IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Anyawuike Ikenna Omedobi

24/3/2024

# Outline

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

# 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 (ML) Prediction

- Summary of all results

- Exploratory data analysis result
- Interactive analytics in screenshots
- Predictive analysis result from ML Lab

# Introduction

SpaceX, short for Space Exploration Technologies Corp, is a revolutionary company that stands at the forefront of revolutionizing the space industry with its groundbreaking achievements and transformative projects. One of its most notable contributions is the Falcon 9 rocket, which has reshaped the landscape of space travel by offering launches at a fraction of the cost compared to traditional providers. While competitors often charge upwards of 165 million dollars per launch, SpaceX has successfully lowered the cost to as low as 62 million dollars. This remarkable cost reduction can be attributed to SpaceX's pioneering concept of reusing the first stage of the rocket by re-landing it, thus enabling it to be used in subsequent missions. With each successful reuse, the price of launches is expected to decrease even further, making space exploration more accessible and cost-effective than ever before. As a data scientist working for a startup aiming to compete with SpaceX, there is a critical need to develop predictive models to forecast the outcome of first-stage landings in future missions. By leveraging machine learning pipelines, this project aims to accurately predict whether a first stage will successfully land or not. Such predictions are essential for effectively bidding against SpaceX for rocket launches, ensuring that resources are allocated efficiently and competitively.

The problems include;

- Identifying all factors that influence the landing outcome.
- The relationship between each variable and how it is affecting the outcome.
- The vest condition needed to increase the probability of a successful landing.

Section 1

# Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX REST API and web scrapping from Wikipedia.
- Perform data wrangling
  - Data was processed using one-hot encoding for 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, and evaluate classification models

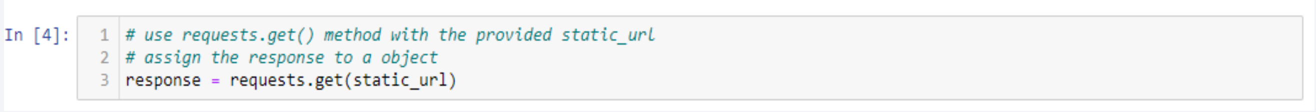
# Data Collection

Data collection encompasses the gathering and measurement of pertinent information from diverse sources regarding specific variables within a defined system. This process enables individuals or organizations to address inquiries, assess outcomes, and inform decision-making effectively. As mentioned, the dataset was collected by using SpaceX REST API and web scrapping from Wikipedia.

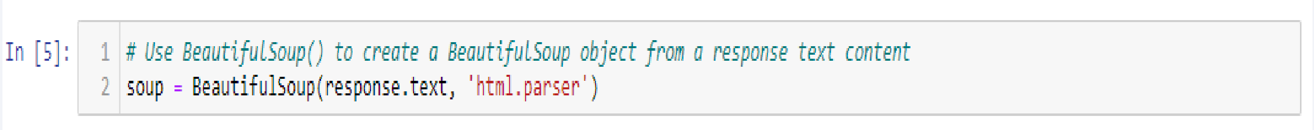
- To utilize a REST API, we initiated by employing the get request function to retrieve data. Next, we decoded the response content as JSON and transformed it into a pandas dataframe utilizing the json_normalize() function. Subsequently, we performed data cleaning procedures, including checking for missing values and appropriately filling them as necessary.

- When performing web scraping, I utilized BeautifulSoup to extract launch records from an HTML table. Subsequently, I parsed the table and converted it into a pandas dataframe, facilitating further analysis of the data.You need to present your data collection process use key phrases and flowcharts

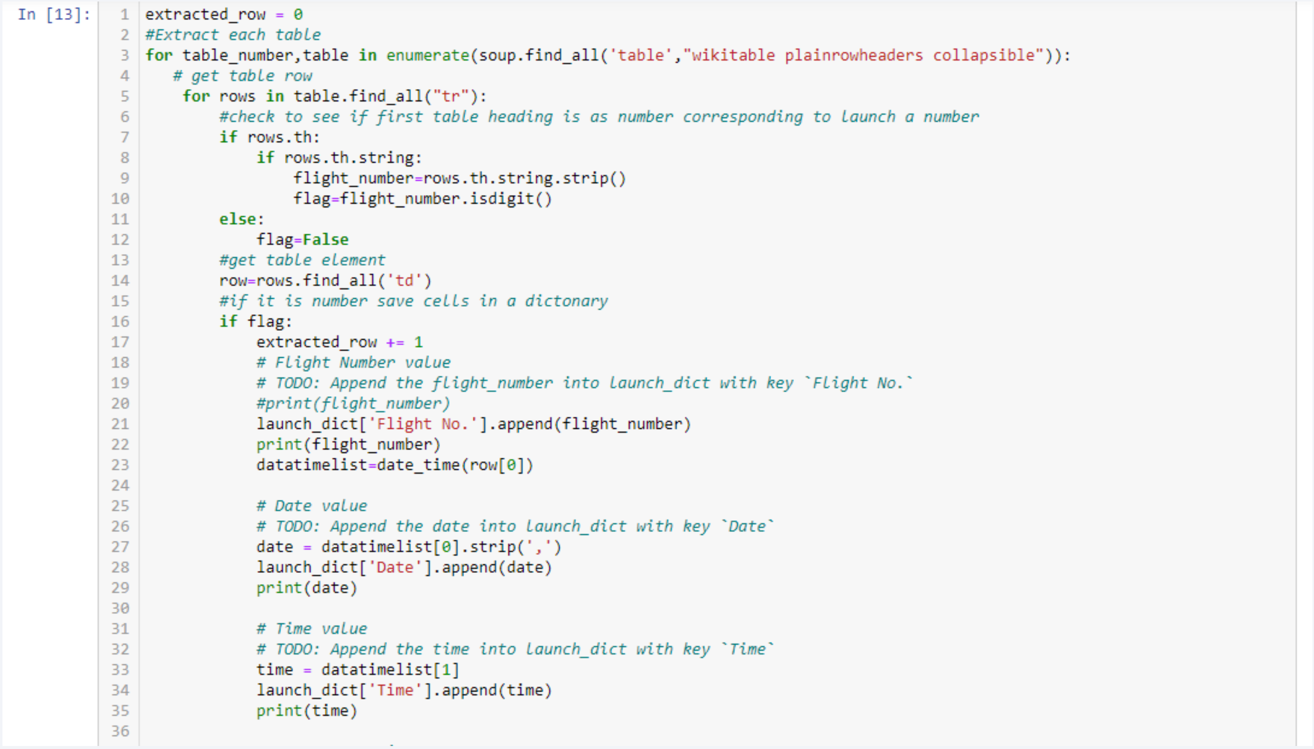
# Data Collection – Scraping



```
In [4]:
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```



```
In [5]:
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
```



```
In [13]:
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            launch_dict['Flight No.'].append(flight_number)
            print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            print(time)
```

Request the Falcon 9 launch data from Wiki page from url

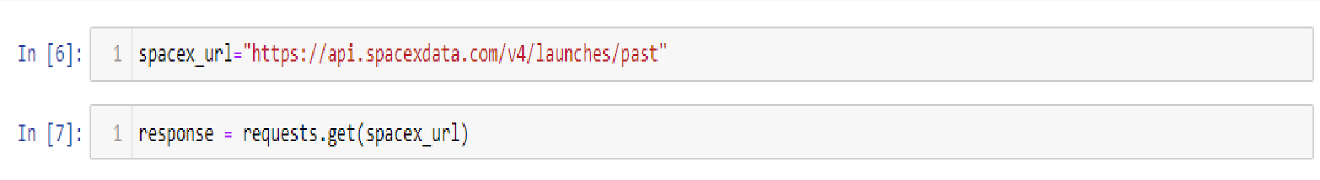
Create a BeautifulSoup from the HTML response

Extract all column or variable names from the HTML header

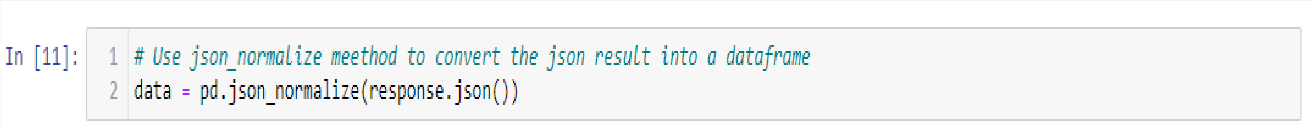
From:

https://github.com/

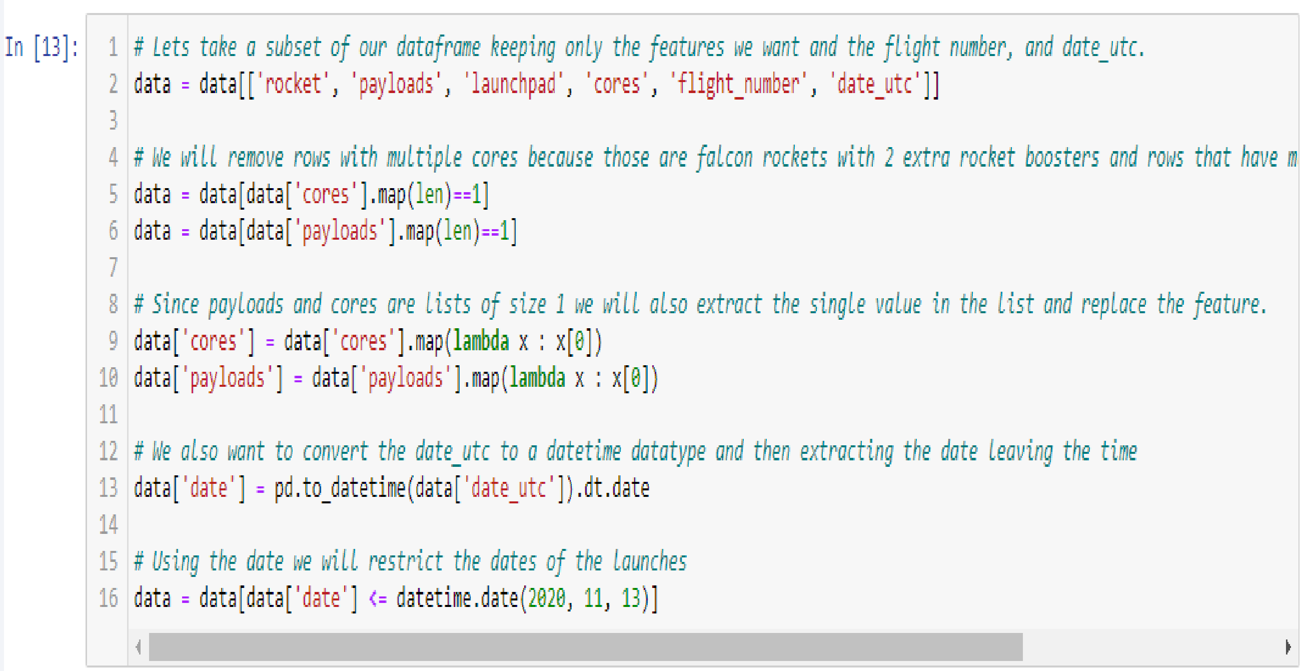
# Data Collection – SpaceX API



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

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



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



```
In [13]: # Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
         data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

         # We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have m
         data = data[data['cores'].map(len)==1]
         data = data[data['payloads'].map(len)==1]

         # Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
         data['cores'] = data['cores'].map(lambda x : x[0])
         data['payloads'] = data['payloads'].map(lambda x : x[0])

         # We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
         data['date'] = pd.to_datetime(data['date_utc']).dt.date

         # Using the date we will restrict the dates of the launches
         data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Get request for rocket launch data using API

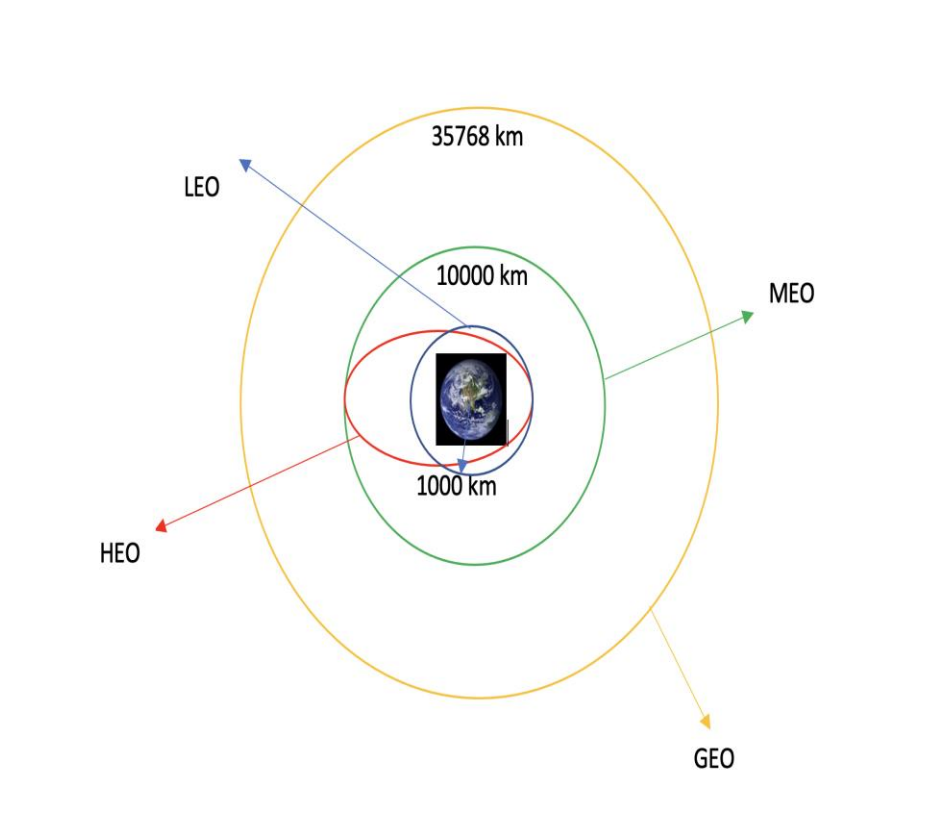
Use `json_normalize` function to convert json result to pandas dataframe

Performed data cleaning and fill the missing value

From:

https://github.com/

# Data Wrangling



From:

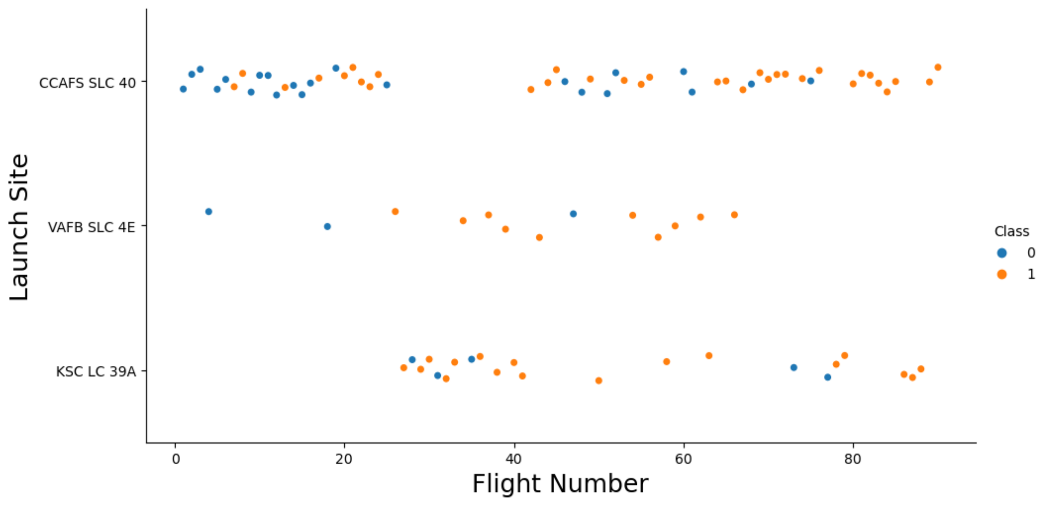
https://github.com/

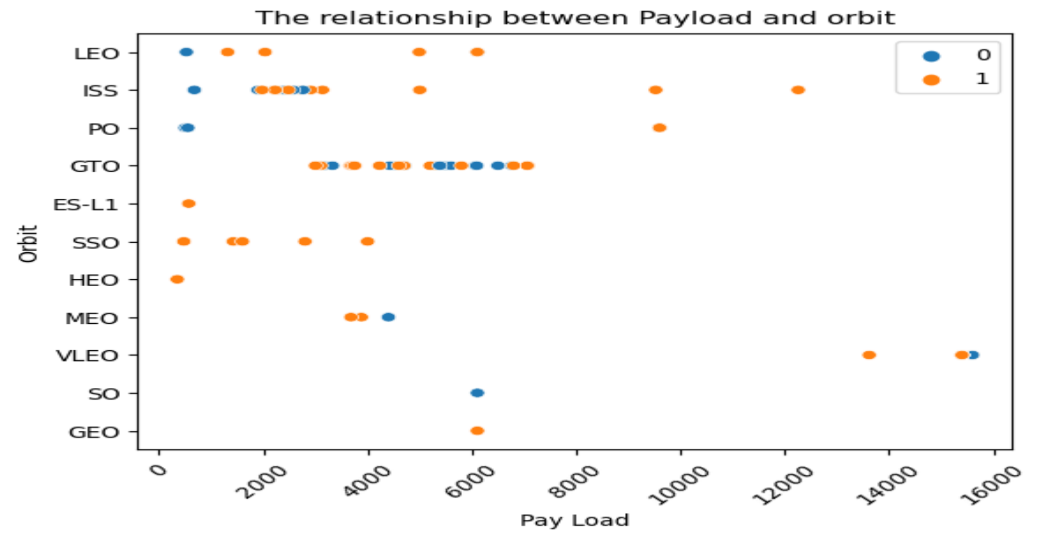
Data wrangling refers to the process of collecting, cleaning, transforming, and organizing raw data to ensure it is in a suitable format for easy access and exploratory data analysis (EDA).

