IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

JS
03.10.2023

Outline

Executive Summary

Summary of methodologies

- Data Collection through API
- Data Collection with Web Scraping
- Data Wrangling
- Exploratory Data Analysis with SQL
- Exploratory Data Analysis with Data Visualization
- Interactive Visual Analytics with Folium
- Machine Learning Prediction

Summary of all results

- Exploratory Data Analysis
- Interactive analytics
- Predictive Analytics

Introduction

SpaceX's success is rooted in cost-effective rocket launches. They advertise Falcon 9 launches for $62 million, significantly cheaper than competitors charging $165 million or more. The key to these savings lies in SpaceX's first stage recovery. However, this approach faces challenges:

1. **Landing Reliability:** The first stage doesn't always land successfully; sometimes, it crashes.
2. **Mission Constraints:** SpaceX might sacrifice the first stage based on mission factors like payload, orbit, or customer needs.

In essence, SpaceX's cost-effectiveness depends on the first stage's landing, a variable influenced by technical hurdles and mission-specific requirements.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - Data was collected using SpaceX API and web scraping
- Perform data wrangling
    - One-hot encoding was applied to 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

1. **Data Collection:**
   Gathered data from two sources: SpaceX API and web scraping Wikipedia.

2. **Data Wrangling:**
   Enriched the collected data by creating a landing outcome label based on feature analysis.

3. **Exploratory Data Analysis (EDA):**
   Conducted EDA using SQL and visualizations to understand data patterns.

4. **Interactive Visual Analysis:**
   Developed a dashboard for a user-friendly exploration of the dataset.

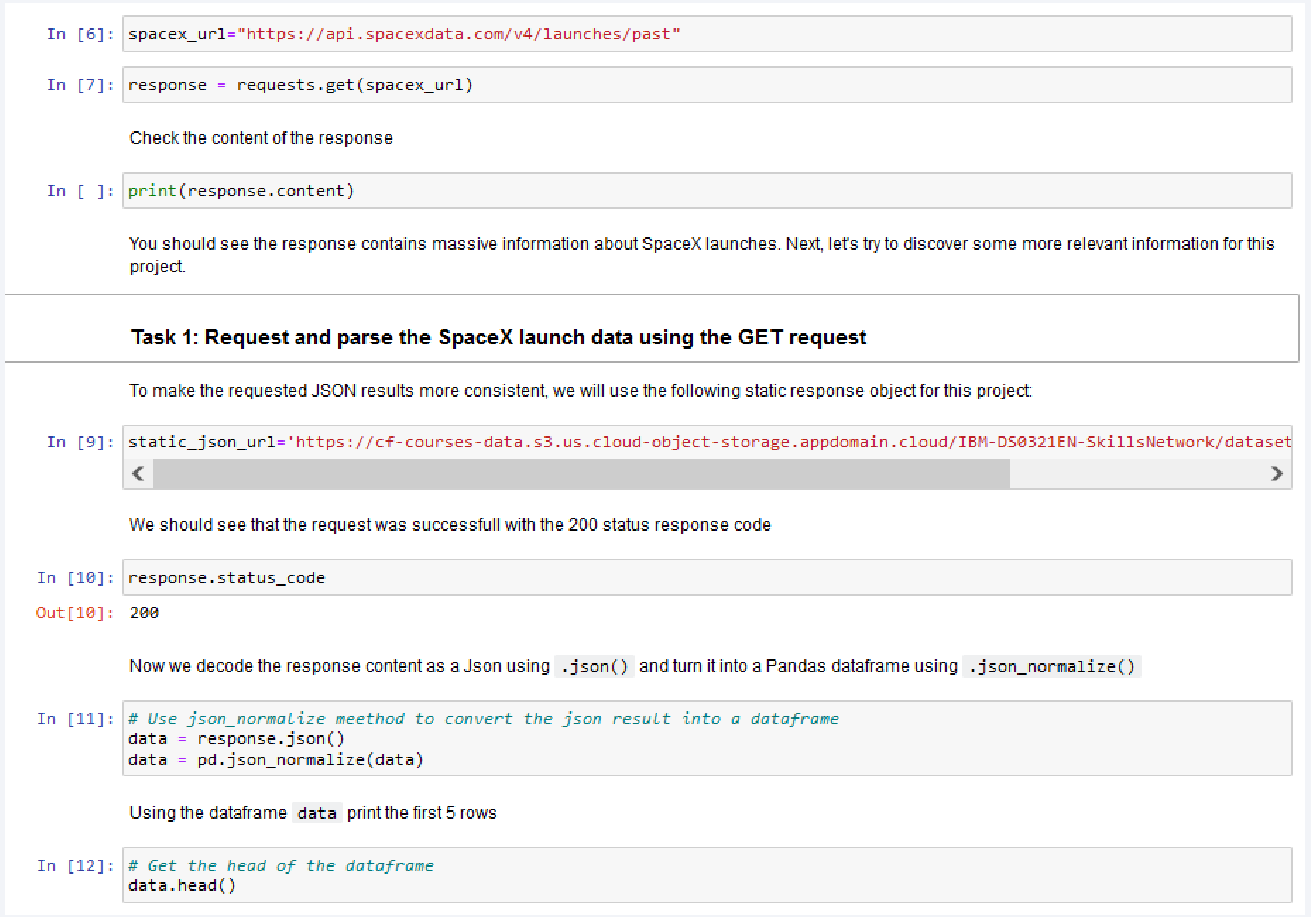
5. **Machine Learning Predictive Analysis:**
   Prepared data by normalization and splitting into training and test sets.
   Employed four classification models (Logistic Regression, SVM, Decision Tree, KNN).
   Evaluated model accuracy using different parameter combinations.

Data Collection – SpaceX API

We utilized a GET request to access the SpaceX API.

GitHub URL:

https://github.com/jseelhoefer/Winning-Space-Race-with-Data-Science/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)
```

Check the content of the response

```
In [ ]: print(response.content)
```

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

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/dataset
```

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

```
In [10]: response.status_code
Out[10]: 200
```

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

```
In [11]: # Use json_normalize meethod to convert the json result into a dataframe
         data = response.json()
         data = pd.json_normalize(data)
```

Using the dataframe `data` print the first 5 rows

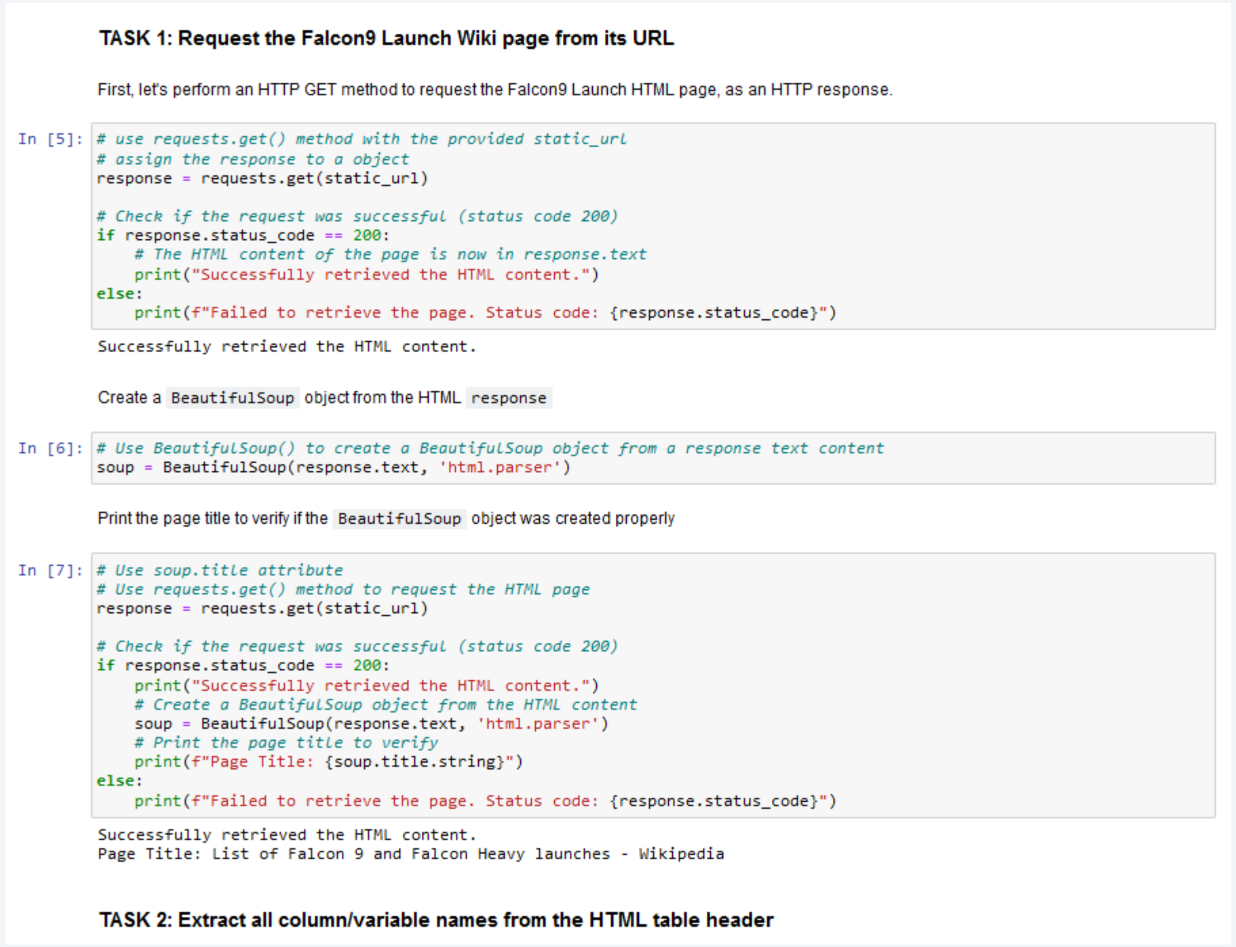
```
In [12]: # Get the head of the dataframe
         data.head()
```

Data Collection - Scraping

We used Webscraping from Wikipedia to get the data and did some further Data Wrangling

GitHub URL:

https://github.com/jseelhoefer/Winning-Space-Race-with-Data-Science/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 [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        response = requests.get(static_url)

        # Check if the request was successful (status code 200)
        if response.status_code == 200:
            # The HTML content of the page is now in response.text
            print("Successfully retrieved the HTML content.")
        else:
            print(f"Failed to retrieve the page. Status code: {response.status_code}")

        Successfully retrieved the HTML content.
```

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

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

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

```
In [7]: # Use soup.title attribute
        # Use requests.get() method to request the HTML page
        response = requests.get(static_url)

        # Check if the request was successful (status code 200)
        if response.status_code == 200:
            print("Successfully retrieved the HTML content.")
            # Create a BeautifulSoup object from the HTML content
            soup = BeautifulSoup(response.text, 'html.parser')
            # Print the page title to verify
            print(f"Page Title: {soup.title.string}")
        else:
            print(f"Failed to retrieve the page. Status code: {response.status_code}")

