IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Gabriel Rodríguez
1/23/2024

Outline

Executive Summary

- Summary of methodologies
    - Data Collection API
    - Data Collection API with Webscraping
    - Data Wrangling
    - Exploratory Data Analysis Using SQL
    - Exploratory Data Analysis for Data Visualization
    - Interactive Visual Analytics with Folium lab
    - Build an Interactive Dashboard with Ploty Dash
    - Machine Learning predictive Analysis

Introduction

- This project delves into the dynamic landscape of commercial space travel, with a particular focus on SpaceX's Falcon 9 rocket, SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. The project's core challenge involves predicting the reusability of the first stage of Falcon 9 launches using machine learning models, transcending traditional rocket science methods. Through the analysis of public information

- Key problems need to be addressed:
  - Pricing Strategy
  - First Stage Reusability Prediction
  - Data Gathering

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - Data was collected using SPACEX REST API and web scrapping from Wikipedia
- Perform data wrangling
    - The data was processed reviewing attributes, categorizing Launch Site and Orbit and classifying outcome
    - Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - How to build, tune, evaluate classification models

Data Collection

- The data for this capstone assignment was collected using the SpaceX REST API. The API, specifically the **SpaceX REST API**, provides comprehensive information about SpaceX launches, including details about rockets, payloads, launch specifications, landing specifications, and landing outcomes. Also, the data was retrieved through a GET request using the requests library, and the response is in the form of **JSON**, which consists of a list of JSON objects representing each launch.

- Additionally, we used an alternative data source: web scraping Wiki pages related to Falcon 9 launches using the Python **BeautifulSoup** package.

Data Collection – SpaceX API

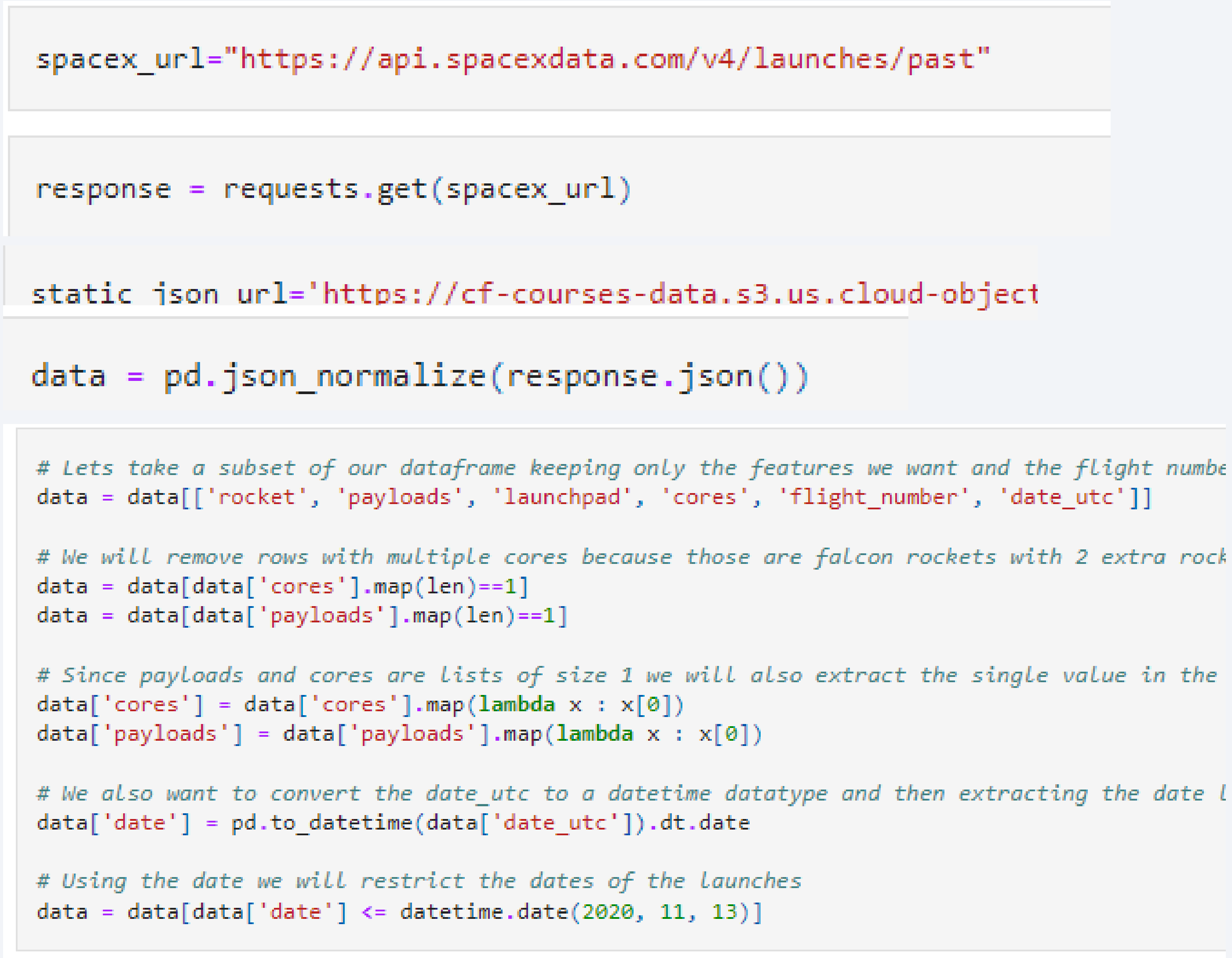
Use of **GET request** to retrieve data about SpaceX launches