First, we will compute the number of launches on each site. Next, we will determine the frequency and occurrence of mission outcomes per orbit type.

Subsequently, we generate a landing outcome label based on the outcome column. This will streamline subsequent analysis, visualization, and machine-learning tasks. Finally, we export the results to a CSV file for reference.

# EDA with Data Visualization





We began by employing scatter plots to explore relationships between attributes, including:

- Payload and Flight Number.

- Flight Number and Launch Site.

- Payload and Launch Site.

- Flight Number and Orbit Type.

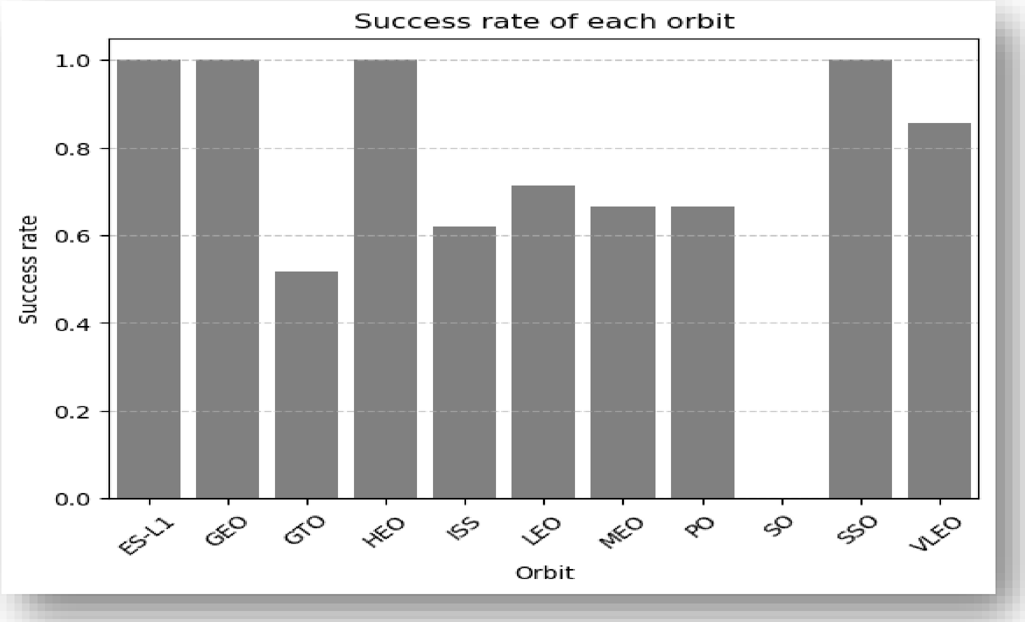
- Payload and Orbit Type.

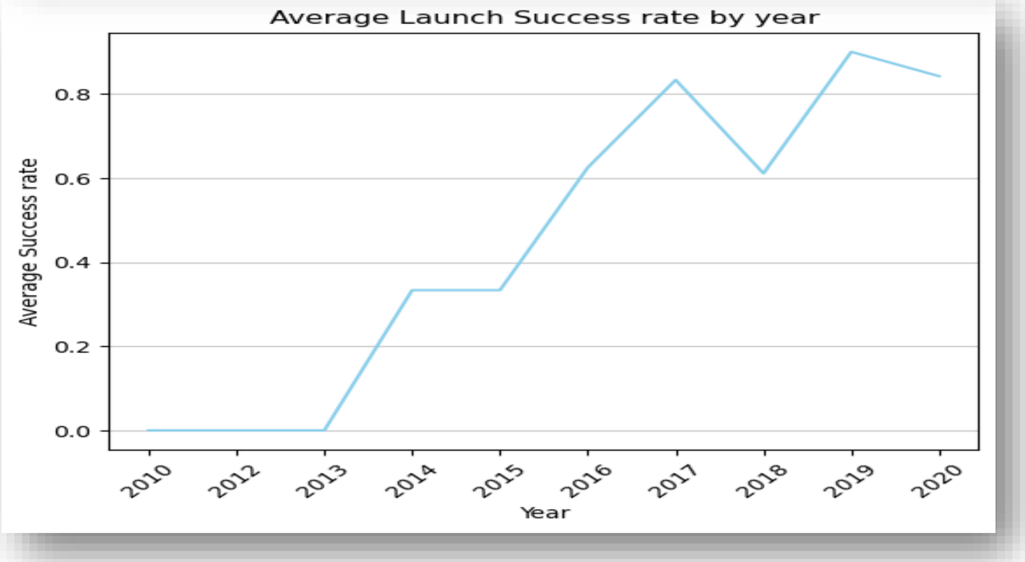
Scatter plots reveal dependencies between attributes, allowing us to discern patterns. This enables us to identify the factors that most strongly influence the success of landing outcomes.

From:

https://github.com/

# EDA with Data Visualization





After obtaining insights from the scatter plots, we will employ additional visualization tools such as bar graphs and line plots for further analysis.

Bar graphs provide a straightforward method for interpreting relationships between attributes. In this context, we will utilize bar graphs to identify which orbits exhibit the highest probability of success.

Subsequently, we will utilize line graphs to illustrate trends or patterns of attributes over time. Specifically, we will use them to visualize the yearly trend of launch success.

Lastly, we will perform feature engineering to prepare for future success prediction modules. This involves creating dummy variables for categorical columns and enhancing the dataset for predictive modeling.

From:

https://github.com/

# EDA with SQL

Using SQL, we executed various queries to gain a better understanding of the dataset, including:

- Displaying the names of the launch sites.

- Retrieving 5 records where launch sites begin with the string 'X’.

- Determining the total payload mass carried by boosters launched by NASA.

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

- Listing the date when the first successful landing outcome on a ground pad was achieved.

- Identifying the names of boosters that successfully landed on a drone ship and carried a payload mass greater than 4000 but less than 6000.

- Listing the total number of successful and failed mission outcomes.

- Determining the booster versions that carried the maximum payload mass.

- Listing the failed landing outcomes on drone ships, including their booster versions and launch site names, for the year 2015.

- Ranking the count of landing outcomes or successes between the dates June 4, 2010, and March 20, 2017, in descending order.

https://github.com/

# Build an Interactive Map with Folium

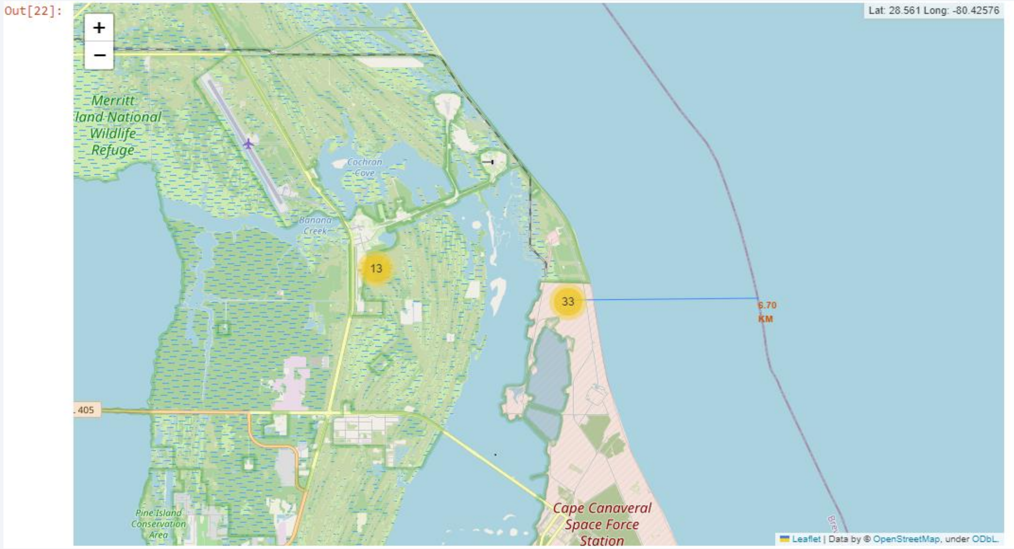
To visualize the launch data on an interactive map, we utilized latitude and longitude coordinates for each launch site. We added a circle marker around each launch site, accompanied by a label indicating the name of the launch site.

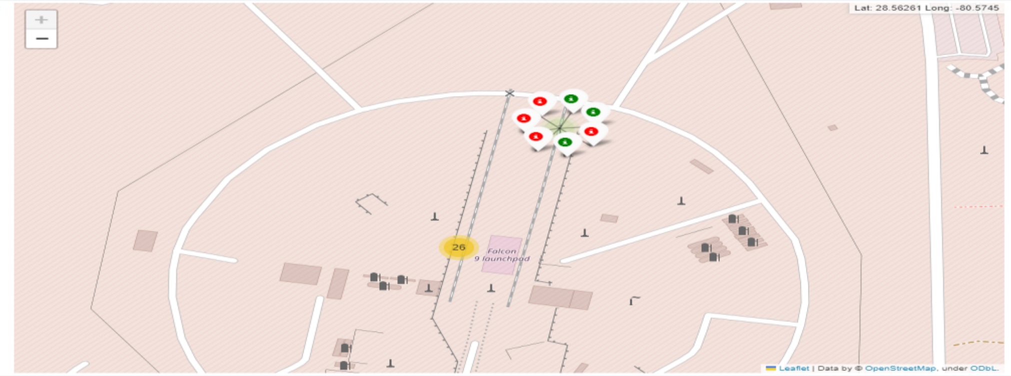
Subsequently, we assigned the launch outcomes (failure and success) from the data frame to classes 0 and 1, respectively. These were represented on the map with red and green markers, organized using MarkerCluster.

To address questions regarding the proximity of launch sites to various landmarks, we employed Haversine's formula. This allowed us to calculate the distance between launch sites and landmarks such as railways, highways, coastlines, and nearby cities. This analysis provided insights into the spatial relationships between launch sites and key geographical features.

- How close are the launch sites with railways, highways, and coastlines?
- How close are the launch sites to nearby cities?

https://github.com/


Out[22]:
+
−
Lat: 28.561 Long: -80.42576
Merritt Island National Wildlife Refuge
Cochran Cove
Banana Creek
13
33
6.70 KM
405
Pine Island Conservation Area
Cape Canaveral Space Force Station
Leaflet | Data by © OpenStreetMap, under ODbL.


+
−
Lat: 28.56261 Long: -80.5745
26
Falcon 9 launchpad
Leaflet | Data by © OpenStreetMap, under ODbL.

# Build a Dashboard with Plotly Dash

We developed an interactive dashboard using Plotly Dash, empowering users to explore the data dynamically according to their preferences.

Firstly, we created pie charts illustrating the total number of launches from specific sites. This visualization provides users with an overview of launch activity across different launch sites.

Additionally, we generated scatter plots to examine the relationship between launch outcomes and payload mass for various booster versions. By plotting Outcome against Payload Mass (Kg), users can discern patterns and trends in mission success relative to payload mass, allowing for deeper insights into the performance of different booster versions.

https://github.com/

# Predictive Analysis (Classification)

**Building the Model**:

- Load the dataset using NumPy and Pandas.
- Perform data transformation tasks such as cleaning, encoding categorical variables, and scaling numerical features.

Split the dataset into training and test sets.

- Decide on the appropriate type of machine learning (ML) algorithm based on the problem at hand.
- Set the parameters and algorithms for GridSearchCV, a technique for hyperparameter optimization.
- Fit the GridSearchCV to the dataset, allowing it to find the optimal parameters and train the model on the training set.

**Evaluating the Model**:

- Assess the accuracy of each model to gauge its performance.
- Obtain tuned hyperparameters for each type of algorithm, ensuring optimal model configuration.
- Visualize the confusion matrix to gain insights into the model's classification performance.

**Improving the Model**:

- Implement feature engineering techniques to enhance the predictive power of the model.

**Finding the Best Model**:

- Determine the best-performing model based on the accuracy score, selecting the one with the highest accuracy as the best model.