        Successfully retrieved the HTML content.
        Page Title: List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

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

Data Wrangling

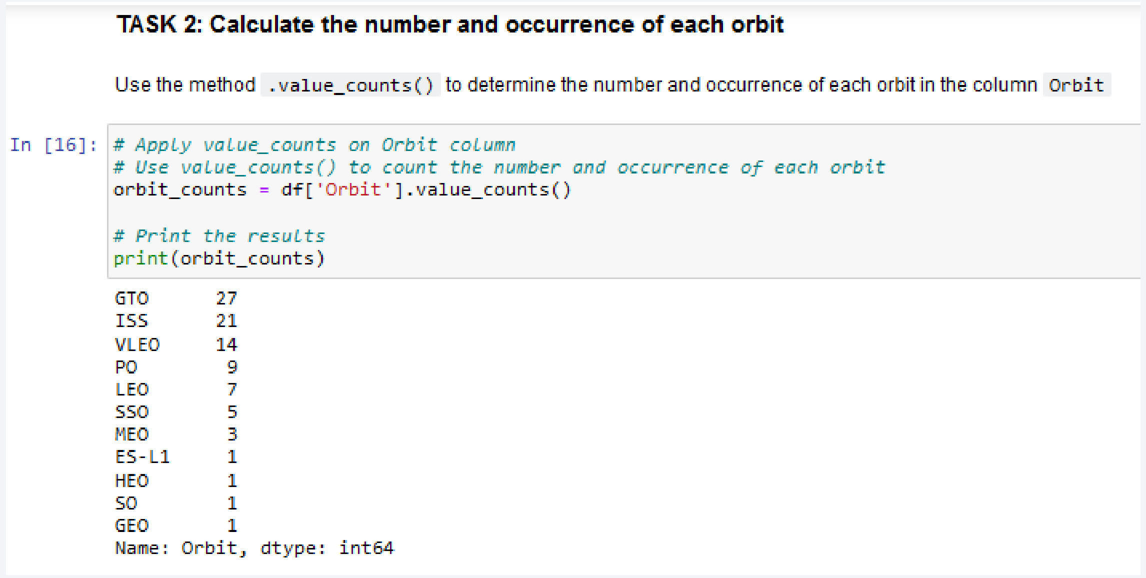
We performed exploratory data analysis by conducting an analysis to determine the frequency of launches at various sites, as well as the distribution and occurrence of different orbits.

Subsequently, we generated a landing outcome label based on the data in the outcome column and then exported the findings to a CSV file.

As you can see on the right side we calculated the number and occurrence of each orbit.

GitHub URL for:

https://github.com/jseelhoefer/Winning-Space-Race-with-Data-Science/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_1_L3_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb



TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

In [16]:
```
# Apply value_counts on Orbit column
# Use value_counts() to count the number and occurrence of each orbit
orbit_counts = df['Orbit'].value_counts()

# Print the results
print(orbit_counts)
```

