IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nathan Setiawan
18-02-2024

Outline

Executive Summary

- Summary of methodologies
  - Data Collection through API and with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL and Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - EDA results
  - Interactive analytics and Predictive Analytics result from Machine Learning Lab

Introduction

- SpaceX has transformed the space industry by offering Falcon 9 rocket launches at significantly reduced costs compared to other providers due their approach of reusing the first stage of the rocket.
- As a data scientist working for a startup competing with SpaceX, the objective of this project is to develop a machine learning pipeline capable of predicting the landing outcomes of the rocket's first stage, determining a competitive pricing strategy against SpaceX for rocket launches.
- The problems aimed to be solved include:
  - Identifying the variables that affect the outcome of the landing.
  - Determine the relationship between variables and their impact on the likelihood of a successful landing.

Section 1

Methodology

Methodology

Executive Summary

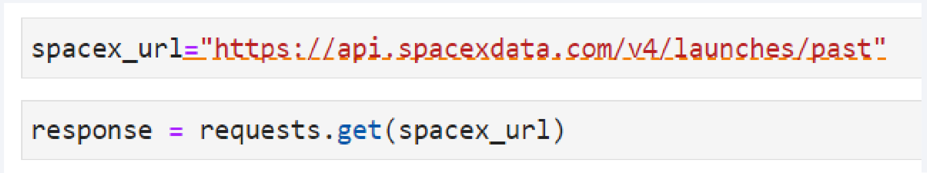
- Data collection methodology:
  - Making use of SpaceX REST API and web scrapping from Wikipedia.
- Perform data wrangling
  - Processing using one-hot encoding to create categorical features.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models.

Data Collection

- For the REST API, the data collection process began with initiating a GET request before the response content was decoded into JSON format and converted into a pandas DataFrame using the json_normalize() function. Afterwards, any missing values were identified and appropriately filled.
- The BeautifulSoup library is used to retrieve launch records in the form of an HTML table during web scraping. This table is then parsed and transformed into a pandas DataFrame to enable further analysis.

Data Collection – SpaceX API

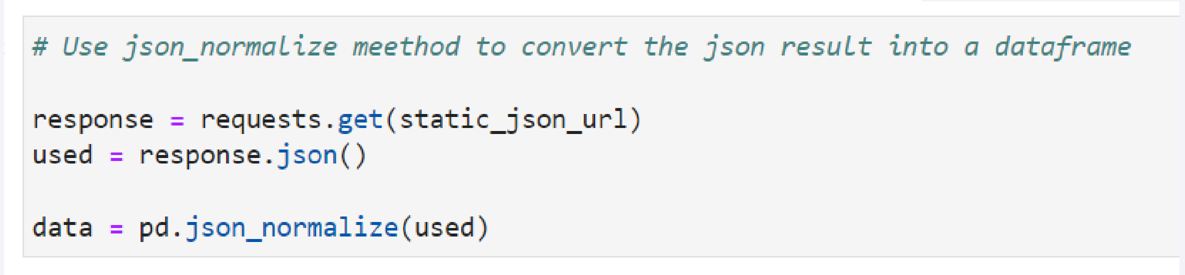
- Here, a GET request for the rocket launch data was done using the API (1) before json_normalize was used to convert the json result into a dataframe (2).
- Afterwards, data cleaning including the handling of missing data is performed on the dataframe (3).
- GitHub URL: https://github.com/Foxy0309/Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



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

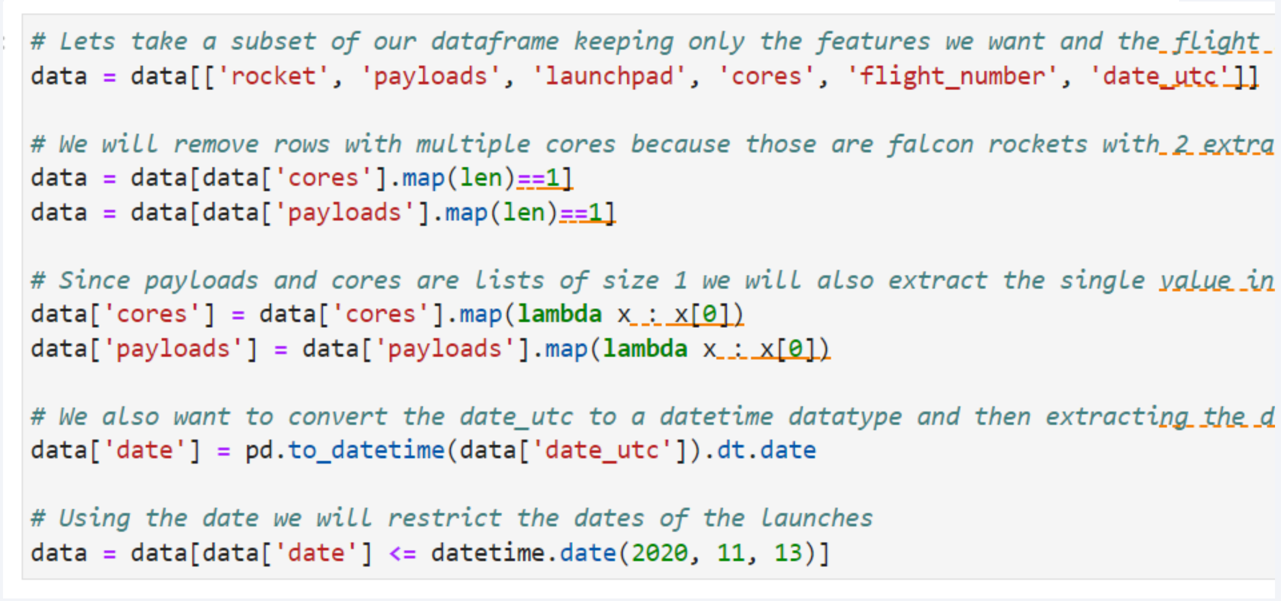
```
response = requests.get(spacex_url)
```

1



```
# Use json_normalize meethod to convert the json result into a dataframe

response = requests.get(static_json_url)
used = response.json()

data = pd.json_normalize(used)
```

2

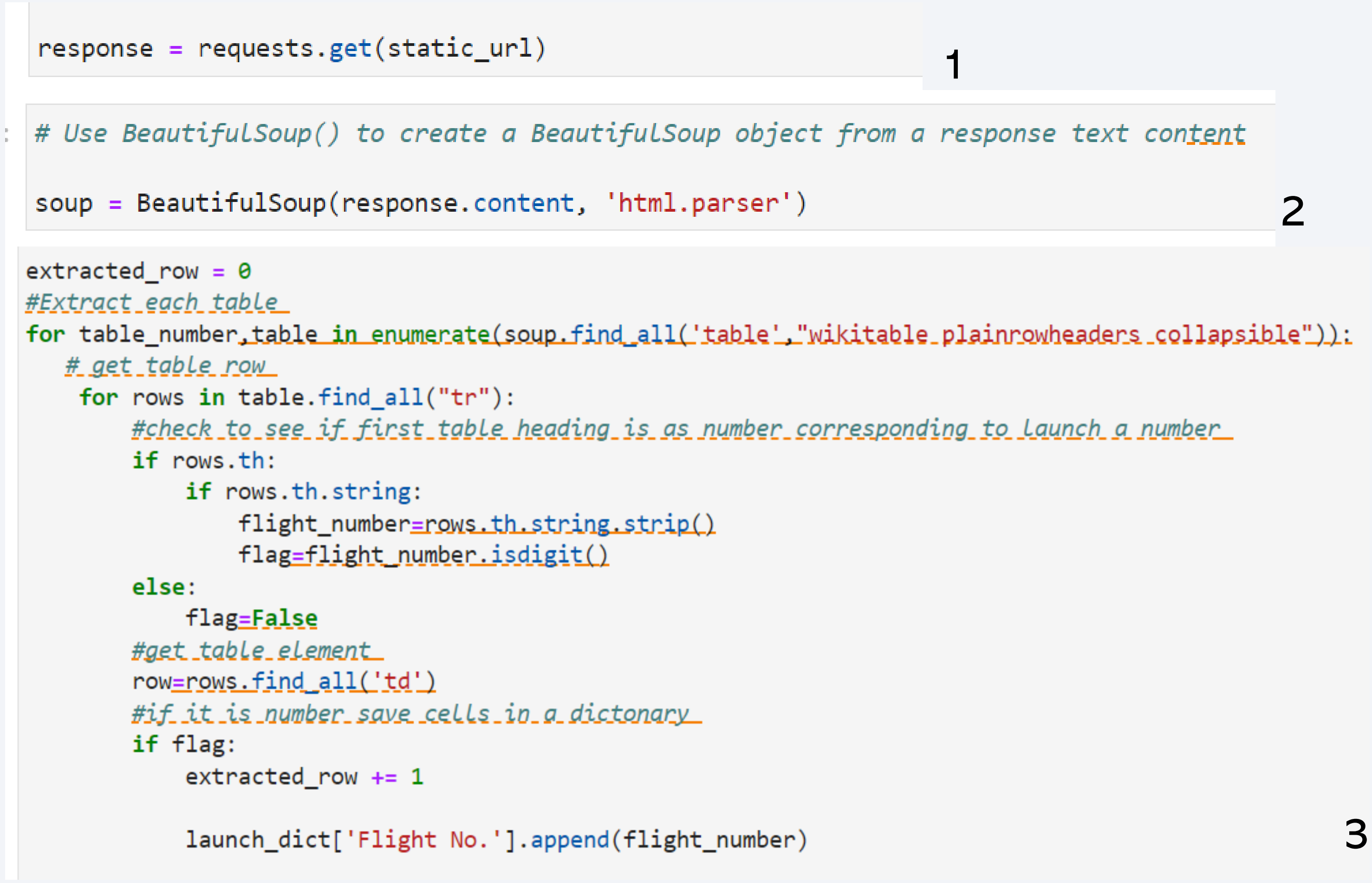


```
# Lets take a subset of our dataframe keeping only the features we want and the flight
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra
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
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 d
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)]
```

3

Data Collection - Scraping

- Here, the launch wiki page was requested via url (1) before a BeautifulSoup object was created from the HTML response (2).
- Afterwards, all column or variable names were extracted from the HTML header (3).
- GitHub URL: https://github.com/Foxy0309/Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb



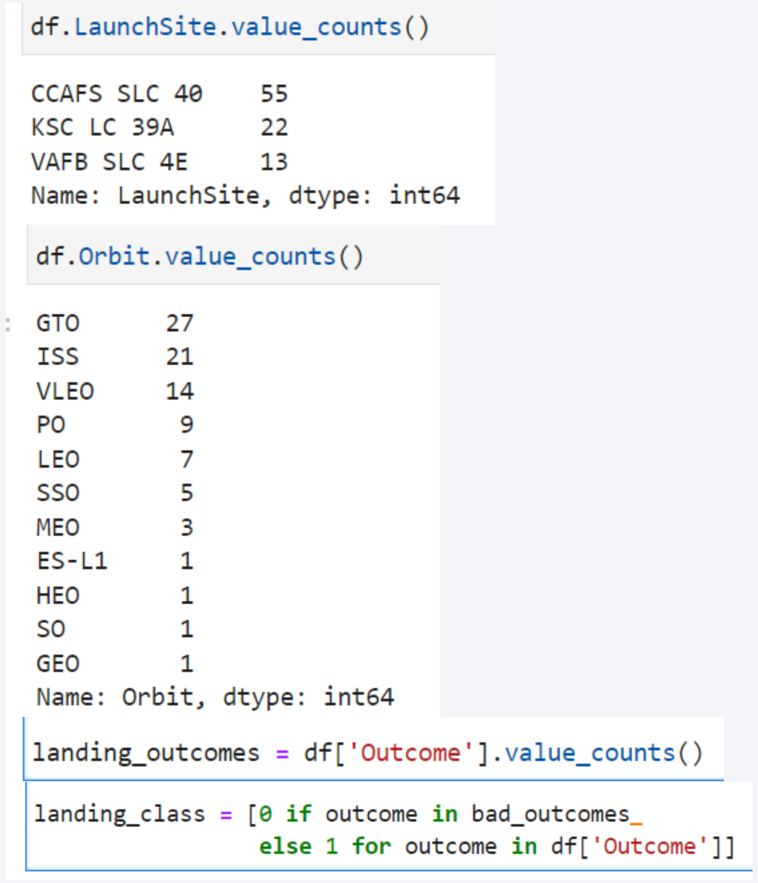
```
response = requests.get(static_url)
```
1