# Results

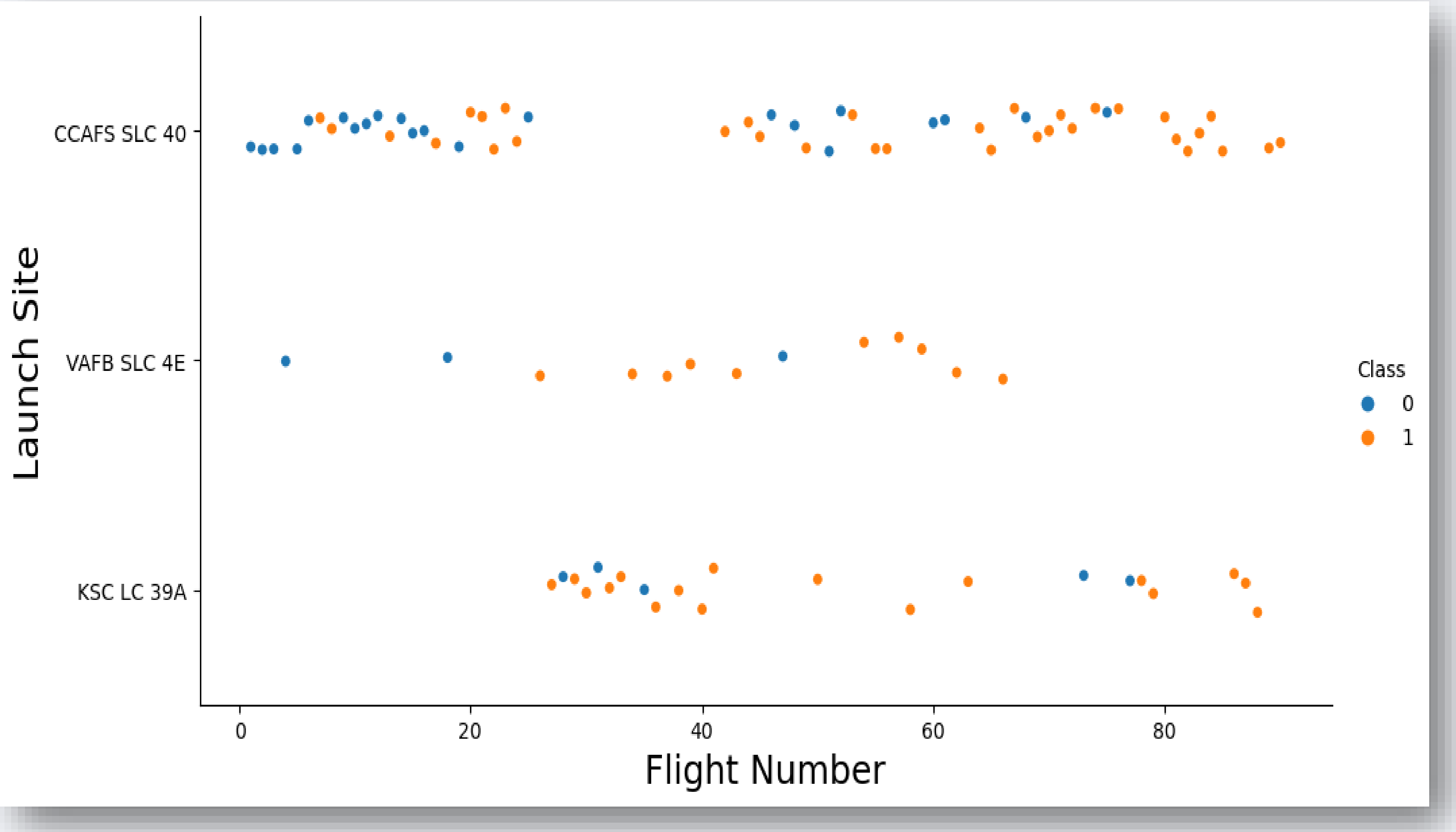
The results will be categorized into 3 main results which is:

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

Section 2

# Insights drawn from EDA

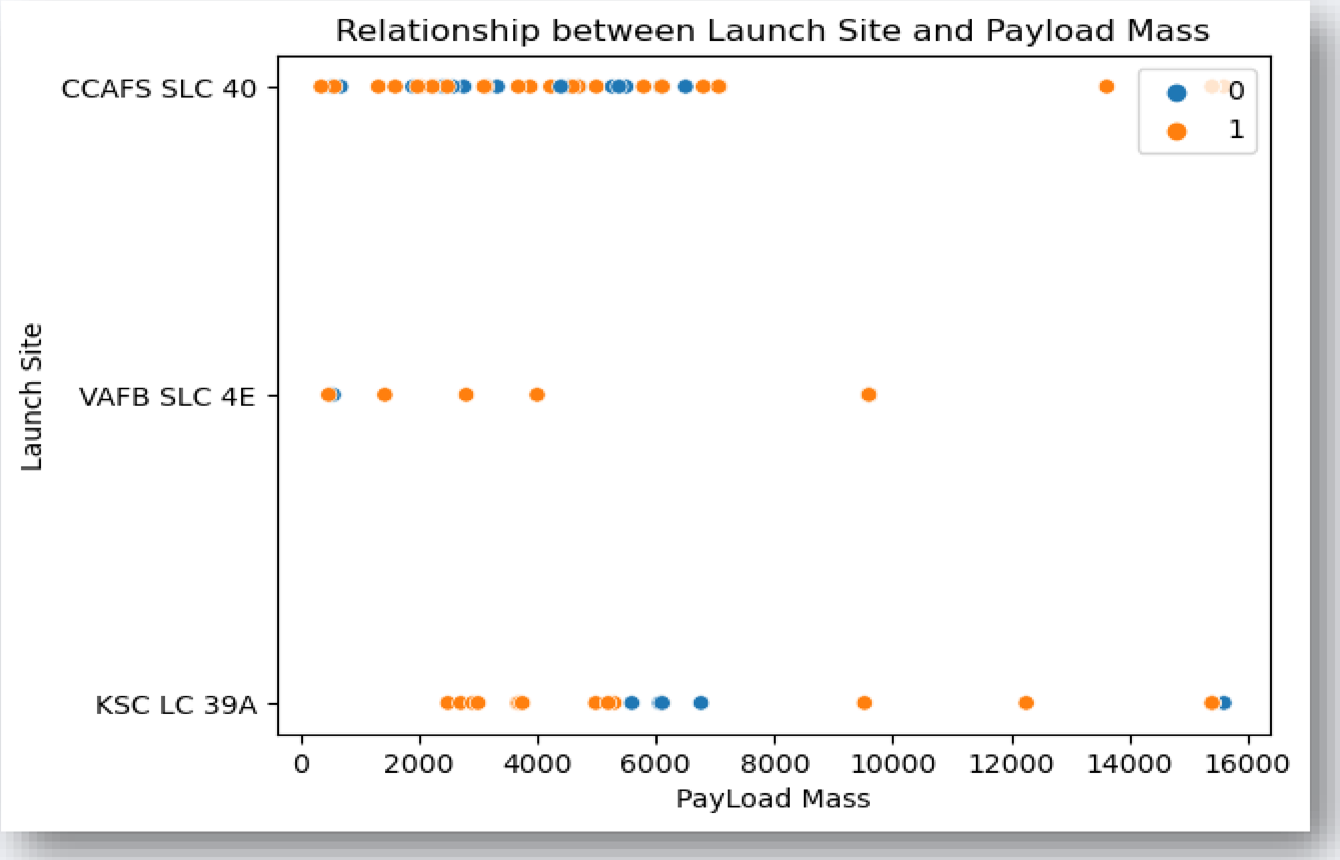
# Flight Number vs. Launch Site



This scatter plot indicates that as the number of flights from a launch site increases, the success rate also tends to increase.

However, the launch site CCAFS SLC40 exhibits a less pronounced pattern in this regard.

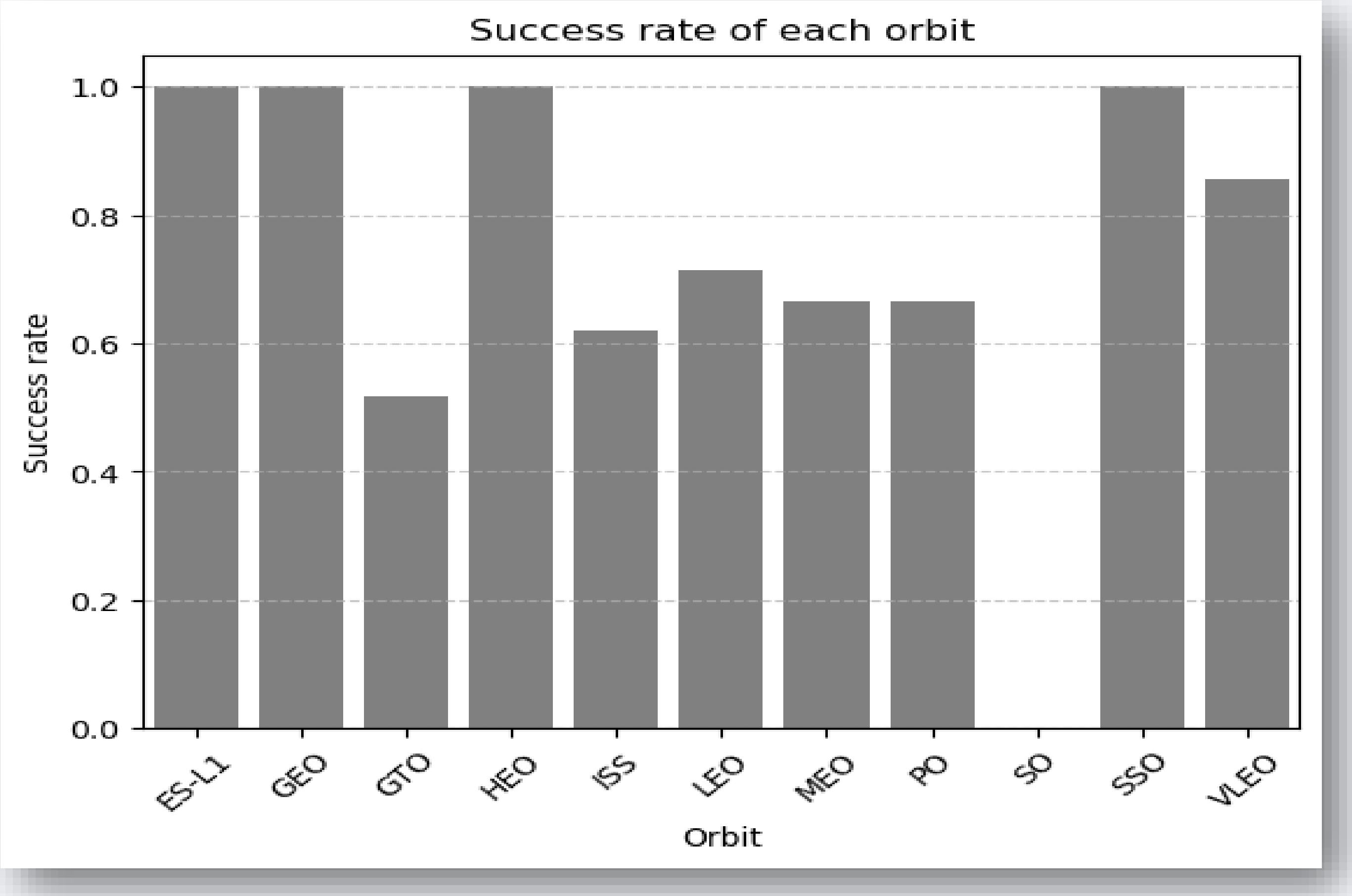
# Payload vs. Launch Site



This scatter plot reveals that when the payload mass exceeds 7000 kg, the probability of success significantly increases.

However, there is no discernible pattern indicating that the launch site is dependent on the payload mass for the success rate.

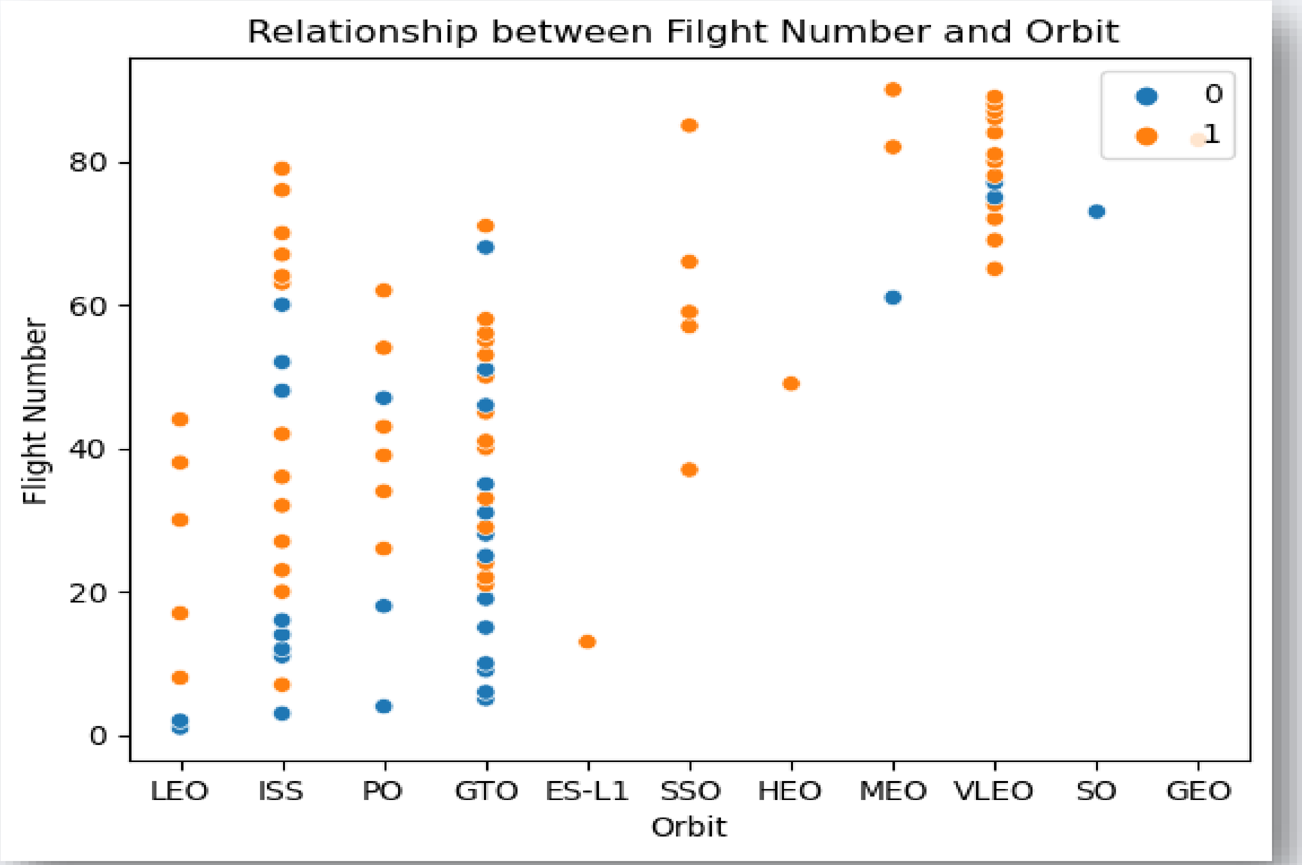
# Success Rate vs. Orbit Type



This figure illustrates the potential influence of orbits on landing outcomes, with some orbits exhibiting a 100% success rate, such as SSO, HEO, GEO, and ES L1, while SO orbit shows a 0% success rate.

However, further analysis reveals that some of these orbits have only one occurrence in the dataset, such as GEO, SO, HEO, and ES L1. This suggests that additional data is needed to observe any meaningful patterns or trends before concluding.

