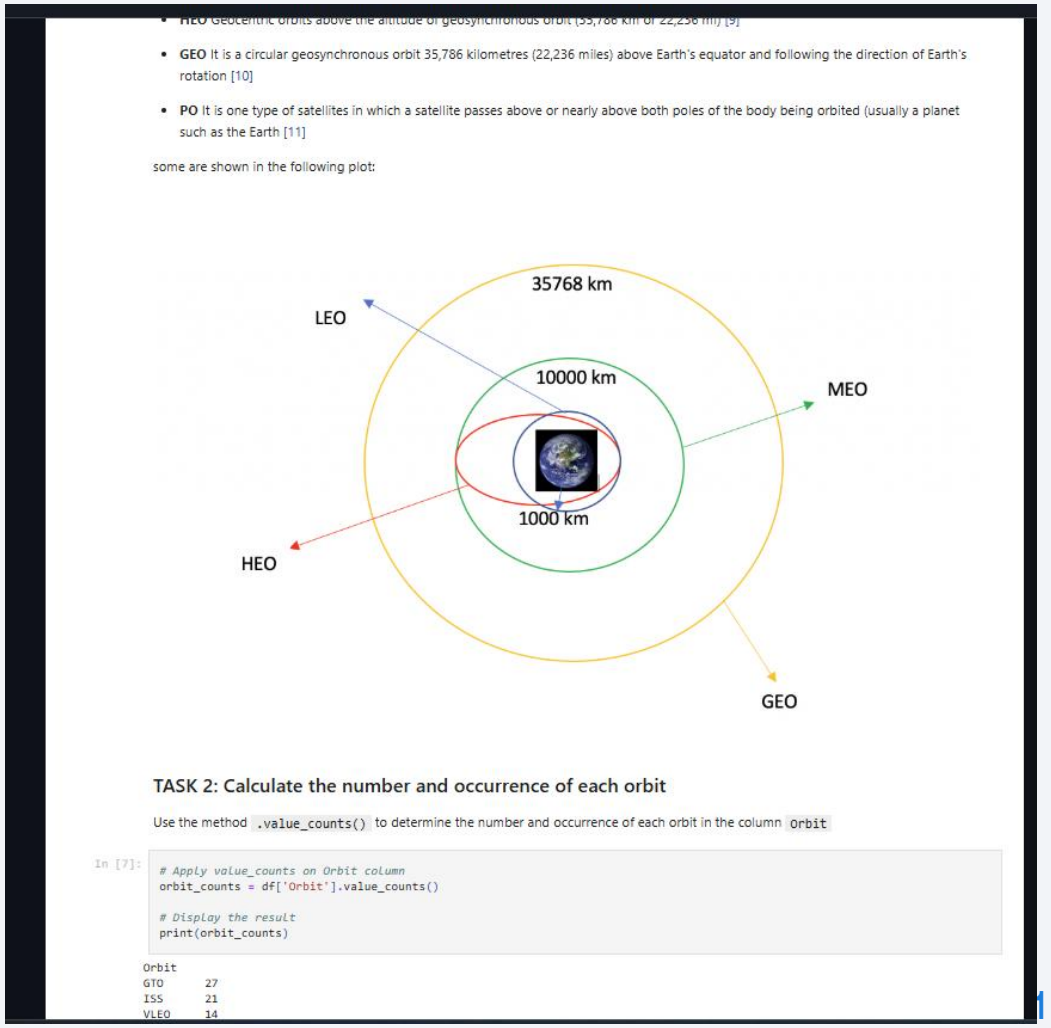
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Data Wrangling

- Determine the number of launches for each site.
- Count the occurrences and types of each orbit.
- Analyze the number and occurrences of mission outcomes based on orbit type.
- Generate a landing outcome label from the Outcome column using one-hot encoding.
- Save the data to a CSV file.
- Add the GitHub URL of your completed data wrangling related notebooks:

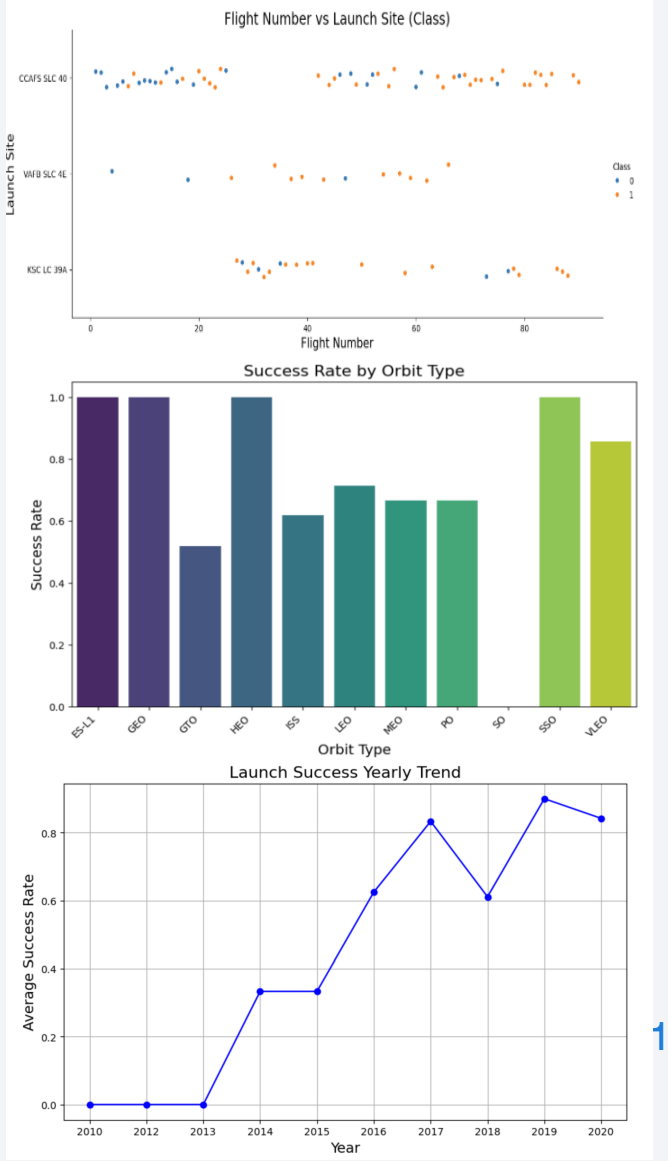
https://github.com/AhmedZayed8/Space-X/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



EDA with Data Visualization

- Scatter plots: Scatter plots illustrated the relationships between two variables. Various feature combinations were analyzed, such as Flight Number vs. Launch Site, Payload vs. Launch Site, Flight Number vs. Orbit Type, and Payload vs. Orbit Type.
- Bar charts: Bar charts provided a clear visual comparison of values across multiple groups. The x-axis represented different categories while the y-axis displayed discrete values. These charts were employed to compare the Success Rate across various Orbit Types.
- Line charts: Line charts effectively showcased data trends over time. A line chart was utilized to depict the Success Rate over a given number of years.
- Add the GitHub URL of your completed EDA with data visualization notebook:

https://github.com/AhmedZayed8/Space-X/blob/main/edadataviz.ipynb



EDA with SQL

- Show the names of the unique launch sites in the space mission.
- Display 5 records where launch sites start with 'CCA'.
- Calculate the total payload mass carried by NASA (CRS) boosters.
- Show the average payload mass carried by booster version F9 v1.1.List the date of the first successful landing outcome on the ground pad.
- Identify boosters that successfully landed on a drone ship with a payload mass between 4000 and 6000.Count the total number of successful and failed mission outcomes.
- Identify the booster versions that carried the maximum payload mass.
- List the failed landing outcomes on drone ships, including their booster versions and launch site names for the year 2015.Rank the count of landing outcomes between 2010-06-04 and 2017-03-20 in descending order.
- Add the GitHub URL of your completed EDA with SQL notebook:

https://github.com/AhmedZayed8/Space-X/blob/main/eda_sql.ipynb

Build an Interactive Map with Folium

- Markers: I added markers to represent each launch site on the map. This helps to easily identify and locate the launch sites.
- Circles: I used circles to indicate the launch outcomes. Different colors were assigned: one color for successful launches and another for failed launches. This visual differentiation makes it clear at a glance which sites had successful outcomes and which did not.
- Lines: I drew lines to connect launch sites with nearby features or landmarks, such as the distance from the launch site to the nearest populated area or ocean. This helps in visualizing the geographic context of the launch sites and understanding their proximity to significant landmarks.
- Color-coded Marker Clusters: Marker clusters were used to group nearby launch sites, with different colors signifying clusters with high success rates. This allows for an easier overview of areas with higher or lower success rates.
- The primary reason for adding these map objects is to enhance the visualization and interpretability of the launch data. They provide a clear, visual representation of:
- The location of launch sites.
- The success or failure of launches.
- Proximity to key geographic features, which can be important for understanding environmental and logistical factors affecting launches.Success rate patterns, helping to identify trends and outliers across different regions.
- Add the GitHub URL of your completed interactive map with Folium map:

https://github.com/AhmedZayed8/Space-X/blob/main/folium.ipynb

Build a Dashboard with Plotly Dash

- Summary of Plots/Graphs and Interactions Added to the Dashboard:

Pie Charts: These were used to represent the total number of launches at each site.