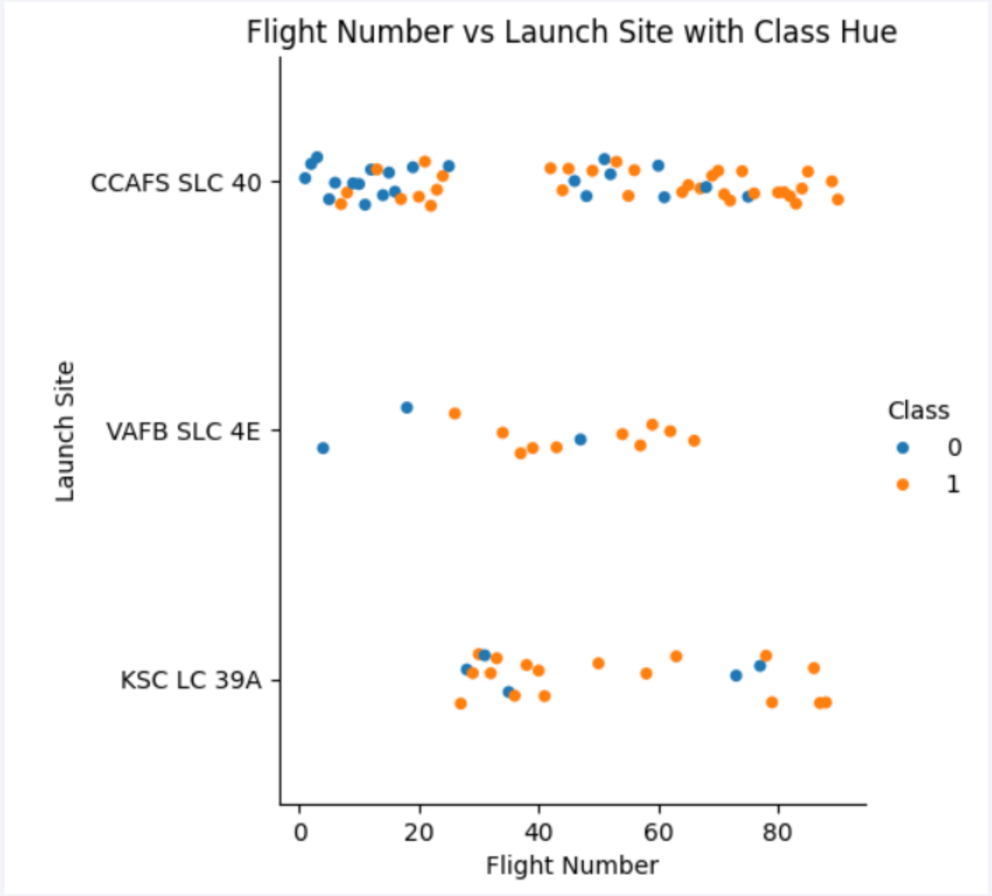
```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

EDA with Data Visualization

We visually analyzed the data to examine the relationships between flight numbers and launch sites, payloads and launch sites, success rates for different orbit types, flight numbers and orbit types, as well as the annual trend in launch success.

GitHub URL:

https://github.com/jseelhoefer/Winning-Space-Race-with-Data-Science/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb



EDA with SQL

We imported the SpaceX dataset into a PostgreSQL database within our Jupyter notebook. Using SQL, we extracted key insights, including:

- Unique launch site names.
- Total payload mass for NASA (CRS) launches.
- Average payload mass for F9 v1.1 boosters.
- Total successful and failed mission outcomes.
- Details of failed drone ship landings, including booster version and launch site.

These queries provided valuable insights into the SpaceX dataset, enhancing our understanding of the missions and their results.

GitHub URL:

https://github.com/jseelhoefer/Winning-Space-Race-with-Data-Science/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

Our tasks were:

1. **Mapping Launch Sites:** We've mapped our launch sites with markers, circles, and lines.
2. **Outcome Classification:** We've categorized launches as "failure" (Class 0) or "success" (Class 1).
3. **Success Rate Enumeration:** Using color-coded markers, we've identified high-success launch sites.
4. **Distance Analysis:** We've calculated distances to railways, highways, coastlines, and cities.

In summary, our mapping, outcome classification, success rate assessment, and distance analysis provide insights into launch site performance and proximity to critical features.

GitHub URL:

https://github.com/jseelhoefer/Winning-Space-Race-with-Data-Science/blob/main/module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

1. We've displayed pie charts that illustrate the total launches at specific sites.
2. We've also included scatter plots that depict the connection between launch outcomes and payload mass (in kilograms) across various booster versions.

- Explain why you added those plots and interactions

GitHub URL:

https://github.com/jseelhoefer/Winning-Space-Race-with-Data-Science/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

First, we enumerated the steps in our data analysis process:

1. Data loading was carried out using numpy and pandas.
2. Data transformation and preprocessing followed suit.
3. We divided the data into training and testing subsets.
4. Multiple machine learning models were constructed and fine-tuned using GridSearchCV.
5. Evaluation of model performance was based on accuracy.
6. We iteratively improved model performance through feature engineering and algorithm tuning.
7. Finally, we determined the top-performing classification model.

GitHub URL:

https://github.com/jseelhoefer/Winning-Space-Race-with-Data-Science/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

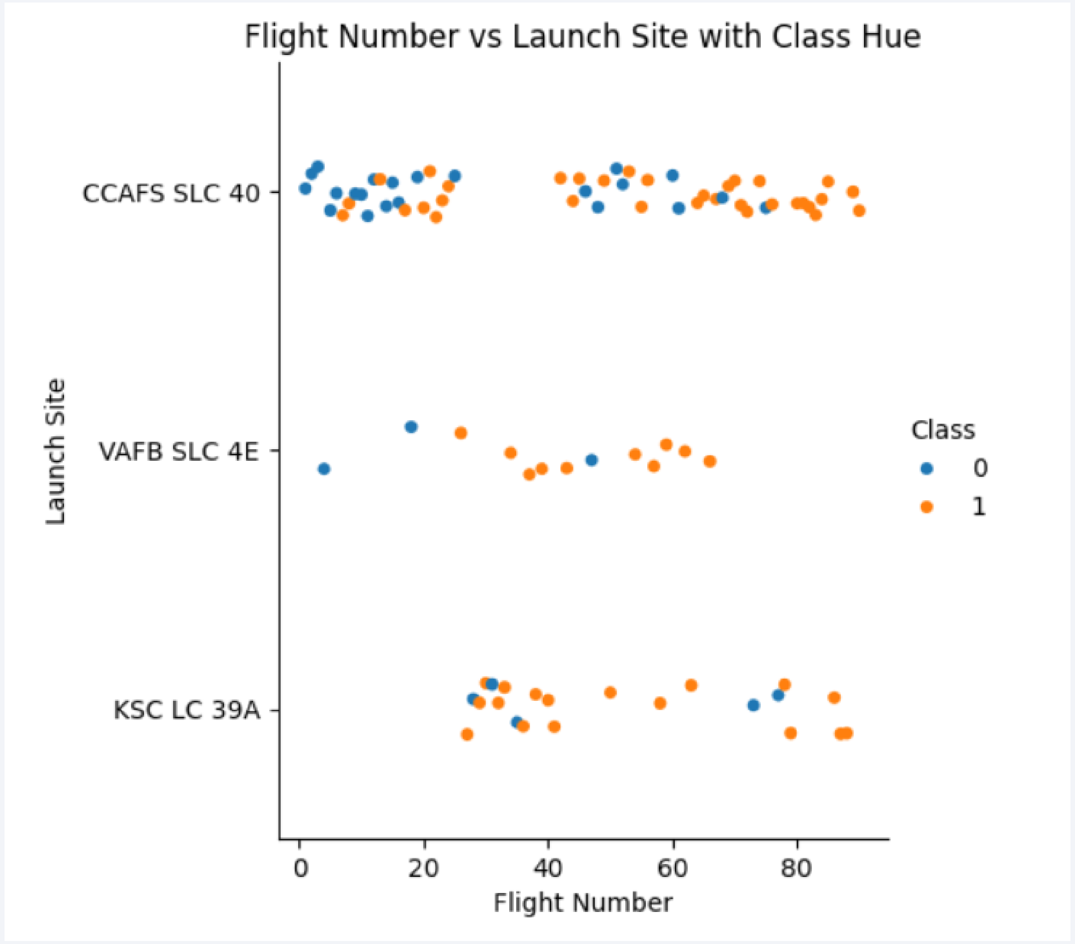
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

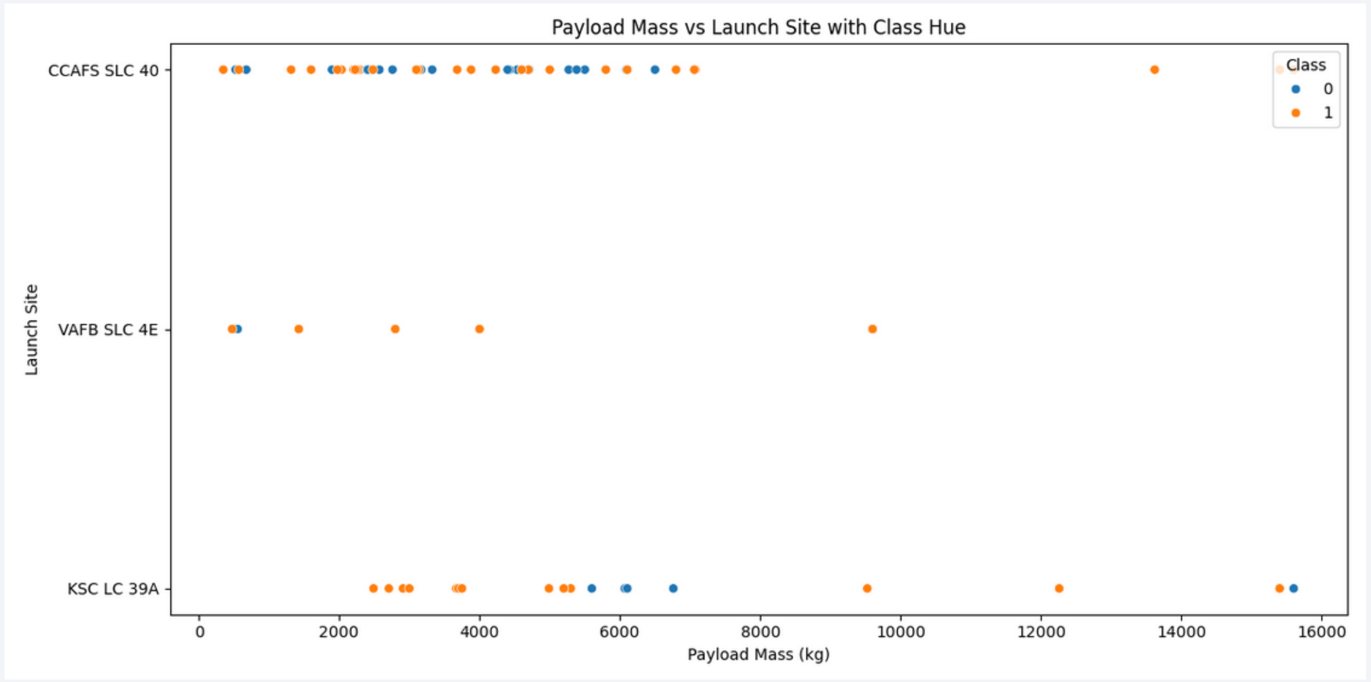
Flight Number vs. Launch Site

- scatter plot of Flight Number vs. Launch Site
- the larger the flight amount at a launch site, the greater the success rate at a launch site.



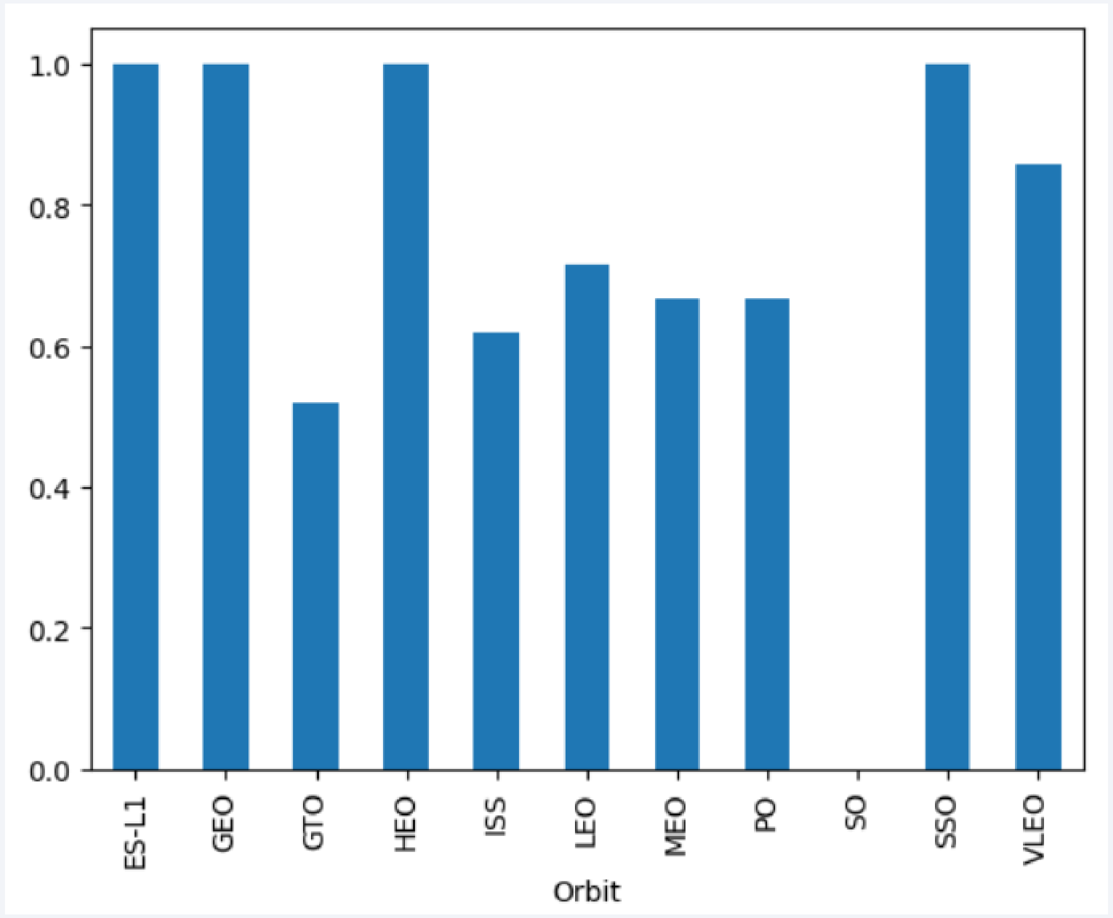
Payload vs. Launch Site

- scatter plot of Payload vs. Launch Site
- for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



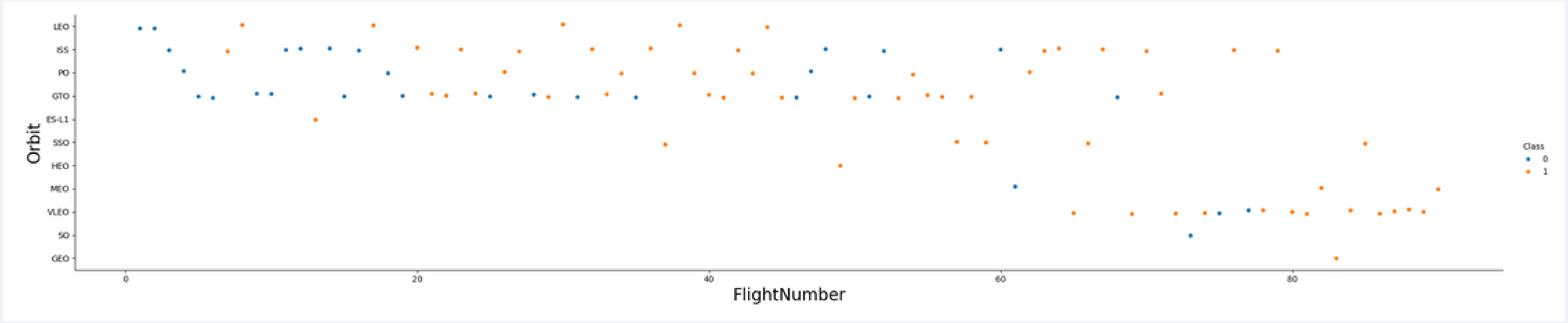
Success Rate vs. Orbit Type

- bar chart for the success rate of each orbit type
- ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



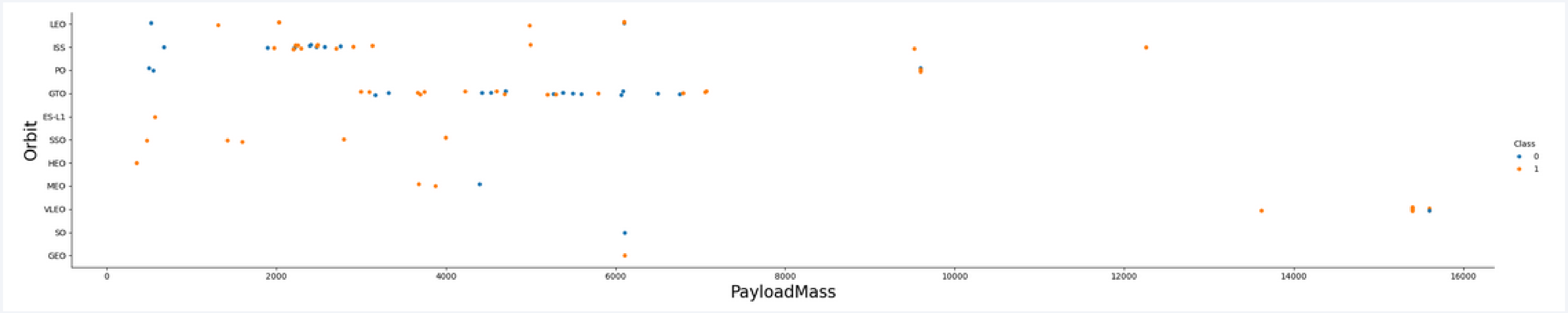
Flight Number vs. Orbit Type

- scatter plot of Flight number vs. Orbit type
- In the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



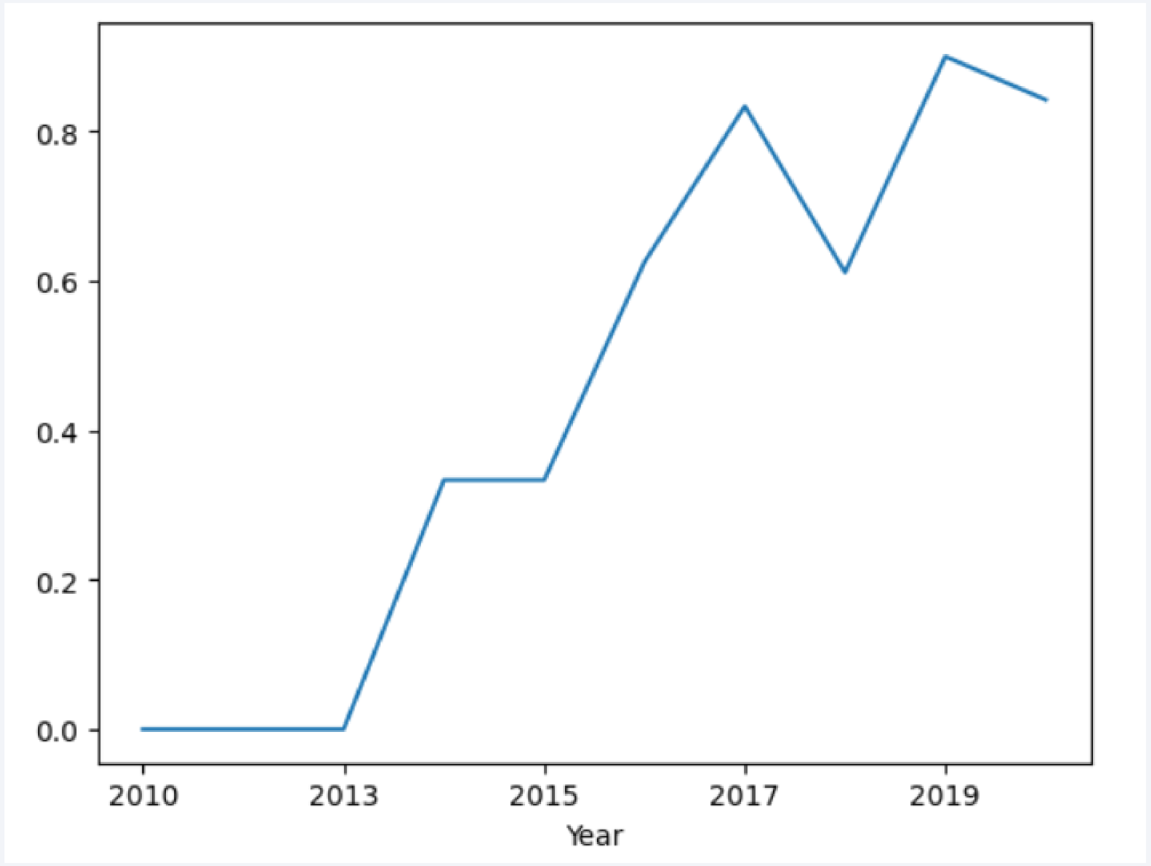
Payload vs. Orbit Type

- scatter plot of payload vs. orbit type
- With heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



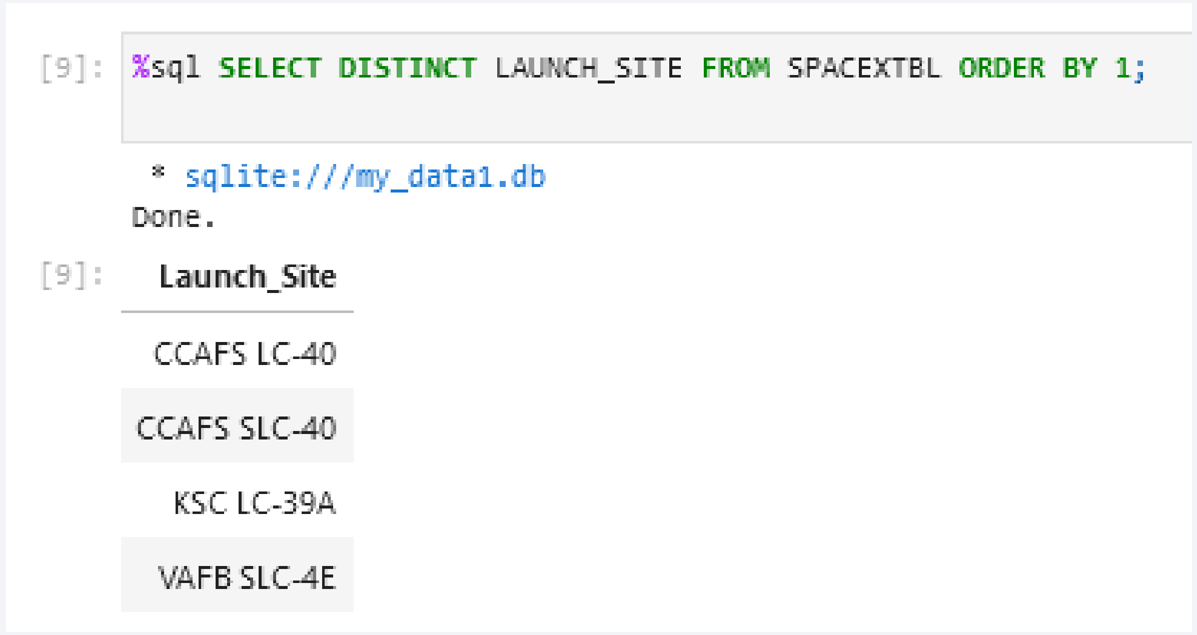
Launch Success Yearly Trend

- Line chart of yearly average success rate
- success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- Find the names of the unique launch sites
- We used the DISTINCT to show only unique launch sites from the SpaceX data.



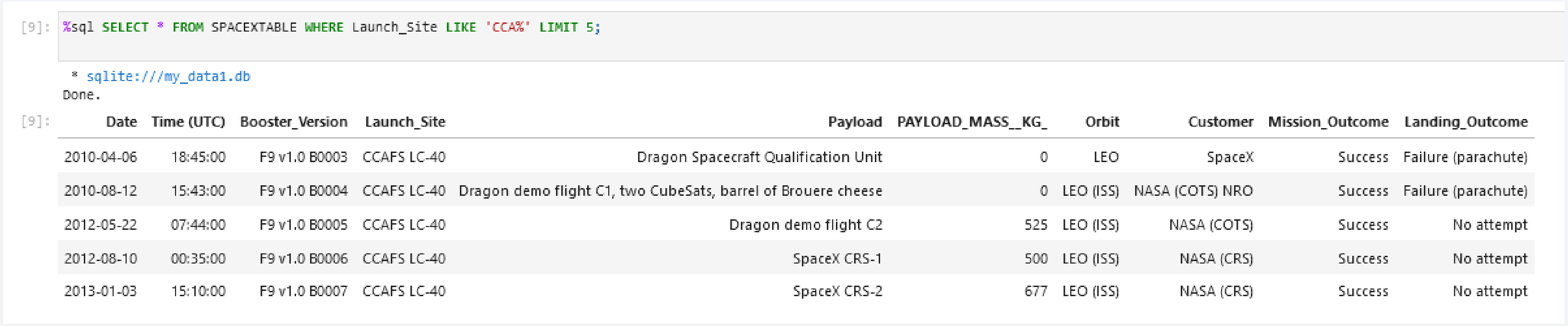
```
[9]: %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;