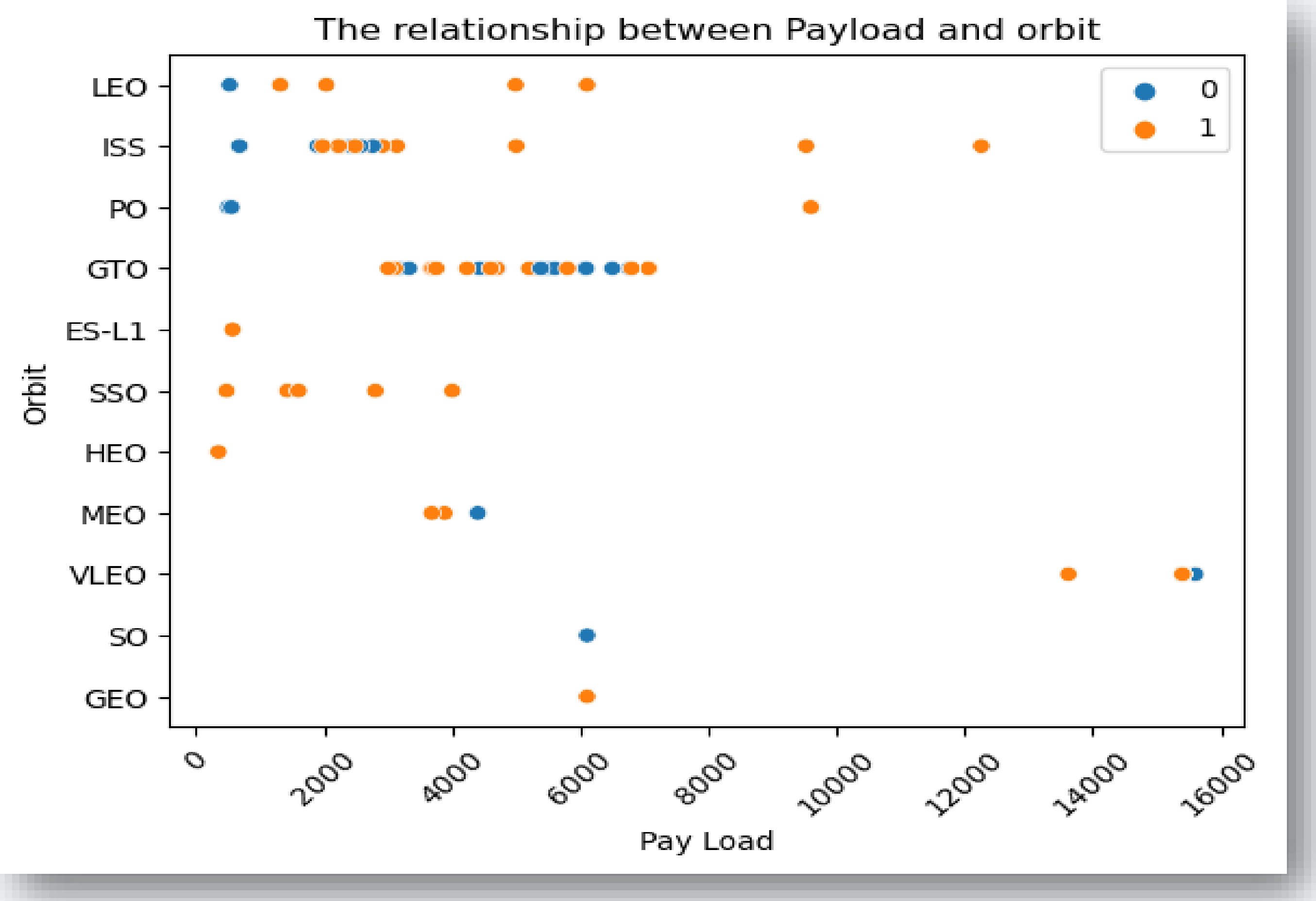
# Flight Number vs. Orbit Type



This scatter plot suggests that, in general, the success rate tends to increase with the larger flight numbers on each orbit, particularly noticeable in the LEO orbit.

However, the GTO orbit shows no clear relationship between these attributes. It's important to note that orbits with only one occurrence should be excluded from the above statement, as they require additional data for meaningful analysis.

# Payload vs. Orbit Type



Heavier payloads have a positive impact on the LEO, ISS, and P0 orbits. However, they exhibit a negative impact on the MEO and VLEO orbits.

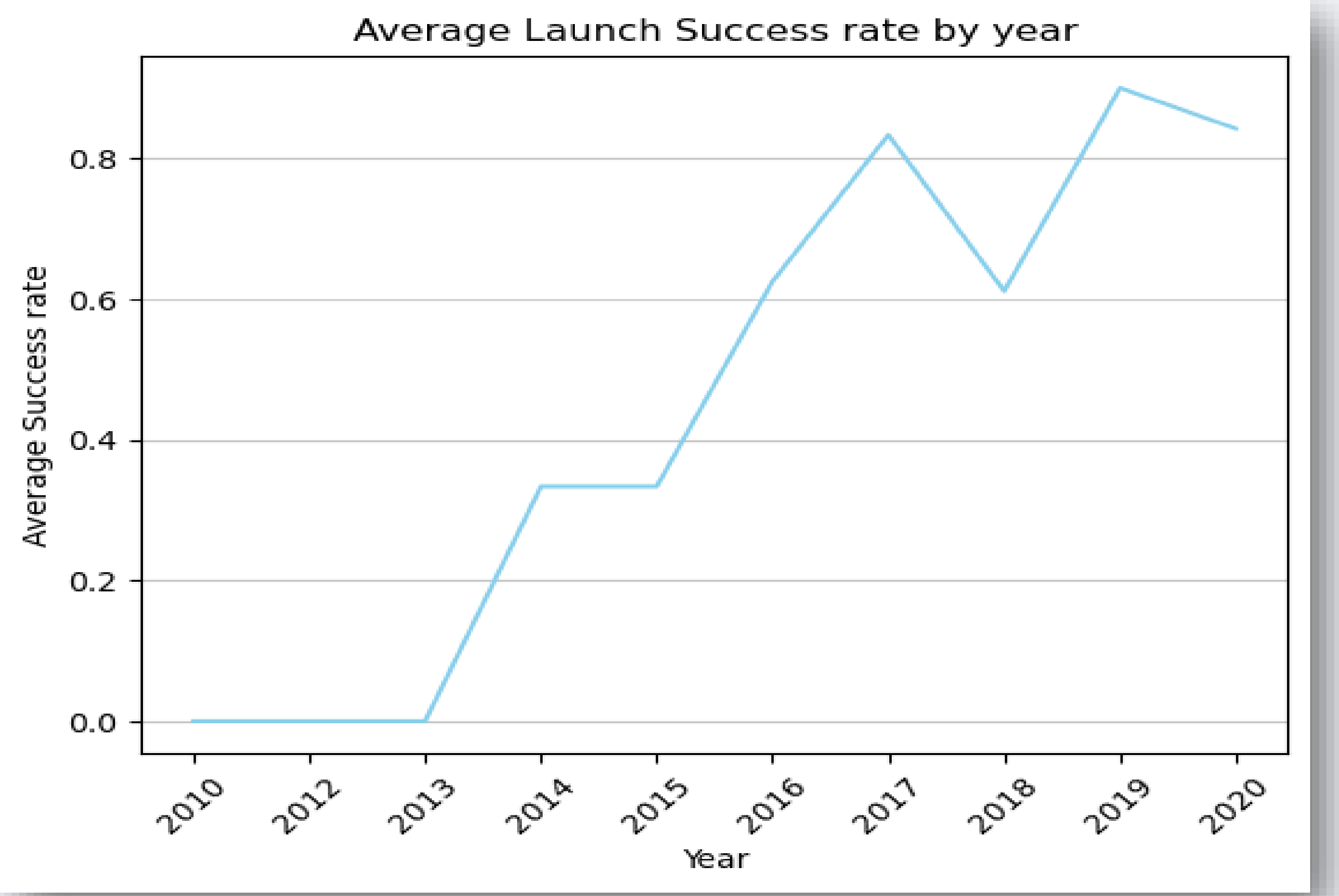
The GTO orbit appears to show no clear relationship between the attributes.

Meanwhile, further data is required to discern any patterns or trends for the SO, GEO, and HEO orbits.

# Launch Success Yearly Trend

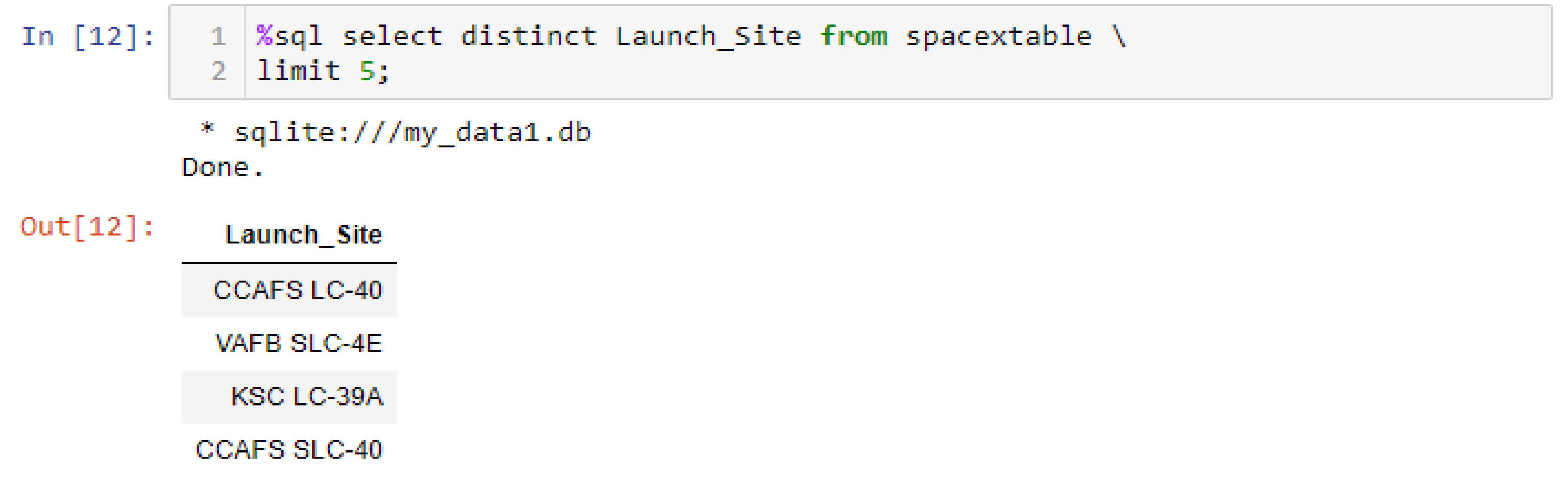
These figures clearly illustrate an increasing trend from the year 2013 until 2020.

If this trend continues into the next year and beyond, the success rate will steadily increase until reaching a 100% success rate.



# All Launch Site Names

We utilized the keyword DISTINCT to display only unique launch sites from the SpaceX data.



In [12]:

```
%sql select distinct Launch_Site from spacextable \
limit 5;
```

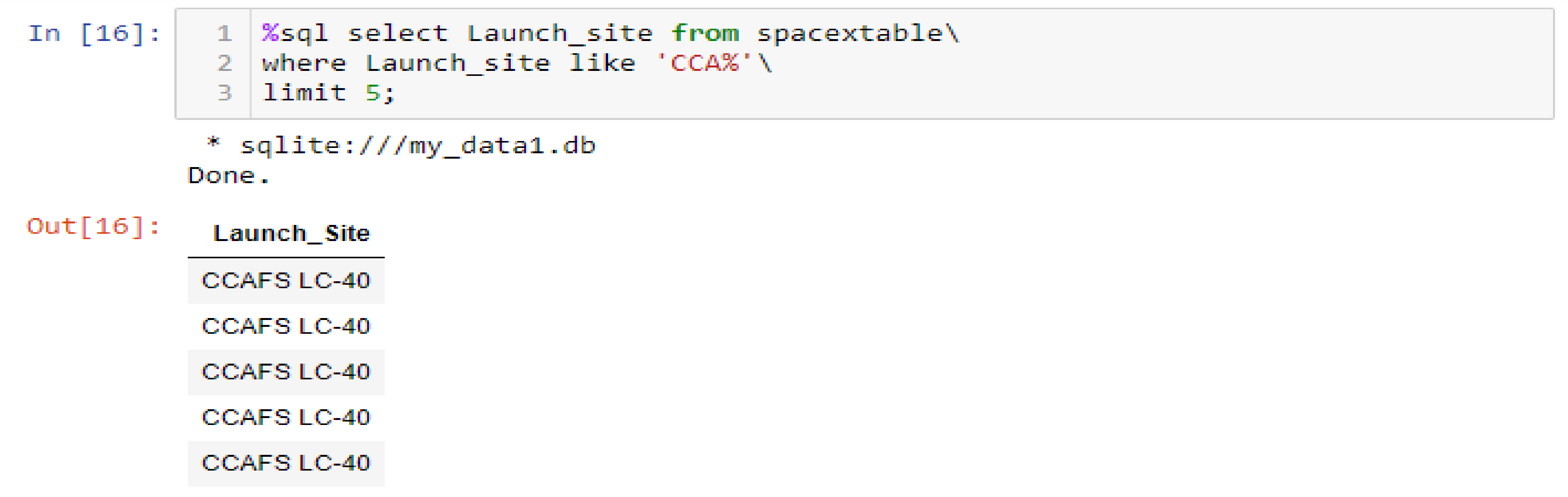
```
 * sqlite:///my_data1.db
Done.
```

Out[12]:

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

# Launch Site Names Begin with 'CCA'

We employed the aforementioned query to retrieve five records where the launch sites start with the specified prefix.



```
In [16]: %sql select Launch_site from spacextable\
         where Launch_site like 'CCA%'\
         limit 5;
```

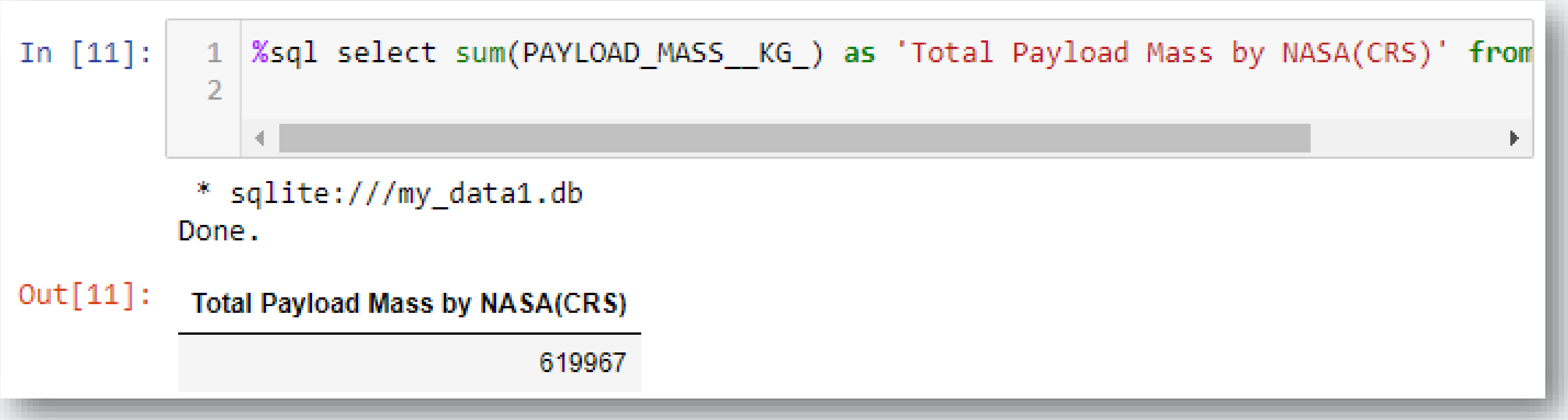
```
 * sqlite:///my_data1.db
Done.
```

Out[16]:

| Launch_Site |
|---|
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |

# Total Payload Mass

The query below allowed us to calculate the total payload carried by boosters from NASA, which amounted to 619,967



In [11]:

```
%sql select sum(PAYLOAD_MASS__KG_) as 'Total Payload Mass by NASA(CRS)' from
```