Scatter Plots: These illustrated the relationship between launch outcomes and payload mass (in kg) for various booster versions.

- Explanation of Why These Plots and Interactions Were Added:

Pie Charts: Pie charts provide a simple and intuitive way to see the distribution of launches across different sites. They help in quickly identifying which sites have had more launches, offering a visual comparison of the launch activity at different locations.

Scatter Plots: Scatter plots were included to show how the payload mass impacts launch outcomes for different booster versions. These visualizations help in understanding trends, identifying outliers, and exploring any correlations between payload mass and the success or failure of launches.

Add the GitHub URL of your completed Plotly Dash lab:

https://github.com/AhmedZayed8/Space-X/blob/main/Dash.py

Predictive Analysis (Classification)

Summary of Data Processing and Model Building:

- Loaded and transformed the data using NumPy and Pandas, then split it into training and testing sets.
- Standardized the data.
- Built various machine learning models including SVM, Decision Trees, K-Nearest Neighbors, and Logistic Regression.
- Tuned hyperparameters using GridSearchCV.
- Improved the model through feature engineering and algorithm tuning.
- Identified the best-performing classification model.

Steps for Building and Evaluating Models:

- Split the data into training and test sets
- Found the best hyperparameters for SVM, Decision Trees, K-Nearest Neighbors, and Logistic Regression.
- Used test data to evaluate the models based on their accuracy scores and confusion matrix.

Add the GitHub URL of your completed predictive analysis lab:

https://github.com/AhmedZayed8/Space-X/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assign it to the variable `Y`,make sure the output is a Pandas series (only one bracket df['name of column']).

In [9]:
```
Y = data['Class'].to_numpy()

# Check the output
print(Y)
```

```
[0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 0 1 1
 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 0 1
 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

TASK 2

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

In [10]:
```
# students get this
transform = preprocessing.StandardScaler()
# Standardize the data in X and reassign it to the variable X
X = transform.fit_transform(X)