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

[9]:

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- We used the query mentioned to display 5 records where launch sites begin with `CCA`



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

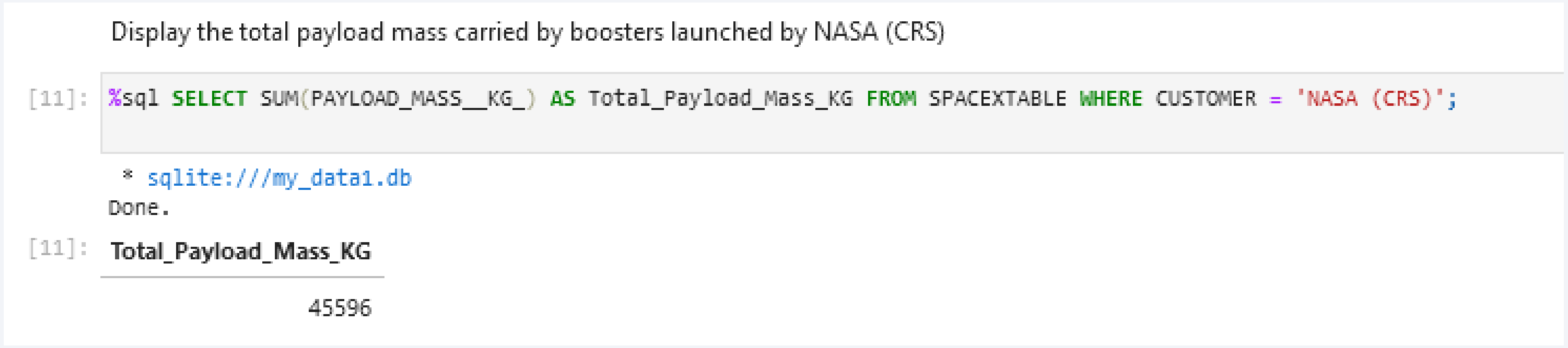
```
 * sqlite:///my_data1.db
Done.
```

[9]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- We calculated the total payload carried by boosters from NASA as 45596kg using the query below

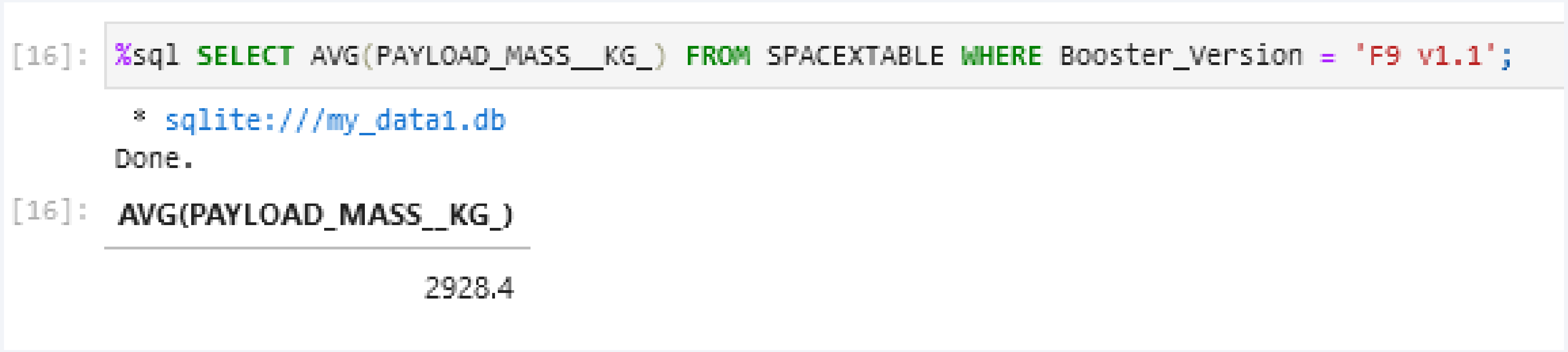


Display the total payload mass carried by boosters launched by NASA (CRS)

```
[11]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Payload_Mass_KG FROM SPACEXTABLE WHERE CUSTOMER = 'NASA (CRS)';

 * sqlite:///my_data1.db
Done.
[11]: Total_Payload_Mass_KG
                     45596
```