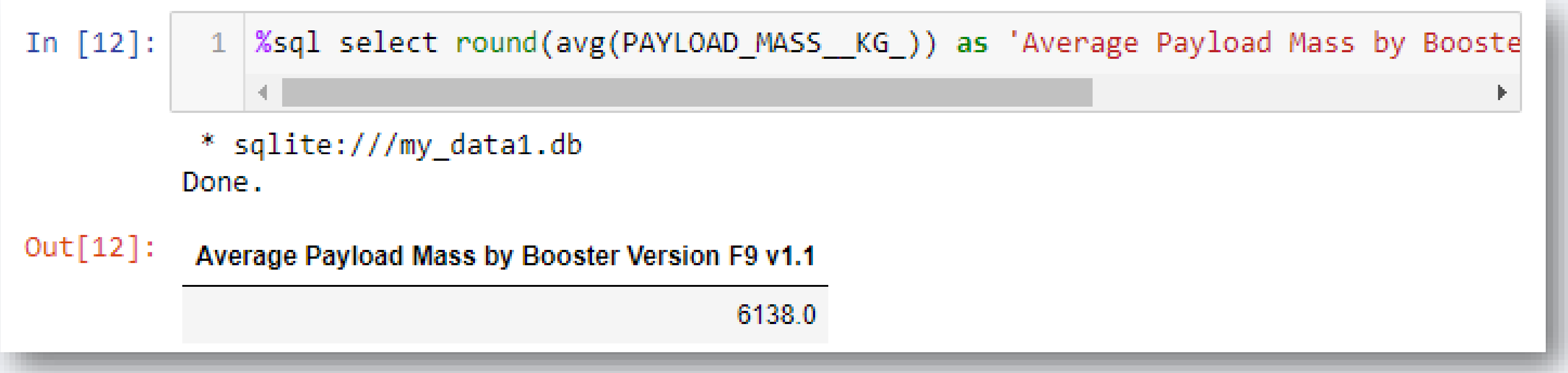
```
 * sqlite:///my_data1.db
Done.
```

Out[11]:

| Total Payload Mass by NASA(CRS) |
|---:|
| 619967 |

# Average Payload Mass by F9 v1.1

We calculated the average payload mass carried by booster version F9 v1.1 as 6138.0



```
In [12]: %sql select round(avg(PAYLOAD_MASS__KG_)) as 'Average Payload Mass by Booste
```

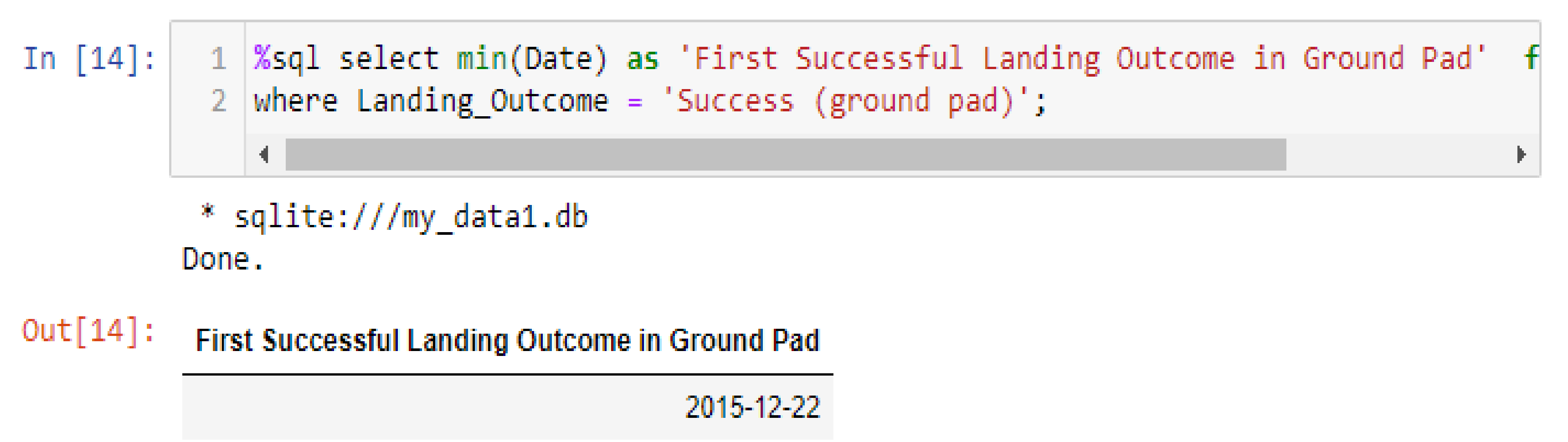
```
 * sqlite:///my_data1.db
Done.
```

Out[12]:

| Average Payload Mass by Booster Version F9 v1.1 |
|---|
| 6138.0 |

# First Successful Ground Landing Date

We utilized the min function to find the result. Upon observation, we noted that the date of the first successful landing outcome on the ground pad was December 22nd, 2015.



In [14]:

```
%sql select min(Date) as 'First Successful Landing Outcome in Ground Pad'  f
where Landing_Outcome = 'Success (ground pad)';
```

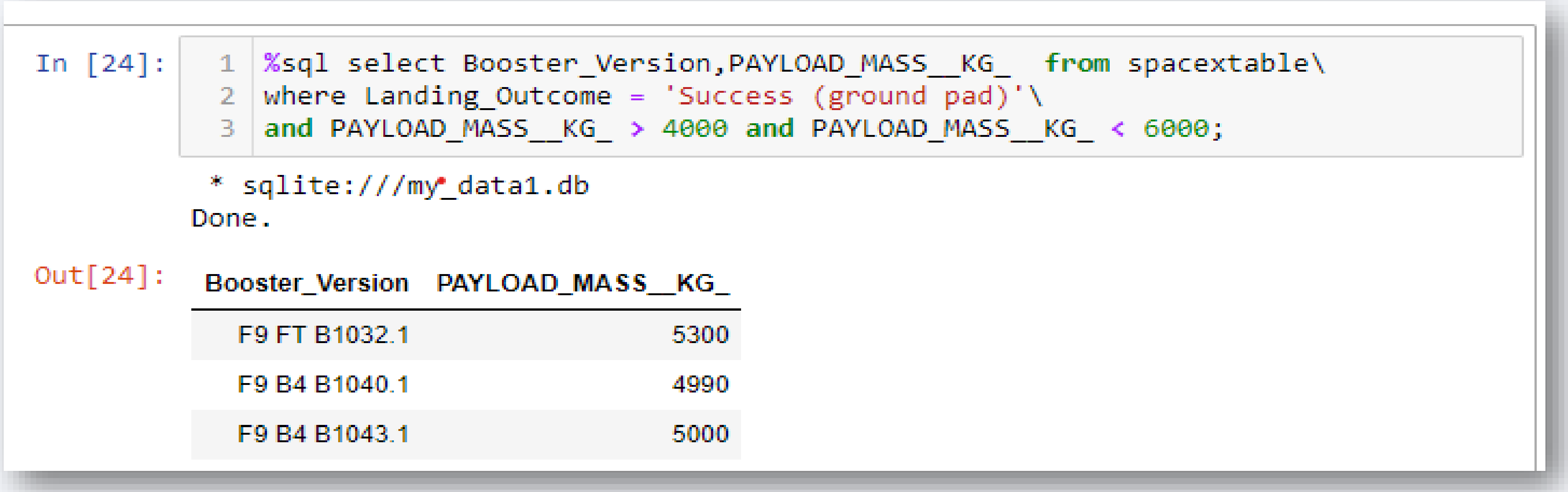
```
 * sqlite:///my_data1.db
Done.
```

Out[14]:

| First Successful Landing Outcome in Ground Pad |
|---:|
| 2015-12-22 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

We utilized the WHERE clause to filter for boosters that have successfully landed on a drone ship. Additionally, we applied the AND condition to identify successful landings with a payload mass greater than 4000 but less than 6000.



```
In [24]:
%sql select Booster_Version,PAYLOAD_MASS__KG_  from spacextable\
where Landing_Outcome = 'Success (ground pad)'\
and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000;
```

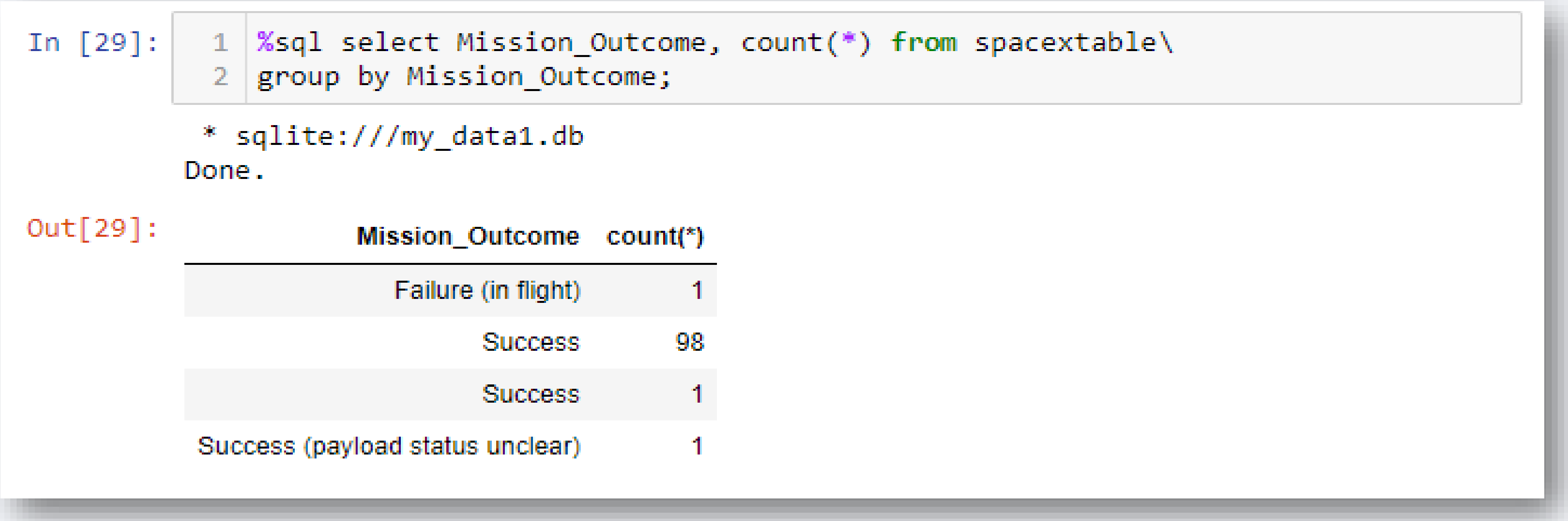
```
 * sqlite:///my_data1.db
Done.
```

Out[24]:

| Booster_Version | PAYLOAD_MASS__KG_ |
|---|---|
| F9 FT B1032.1 | 5300 |
| F9 B4 B1040.1 | 4990 |
| F9 B4 B1043.1 | 5000 |

# Total Number of Successful and Failure Mission Outcomes

We utilized the COUNT and GROUP BY functions on the 'Mission_Outcome' column to generate a table displaying the count of successful missions and failed missions.



In [29]:

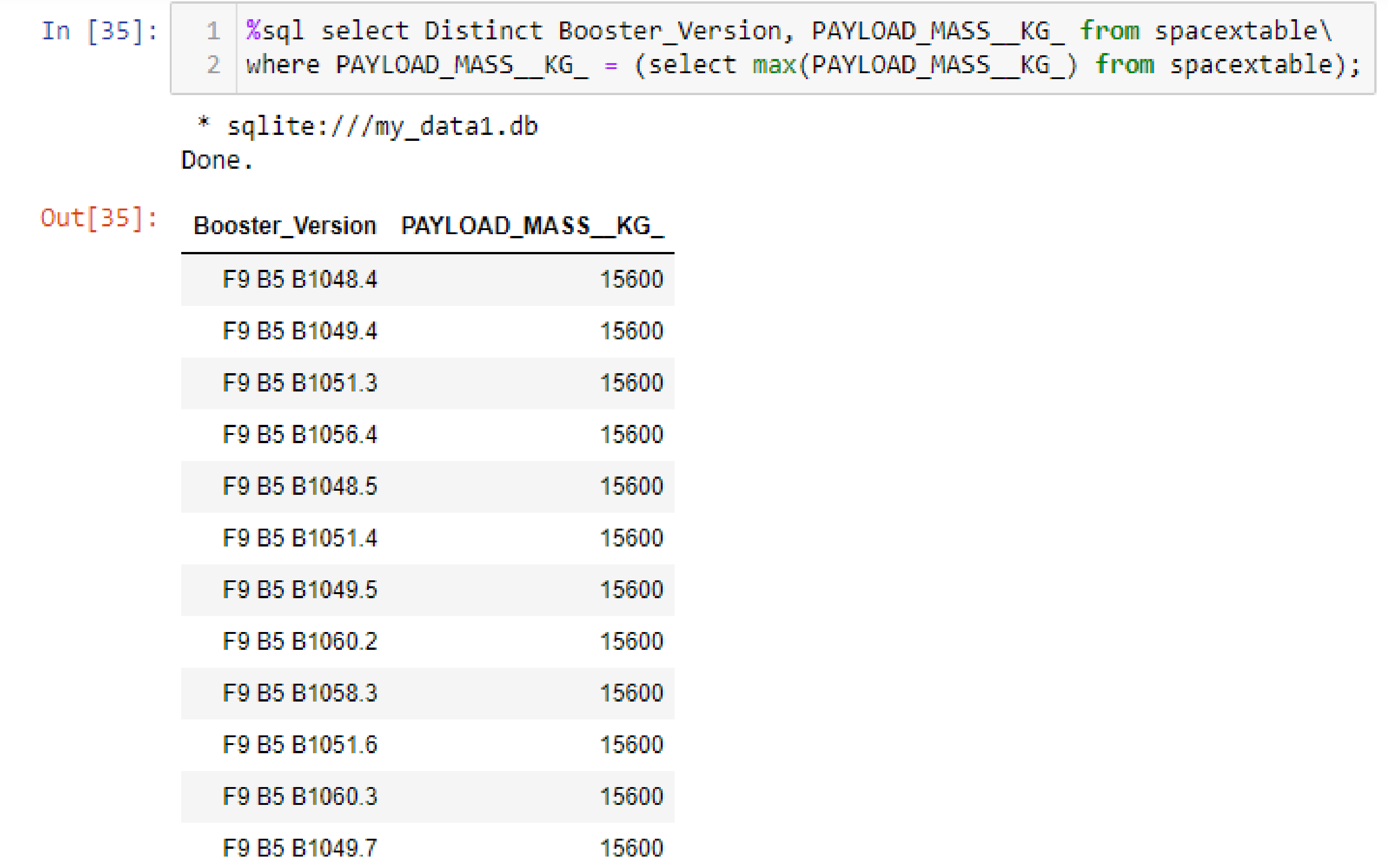
```
%sql select Mission_Outcome, count(*) from spacextable\
group by Mission_Outcome;
```

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

Out[29]:

| Mission_Outcome | count(*) |
|---:|---:|
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload



In [35]:

```
%sql select Distinct Booster_Version, PAYLOAD_MASS__KG_ from spacextable\
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextable);
```