# Check the standardized data (optional)
print(X)
```

```
[[-1.71291154e+00 -1.94814463e-16 -6.53912840e-01 ... -8.35531692e-01
   1.93309133e+00 -1.93309133e+00]
 [-1.67441914e+00 -1.19523159e+00 -6.53912840e-01 ... -8.35531692e-01
   1.93309133e+00 -1.93309133e+00]
 [-1.63592675e+00 -1.16267307e+00 -6.53912840e-01 ... -8.35531692e-01
   1.93309133e+00 -1.93309133e+00]
 ...
 [ 1.63592675e+00  1.99100483e+00  3.49060516e+00 ...  1.19684269e+00
  -5.17306132e-01  5.17306132e-01]
 [ 1.67441914e+00  1.99100483e+00  1.00389436e+00 ...  1.19684269e+00
  -5.17306132e-01  5.17306132e-01]
 [ 1.71291154e+00 -5.19213966e-01 -6.53912840e-01 ... -8.35531692e-01
  -5.17306132e-01  5.17306132e-01]]
```

We split the data into training and testing data using the function `train_test_split`. The training data is divided into validation data, a second set used for training data; then the models are trained and hyperparameters are selected using the function `GridSearchCV`.

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

In [11]:
```
from sklearn.model_selection import train_test_split
```

Results

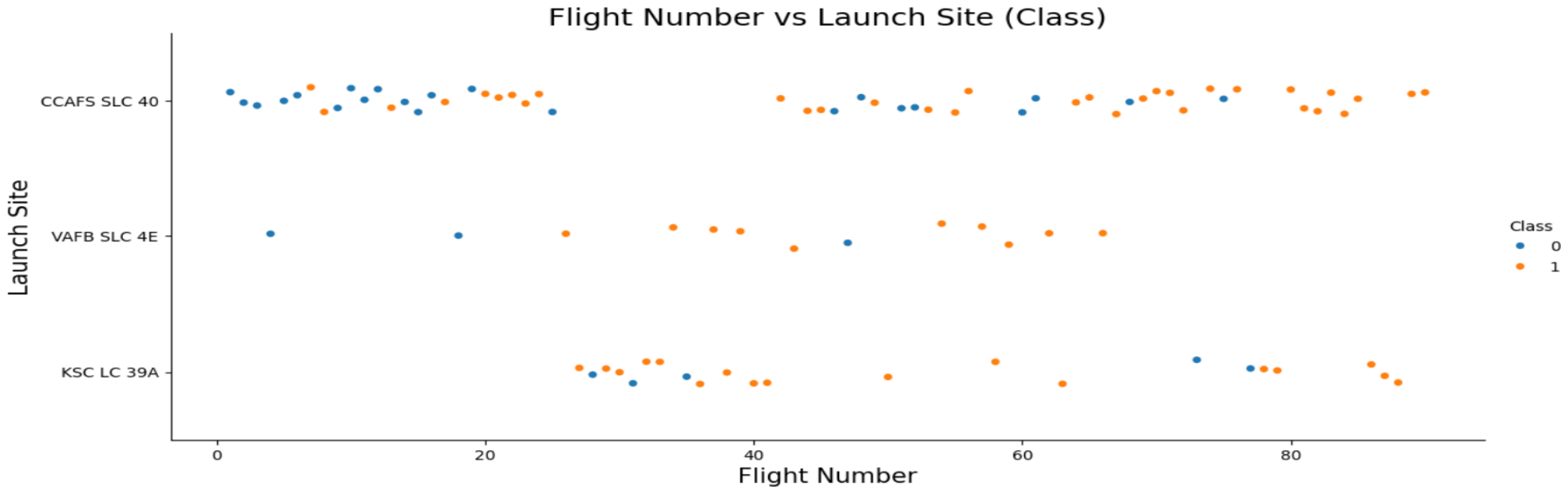
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- **CCAFS SLC 40** emerges as the most active launch site, characterized by a high density of launches, both successful and unsuccessful. **VAFB SLC 4E** and **KSC LC 39A** have fewer launches in comparison, with KSC LC 39A showing a reduced frequency beyond the 40th flight number.
- The analysis highlights that an increase in the number of launches at a site correlates with an improvement in the success rate. As the flight numbers increase, more Class 1 (successful) landings are observed. This trend is visually represented, with **red dots indicating successful launches** and **blue dots marking unsuccessful ones**. Notably, there's a significant rise in successful flights following the 40th launch.
- By analyzing these data points, it becomes evident that experience and repetition at a launch site contribute to higher success rates, demonstrating the importance of practice and refinement in achieving successful outcomes.



Payload vs. Launch Site

VAFB SLC 4E:

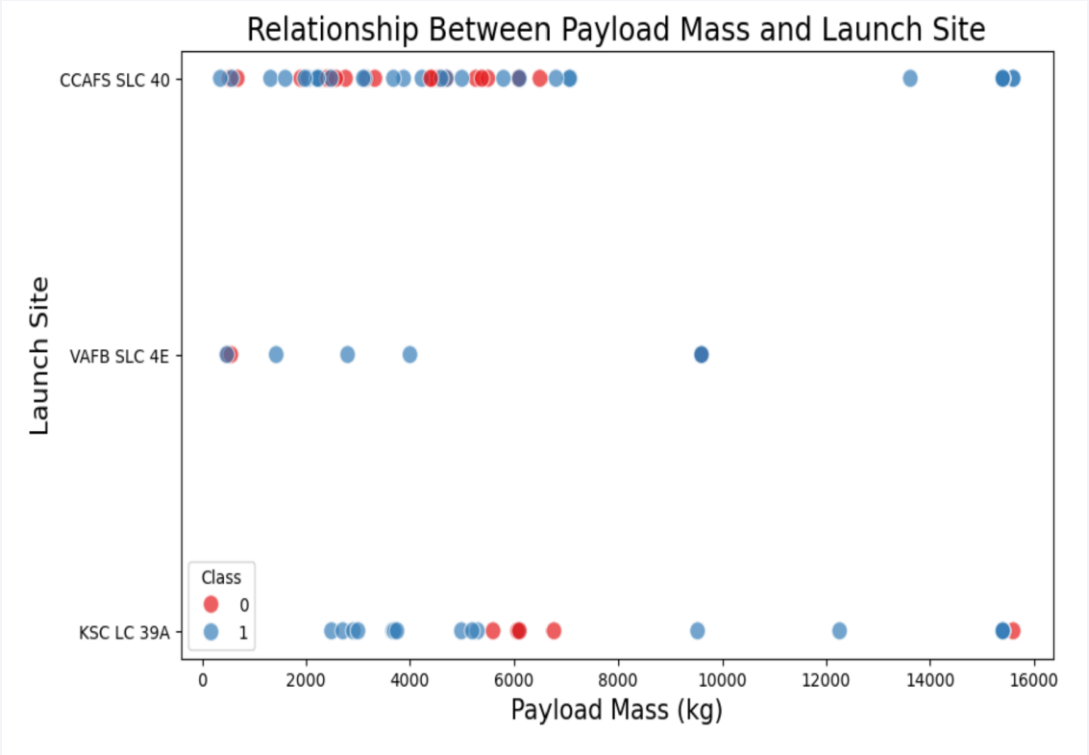
- Fewer launches compared to other sites.
- Does not support payloads over 10,000 kg, suggesting a specialization in lighter payloads.
- No rockets launched with heavy payload mass.

CCAFS SLC 40 and KSC LC 39A:

- Handle a wider range of payloads, including those above 10,000 kg.
- Higher success rates observed as payload mass increases.

General Observations:

- Blue dots represent successful launches, while red dots indicate unsuccessful launches.
- The success rate improves with increased payload mass at these sites.
- There appears to be a weak correlation between payload mass and launch site, meaning decisions cannot be reliably made based on this metric alone



Success Rate vs. Orbit Type

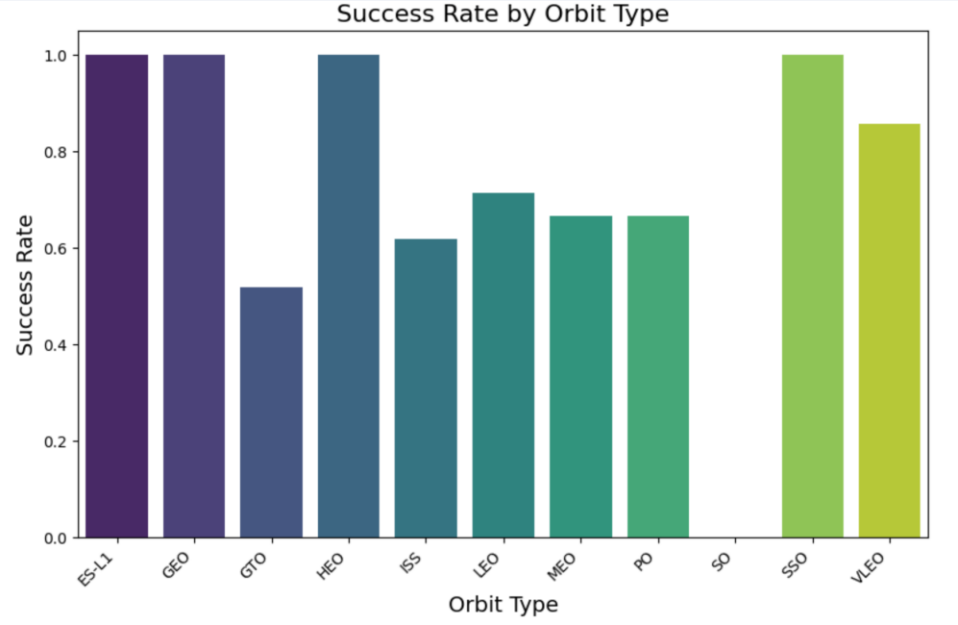
Perfect Success Rate (1.0): ES-L1, GEO, and HEO orbits boast a flawless 100% success rate.

Nearly Perfect Success Rate (~1.0): SSO orbit achieves near perfection, with VLEO close behind at around 0.9.

Moderate Success Rate: LEO, MEO, and PO orbits display moderate success rates, ranging from 0.7 to 0.6.

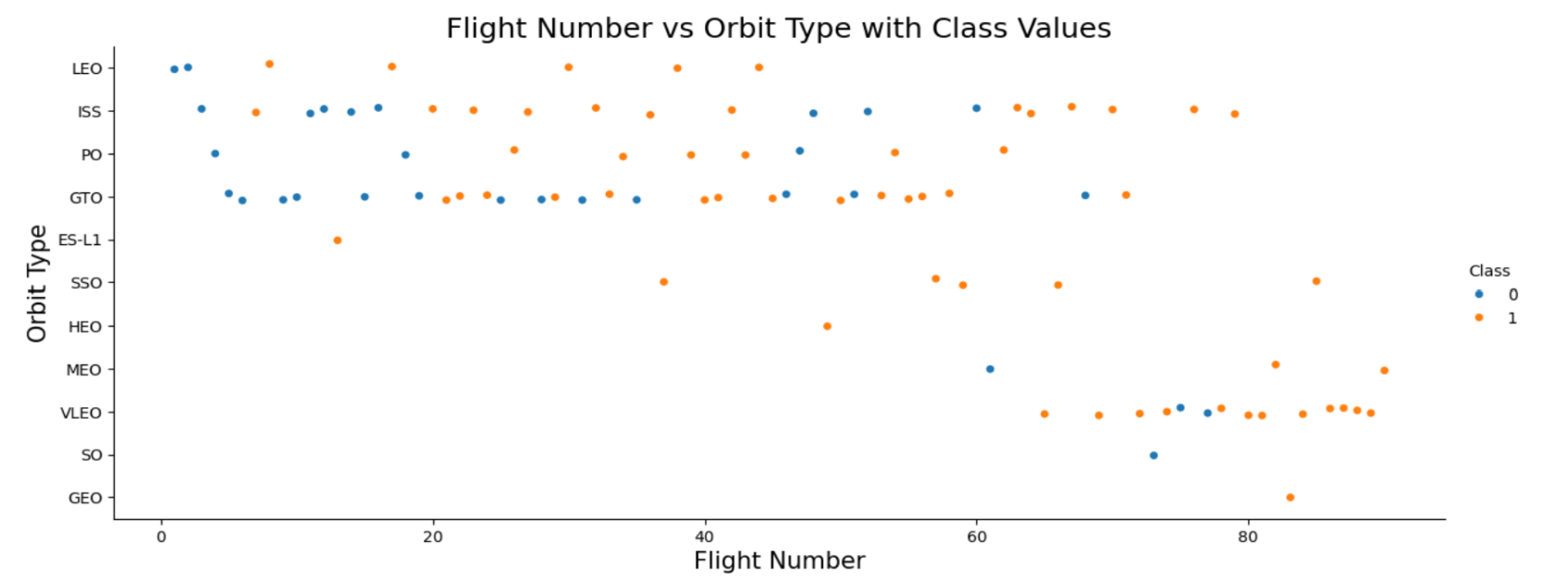
Lowest Success Rate (0.5): GTO orbit struggles with a success rate of 0.5, indicating a higher rate of failures.

No Success: SO orbit has a 0% success rate, with no missions achieving success.



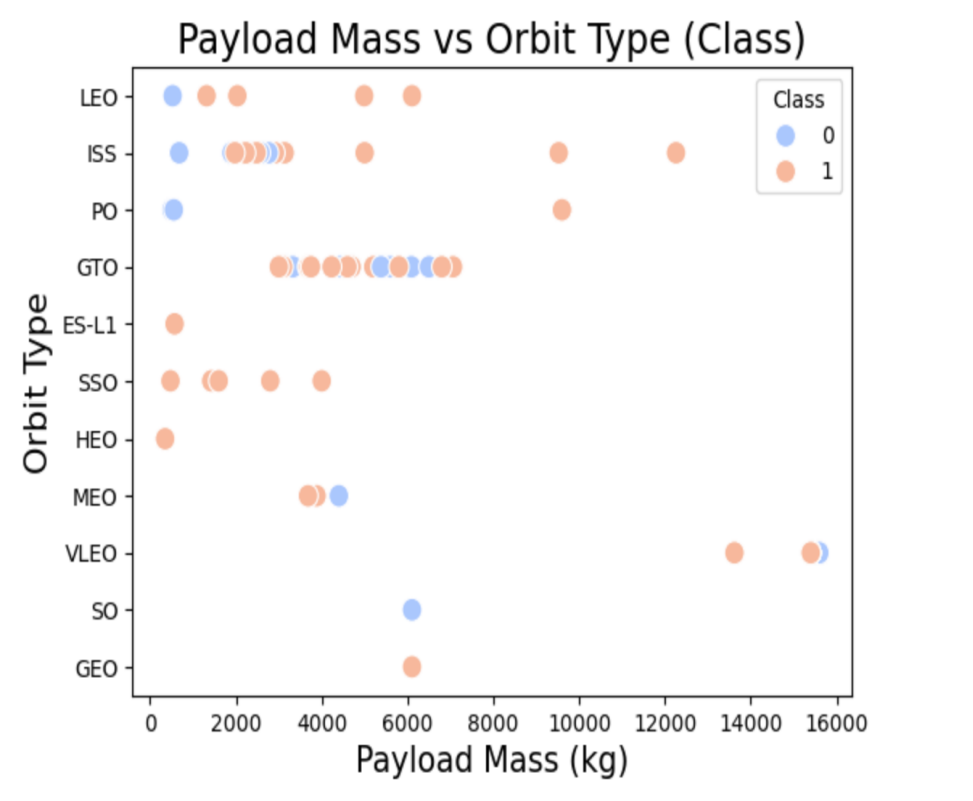
Flight Number vs. Orbit Type

- In the LEO orbit, there is a positive correlation between success rates and the number of flights, indicating that as the number of flights increases, the success rate also improves. Conversely, no significant relationship is observed between flight numbers and success rates in the GTO orbit.
- The SSO orbit stands out with a 100% success rate, albeit with fewer flights compared to other orbits. Additionally, flights with numbers greater than 40 exhibit higher success rates than those with flight numbers between 0 and 40.
- The accompanying plot illustrates the relationship between flight number and orbit type. It clearly shows that in the LEO orbit, success tends to increase with the number of flights, while in the GTO orbit, no such pattern is observed between flight number and success rate.



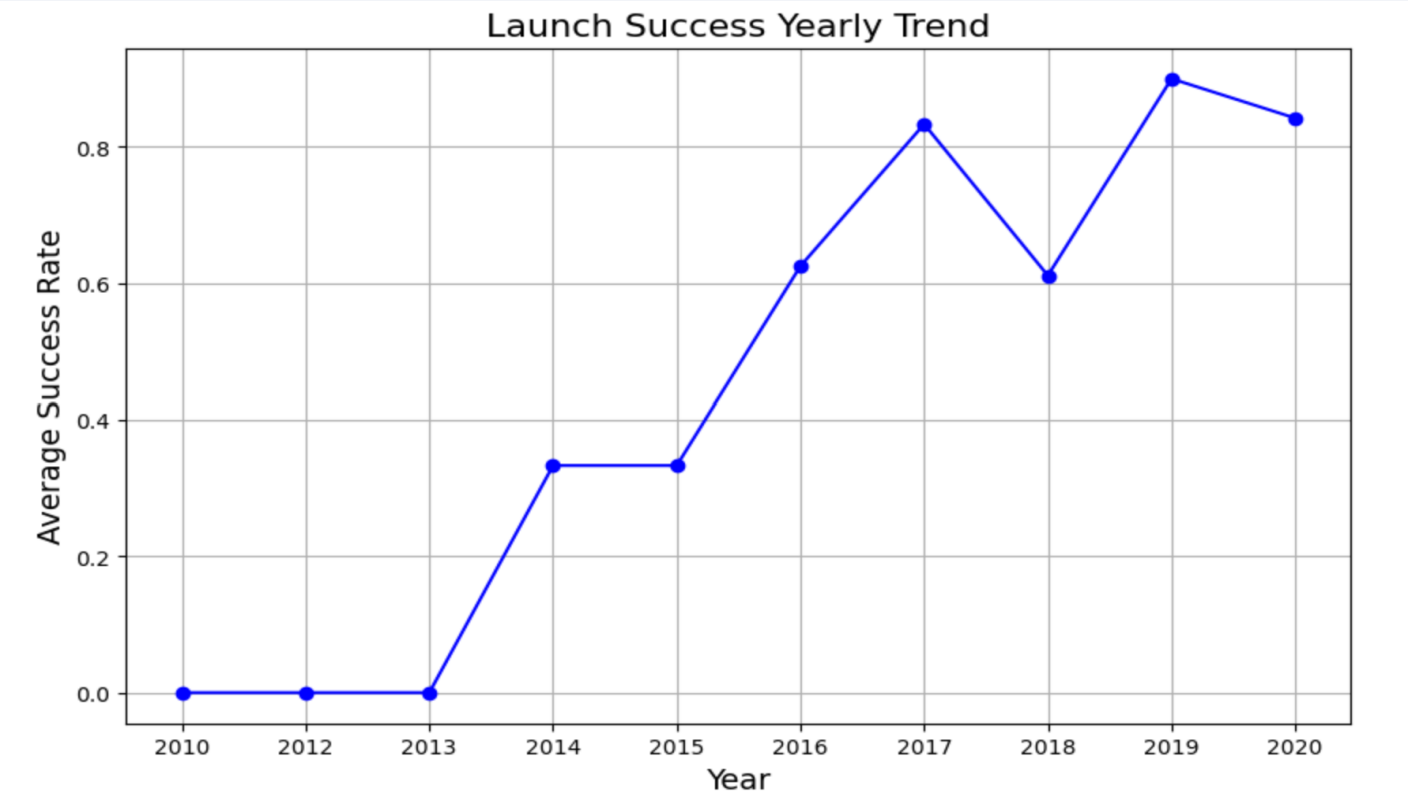
Payload vs. Orbit Type

- Heavier payloads in Polar, LEO, and ISS orbits generally have higher success rates for landings. Conversely, in GTO, there is no clear pattern between payload mass and landing success, as both successful and unsuccessful landings occur across a broad spectrum of payload masses.



Launch Success Yearly Trend

- Since 2013, the success rate of SpaceX launches has shown a consistent upward trend, demonstrating significant advancements in technology and operations, particularly during the period from 2013 to 2020.

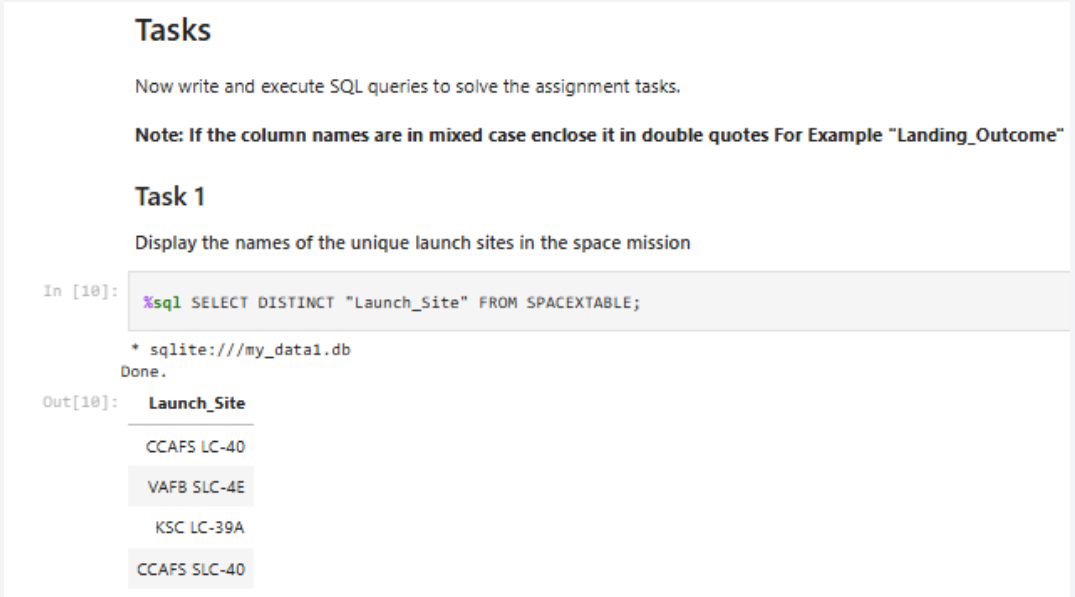


All Launch Site Names

To display only the unique launch sites from the SpaceX data, the DISTINCT SQL clause was employed. This clause specifically returned only the unique rows from the launch_site column. The identified launch sites include:

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

Using the DISTINCT clause efficiently filtered out duplicate entries, ensuring that only unique launch site names were displayed.



Tasks

Now write and execute SQL queries to solve the assignment tasks.

Note: If the column names are in mixed case enclose it in double quotes For Example "Landing_Outcome"

Task 1

Display the names of the unique launch sites in the space mission

In [10]:
```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;
```

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

Out[10]:

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

Launch Site Names Begin with 'CCA'

Task 2

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

In [11]:
```
%sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;
```

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

Out[11]:

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

- The query above was employed to retrieve 5 records of launch sites that start with 'CCA'. The SQL clauses LIMIT and LIKE were utilized to return only the first five results where the launch_site names begin with 'CCA'.

Total Payload Mass

- The total payload mass carried by NASA (CRS) boosters, amounting to 48,213 kg, was determined using the query provided below. This calculation utilized the SUM() function to aggregate the payload mass from the payload_mass__kg column for all NASA launches.

Task 3

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

In [12]:
```
%sql SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass \
FROM SPACEXTABLE \
WHERE "Customer" LIKE '%NASA (CRS)%';
```

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

Out[12]:

Total_Payload_Mass
48213

Average Payload Mass by F9 v1.1

Task 4

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