Average Payload Mass by F9 v1.1

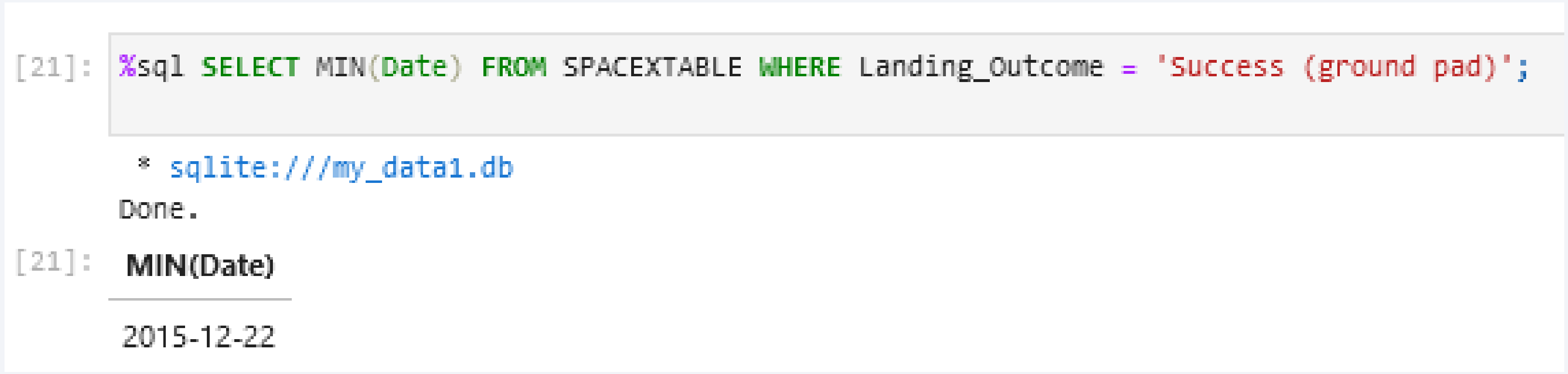
- Calculate the average payload mass carried by booster version F9 v1.1
- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4kg



```
[16]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1';

 * sqlite:///my_data1.db
Done.

[16]: AVG(PAYLOAD_MASS__KG_)
                     2928.4
```

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- We observed that the dates of the first successful landing outcome on ground pad was 15.12.2022

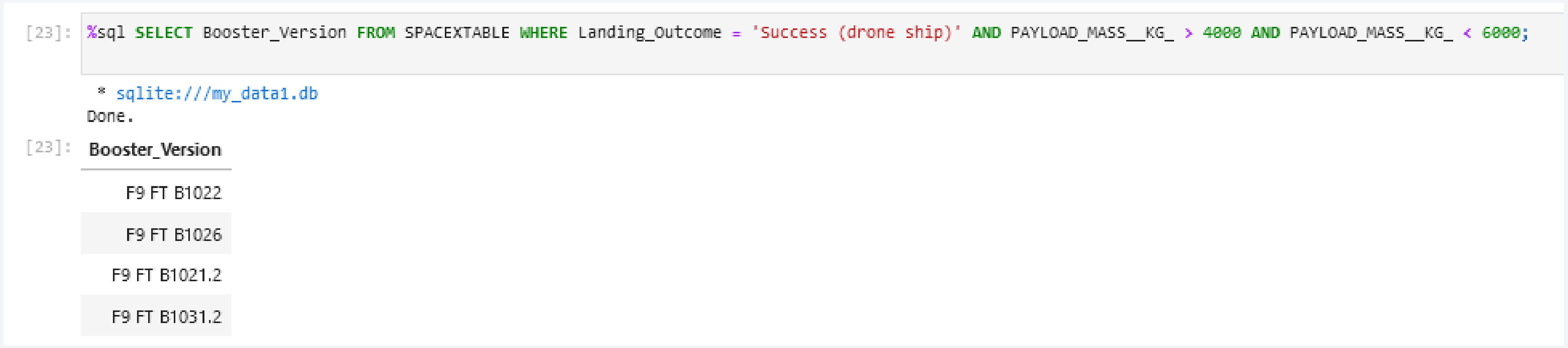


```
[21]: %sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)';

 * sqlite:///my_data1.db
Done.

[21]: MIN(Date)
      2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

- We used the WHERE clause to filter for boosters which have successfully landed on droneship and applied the AND condition to determine successful landing with payload mass greater than 4000 and less than 6000



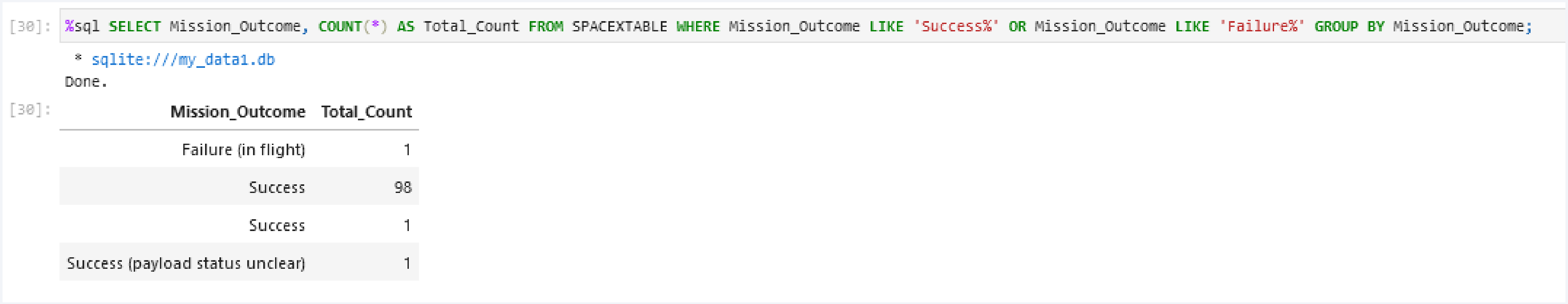
```
[23]: %sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;

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

[23]:

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- We used wildcard like '%' to filter for WHERE mission outcome was a success or a failure.



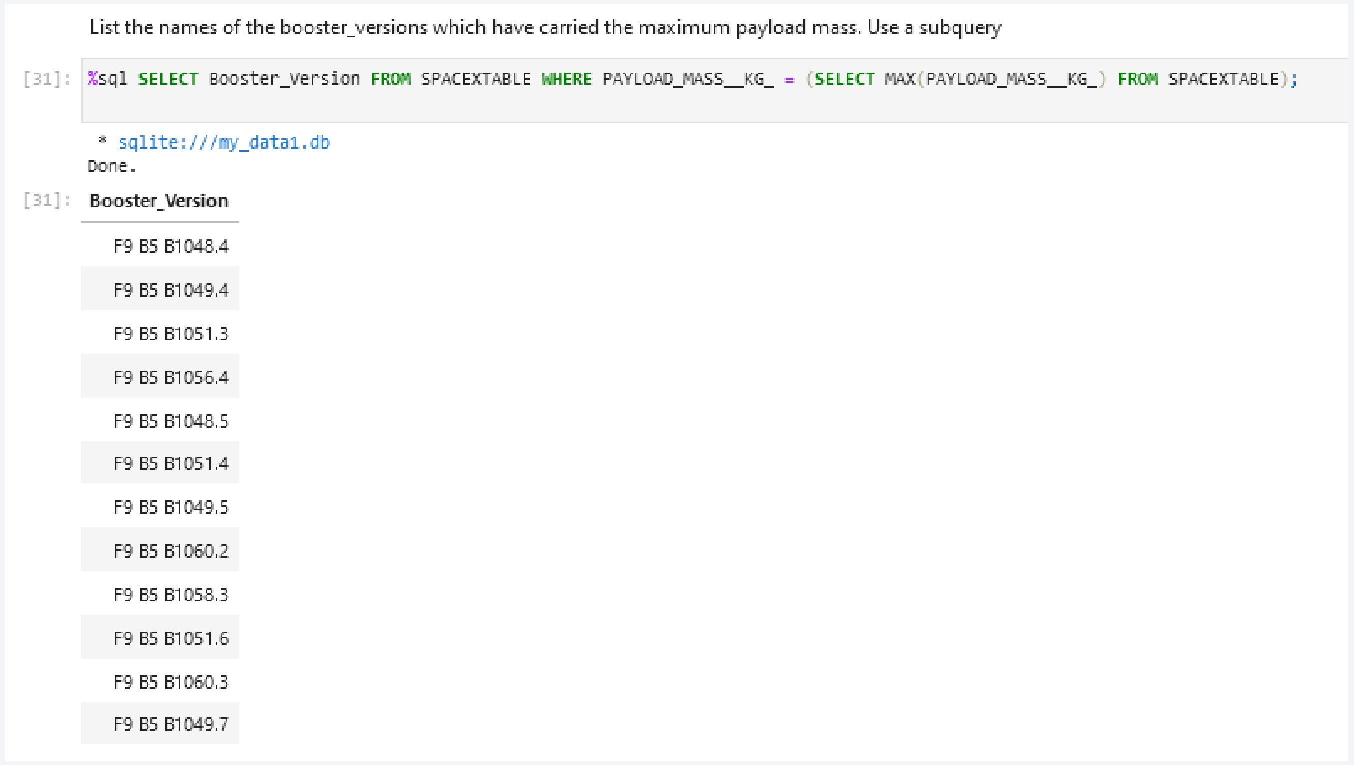
```
[30]: %sql SELECT Mission_Outcome, COUNT(*) AS Total_Count FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Success%' OR Mission_Outcome LIKE 'Failure%' GROUP BY Mission_Outcome;

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

[30]:

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- We determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.


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

```
[31]: %sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
```

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

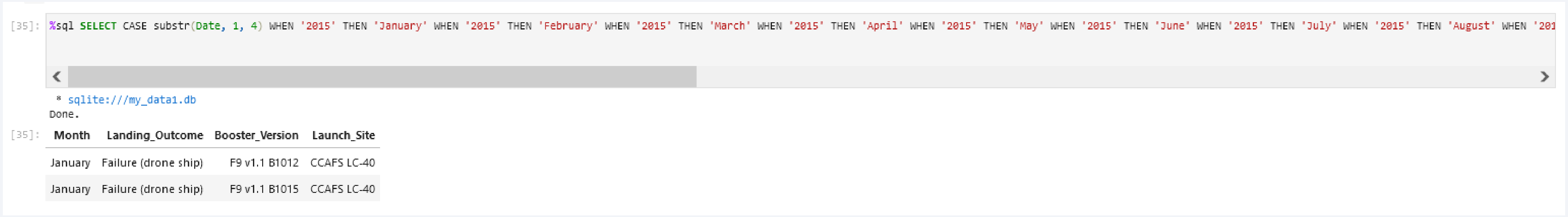
[31]:

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

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

- We used a combinations of the WHEN clause, LIKE and AND conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015