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

soup = BeautifulSoup(response.content, 'html.parser')
```
2

```
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

            launch_dict['Flight No.'].append(flight_number)
```
3

Data Wrangling

- Transforming the string variables into categorical variables is needed here (for example True Ocean, True RTLS, True ASDS all point to a successful mission) by making these variables into 1 if the mission was successful or 0 when the mission was a failure.
- The process begins by calculating launch numbers for each site, calculating the numbers and occurrences on each orbit as well as the mission outcomes for each type of orbit.
- A landing outcome label is then created from the outcome column for easier visualization.
- GitHub URL: https://github.com/Foxy0309/Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



```
df.LaunchSite.value_counts()

CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

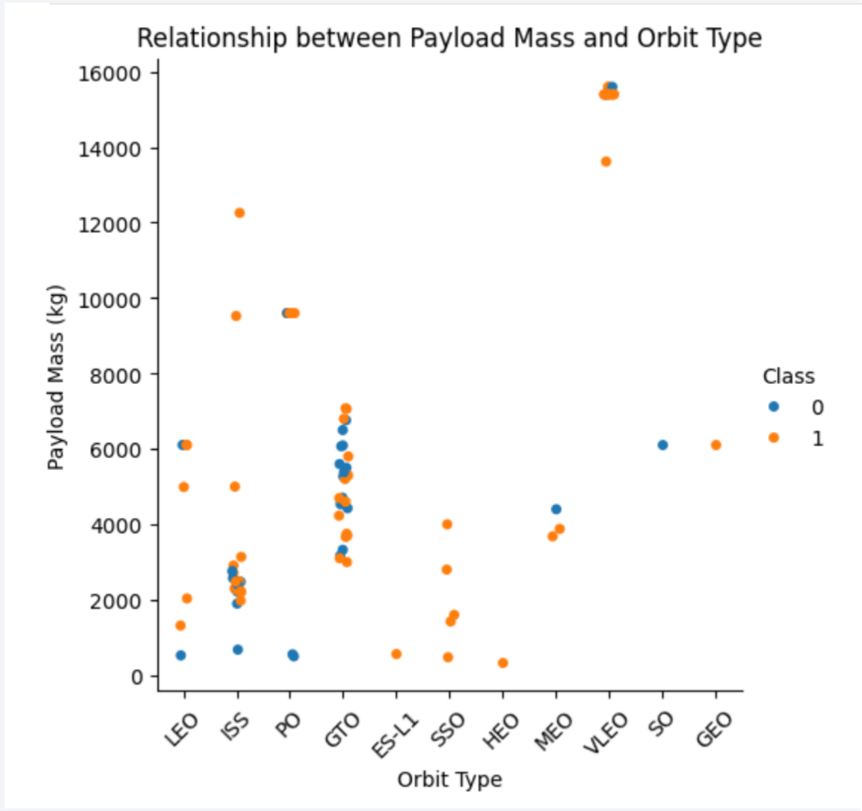
```
df.Orbit.value_counts()

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
```

```
landing_outcomes = df['Outcome'].value_counts()
```

```
landing_class = [0 if outcome in bad_outcomes_
                else 1 for outcome in df['Outcome']]
```

EDA with Data Visualization

- Among the charts that were plotted, scatter plots were the most aplenty. These plots showed the relationship between Orbit vs. Payload Mass, Flight Number vs. Launch Site, Payload vs. Launch Site and some other relationships.
- A bar graph showing the success rate by orbit and a line graph showing the yearly trend of launch success were also plotted.
- Github URL: https://github.com/Foxy0309/Data-Science-Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb



EDA with SQL

- The following SQL queries were performed:
  - Retrieving the distinct names of launch sites used in space missions.
  - Displaying 5 entries where launch sites start with 'CCA'.
  - Calculating the total payload mass transported by boosters for missions launched by NASA (CRS).
  - Calculating the average payload mass transported by the booster version F9 v1.1.
  - Identifying the date of the first successful landing on a ground pad.
  - Identifying booster names that achieved success on a drone ship and carried a payload mass of more than 4000 but less than 6000 kilograms.
- Counting the total successful and unsuccessful mission outcomes.
- Identifying booster versions that transported the highest payload mass.
- Retrieving records showing month names, failed landing outcomes on drone ships, booster versions, and launch sites for each month in the year 2015.
- Ranking the number of successful landing outcomes from 04 June 2010 to 20 March 2017, ordered by descending frequency.
- Github URL: https://github.com/Foxy0309/Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(2).ipynb

Build an Interactive Map with Folium

- Folium was used to create an interactive map centered on NASA Johnson Space Center in Houston, Texas, showcasing key data points:
    - Red circle markers for NASA Johnson Space Center and each launch site, with labels for identification.
    - Marker clusters to display launch outcomes, using green for successful and red for unsuccessful landings.
    - Lines and markers to illustrate distances from launch sites to significant landmarks like railways, highways, coastlines, and cities.
- This visualization aids in understanding the spatial aspects of launch sites.
- GitHub URL: https://github.com/Foxy0309/Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- The Plotly Dash interactive dashboard that allows users to analyze data through:
  - Pie charts for total launches by site.
  - Scatter plots correlating Outcome with Payload Mass for various booster versions.
- Dashboard features:
  - A dropdown to select launch sites (dash_core_components.Dropdown).
  - Pie charts for success/failure rates per selected site (plotly.express.pie).
  - A range slider for filtering payload mass (dash_core_components.RangeSlider).
  - Scatter plots showing Success vs. Payload Mass relationship (plotly.express.scatter).
- Github URL: https://github.com/Foxy0309/Data-Science-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Data Preparation and Model Building:
  - Import the dataset.
  - Perform data normalization.
  - Divide the dataset into training and testing subsets.
- Model Preparation:
  - Choose appropriate machine learning algorithms.
  - Apply GridSearchCV to determine optimal parameters for each algorithm.
  - Train models using the GridSearchCV with the training dataset.
- Model Evaluation:
  - Identify the optimal hyperparameters for each model.
  - Calculate the accuracy of each model on the test dataset.
  - Generate and examine the Confusion Matrix for each model.
- Model Comparison
  - Evaluate and compare the models based on their accuracy.
  - Select the model with the highest accuracy as the best model.

Github URL: https://github.com/Foxy0309/Data-Science-Capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5_jupyterlite%20(1).ipynb

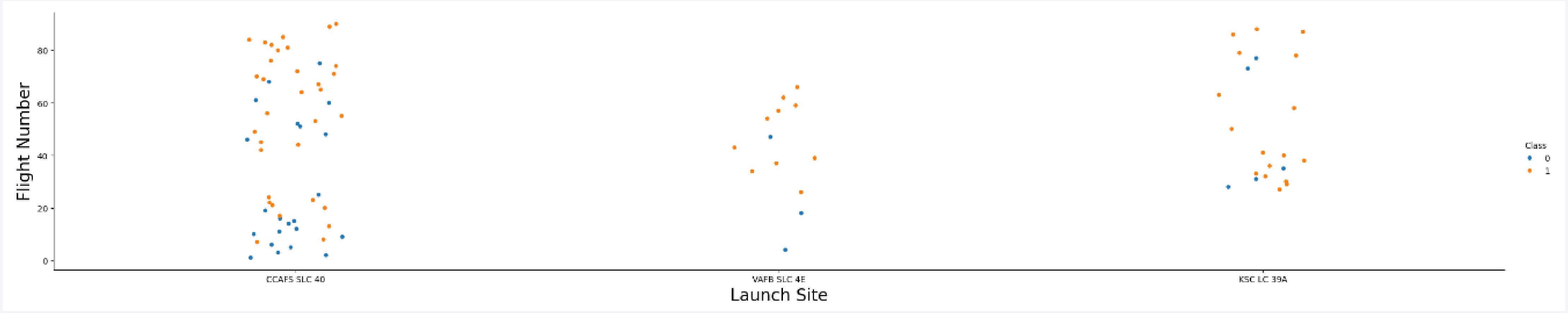
Results

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

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

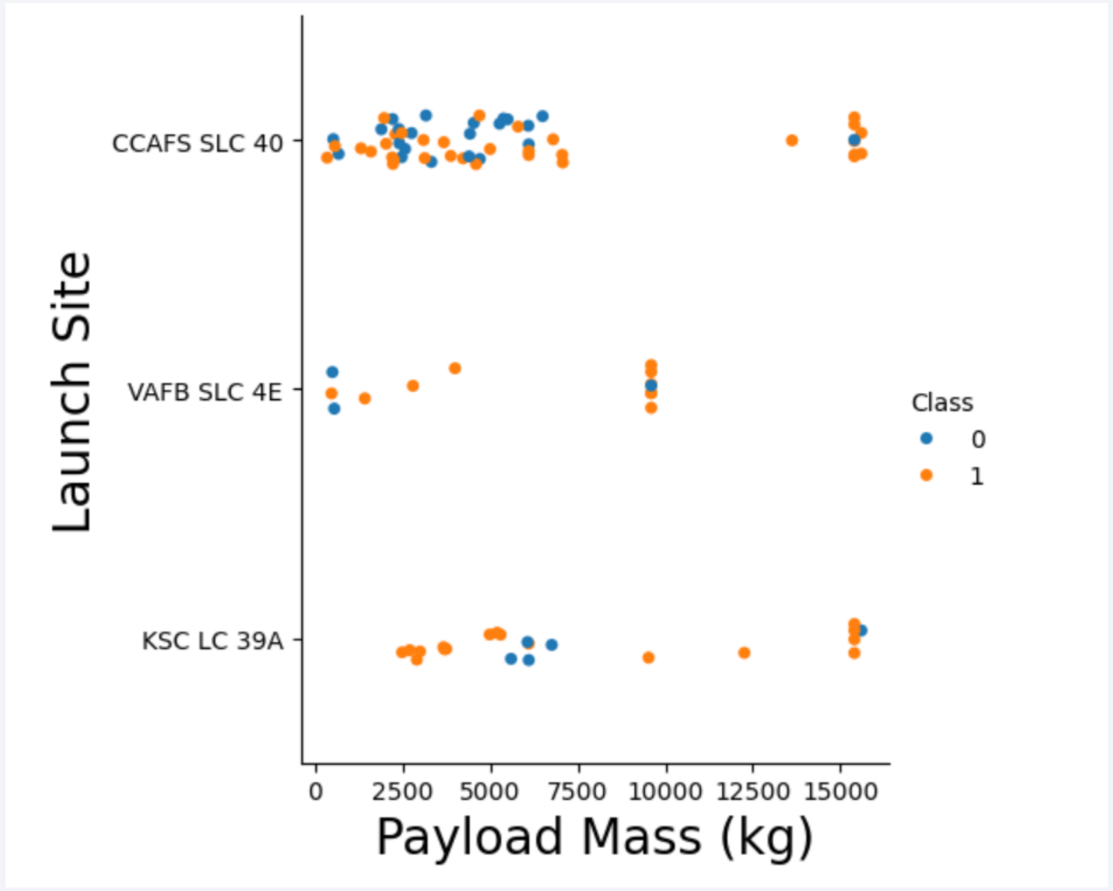


The plot shows that the higher the number of flights the progressively greater the success rate will be.