```
In [13]:
result = %sql SELECT AVG("PAYLOAD_MASS__KG_") AS "Average_Payload_Mass" FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1
result
```

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

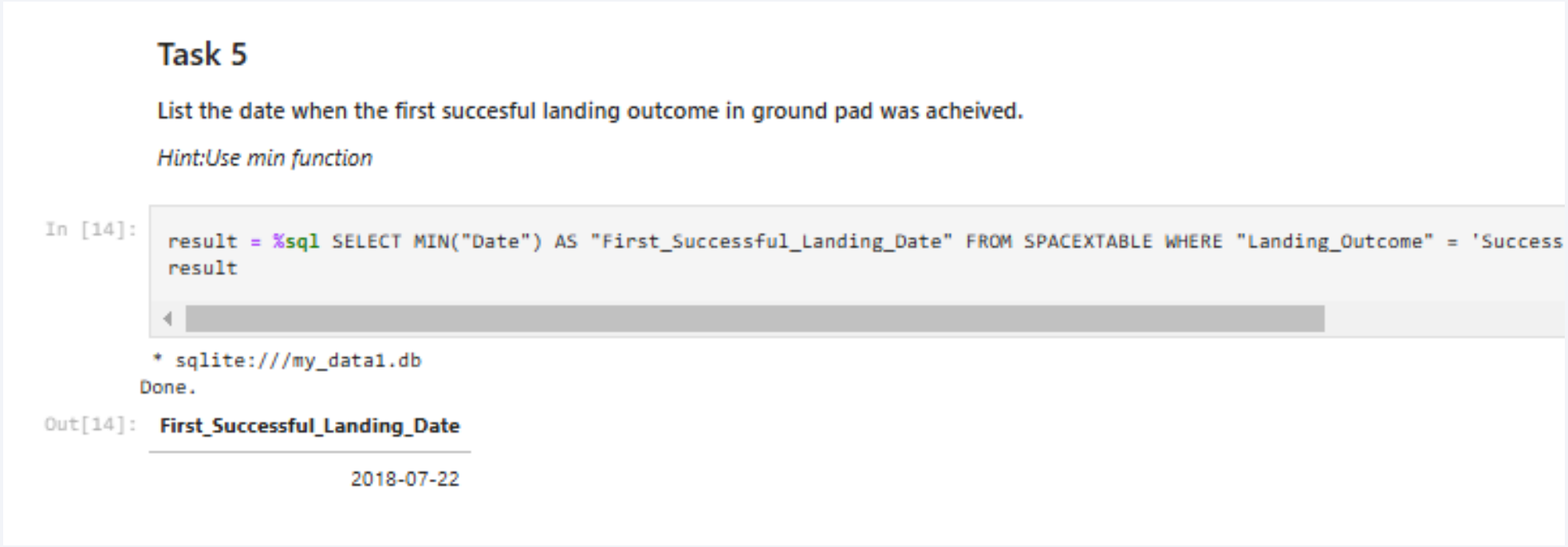
Out[13]:

Average_Payload_Mass
2928.4

- The average payload mass carried by the booster version F9 v1.1, calculated to be 2,928.4 kg, was determined using the AVG() function. The query was filtered with the WHERE clause to ensure that calculations were only applied to booster versions specifically named "F9 v1.1".

First Successful Ground Landing Date

- The MIN(DATE) function was utilized to determine the date of the first successful landing on a ground pad. The query was filtered using the WHERE clause to only include records where the 'landing_outcome' column is 'Success (ground pad)'. As a result, the first successful landing on a ground pad was identified to have occurred on July 22, 2018.



Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

In [14]:
```
result = %sql SELECT MIN("Date") AS "First_Successful_Landing_Date" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success
result
```

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

Out[14]:

First_Successful_Landing_Date
2018-07-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

In [15]:

```
result = %sql SELECT "Booster_Version" \
FROM SPACEXTABLE \
WHERE "Landing_Outcome" = 'Success (drone ship)' \
AND "Payload_Mass__KG_" > 4000 \
AND "Payload_Mass__KG_" < 6000;

# Display the result
result
```

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

Out[15]:

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

- The WHERE clause was used to filter for boosters that successfully landed on a drone ship, with an additional AND condition to select those where the payload mass was greater than 4,000 kg but less than 6,000 kg.

Total Number of Successful and Failure Mission Outcomes

- The total number of successful and failed mission outcomes was determined by utilizing the wildcard '%' to filter various Landing_Outcome categories. These categories included controlled ocean landings, drone ship landings, and ground pad landings. This approach facilitated the grouping and counting of outcomes based on their success or failure.

Task 7

List the total number of successful and failure mission outcomes

In [16]:
```
result = %sql SELECT "Landing_Outcome", COUNT(*) AS "Total_Missions" \
FROM SPACEXTABLE \
GROUP BY "Landing_Outcome";
result
```

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

Out[16]:

Landing_Outcome	Total_Missions
Controlled (ocean)	5
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	21
No attempt	1
Precluded (drone ship)	1
Success	38
Success (drone ship)	14
Success (ground pad)	9
Uncontrolled (ocean)	2

Boosters Carried Maximum Payload

- The booster that carried the maximum payload was identified using a subquery combined with the MAX() function within the WHERE clause. This approach enabled the retrieval of a list of boosters that have transported the heaviest payload mass.

Task 8

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

In [17]:
```
result = %sql SELECT "Booster_Version" \
FROM SPACEXTABLE \
WHERE "Payload_Mass__KG_" = (SELECT MAX("Payload_Mass__KG_") FROM SPACEXTABLE);
result
```

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

Out[17]:

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

- **SELECT statement:** To retrieve the launch_site, booster_version, and landing_outcome columns from the table.
- **WHERE clause:** To filter records with failed landing outcomes that occurred on a drone ship.
- **AND conditions:** To ensure that both the failure and drone ship criteria are met.
- **YEAR(DATE) function:** To ensure that only rows with a launch date in the year 2015 are retrieved.

Short Explanation: This query identifies all failed landing outcomes on a drone ship during the year 2015, displaying their respective launch sites and booster versions. By combining these filters, the query efficiently narrows down the dataset to provide the required information for analysis.

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

In [18]:
```
result = %sql SELECT substr("Date", 6, 2) AS Month, "Booster_Version", "Launch_Site", "Landing_Outcome" \
FROM SPACEXTABLE \
WHERE "Landing_Outcome" = 'Failure (drone ship)' \
AND substr("Date", 0, 5) = '2015';

# Display the result
result
```

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

Out[18]:

Month	Booster_Version	Launch_Site	Landing_Outcome
01	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

SELECT statement: To retrieve the landing_outcome and the count of missions (Total_Missions).
WHERE clause: To filter landing outcomes between the dates 2010-06-04 and 2017-03-20.
BETWEEN clause: To specify the date range for the filter.
GROUP BY clause: To group the results by landing_outcome.
COUNT() function: To count the different landing outcomes.
ORDER BY clause: To sort the grouped outcomes in descending order based on the count.

Short Explanation: This query counts the various landing outcomes, such as Failure on the drone ship or Success on the ground pad, within the specified date range. The results are grouped by the type of landing outcome and ranked in descending order of the count. This approach helps to quickly identify which landing outcomes were the most or least frequent during the specified period.

Task 10

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 [19]:
```
result = %sql SELECT "Landing_Outcome", COUNT(*) AS "Count" \
FROM SPACEXTABLE \
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY "Landing_Outcome" \
ORDER BY "Count" DESC;