```
[35]: %sql SELECT CASE substr(Date, 1, 4) WHEN '2015' THEN 'January' WHEN '2015' THEN 'February' WHEN '2015' THEN 'March' WHEN '2015' THEN 'April' WHEN '2015' THEN 'May' WHEN '2015' THEN 'June' WHEN '2015' THEN 'July' WHEN '2015' THEN 'August' WHEN '201
```

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

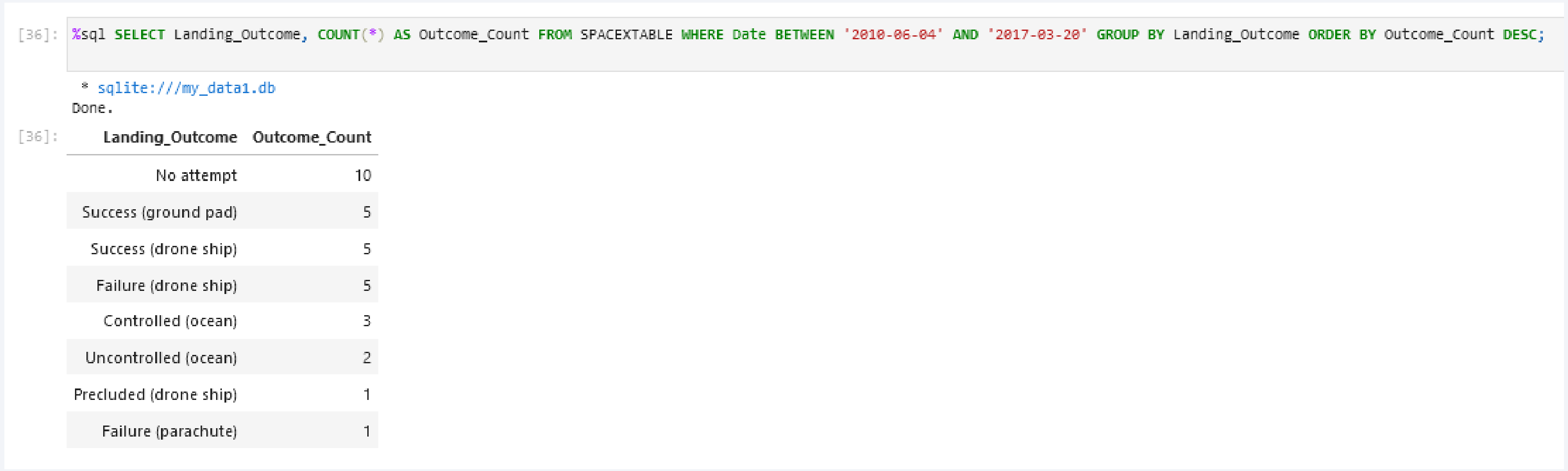
[35]:

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

- We filtered landing outcomes between 04.06.2010, and 20.03.2017, using the WHERE clause. Then, we grouped and ordered these outcomes in descending order using the GROUP BY and ORDER BY clauses.



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

 * sqlite:///my_data1.db
Done.
[36]:
```

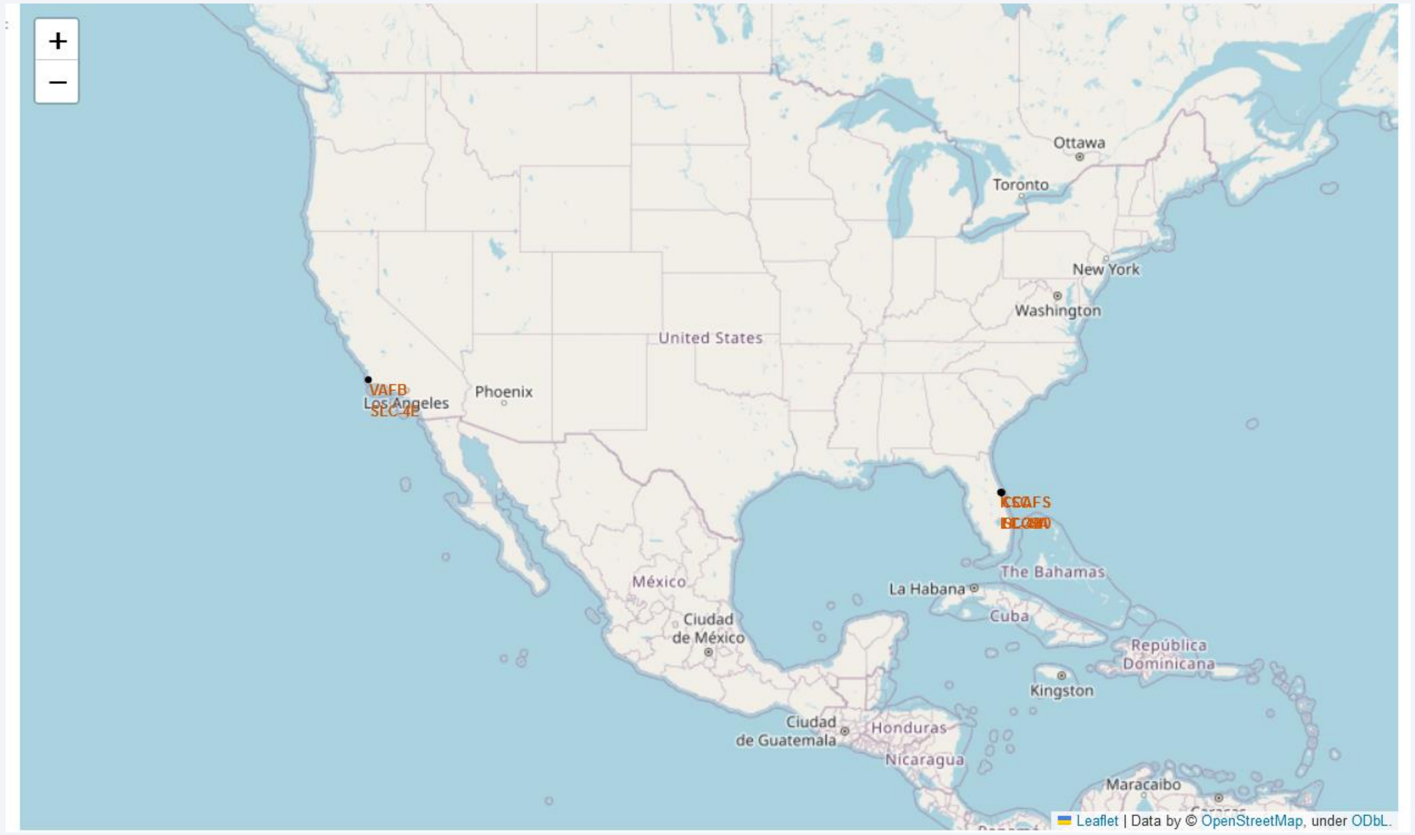
Landing_Outcome	Outcome_Count
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