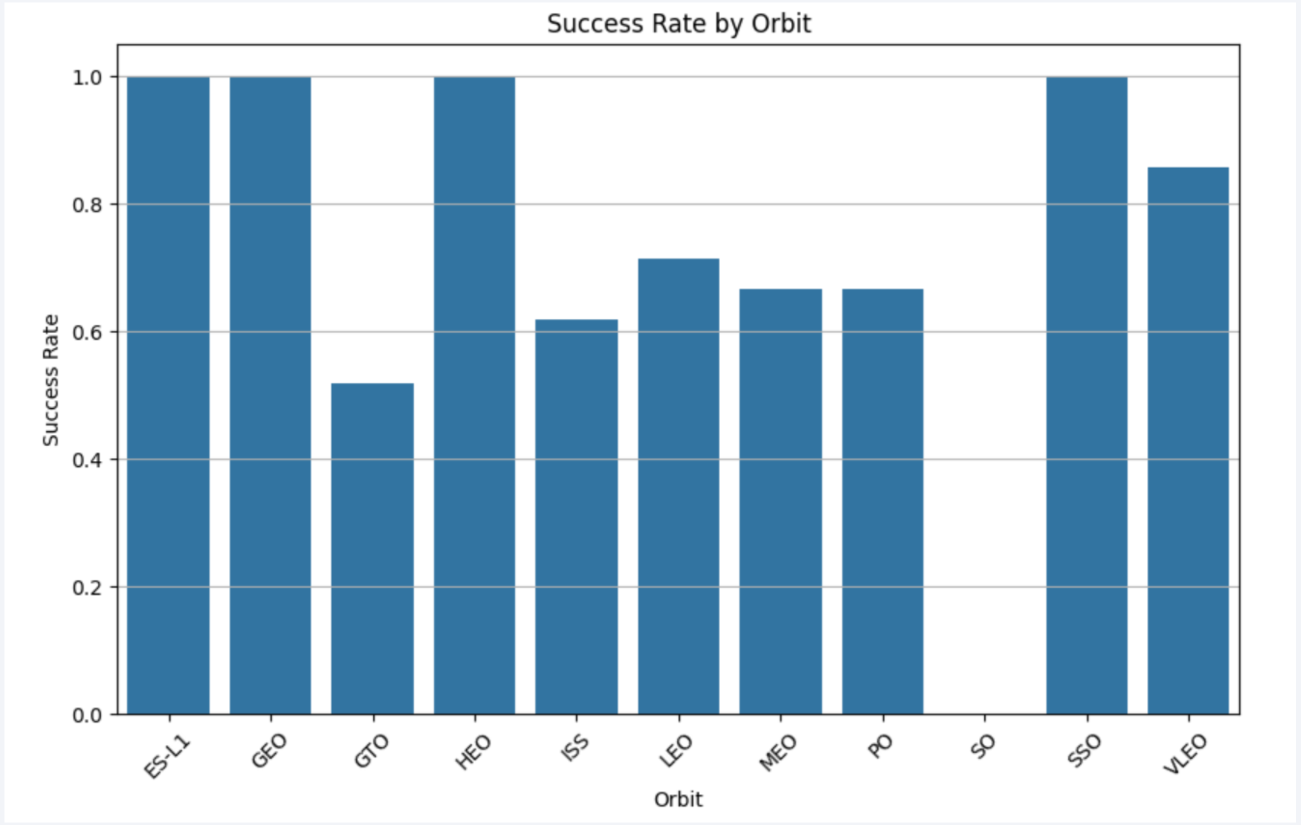
Payload vs. Launch Site

- The plot shows that a payload mass exceeding 7000kg significantly enhances the likelihood of a successful landing.

- However, although a heavier payload appears to improve success rates beyond a certain threshold, it isn't a 100% probability either



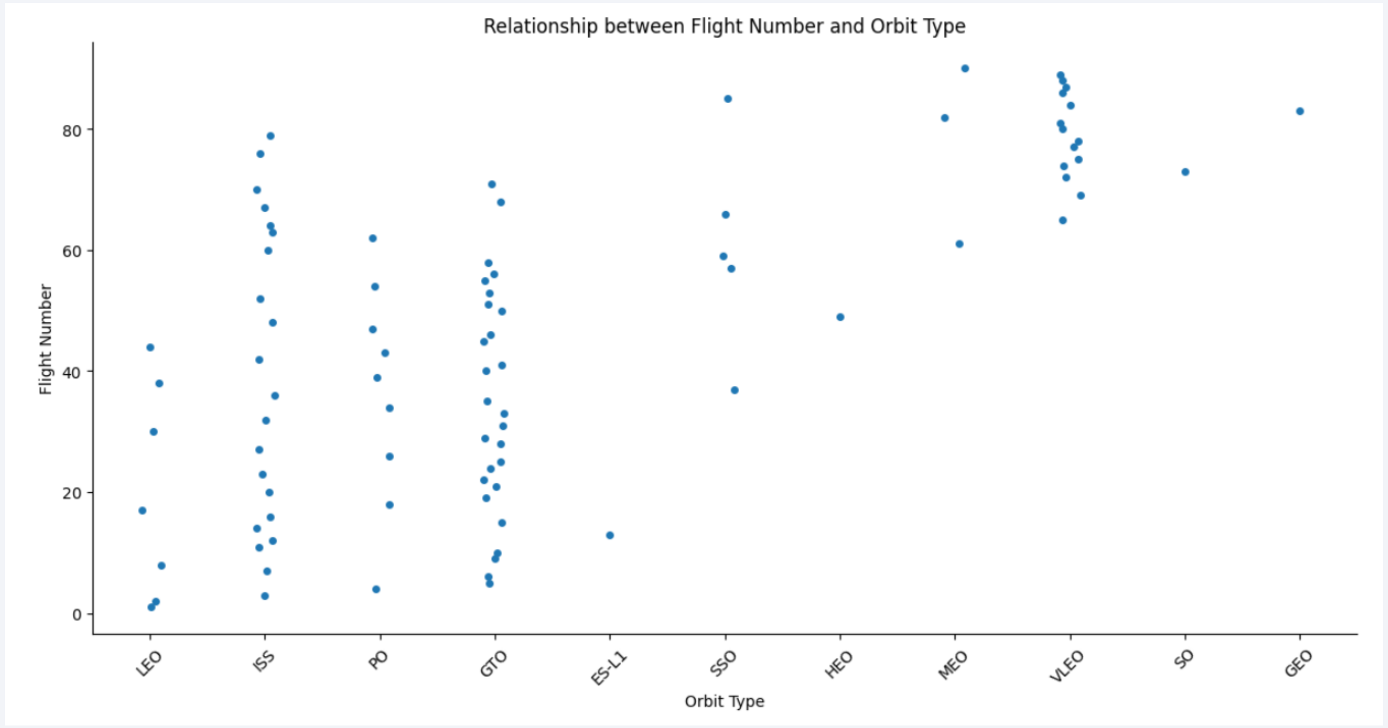
Success Rate vs. Orbit Type



The plot shows that the ES-L1, GEO, HEO, SSO orbits have the highest success rates.

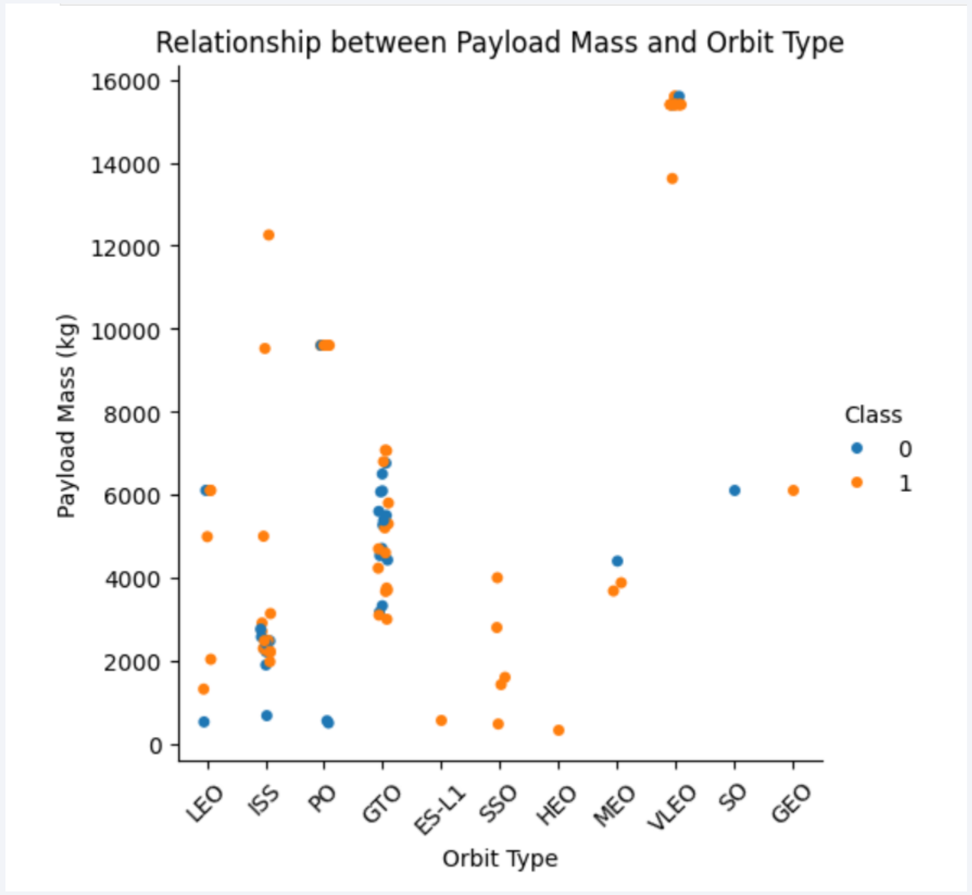
Flight Number vs. Orbit Type

The plot shows that the success rates generally increase with the flight number, suggesting that the knowledge gained from previous launches across different orbits may contribute to higher success rates

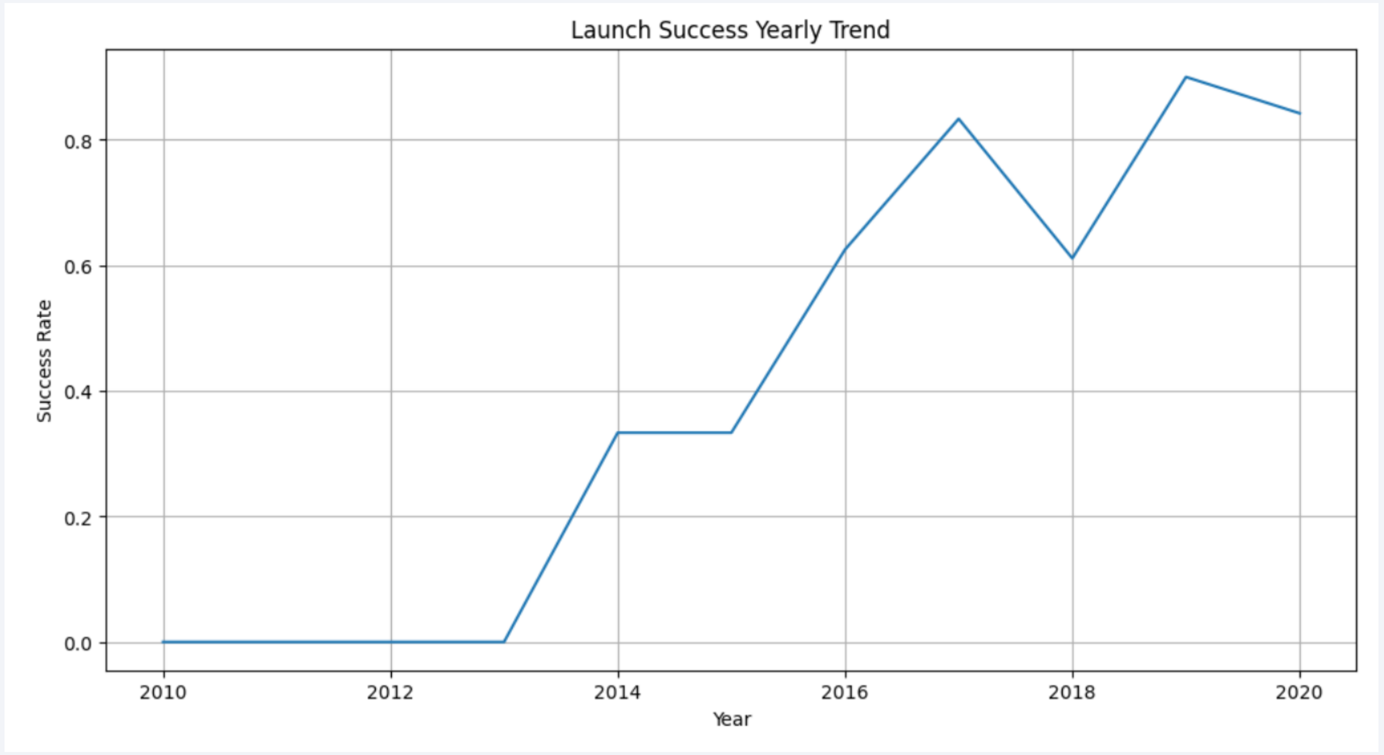


Payload vs. Orbit Type

- The plot shows that payload weight significantly affects launch success rates across different orbits.
- Heavier payloads increase success rates in LEO, ISS, and PO orbits, while adversely affecting MEO and VLEO orbits.