Json_normalize function is employed for the normalization of the JSON data into a flat table

Data cleaning

GitHub

https://github.com/Gabrodul/Applied-IBM-Data-Science-Capstone/blob/9b9ef63dd2f8ea8a91c95f7f36982622b5d825a4/1_jupyter-labs-spacex-data-collection-api.ipynb



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

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

```
static_json_url='https://cf-courses-data.s3.us.cloud-object
```

```
data = pd.json_normalize(response.json())
```

```
# Lets take a subset of our dataframe keeping only the features we want and the flight numbe
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 rock
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
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 l
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)]
```

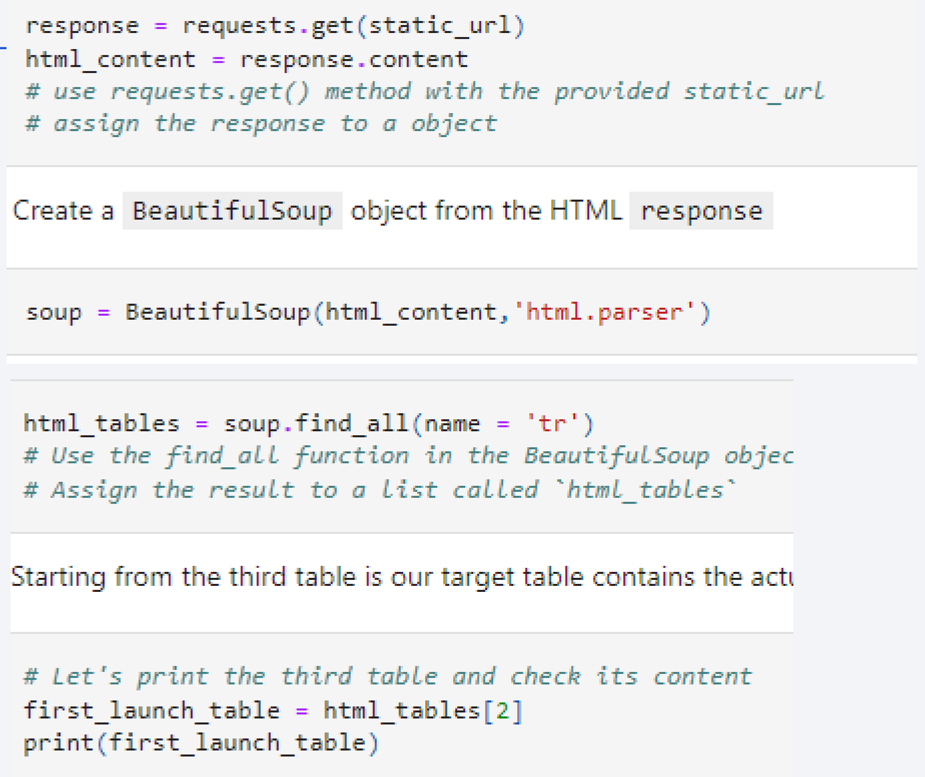
Data Collection - Scraping

Performed an HTTP **GET** method to request the Falcon9 Launch HTML page, as an HTTP response.