Section 3

Launch Sites Proximities Analysis

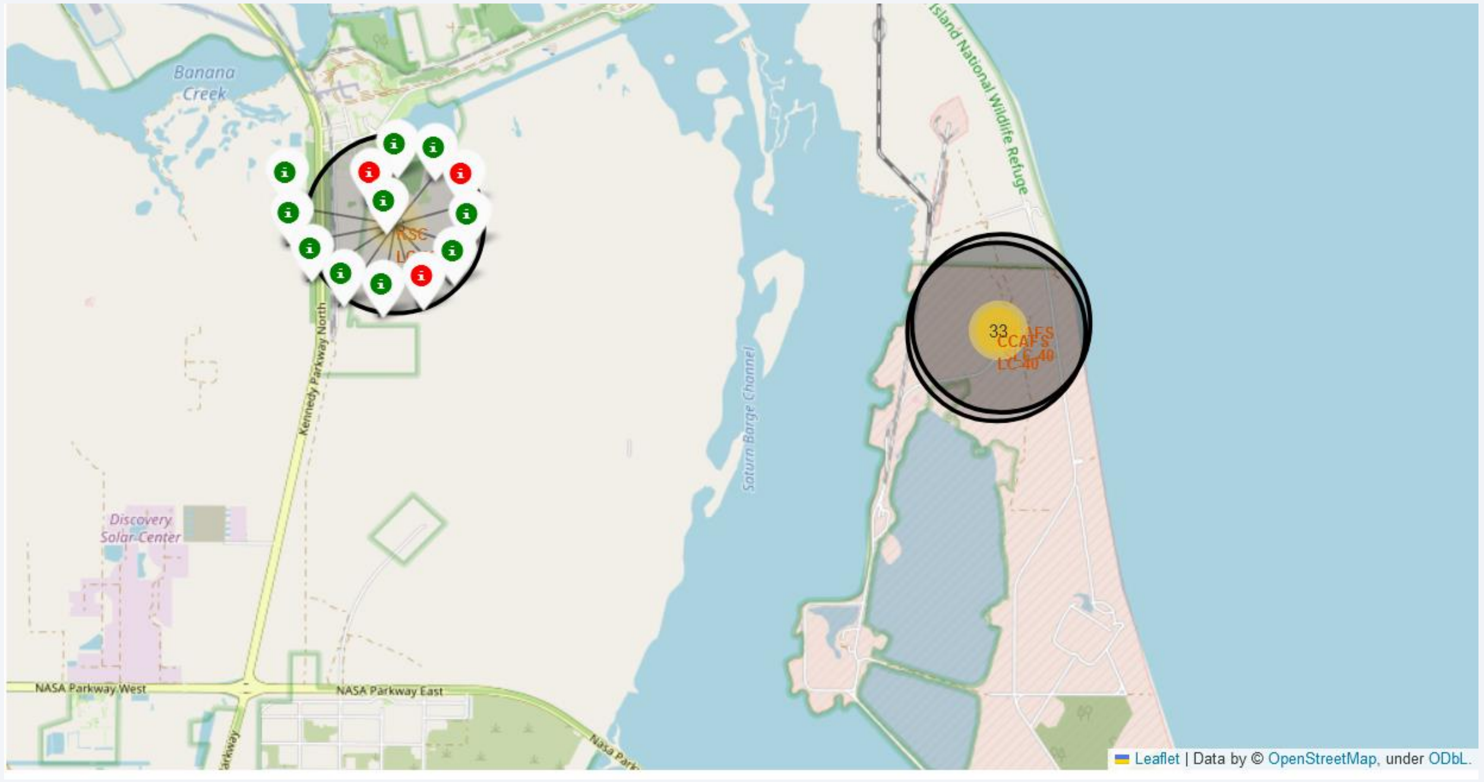
Space X Launch Sites

- The launch sites are on the US coasts



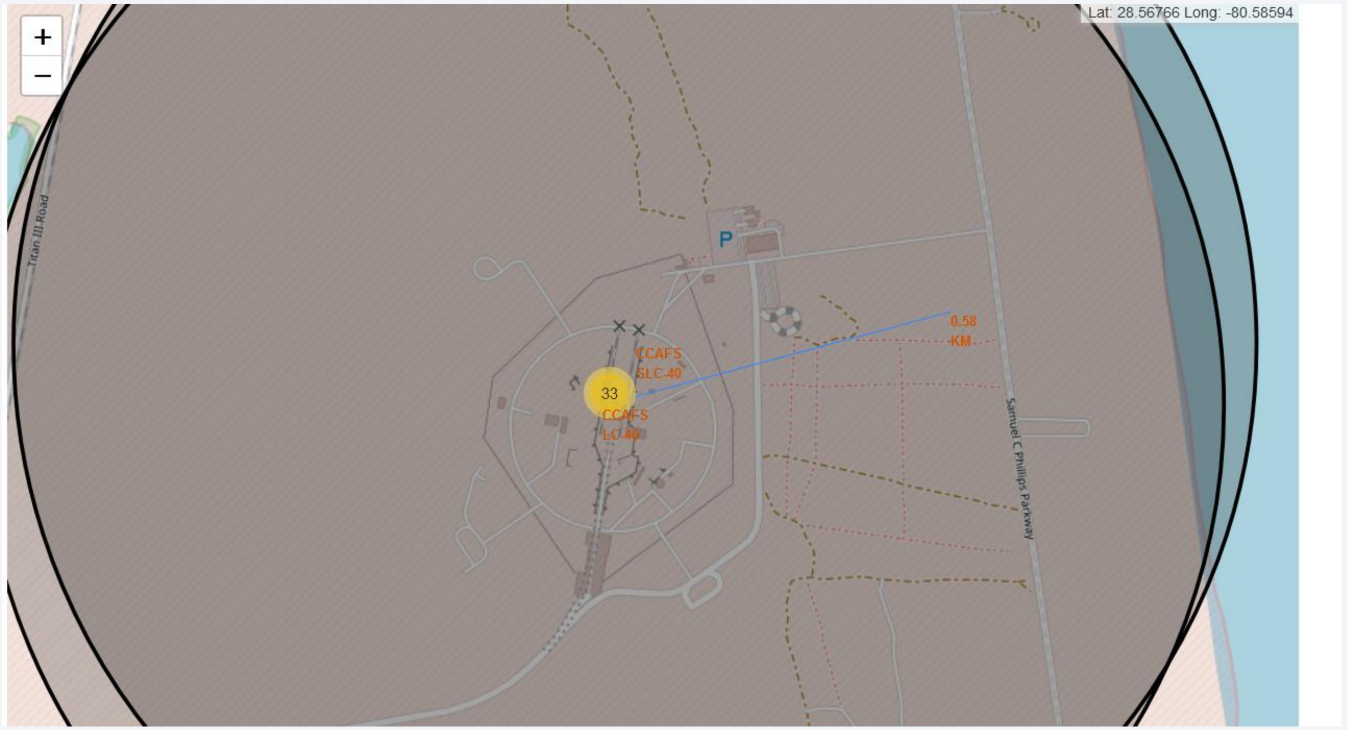
Coloured Launch Outcomes on Map

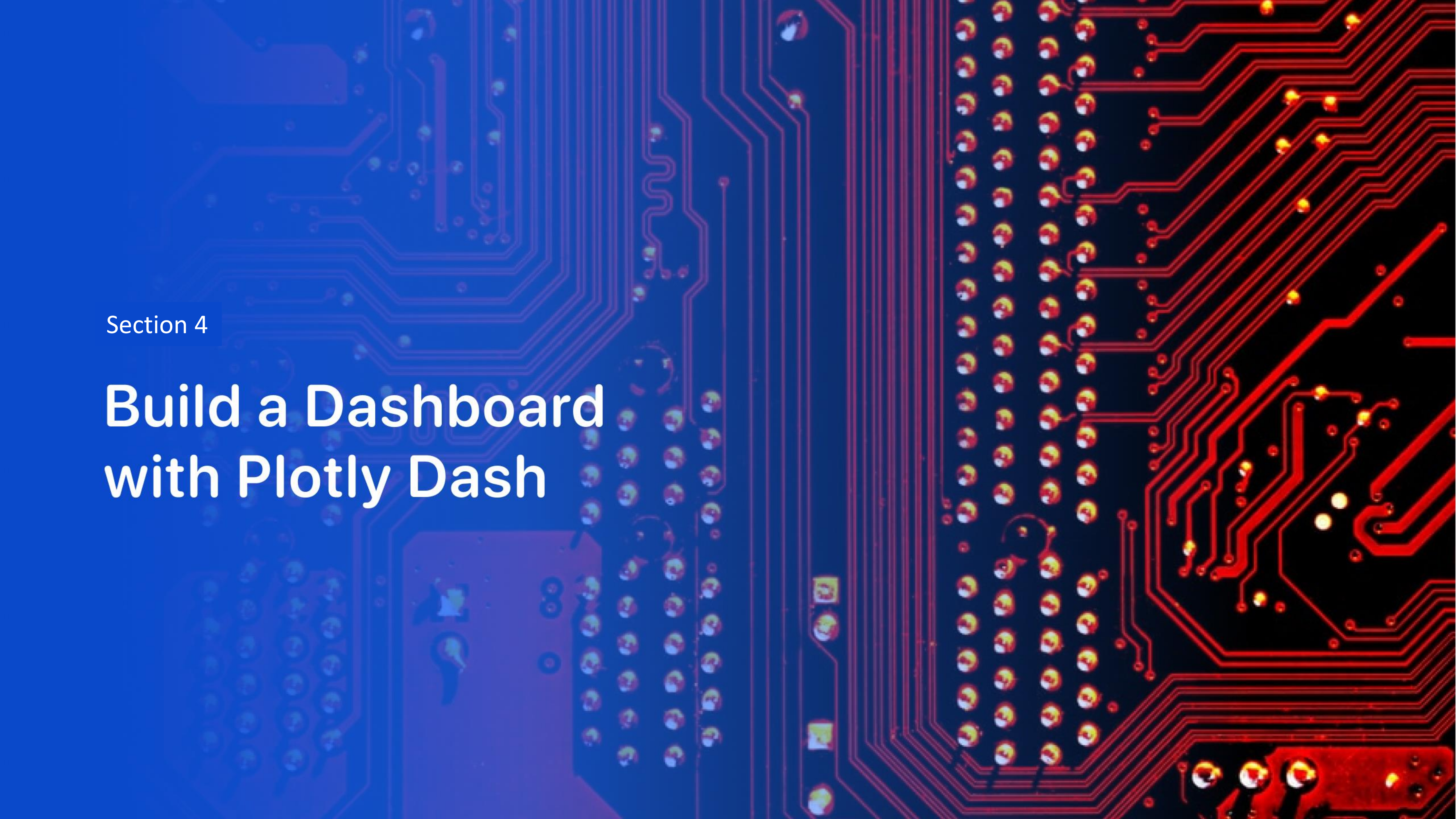
- Green for successful lanndings vs red for failed landings



Distance to Launch

- Is it possible to determine distances to particular facilities, such as railways