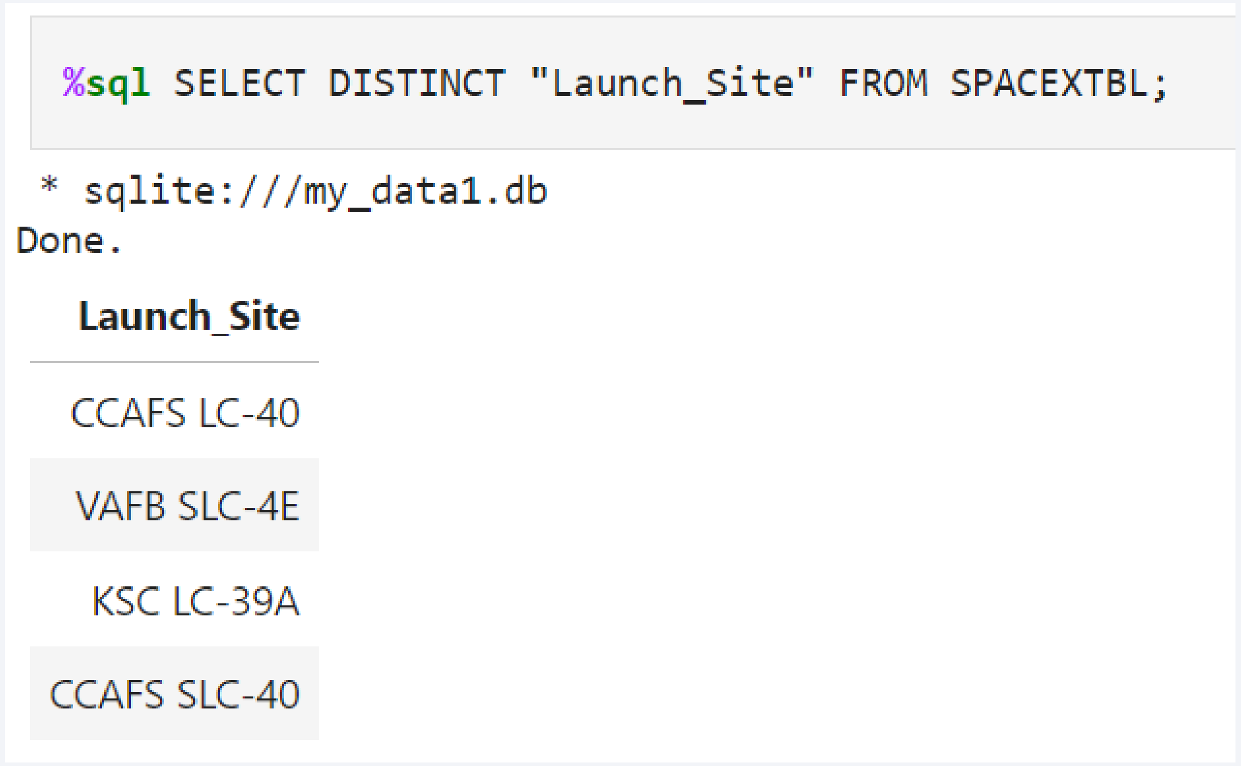
Launch Success Yearly Trend



The plot shows that success rate of launches since 2013 have gradually been increasing with a slight setback in 2018.

All Launch Site Names

DISTINCT was used alongside SELECT to make sure only unique elements were outputted

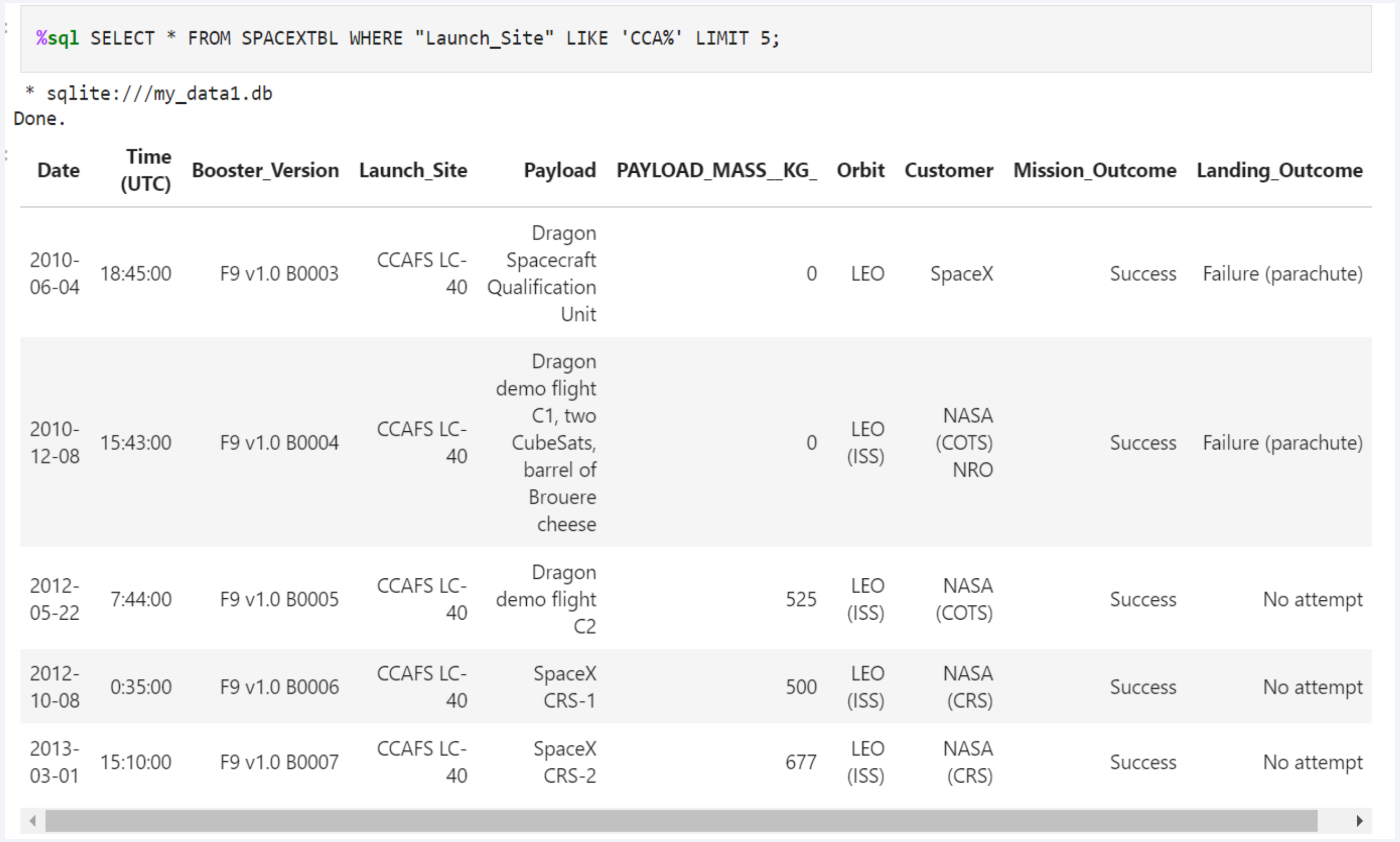


```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTBL;
```

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

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

Launch Site Names Begin with 'CCA'