Created a **BeautifulSoup object** from the HTML response

Used the **find_all** function in the **BeautifulSoup** object, with element type **`tr`** and targeting the third table

Finally, Iterated through the **<th>** elements and apply the **extract_column_from_header()** to extract column name one by one



```
response = requests.get(static_url)
html_content = response.content
# use requests.get() method with the provided static_url
# assign the response to a object
```

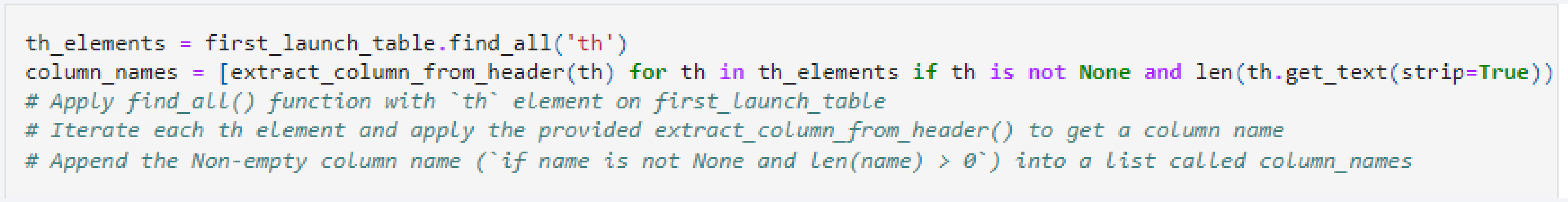
Create a `BeautifulSoup` object from the HTML `response`

```
soup = BeautifulSoup(html_content,'html.parser')
```

```
html_tables = soup.find_all(name = 'tr')
# Use the find_all function in the BeautifulSoup objec
# Assign the result to a list called `html_tables`
```

Starting from the third table is our target table contains the act

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```



```
th_elements = first_launch_table.find_all('th')
column_names = [extract_column_from_header(th) for th in th_elements if th is not None and len(th.get_text(strip=True))
# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
```

GitHub

- https://github.com/Gabrodul/Applied-IBM-Data-Science-Capstone/blob/9b9ef63dd2f8ea8a91c95f7f36982622b5d825a4/2_jupyter-labs-webscraping.ipynb

Data Wrangling

- Data Wrangling had a focus on reviewing key attributes associated with SpaceX launches. These attributes include Flight Number, Date, Booster version, Payload mass, Orbit, Launch Site, etc. The goal was to convert landing outcomes into classes, where 0 signifies a failed landing and 1 indicates a successful landing, streamlining the classification variable for subsequent modeling. This, involves the following steps:

  - **Attribute Identification**
  - **Categorization and Explanation**
  - **Outcome Classification**
  - **Landing Outcome Conversion**