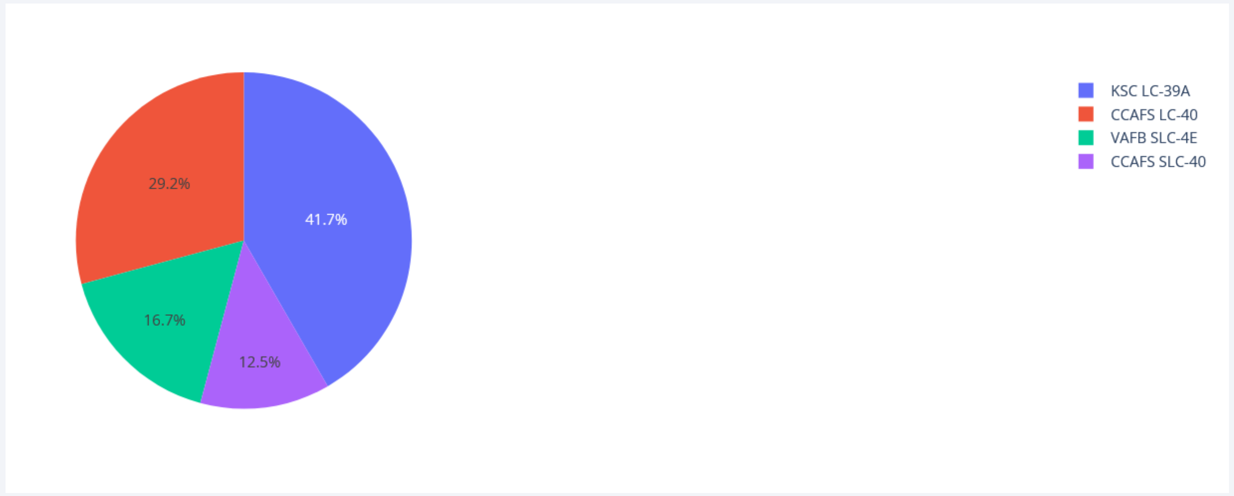


Section 4

Build a Dashboard with Plotly Dash

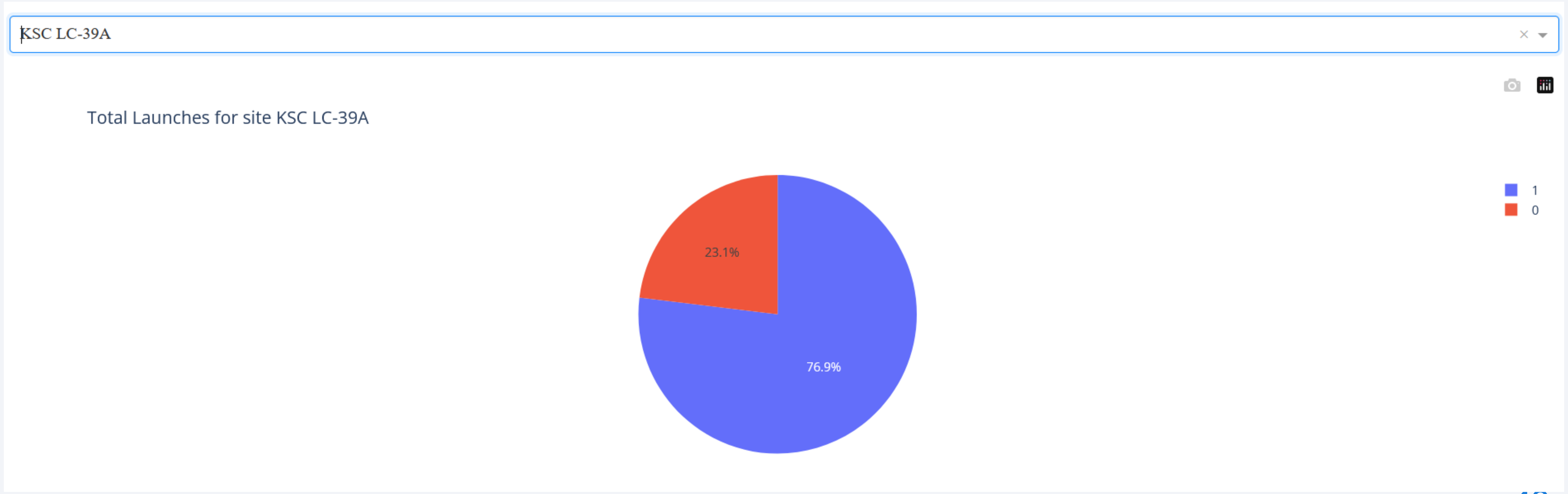
Launch success count for all sites

- KSC LC-39A had the most success



Launch site with highest success ratio

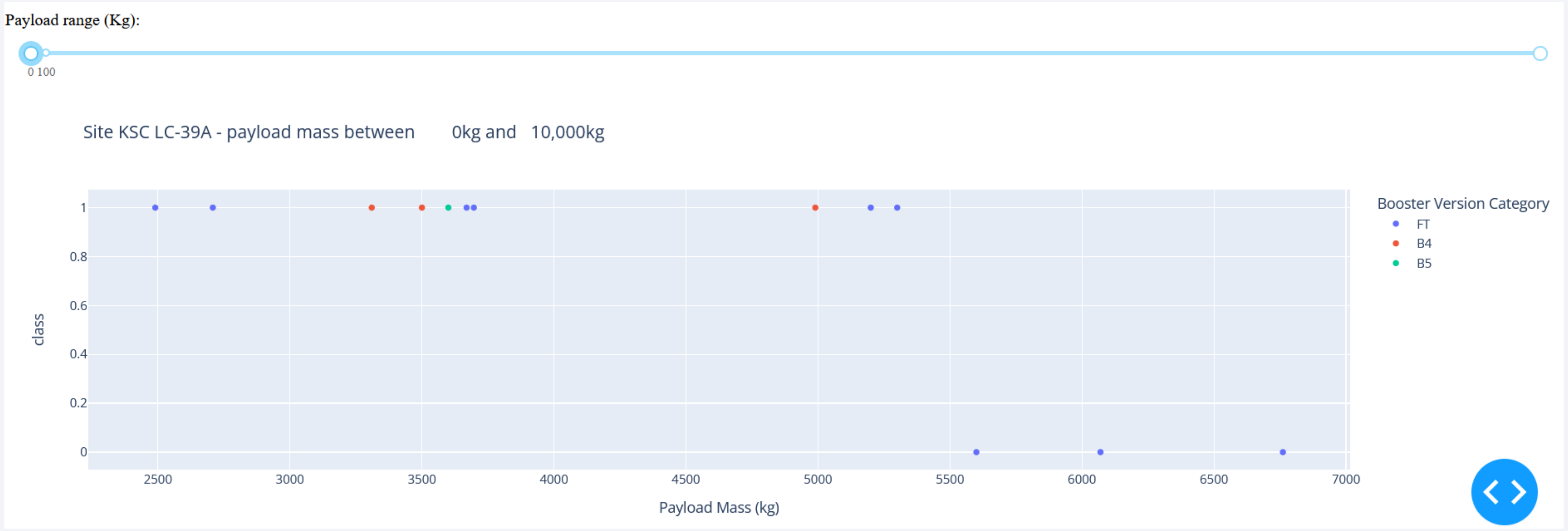
- KSC LC-39A achieved a success ration of 76,9%


KSC LC-39A

Total Launches for site KSC LC-39A

Payload vs Launch Outcome

- Successful launches are more likely at KSC LC-39A, especially with lighter payloads.

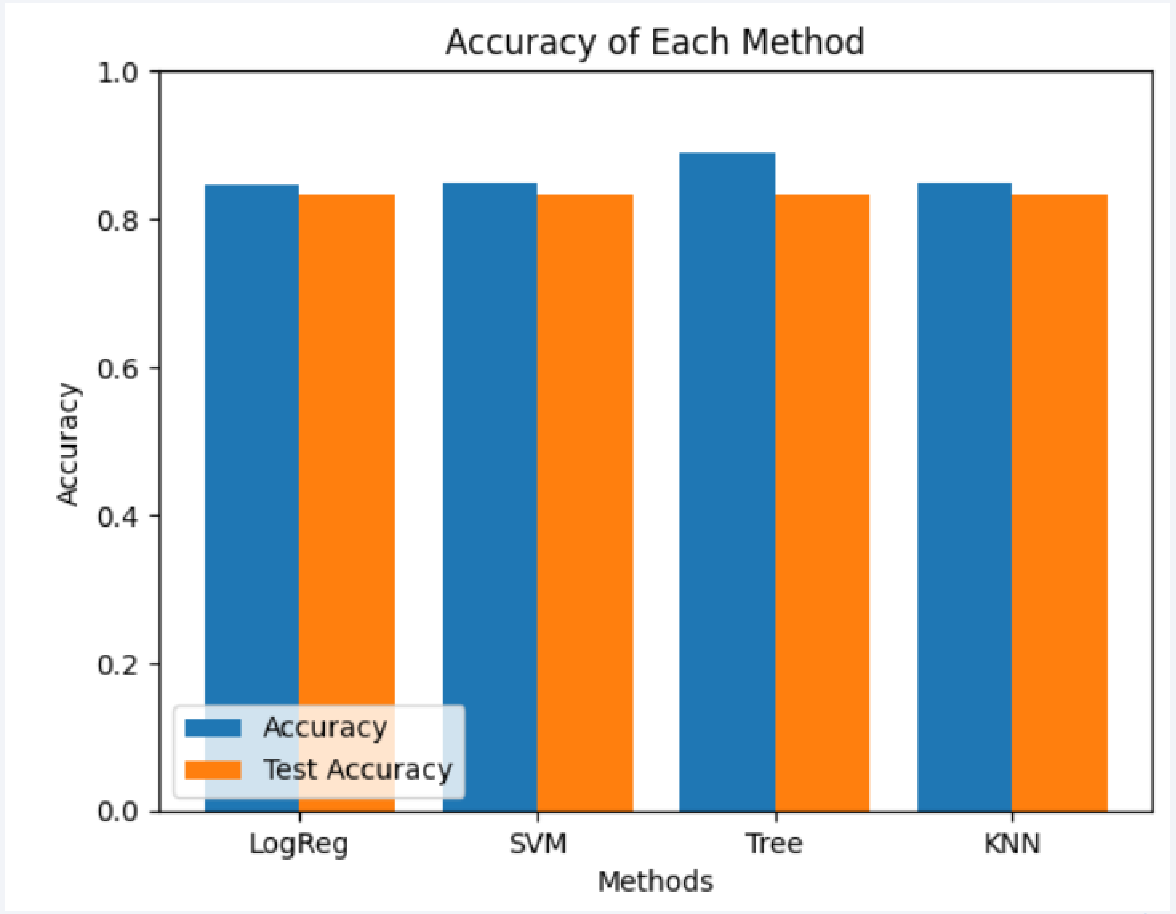


Section 5

Predictive Analysis (Classification)

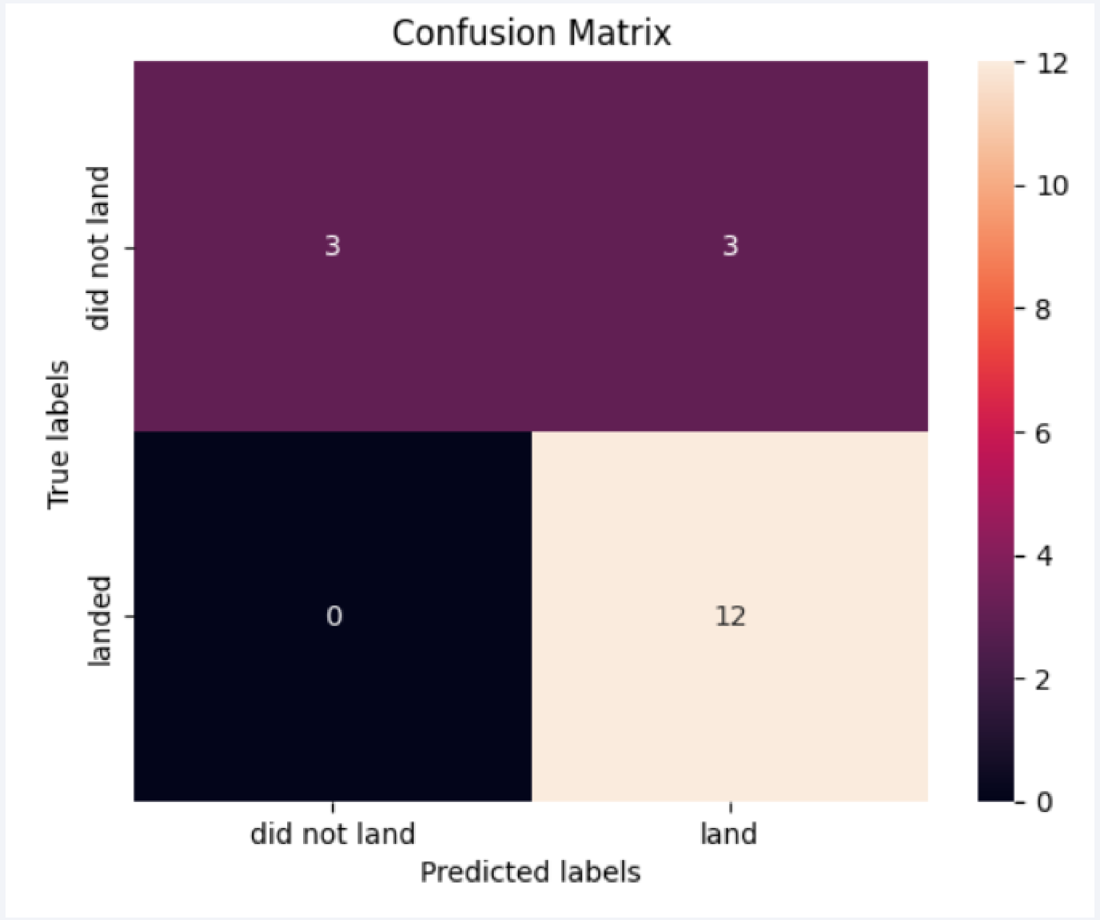
Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy



Confusion Matrix

- The decision tree classifier's confusion matrix indicates its ability to differentiate classes. The main issue lies in false positives, where unsuccessful landings are incorrectly labeled as successful by the classifier.



Conclusions

- Launch success rates tend to rise with more flights at a launch site.
- Success rates increased steadily from 2013 to 2020.
- Orbits like ES-L1, GEO, HEO, SSO, and VLEO showed the highest success rates.
- KSC LC-39A had the most successful launches among all sites.
- The Decision Tree Classifier is the preferred machine learning algorithm for this task.

Appendix

-

Thank you!