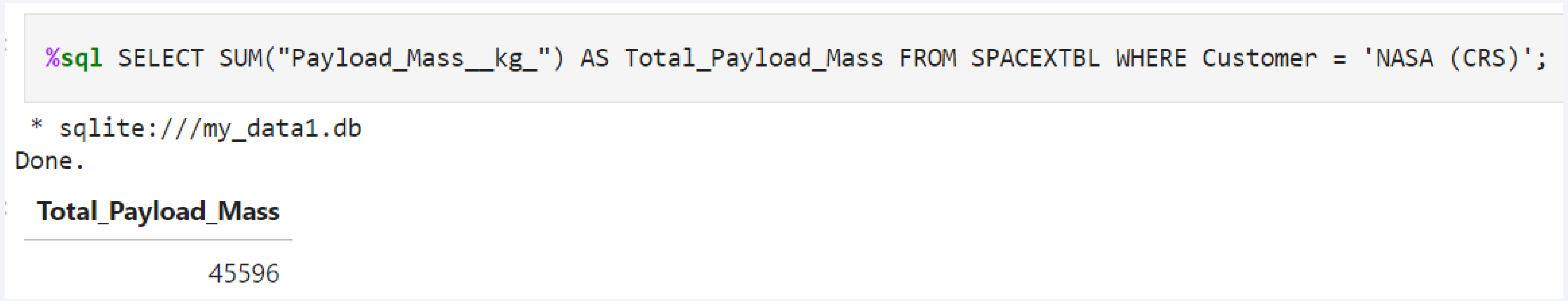


```
%sql SELECT * FROM SPACEXTBL WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;

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

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

WHERE and LIKE were used to set the conditions for the query and LIMIT was used to limited the output to only 5 rows.

Total Payload Mass



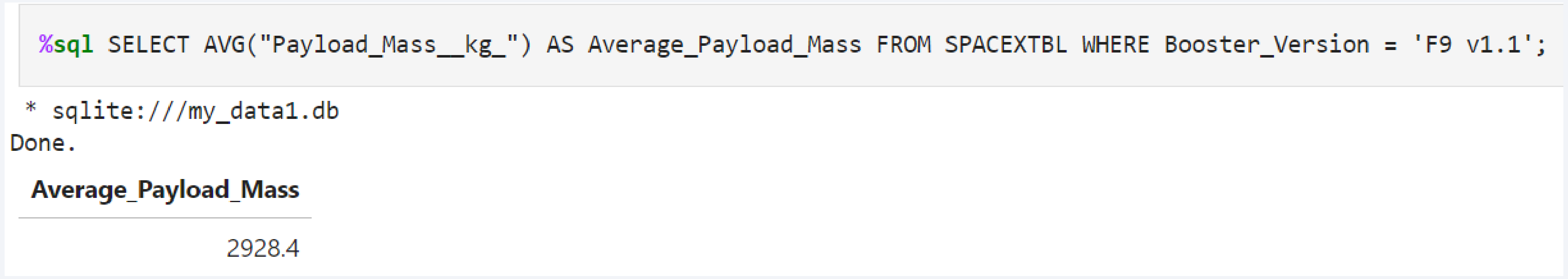
```
%sql SELECT SUM("Payload_Mass__kg_") AS Total_Payload_Mass FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';
```

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

Total_Payload_Mass
45596

The SUM function returns the sum of the column while being limited by the WHERE function.

Average Payload Mass by F9 v1.1



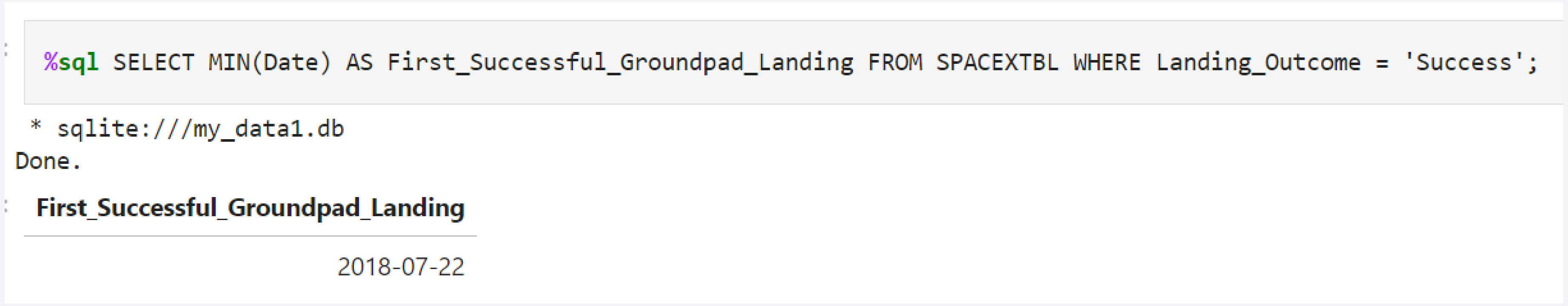
```
%sql SELECT AVG("Payload_Mass__kg_") AS Average_Payload_Mass FROM SPACEXTBL WHERE Booster_Version = 'F9 v1.1';
```

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

Average_Payload_Mass
2928.4

The AVERAGE function returns the average of the column while being limited by the WHERE function.

First Successful Ground Landing Date



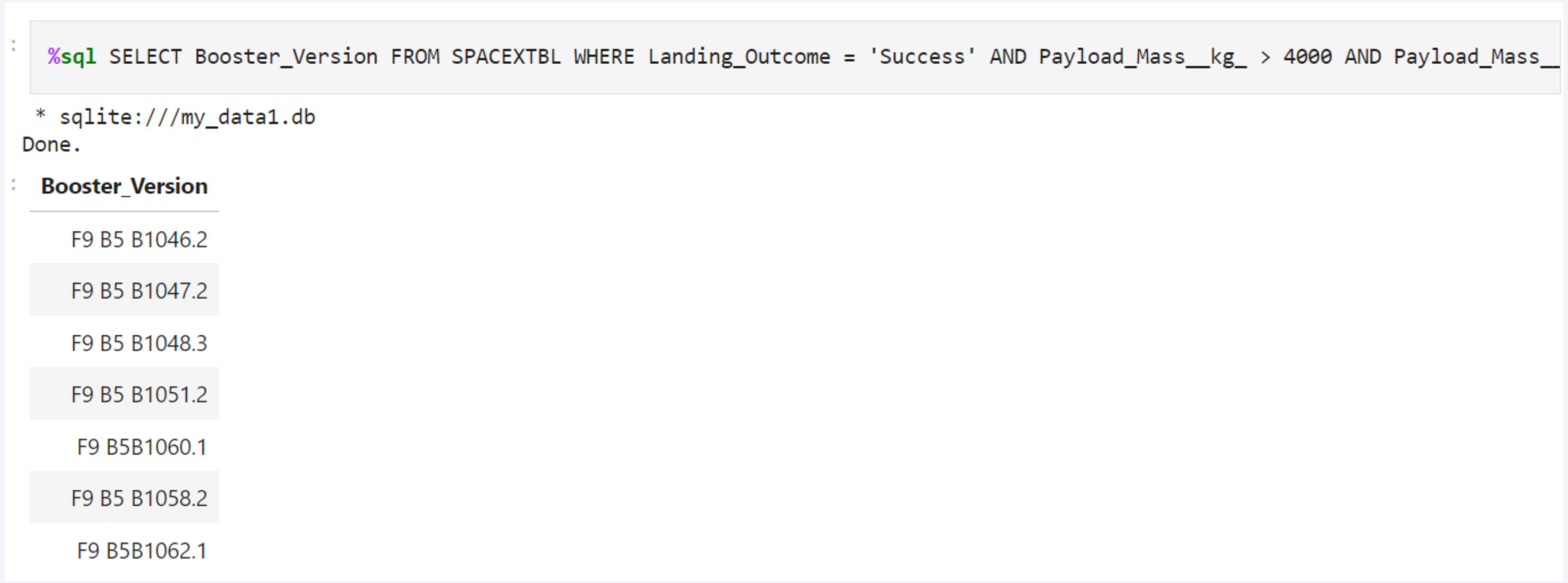
```
%sql SELECT MIN(Date) AS First_Successful_Groundpad_Landing FROM SPACEXTBL WHERE Landing_Outcome = 'Success';
```

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

First_Successful_Groundpad_Landing
2018-07-22

The WHERE function sets the condition for the MIN function used in the query which helps display the earliest date within the column.

Successful Drone Ship Landing with Payload between 4000 and 6000



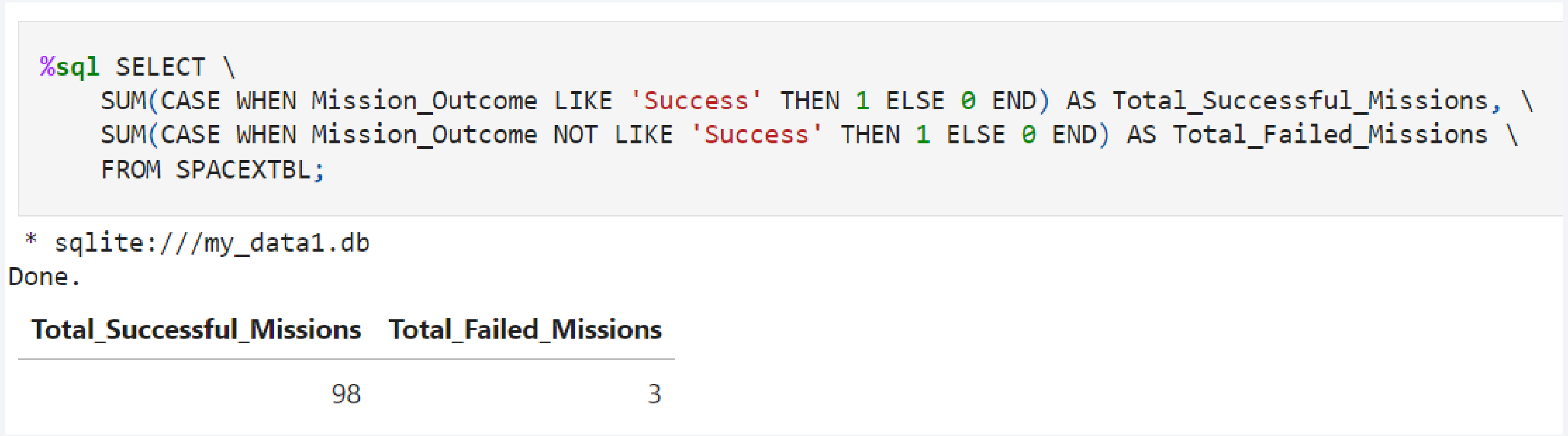
```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome = 'Success' AND Payload_Mass__kg_ > 4000 AND Payload_Mass__
```

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