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

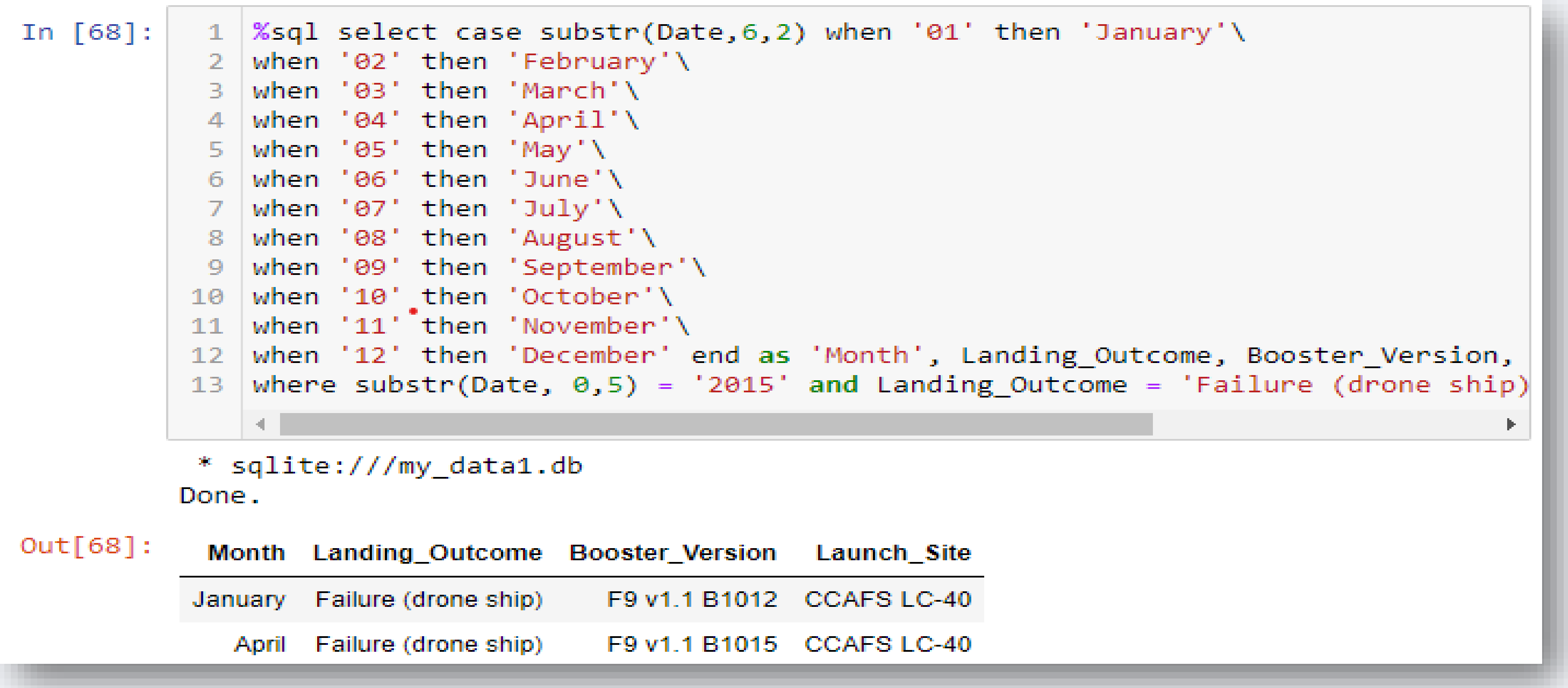
Out[35]:

| Booster_Version | PAYLOAD_MASS__KG_ |
|---|---|
| F9 B5 B1048.4 | 15600 |
| F9 B5 B1049.4 | 15600 |
| F9 B5 B1051.3 | 15600 |
| F9 B5 B1056.4 | 15600 |
| F9 B5 B1048.5 | 15600 |
| F9 B5 B1051.4 | 15600 |
| F9 B5 B1049.5 | 15600 |
| F9 B5 B1060.2 | 15600 |
| F9 B5 B1058.3 | 15600 |
| F9 B5 B1051.6 | 15600 |
| F9 B5 B1060.3 | 15600 |
| F9 B5 B1049.7 | 15600 |

We determined the booster that has carried the maximum payload by using a subquery within the WHERE clause along with the MAX() function.

# 2015 Launch Records

We utilized a combination of the WHERE clause, LIKE operator, AND condition, and BETWEEN condition to filter for failed landing outcomes on a drone ship, including their booster versions and launch site names for the year 2015.



In [68]:

```
%sql select case substr(Date,6,2) when '01' then 'January'\
when '02' then 'February'\
when '03' then 'March'\
when '04' then 'April'\
when '05' then 'May'\
when '06' then 'June'\
when '07' then 'July'\
when '08' then 'August'\
when '09' then 'September'\
when '10' then 'October'\
when '11' then 'November'\
when '12' then 'December' end as 'Month', Landing_Outcome, Booster_Version,
where substr(Date, 0,5) = '2015' and Landing_Outcome = 'Failure (drone ship)
```

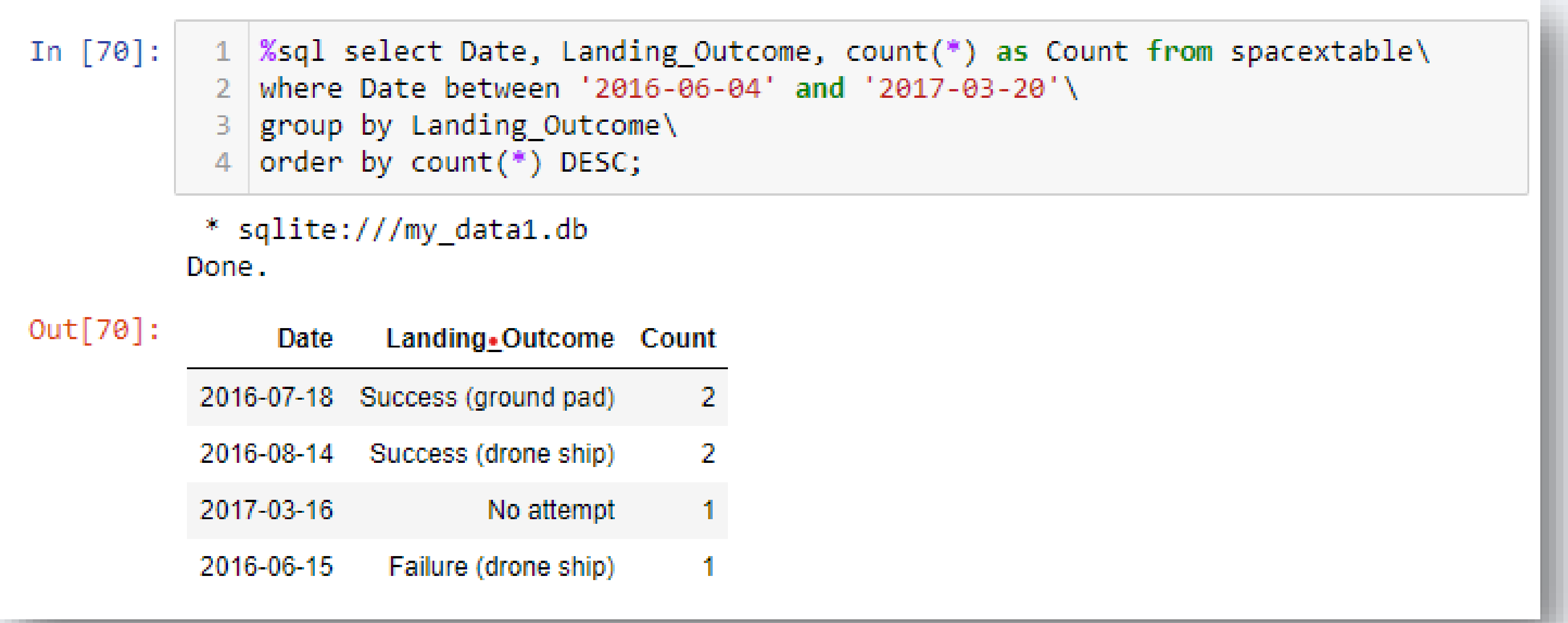
```
 * sqlite:///my_data1.db
Done.
```

Out[68]:

| Month | Landing_Outcome | Booster_Version | Launch_Site |
|---|---|---|---|
| January | Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 |
| April | Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 |

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

We selected landing outcomes and the count of landing outcomes from the data, using the WHERE clause to filter for landing outcomes between June 4, 2010, and March 20, 2010. We then applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to arrange the grouped landing outcomes in descending order.



In [70]:

```
%sql select Date, Landing_Outcome, count(*) as Count from spacextable\
where Date between '2016-06-04' and '2017-03-20'\
group by Landing_Outcome\
order by count(*) DESC;
```

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

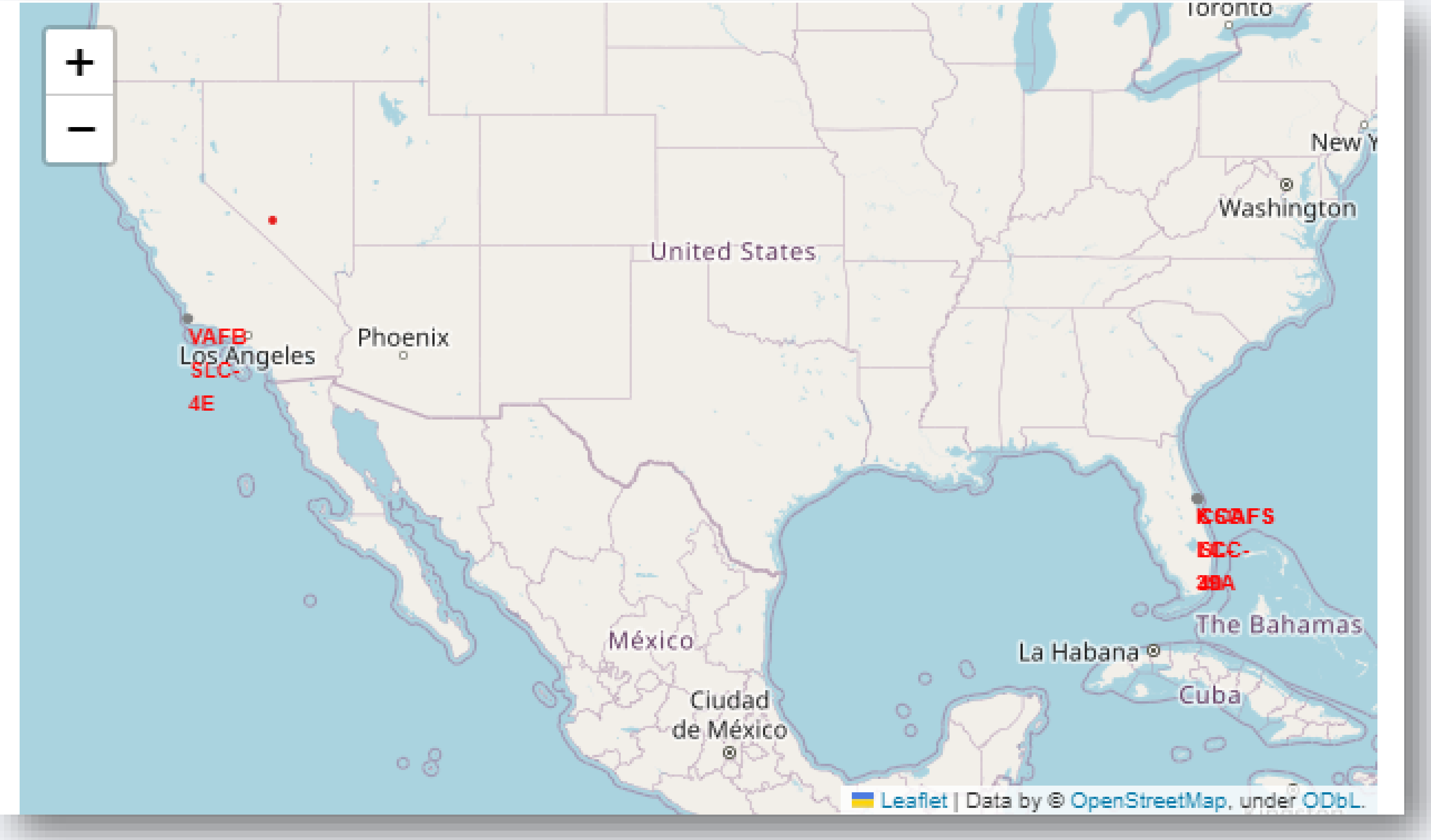
Out[70]:

| Date | Landing_Outcome | Count |
|---|---|---|
| 2016-07-18 | Success (ground pad) | 2 |
| 2016-08-14 | Success (drone ship) | 2 |
| 2017-03-16 | No attempt | 1 |
| 2016-06-15 | Failure (drone ship) | 1 |

Section 3

# Launch Sites Proximities Analysis

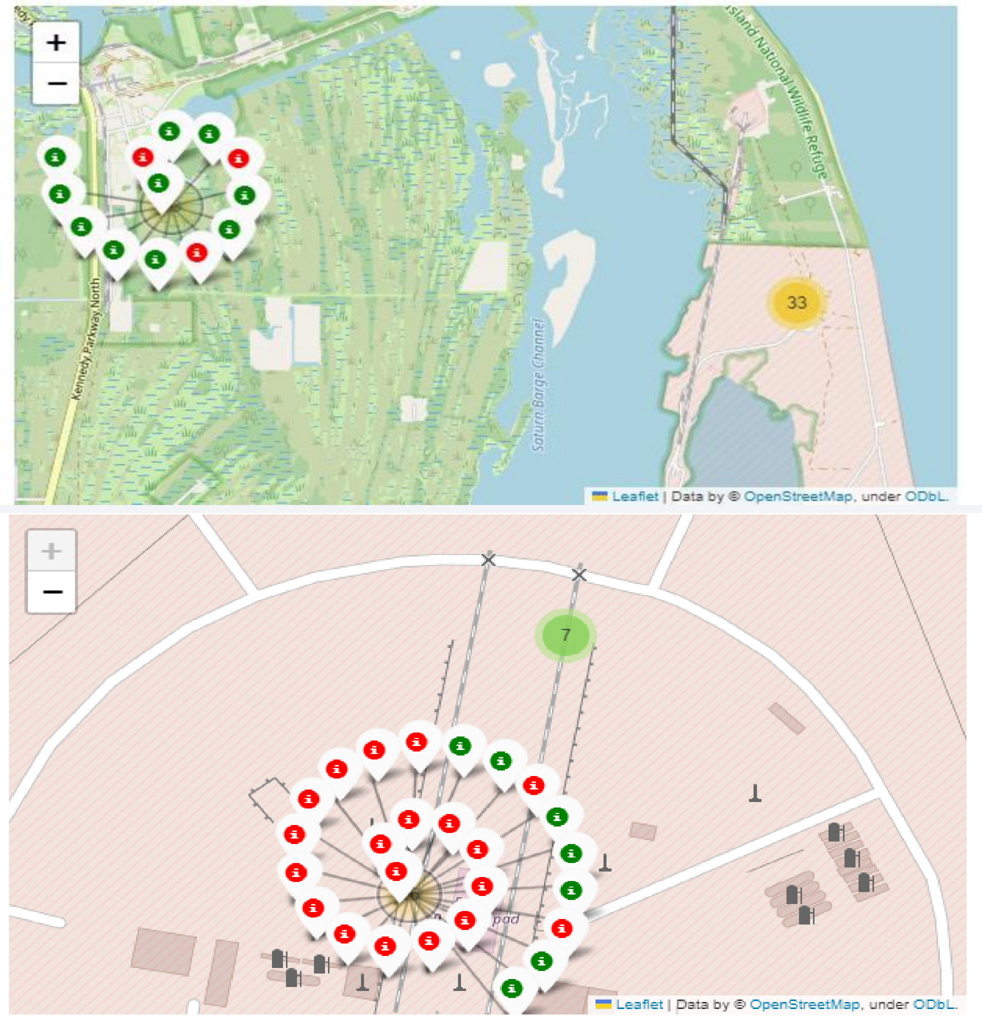
# Location of all the Launch Sites



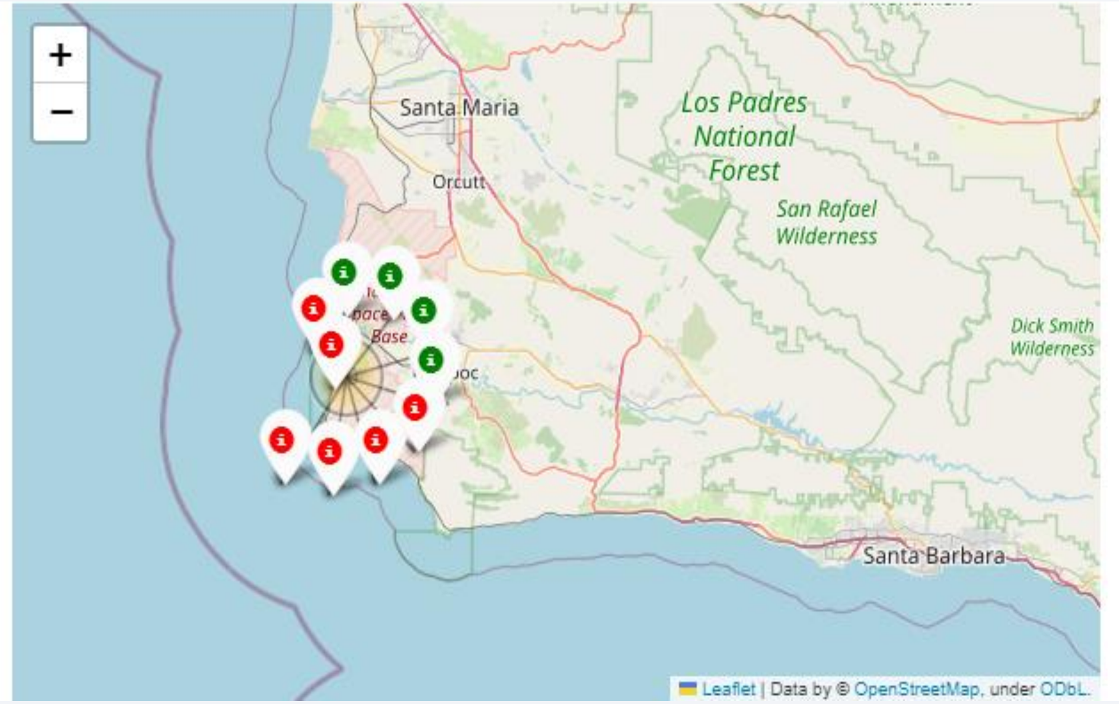
All SpaceX launch sites are situated within the borders of the United States.