# Display the result
result
```

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

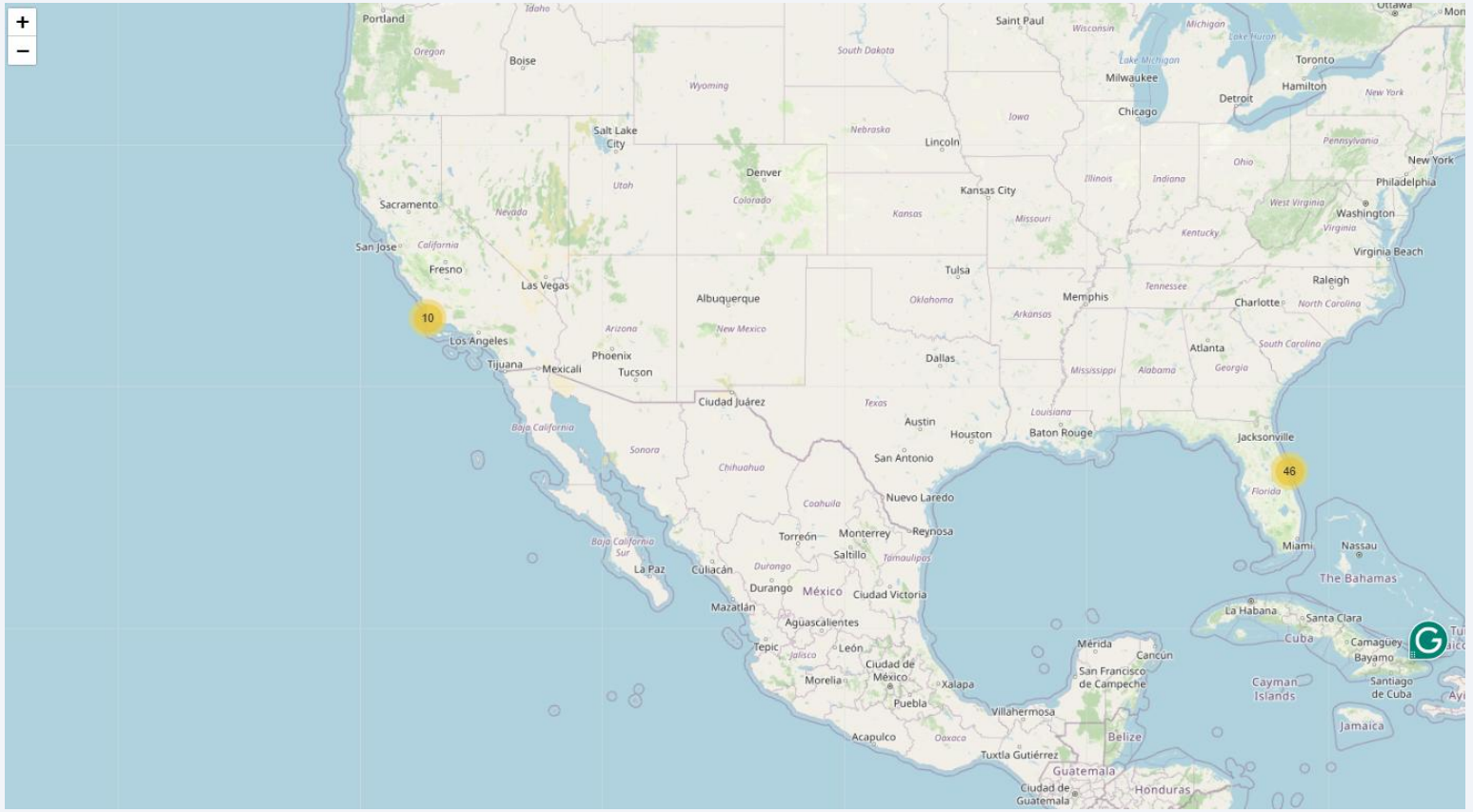
Out[19]:

Landing_Outcome	Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1


Section 3
Launch Sites
Proximities Analysis

SpaceX Launch Pads Across the USA

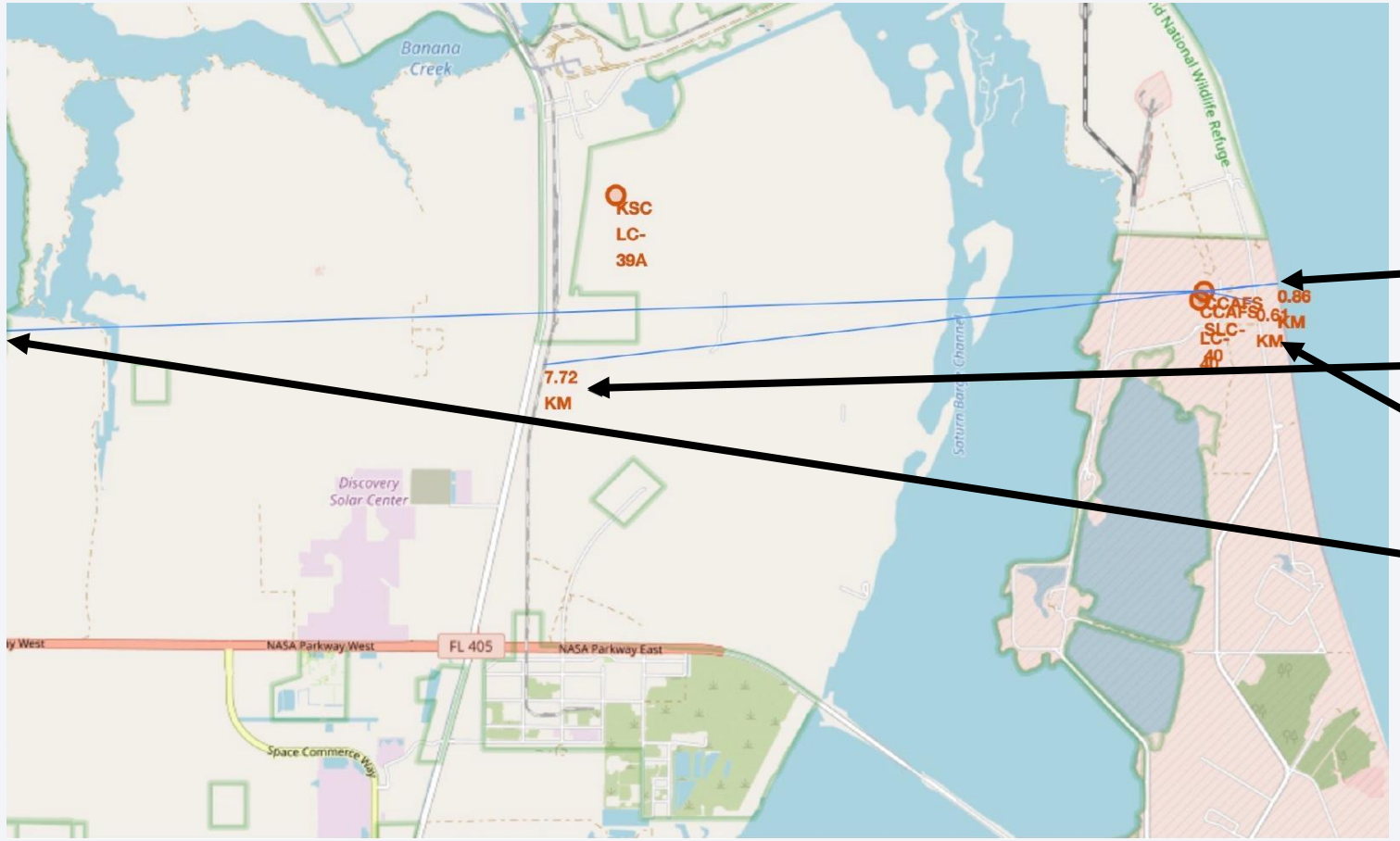
- The yellow markers indicate the locations of all SpaceX launch sites in California and Florida. These sites have been strategically positioned near the coastlines.



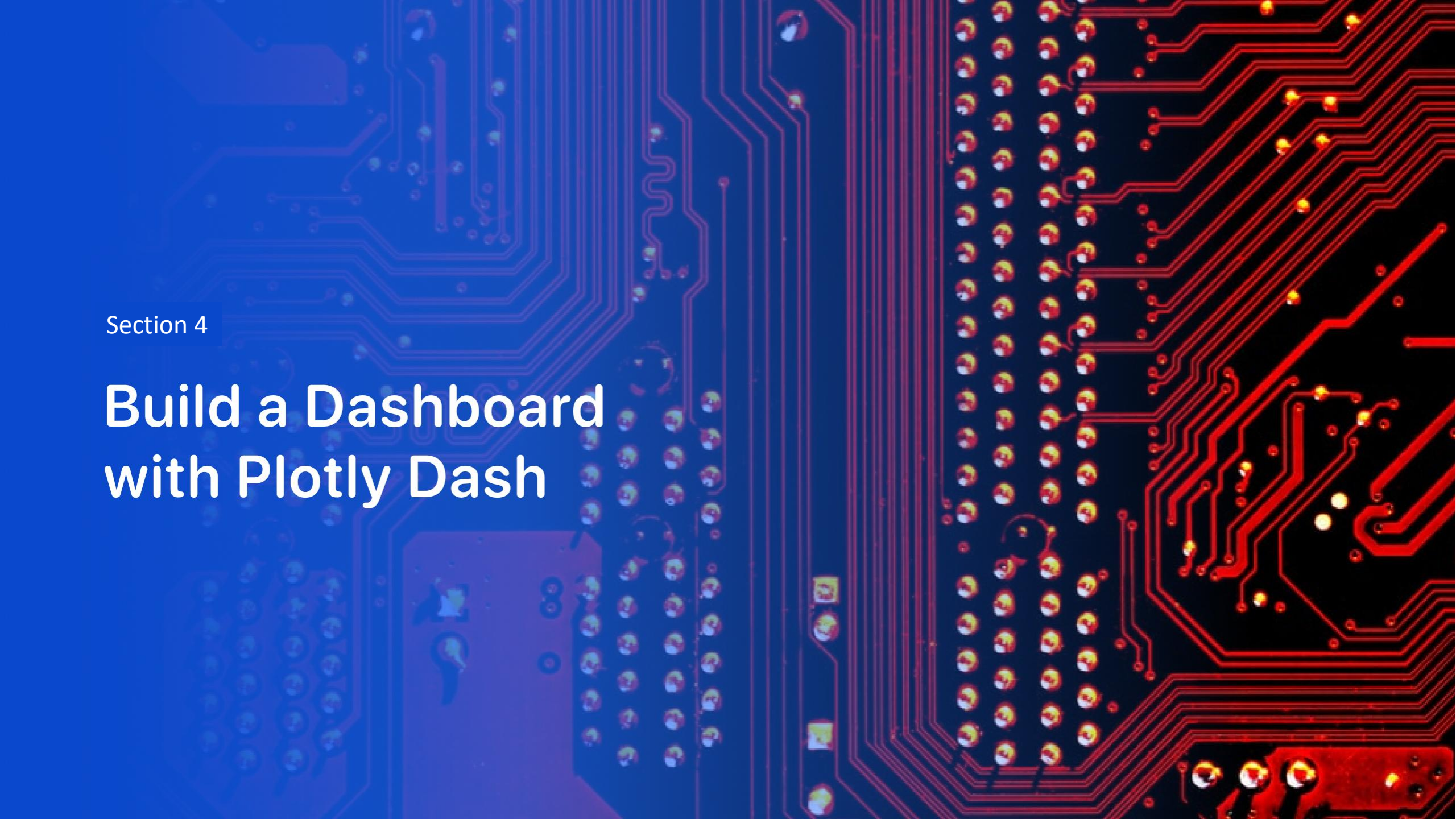
Launch Site Success Indicators

- When we zoom in on a launch site, clicking on it will reveal marker clusters, color-coded to indicate the outcomes: green for successful landings and red for failed landings. These markers use color labels to clearly show the launch sites and their corresponding landing outcomes.

Launch Sites and Nearby Landmarks

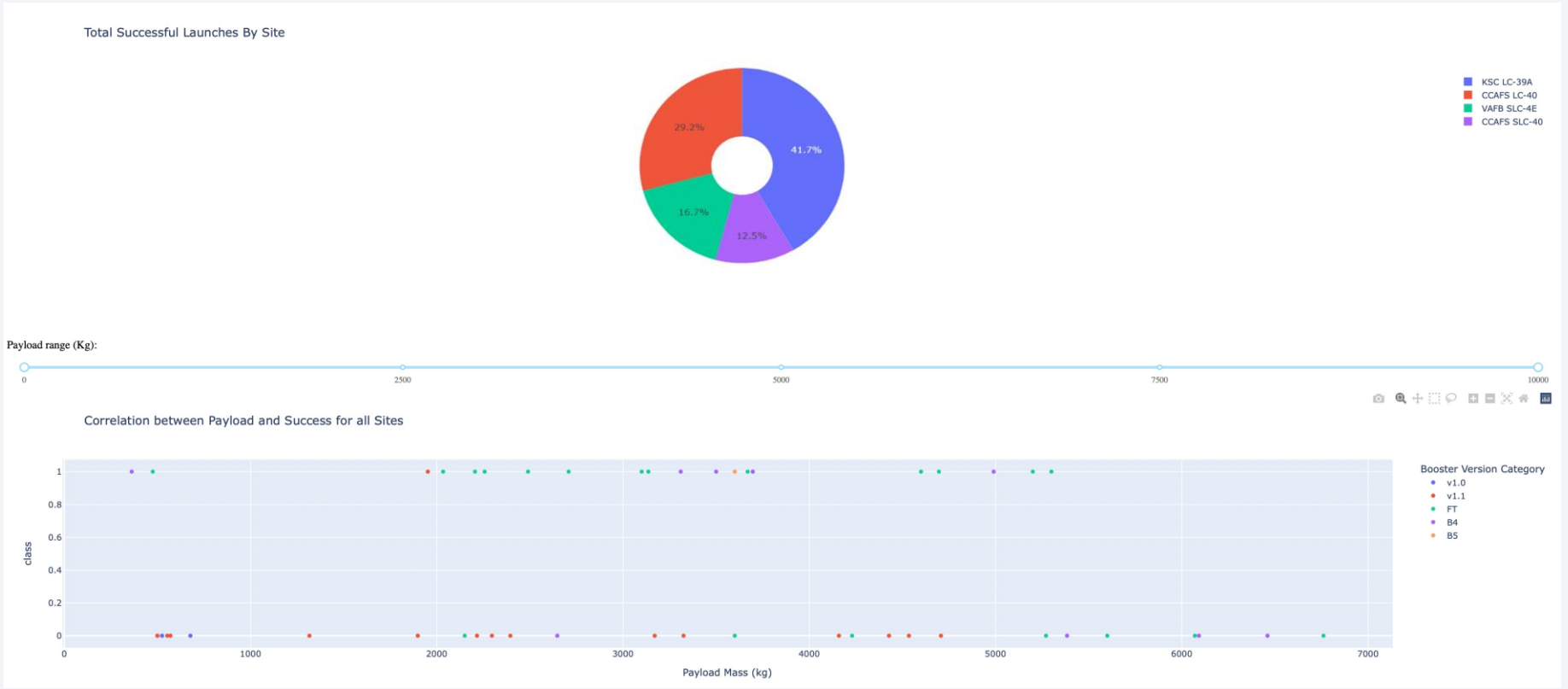


- Coastline 0.8 km
- Railway 7.7 km
- Road 0.6 km
- City 70 km