Booster_Version
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

The WHERE and AND functions help set the conditions for the query to extract the desired information.

Total Number of Successful and Failure Mission Outcomes



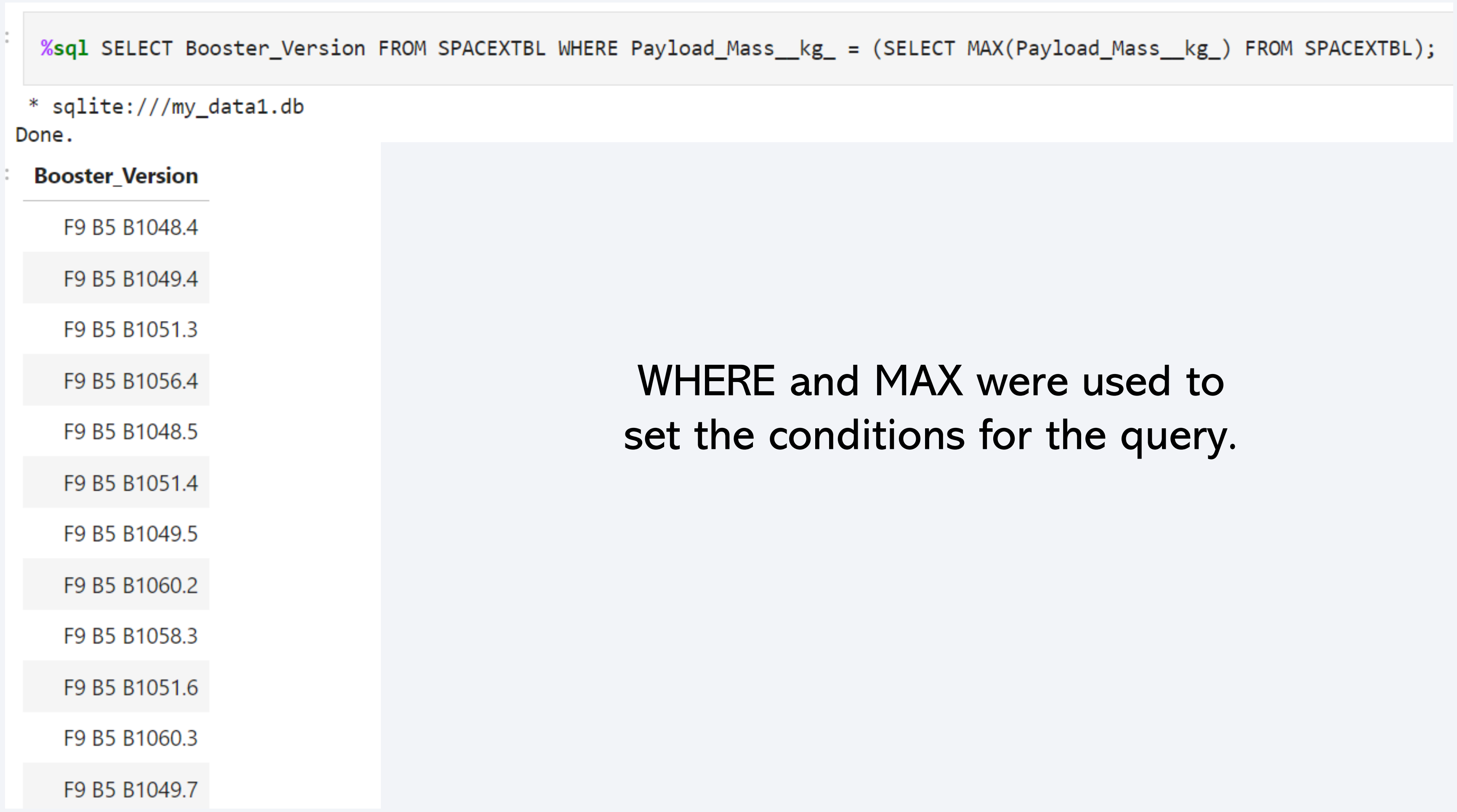
```
%sql SELECT \
    SUM(CASE WHEN Mission_Outcome LIKE 'Success' THEN 1 ELSE 0 END) AS Total_Successful_Missions, \
    SUM(CASE WHEN Mission_Outcome NOT LIKE 'Success' THEN 1 ELSE 0 END) AS Total_Failed_Missions \
    FROM SPACEXTBL;
```

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

Total_Successful_Missions	Total_Failed_Missions
98	3

The query calculates the total number of successful and failed missions from the 'SPACEXTBL' table by checking the 'Mission_Outcome' column for the presence or absence of the word 'Success', assigning a value of 1 or 0 respectively.

Boosters Carried Maximum Payload



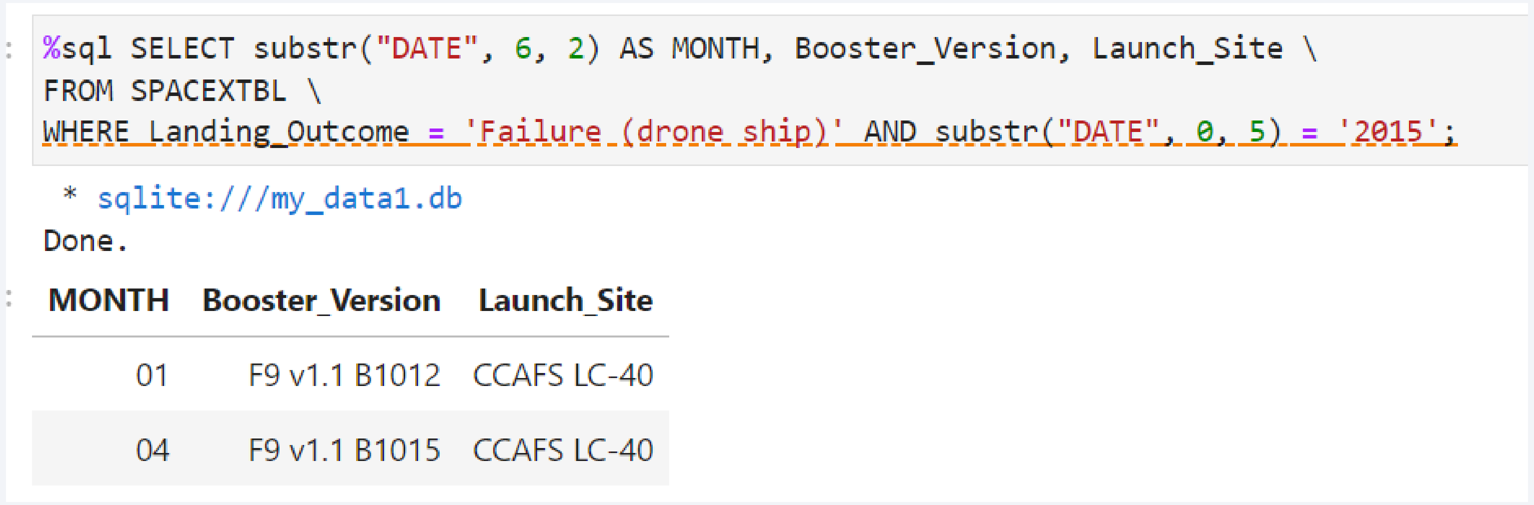
```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE Payload_Mass__kg_ = (SELECT MAX(Payload_Mass__kg_) FROM SPACEXTBL);
```

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

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

WHERE and MAX were used to set the conditions for the query.

2015 Launch Records



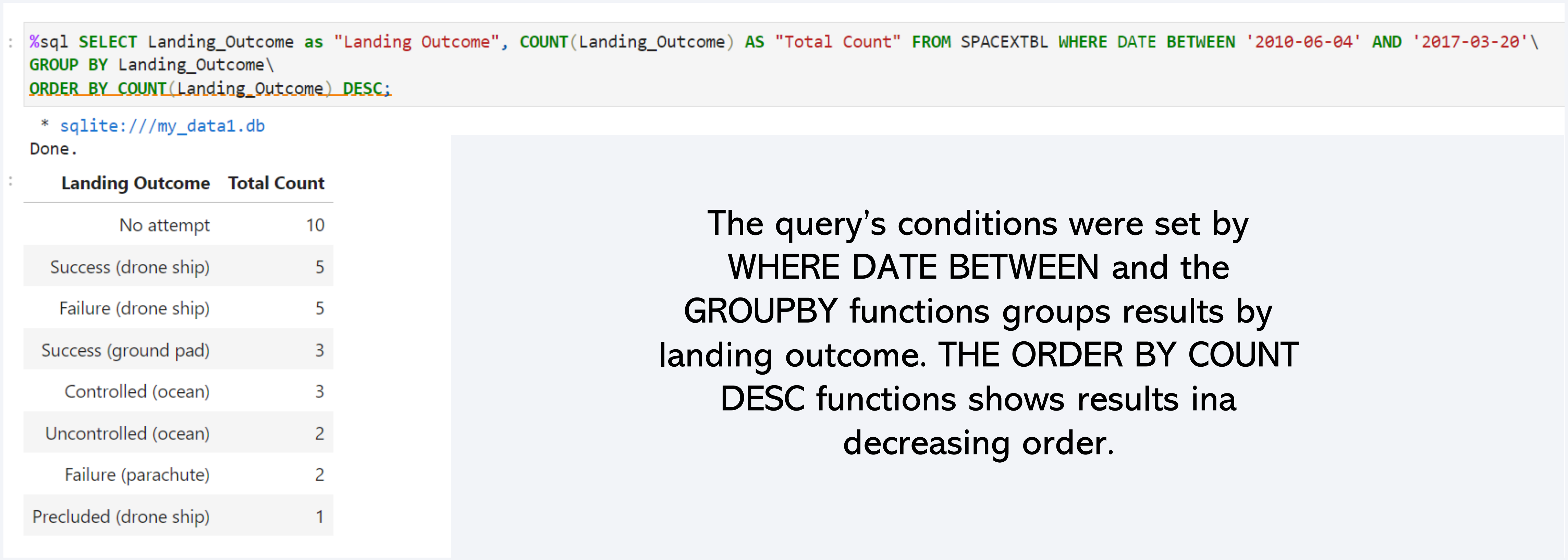
```
%sql SELECT substr("DATE", 6, 2) AS MONTH, Booster_Version, Launch_Site \
FROM SPACEXTBL \
WHERE Landing_Outcome = 'Failure (drone ship)' AND substr("DATE", 0, 5) = '2015';
```

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

MONTH	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

In this query, substr functions process date to take month or year. Substr(DATE, 6, 2) show month while substr(DATE,0, 5) shows year.

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