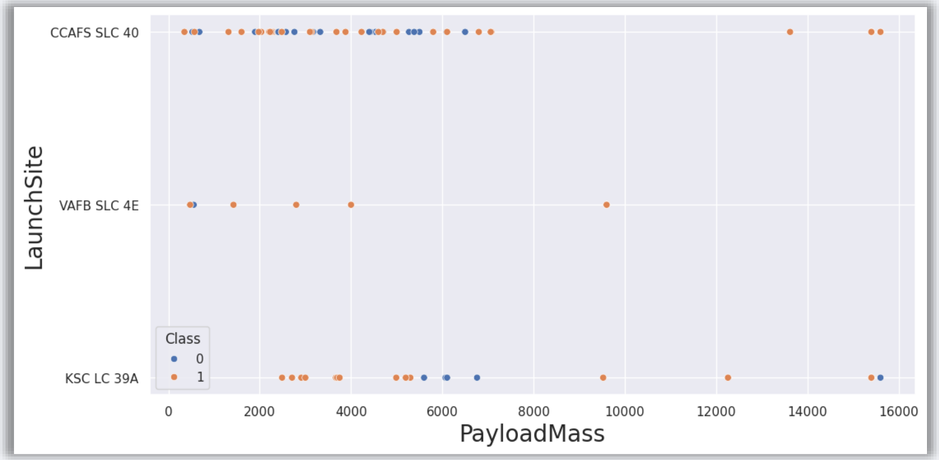
GitHub

https://github.com/Gabrodul/Applied-IBM-Data-Science-Capstone/blob/9b9ef63dd2f8ea8a91c95f7f36982622b5d825a4/3_labs-jupyter-spacex-Data%20wrangling.ipynb

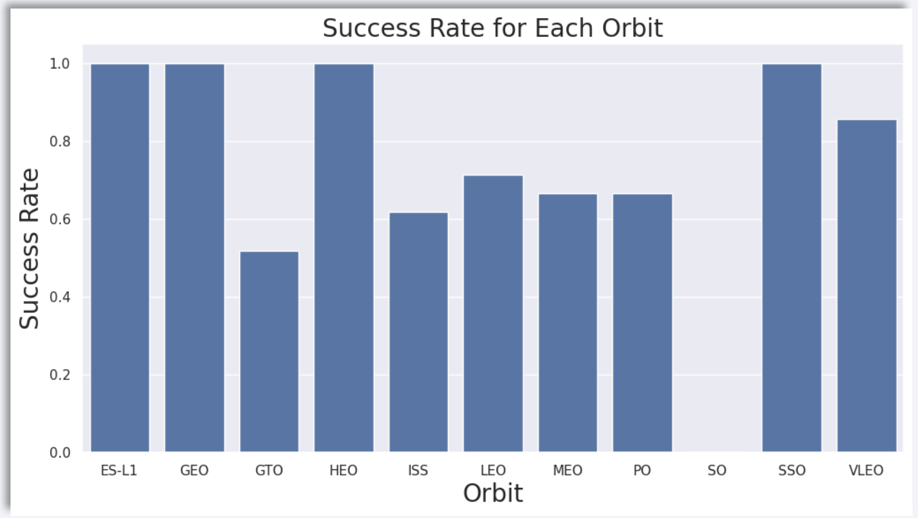
EDA with Data Visualization

In this phase, we used different kinds of plot to determine relationship and dependency between various attributes:

- **LaunchSite vs PayloadMass:**



- **Success Rate and Orbit:**



GitHub

https://github.com/Gabrodul/Applied-IBM-Data-Science-Capstone/blob/9b9ef63dd2f8ea8a91c95f7f36982622b5d825a4/5_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

GitHub https://github.com/Gabrodul/Applied-IBM-Data-Science-Capstone/blob/9b9ef63dd2f8ea8a91c95f7f36982622b5d825a4/4_jupyter-labs-eda-sql-coursera_sqllite.ipynb

- Using SQL we performed covering various aspects of the dataset, including:
- Displaying unique launch sites
- Filtering records based on launch site names
- Calculating the total payload mass for NASA (CRS) launches
- Determining the average payload mass for booster version F9 v1.1
- Finding the date of the first successful ground pad landing
- Listing boosters with success in drone ship and specific payload mass range
- Summarizing the total number of successful and failure mission outcomes
- Identifying booster versions with the maximum payload mass
- Displaying records for specific months and landing outcomes in the year 2015
- Ranking the count of landing outcomes in descending order within a specified date range.

Build an Interactive Map with Folium

- Using Folium, an interactive map library, we performed an analysis of launch site locations. The objectives of the lab included marking all launch sites on a map, marking the success/failed launches for each site on the map, and calculating the distances between a launch site and its proximities.
- We enhanced the map by adding markers for the success/failed launches. Successful launches were **marked in green**, and **failed launches were marked in red**.
- **MarkerCluster** object is utilized to handle multiple markers at the same coordinates efficiently.
- Finally, we calculated and visualized the distances between launch sites and selected points of interest such as coastlines, railways, highways, and cities.