# Folium-Map Markers Identifying Launch sites with color labels

Florida Launch Site


+
−
Kennedy Parkway North
Island National Wildlife Refuge
Saturn Barge Channel
33
Leaflet | Data by © OpenStreetMap, under ODbL.
+
−
7
pad
Leaflet | Data by © OpenStreetMap, under ODbL.

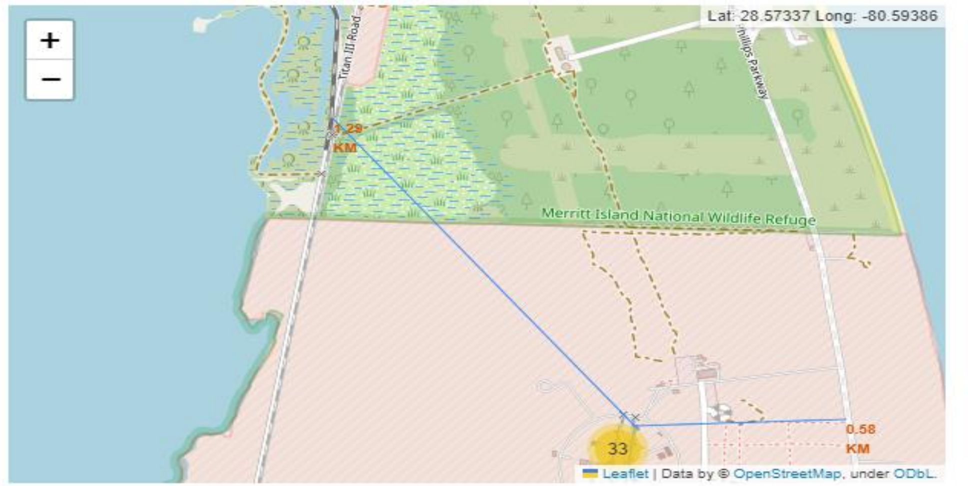
California Launch Site

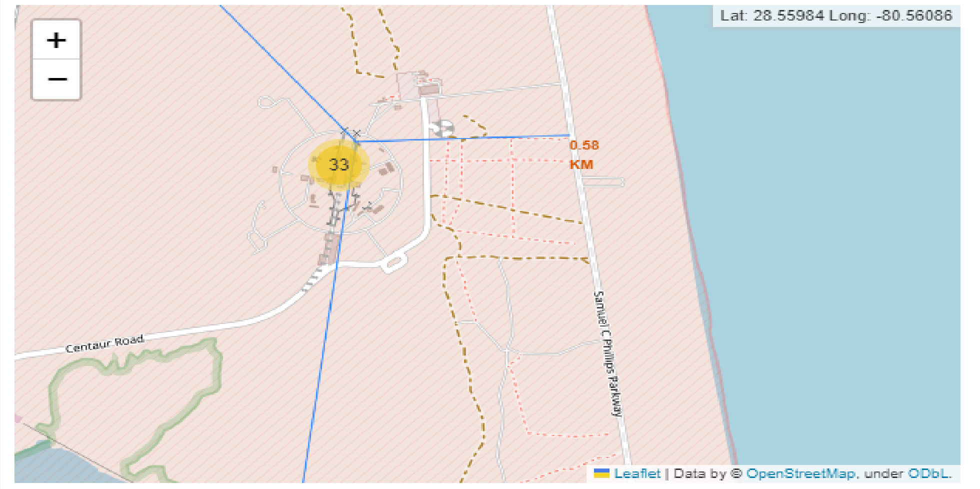


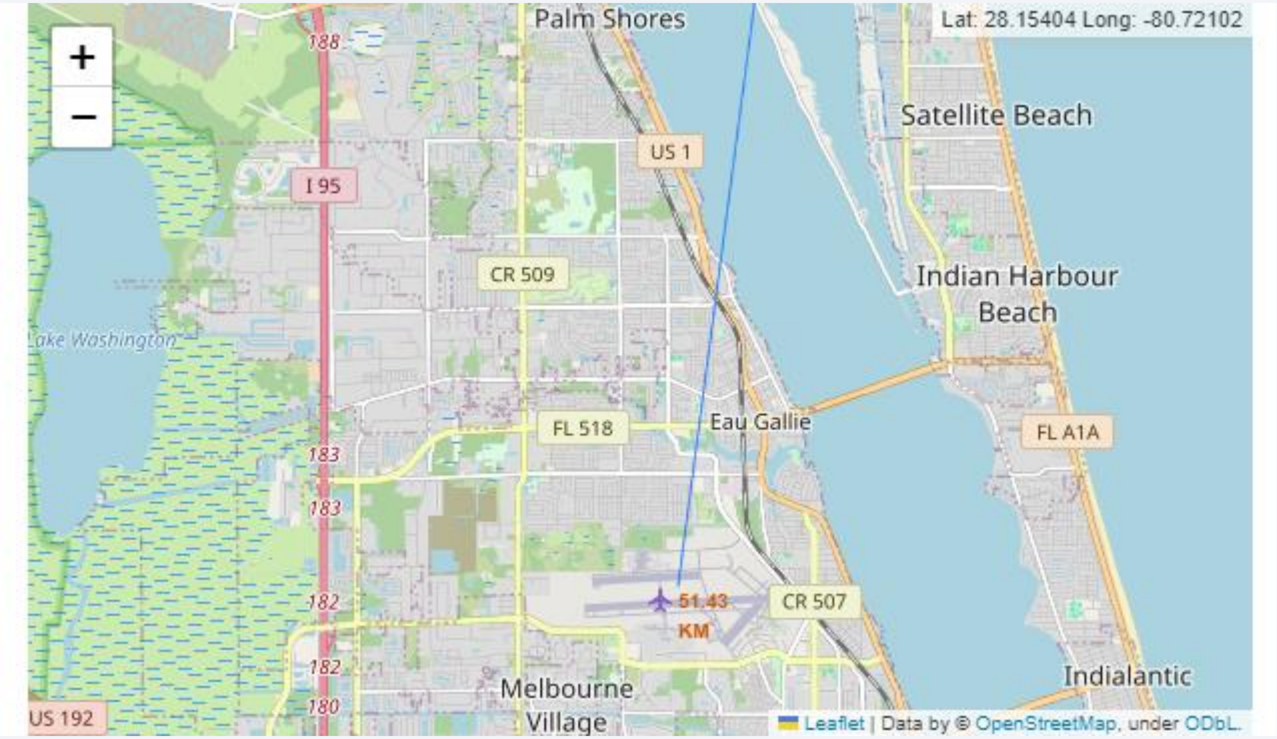
Green Marker showing Successful Launches and Red Marker identifying Failures

# Launch Sites Distance to Landmarks

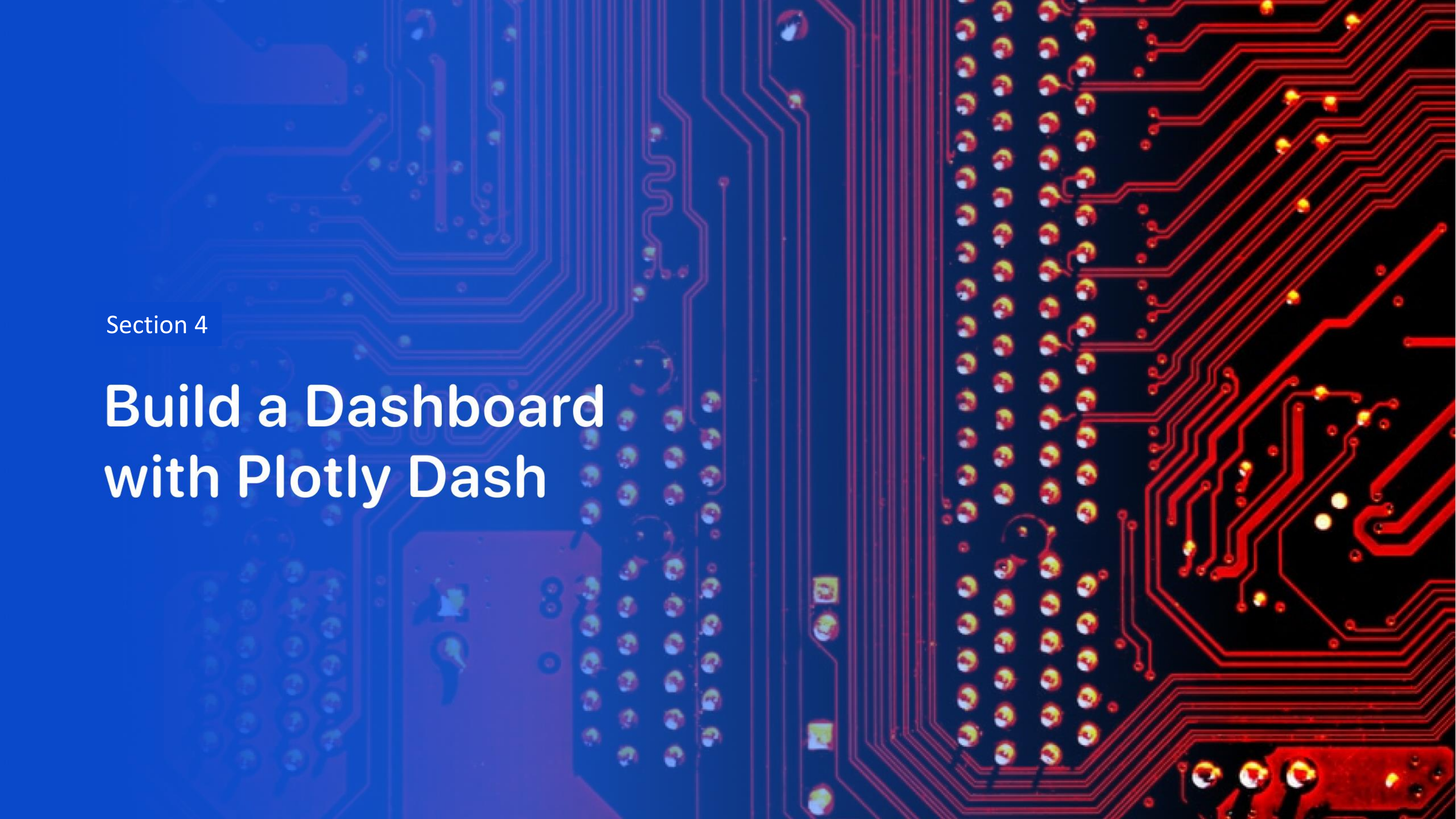
The Florida launch site proximity to railways and highways.


Lat: 28.57337 Long: -80.59386
+
−
Titan III Road
Phillips Parkway
1.29 KM
Merritt Island National Wildlife Refuge
33
0.58 KM
Leaflet | Data by © OpenStreetMap, under ODbL.


Lat: 28.55984 Long: -80.56086
+
−
33
0.58 KM
Centaur Road
Samuel C Phillips Parkway
Leaflet | Data by © OpenStreetMap, under ODbL.



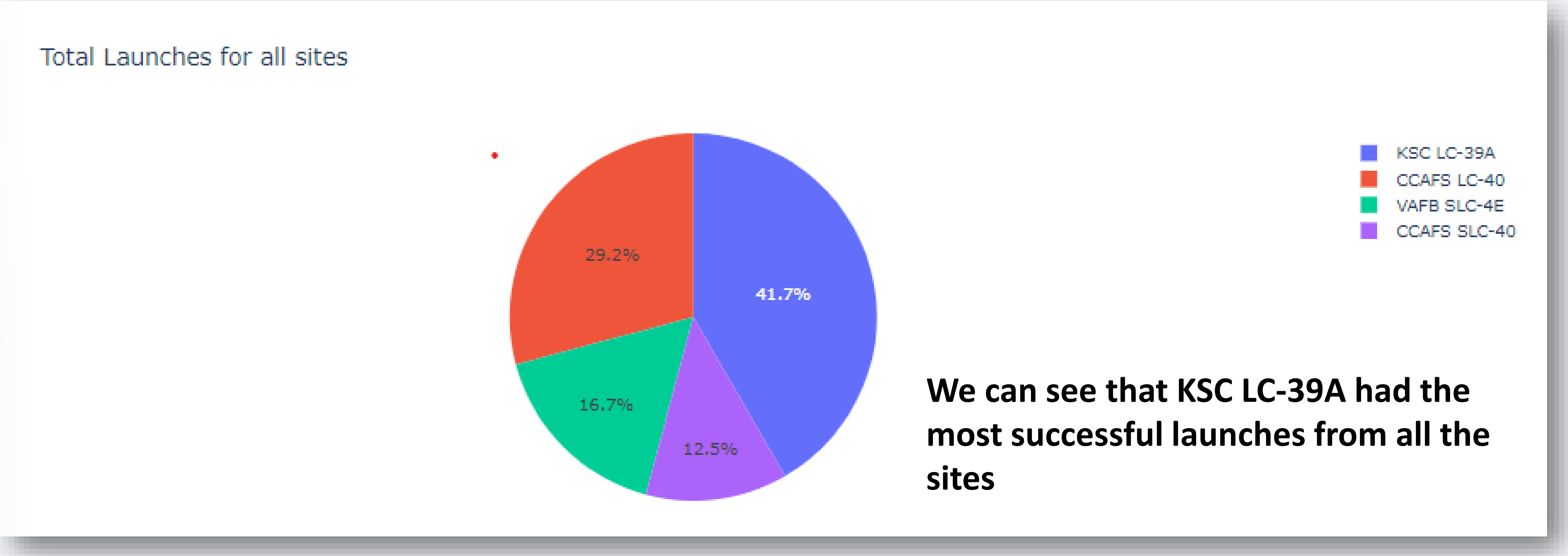
The Florida launch site’s proximity Orlando Melbourne Airport