```
%sql SELECT Landing_Outcome as "Landing Outcome", COUNT(Landing_Outcome) AS "Total Count" FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'\
GROUP BY Landing_Outcome\
ORDER BY COUNT(Landing_Outcome) DESC;
```

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

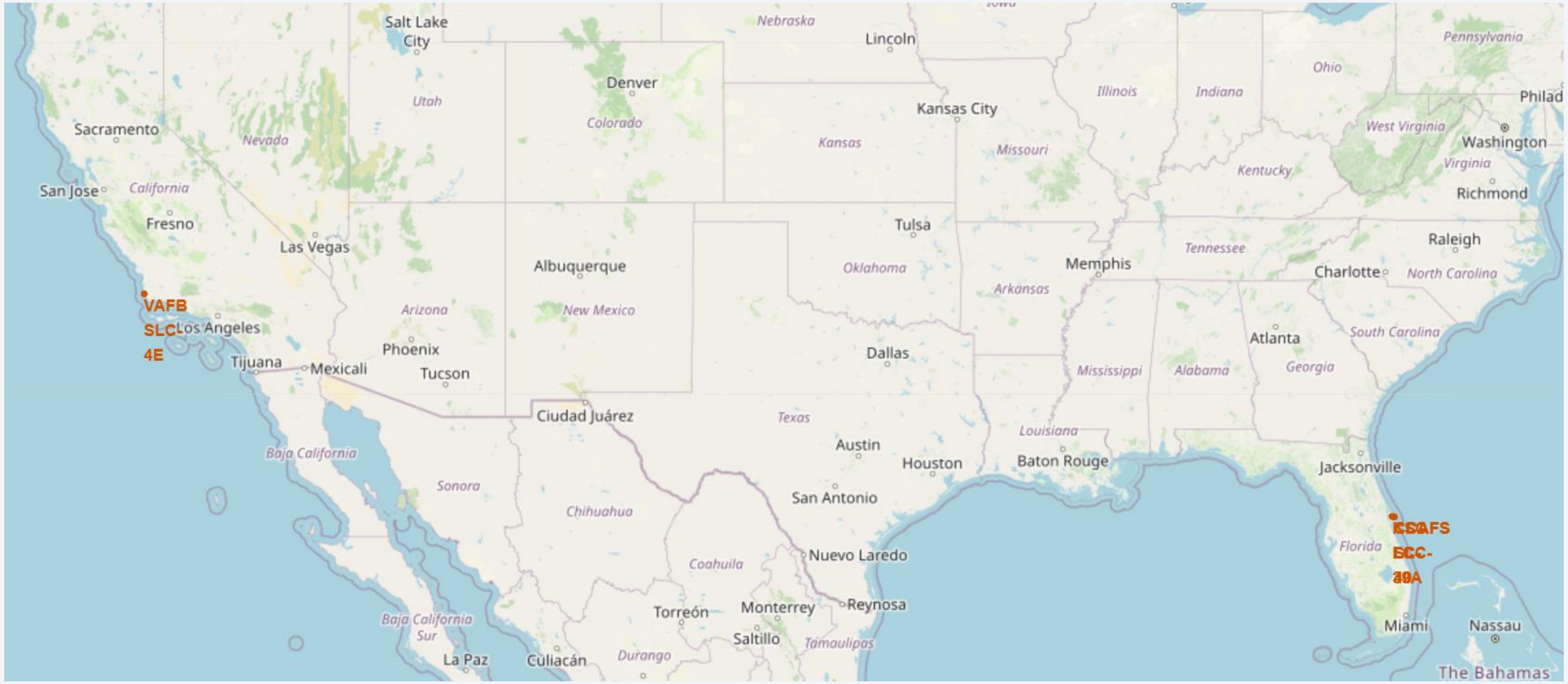
Landing Outcome	Total Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

The query's conditions were set by WHERE DATE BETWEEN and the GROUPBY functions groups results by landing outcome. THE ORDER BY COUNT DESC functions shows results ina decreasing order.

Section 3

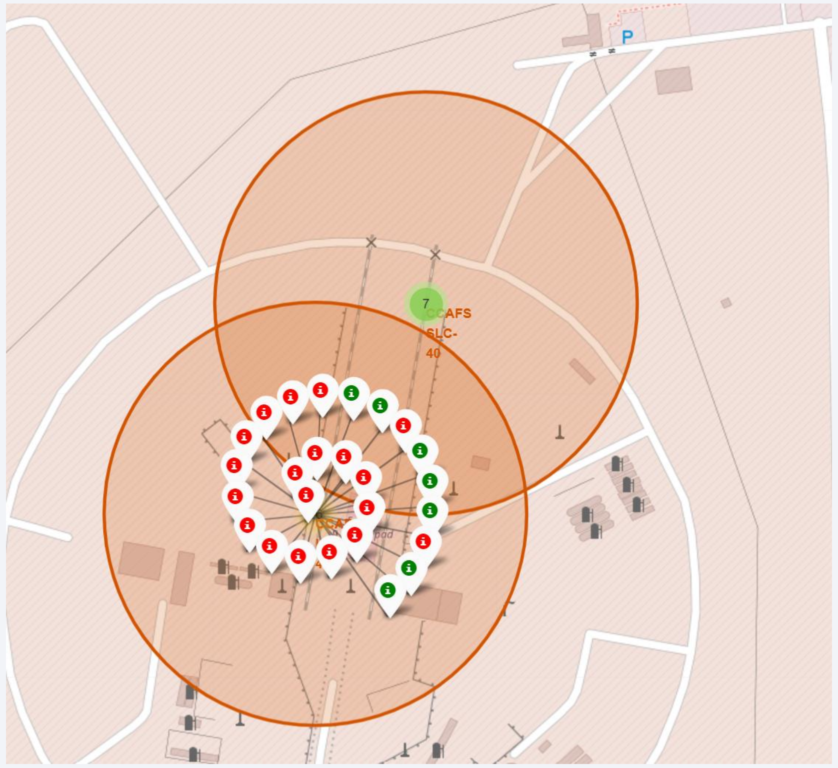
Launch Sites Proximities Analysis

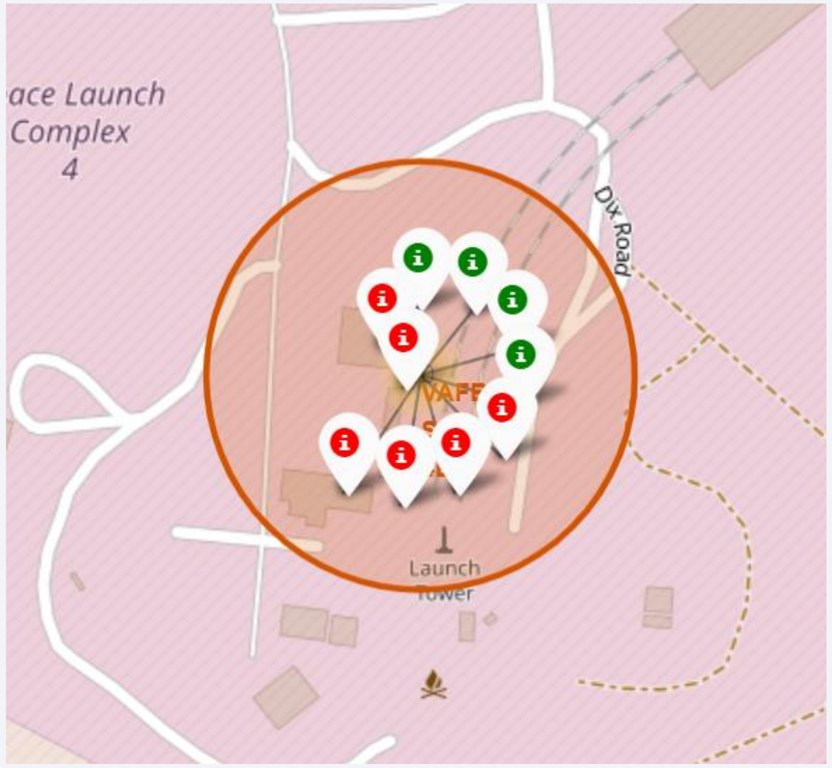
All Launch Site Locations



All SpaceX launch locations are on the coastlines of the USA

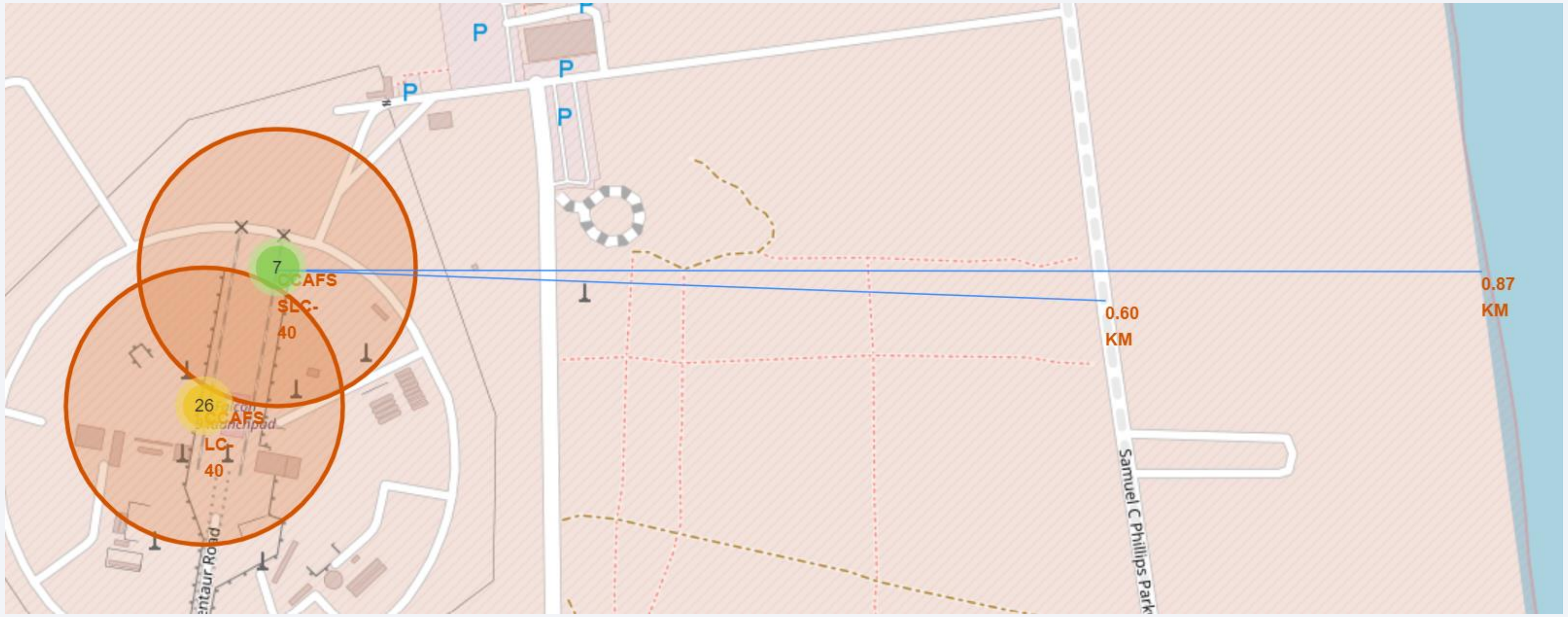
Labeled Launch Sites



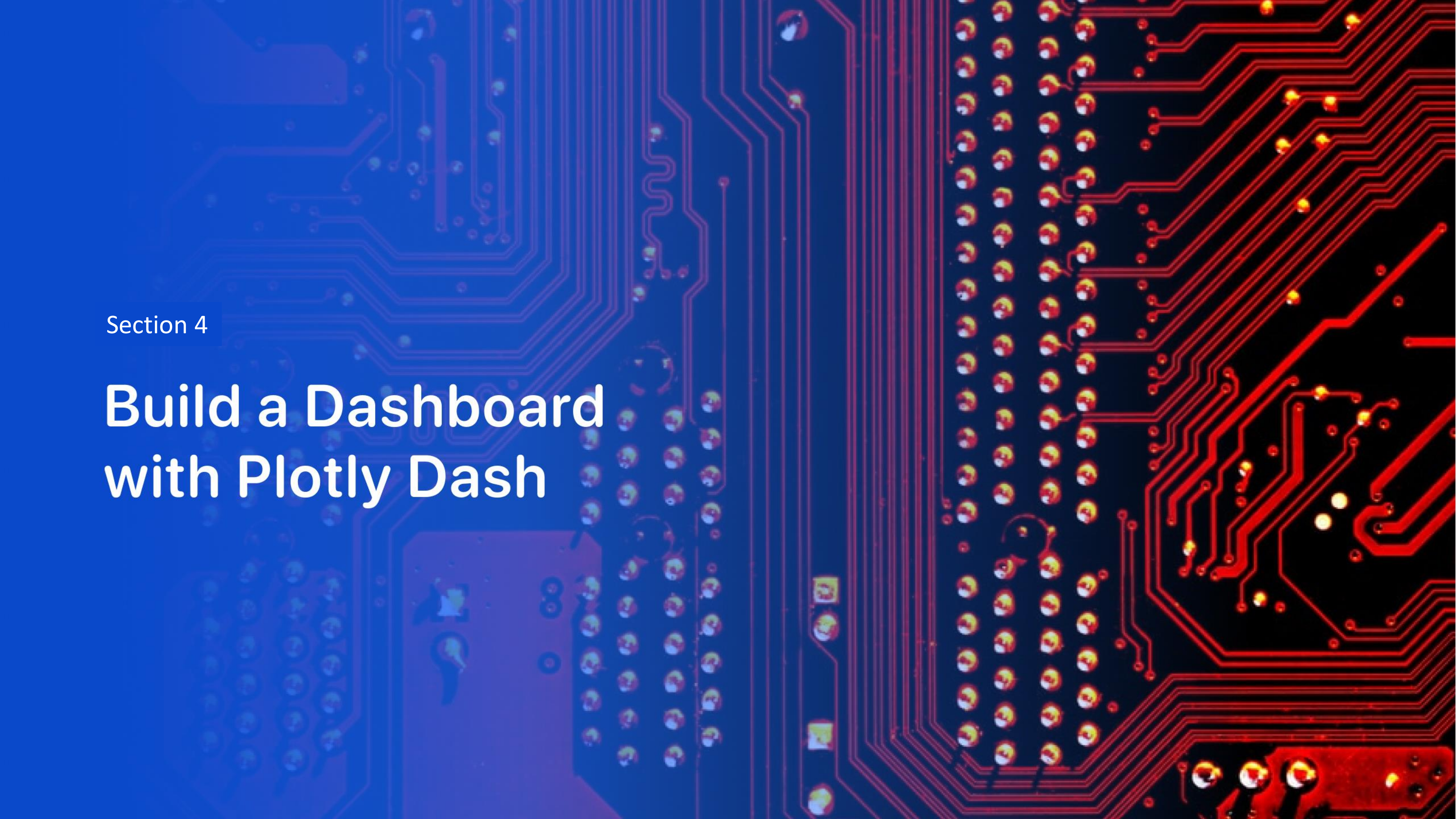


Launches marked green were successful while red markers show failed launches. The left image is taken from the CCAFS SLC-40 launch site while the right is from the VAFS SLC-4E site

Distances from CCAFS SLC-40



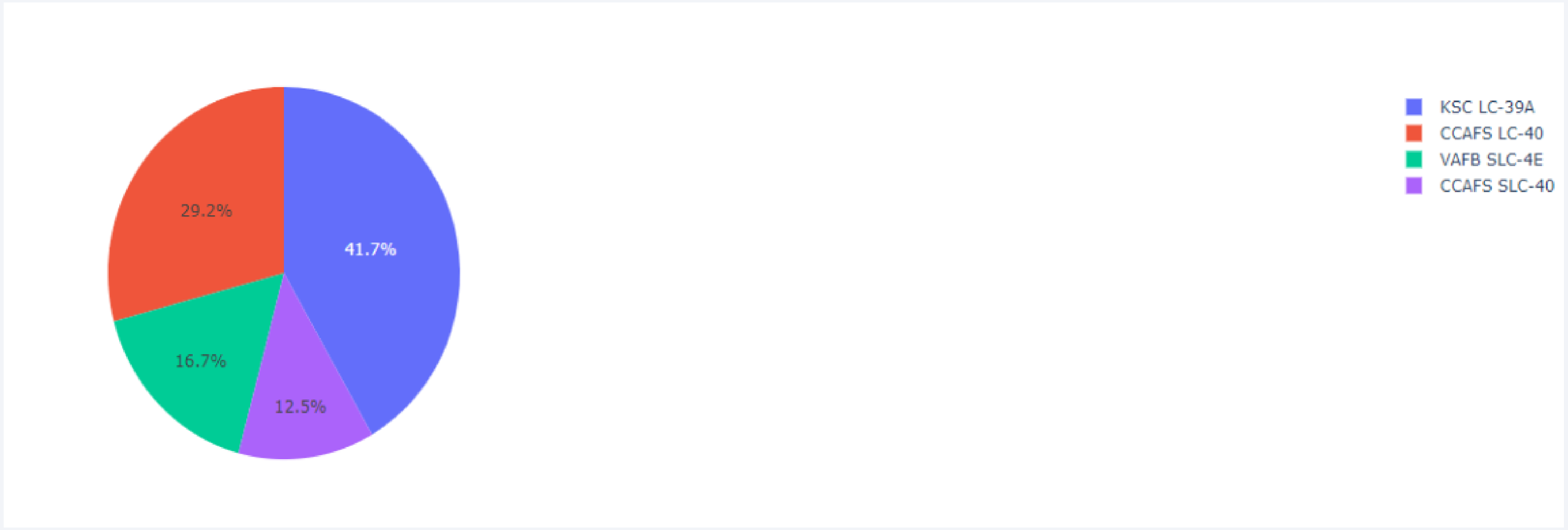
The CCAFS SLC-40 launch is not too far from highways and coastlines



Section 4

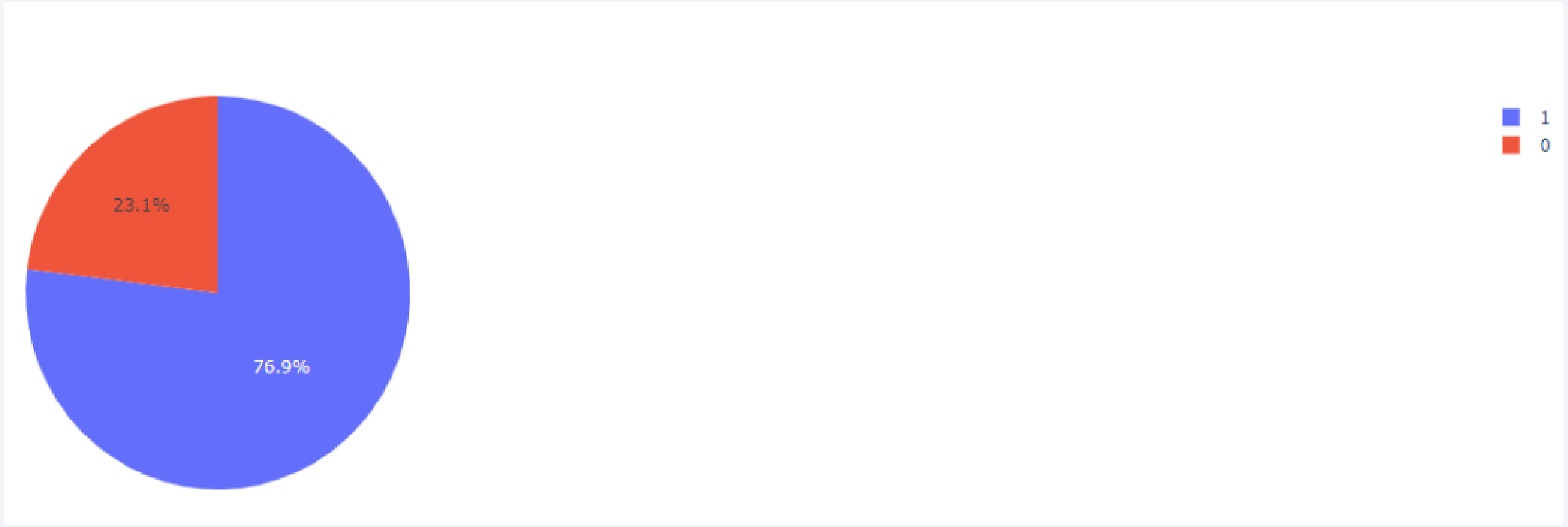
Build a Dashboard with Plotly Dash

Success by Site



KSC LC-39A has the highest overall success rate

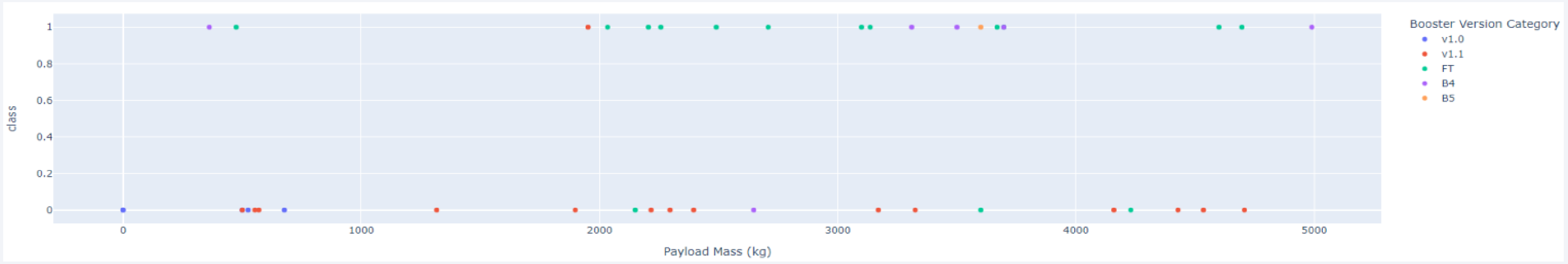