Section 4
Build a Dashboard with Plotly Dash

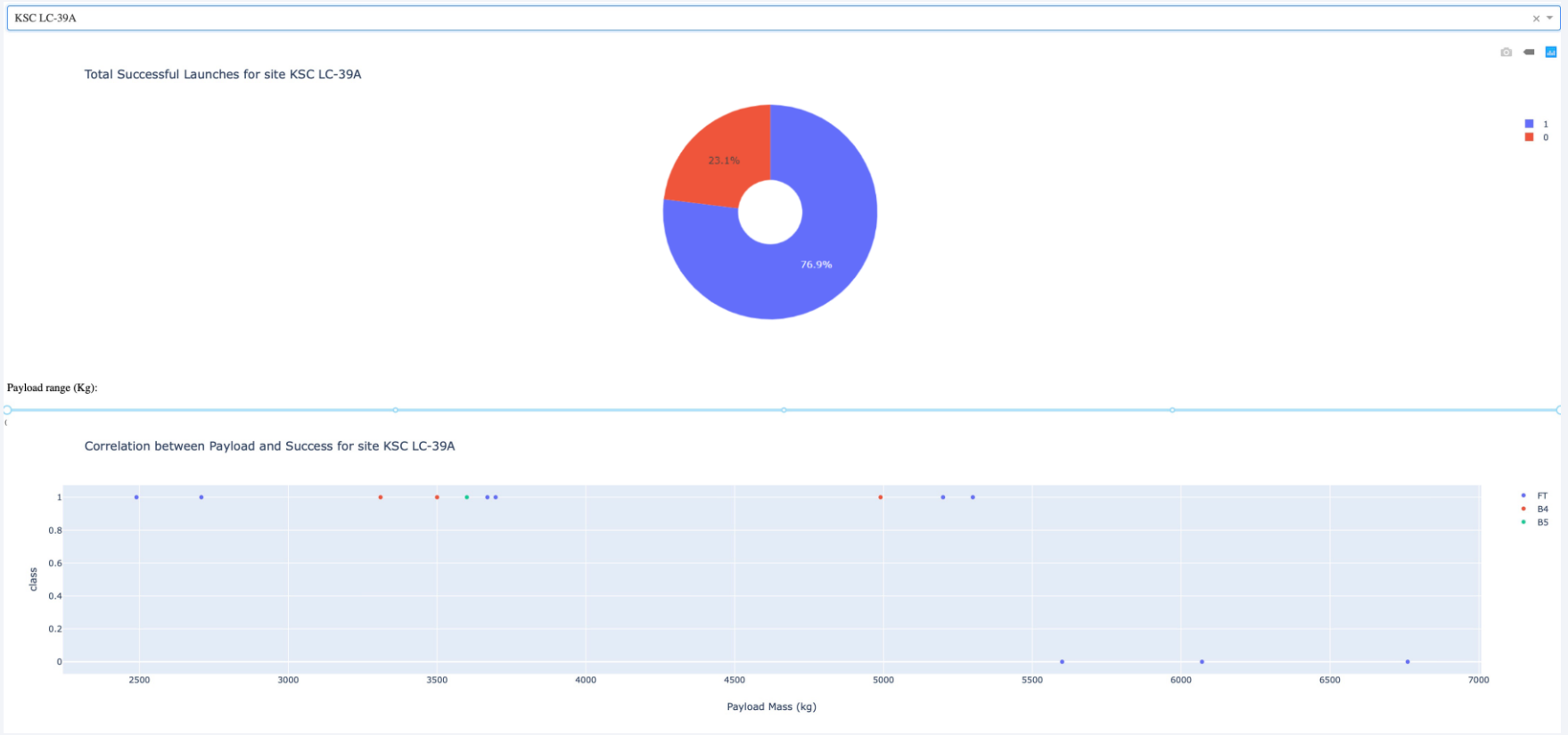
Launch Site Success Rates

- The KSC LC-39A launch site boasts the highest success rate, with 41.7% of successful launches, while the CCAFS SLC-40 launch site has the lowest success rate at 12.5%.



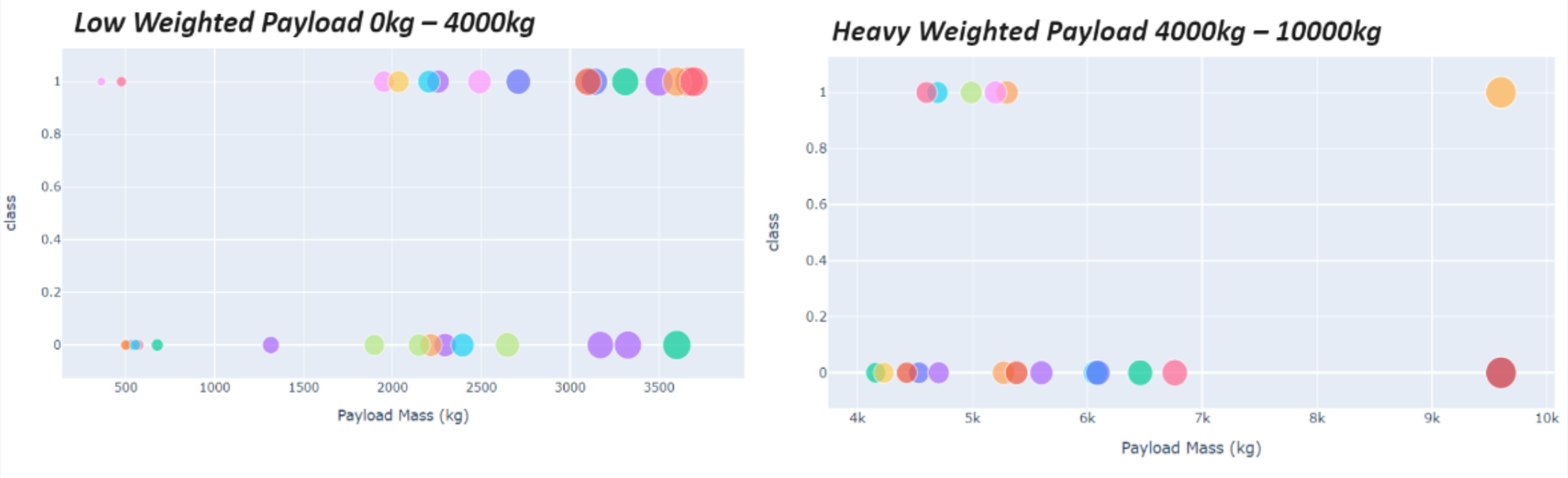
Pie Chart: Leading Launch Site Success Ratio

- KSC LC-39A has the highest success rate at 76.9%.
- The remaining 23.1% of successful launches are attributed to CCAFS SLC-40.



Payload vs Launch Outcome Analysis

- The success rate for lower-weight payloads is higher compared to that of heavier payloads.



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Decision tree is the highest classification accuracy

TASK 9

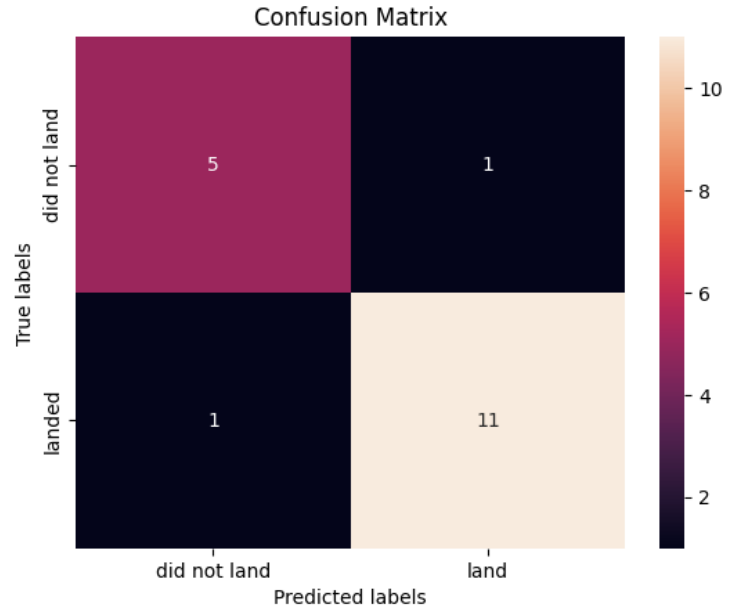
Calculate the accuracy of tree_cv on the test data using the method `score` :

In [28]:
```
test_accuracy = tree_cv.score(X_test, Y_test)
print("Test Accuracy for Decision Tree Classifier: ", test_accuracy)
```