Section 4

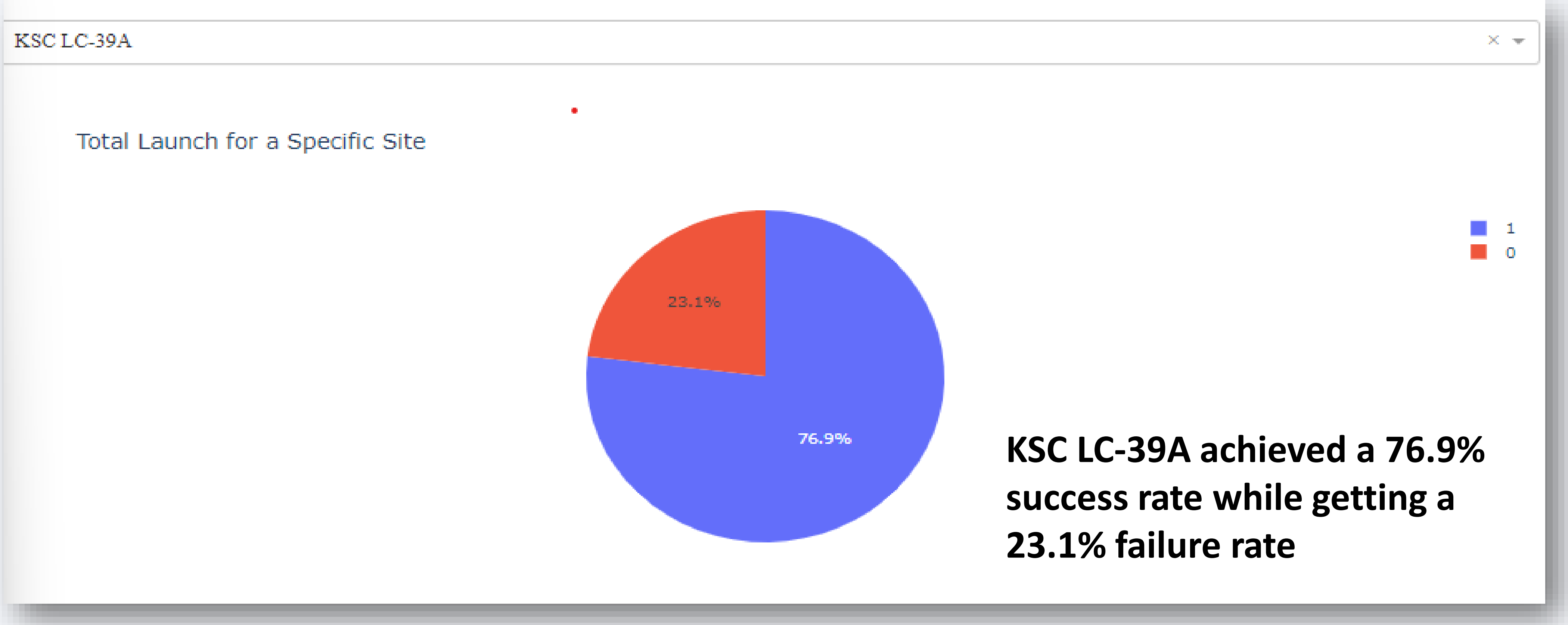
# Build a Dashboard with Plotly Dash

# The success percentage by each sites.



**We can see that KSC LC-39A had the most successful launches from all the sites**

# The highest Launch-success ratio: KSC LC-39A

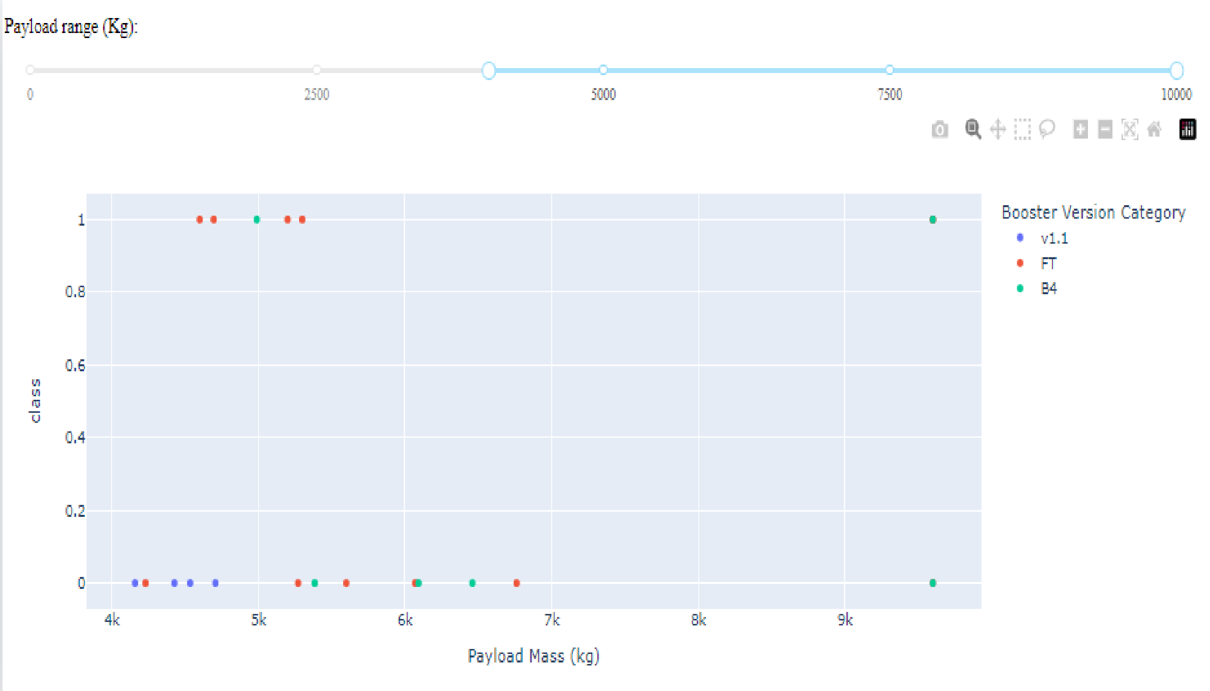


KSC LC-39A

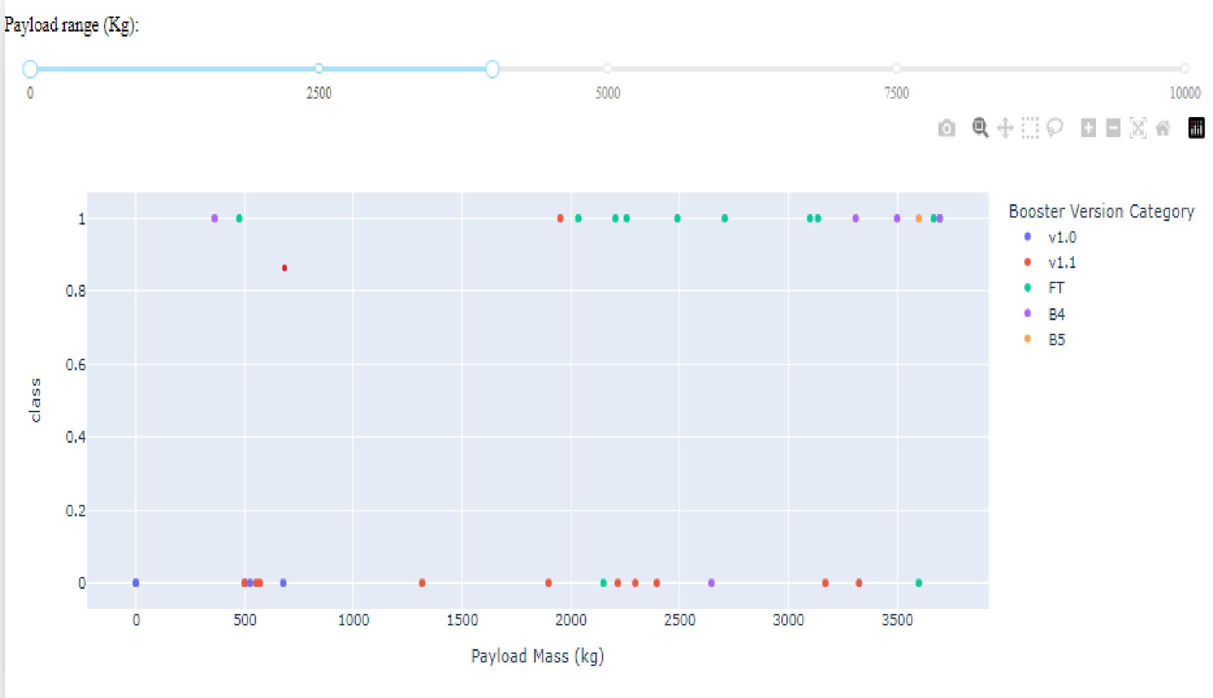
**KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate**

# Launch Outcome vs Payload Scatter plot.

The success rates for payloads with lower weights consistently surpass those of heavier payloads.



**Low weighted Payload 0Kg – 4000kg**



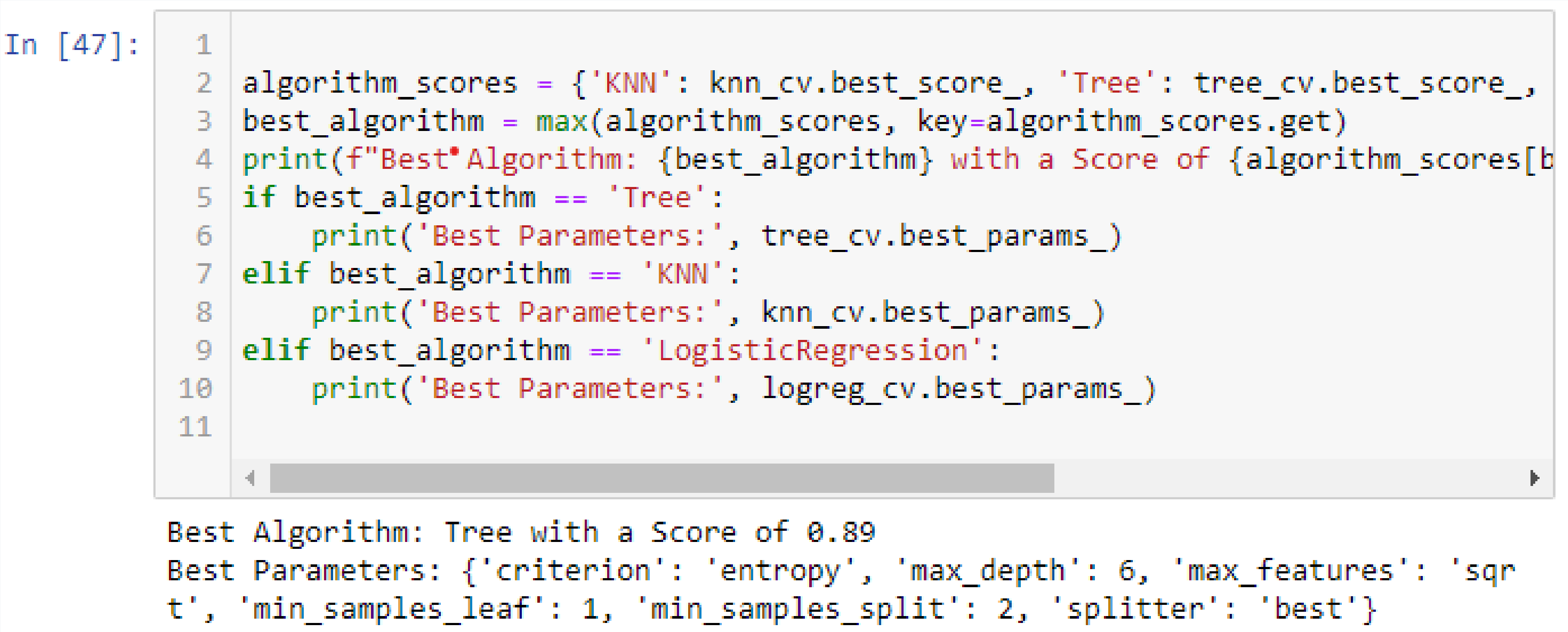
**Heavy weighted Payload 4000Kg – 10000kg**

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

With the implementation of the following code, it becomes evident that the Tree Algorithm emerges as the optimal choice, boasting the highest classification accuracy among the algorithms considered.



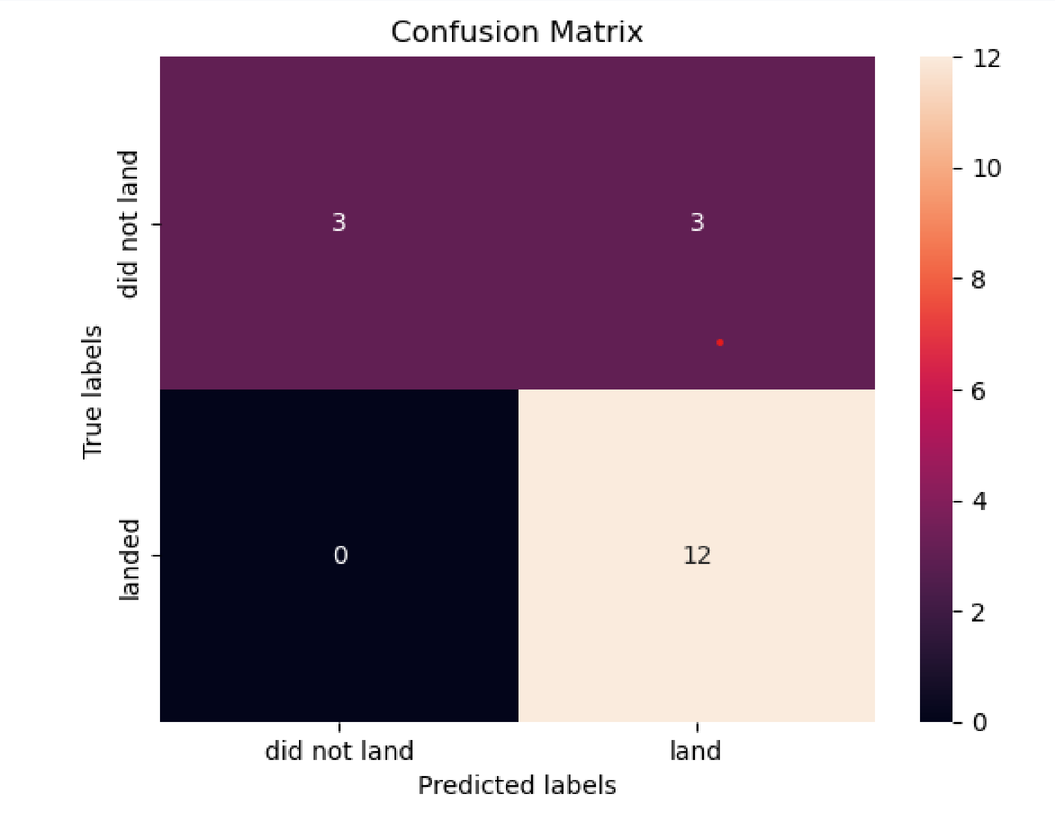
In [47]:

```

algorithm_scores = {'KNN': knn_cv.best_score_, 'Tree': tree_cv.best_score_,
best_algorithm = max(algorithm_scores, key=algorithm_scores.get)
print(f"Best Algorithm: {best_algorithm} with a Score of {algorithm_scores[b
if best_algorithm == 'Tree':
    print('Best Parameters:', tree_cv.best_params_)
elif best_algorithm == 'KNN':
    print('Best Parameters:', knn_cv.best_params_)
elif best_algorithm == 'LogisticRegression':
    print('Best Parameters:', logreg_cv.best_params_)

```

```
Best Algorithm: Tree with a Score of 0.89
Best Parameters: {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqr
t', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
```

# Confusion Matrix

The confusion matrix analysis of the decision tree classifier demonstrates its capability to discern between various classes. However, a notable issue arises with false positives, where unsuccessful landings are incorrectly classified as successful ones by the classifier.



# Conclusions

We can conclude that:

- The Tree Classifier Algorithm is the most effective machine learning approach for this dataset.
- Low-weighted payloads (defined as 4000kg and below) exhibit better performance compared to heavy-weighted payloads.
- Success rates for SpaceX launches have shown an increasing trend since 2013, indicating improvement over time with a projected trajectory toward perfection in future launches.
- KSC LC 39A has the highest success rate among all launch sites, at 76.9%.
- The SSO orbit demonstrates the highest success rate, achieving 100% success with more than one occurrence.

# Thank you!