KSC LC 39A Launches



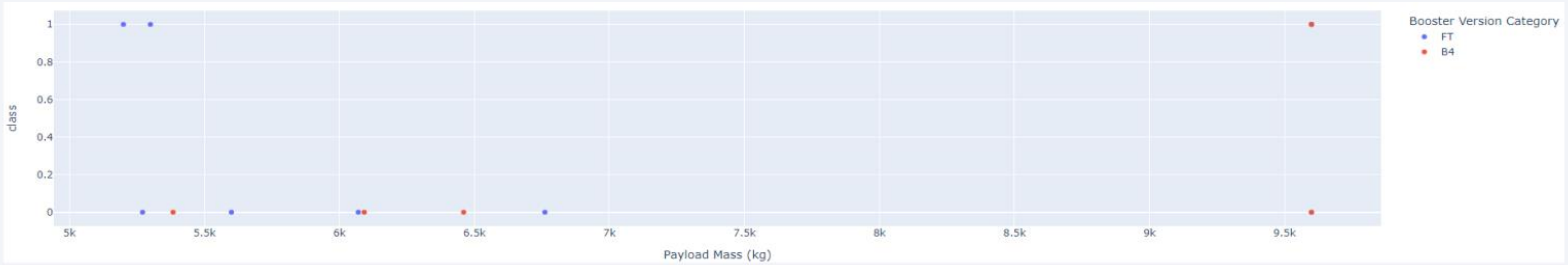
KSC LC-39A has more than a 75% launch success rate

Payload vs Launch Outcome Scatter Plot

Low Weighted Payload



Heavy Weighted Payload



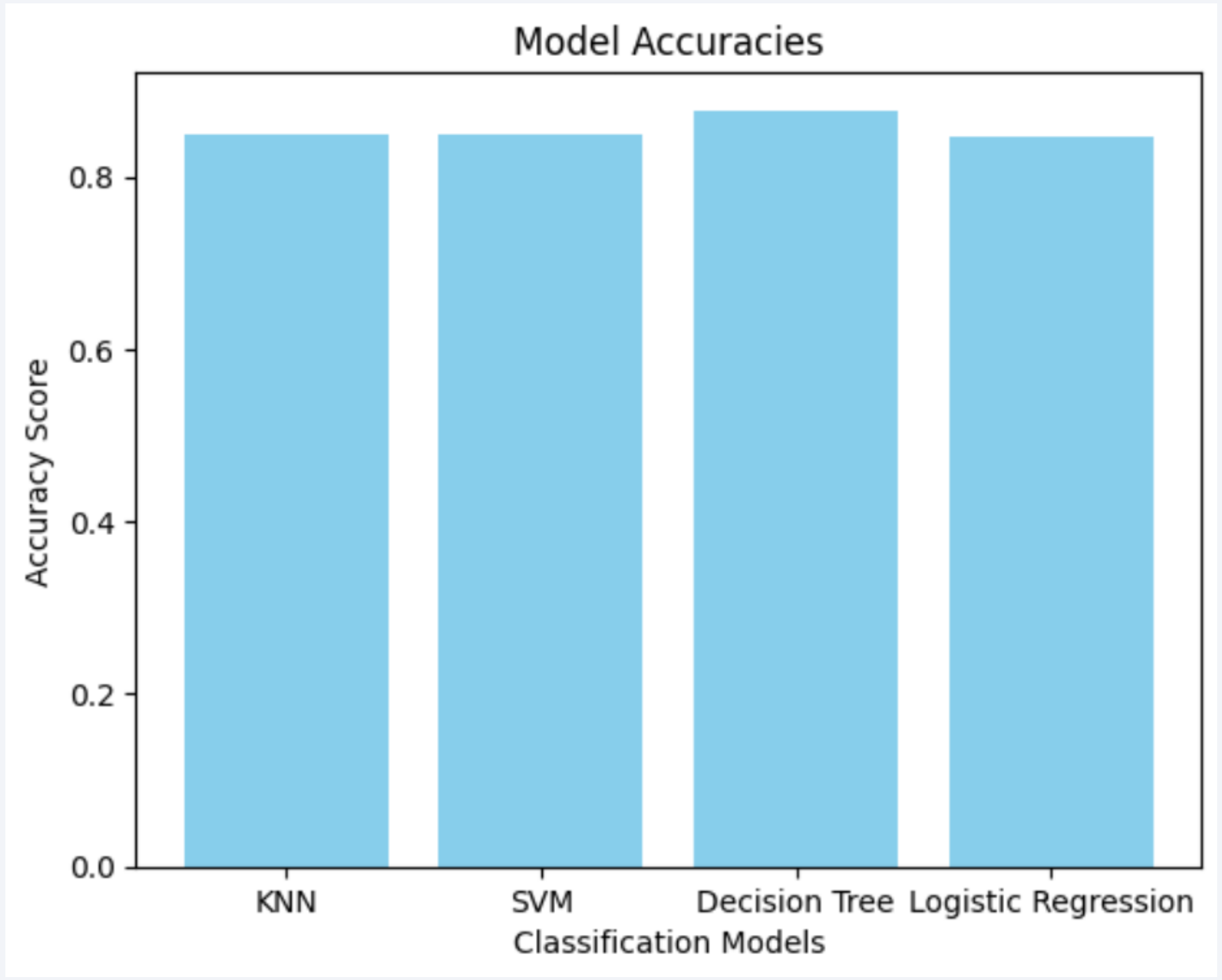
Section 5

Predictive Analysis (Classification)

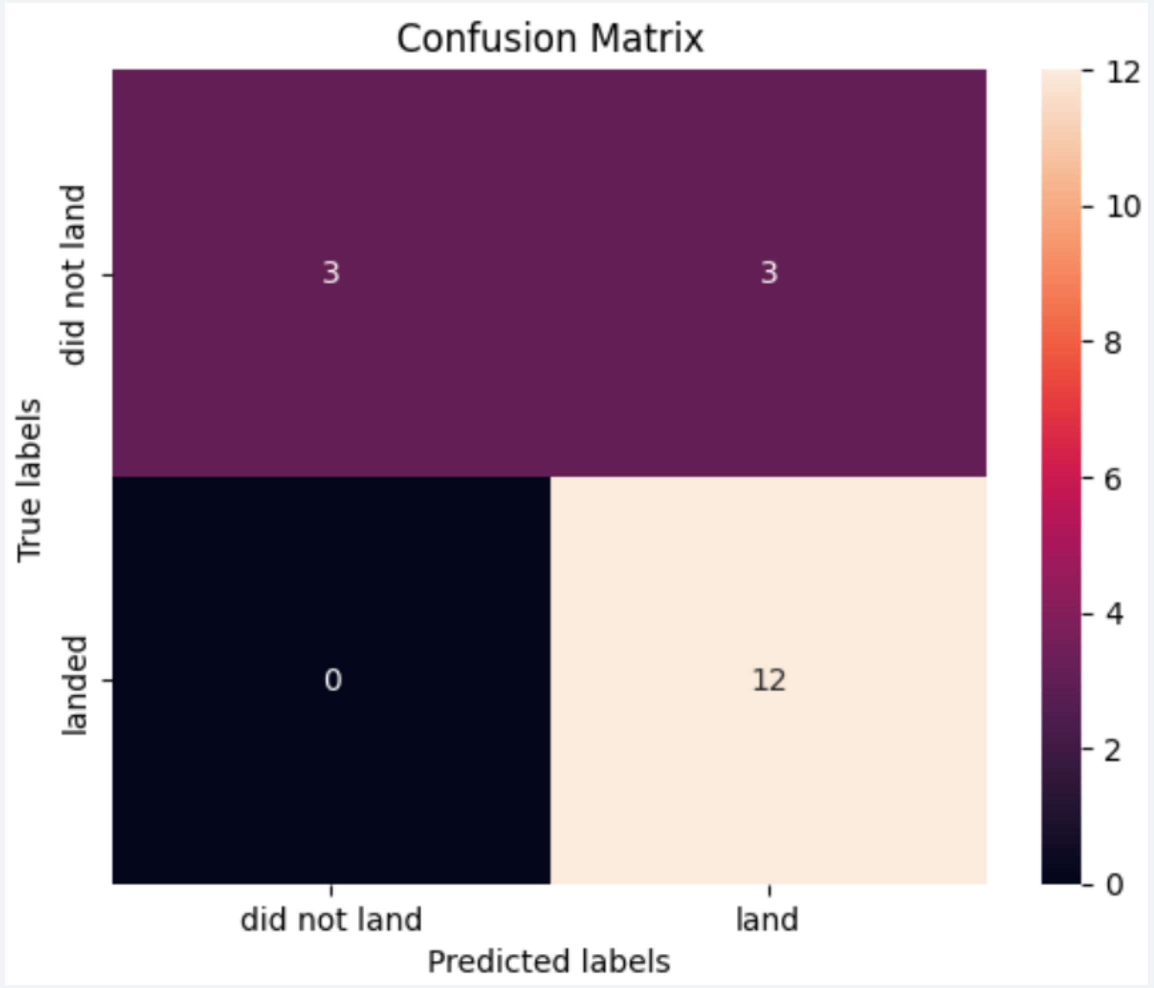
Classification Accuracy

The best classification model is Decision Tree with a score of 0.8767857142857144

Best Parameters: {'criterion': 'gini', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'splitter': 'random'}



Confusion Matrix



The confusion matrix for the decision tree indicates its capability to differentiate between various categories, but the occurrence of false positives, where the classifier incorrectly identifies unsuccessful landings as successful is somewhat of an issue.

Conclusions

- The Decision Tree Classifier emerges as the optimal machine learning method for this dataset.
- Lighter payloads, defined as weighing 4000kg or less, tend to be more successful than heavier ones.
- Since 2013, SpaceX's launch success rate has been on the rise, suggesting a trend towards perfecting launches by 2020.
- KSC LC-39A stands out as the launch site with the most successful launches, with a 76.9% success rate.
- The SSO orbit has a 100% success rate with multiple occurrences, indicating consistent performance.

Thank you!