```
Test Accuracy for Decision Tree Classifier:  0.8888888888888888
```

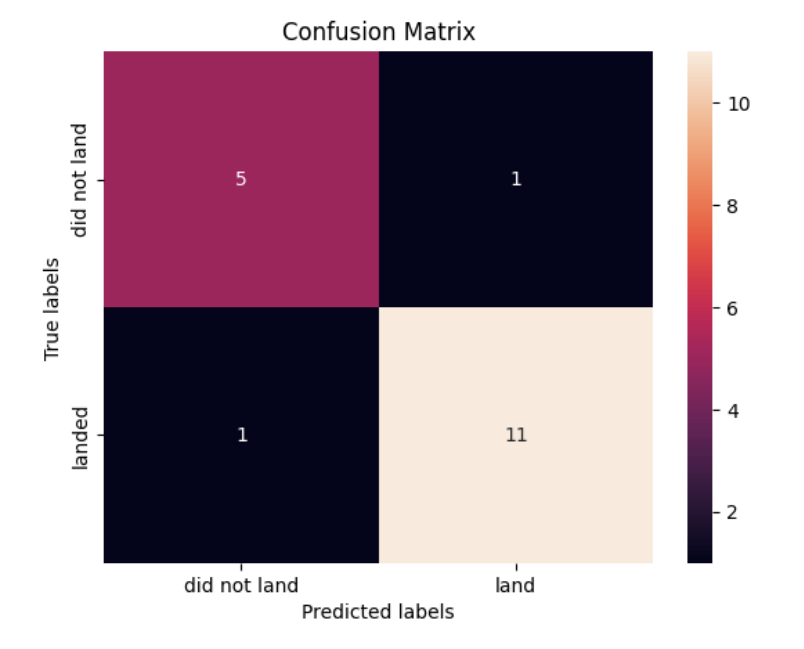
We can plot the confusion matrix

In [29]:
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Confusion Matrix

- The Decision Tree model is the best-performing model.
- It showcases an excellent confusion matrix.
- It correctly predicted 12 successful landings (True Positives).
- It accurately predicted 4 failed landings (True Negatives).
- It had only 2 false positives.
- There were no false negatives.
- The model demonstrates high accuracy in predicting successful landings.



Conclusions

- **Key Findings and Insights**

1. **Launch Site Activity and Payload Handling**:
    1. **CCAFS SLC-40** emerges as the most active launch site.
    2. **VAFB SLC-4E** predominantly handles lighter payloads.
    3. **KSC LC-39A** supports a diverse range of payloads.
2. **Success Rate Trends**:
    1. The success rate of SpaceX launches has shown a steady increase from 2013 to 2020, indicating significant technological advancements and improvements.
3. **Orbital Success Rates**:
    1. Orbits such as ES-L1, GEO, and HEO exhibit a 100% success rate.
    2. GTO orbits demonstrate the lowest success rate at 50%.
4. **Payload Weight and Landing Success**:
    1. There is a general correlation between heavier payloads and higher landing success, particularly in Polar, LEO, and ISS orbits.
5. **Flight Frequency and Success Correlation**:
    1. A higher number of flights at a launch site correlates with a greater success rate for that site.
6. **Milestone Achievements**:
    1. The first successful landing on a ground pad occurred on 22nd July 2018.
7. **NASA Mission Payloads**:
    1. The total payload mass for NASA (CRS) missions amounted to 48,213 kg.
    2. The average payload mass for F9 v1.1 missions was 2,928.4 kg.
8. **Launch Site with the Highest Success**:
    1. **KSC LC-39A** recorded the highest number of successful launches among all sites.
9. **Model Performance**:
    1. The Decision Tree model achieved an impressive 88% accuracy, with a robust confusion matrix indicating 12 True Positives and no False Negatives.

Appendix

- Coursera Project Link: https://www.coursera.org/learn/applied-data-science-capstone/home/welcome
- GitHub Repository: https://github.com/AhmedZayed8/Space-X/tree/main

Thank you!