GitHub

https://github.com/Gabrodul/Applied-IBM-Data-Science-Capstone/blob/9b9ef63dd2f8ea8a91c95f7f36982622b5d825a4/6_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

We created a dashboard to visualize SpaceX launch records. Here's a summary of the plots/graphs and interactions added to the dashboard:

- **Launch Site Dropdown:** Allows users to filter the data based on the selected launch site.
- **Success Pie Chart:** Provides an overall view of success rates for different launch sites.
- **Payload Range Slider:** Allows users to filter the data based on the specified payload mass range.
- **Success Payload Scatter Chart:** Provides insights into how payload mass and booster version correlate with launch success.

Interactions:

- **The Launch Site Dropdown** interacts with both the **Success Pie Chart** and **the Success Payload Scatter Chart**, dynamically updating the visualizations based on the selected launch site.

- **The Payload Range Slider** interacts with **the Success Payload Scatter Chart**, enabling users to filter launches based on payload mass.

GitHub

https://github.com/Gabrodul/Applied-IBM-Data-Science-Capstone/blob/9b9ef63dd2f8ea8a91c95f7f36982622b5d825a4/7_spacex_dash_app.py

Predictive Analysis (Classification)

GitHub https://github.com/Gabrodul/Applied-IBM-Data-Science-Capstone/blob/9b9ef63dd2f8ea8a91c95f7f36982622b5d825a4/8_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Built:

- The data preparation involved loading information from two CSV files to create a comprehensive dataset for analysis.
- Feature engineering included creating a binary target variable
- Standardization was performed on the feature set, and the data was split into training and testing sets for subsequent model evaluation.

Evaluation

- The Logistic Regression model was assessed through hyperparameter tuning with GridSearchCV.
- The model's performance was further analyzed on the test set, displaying accuracy and a confusion matrix.
- Similar steps were taken for the Support Vector Machine (SVM) model.

Improved

- A Decision Tree model was introduced and optimized using GridSearchCV for hyperparameter tuning.
- The best parameters and accuracy on the validation set were highlighted, followed by an assessment of its performance on the test set, including the presentation of a confusion matrix.

Best Performing Model

- The K Nearest Neighbors (KNN) model was implemented and fine-tuned using GridSearchCV.
- Ultimately, the Decision Tree model emerged as the best-performing classification model.

Results

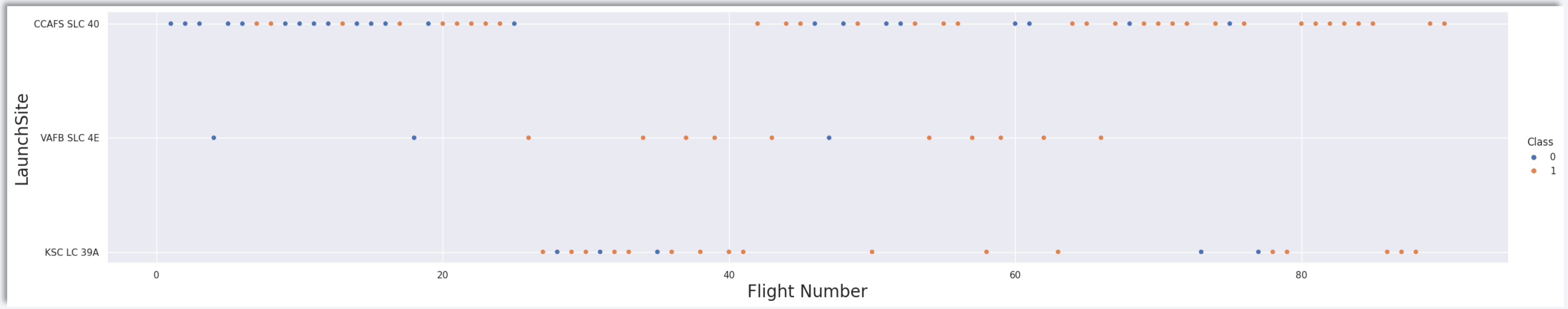
The results will be divided in three main categories:

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

Section 2

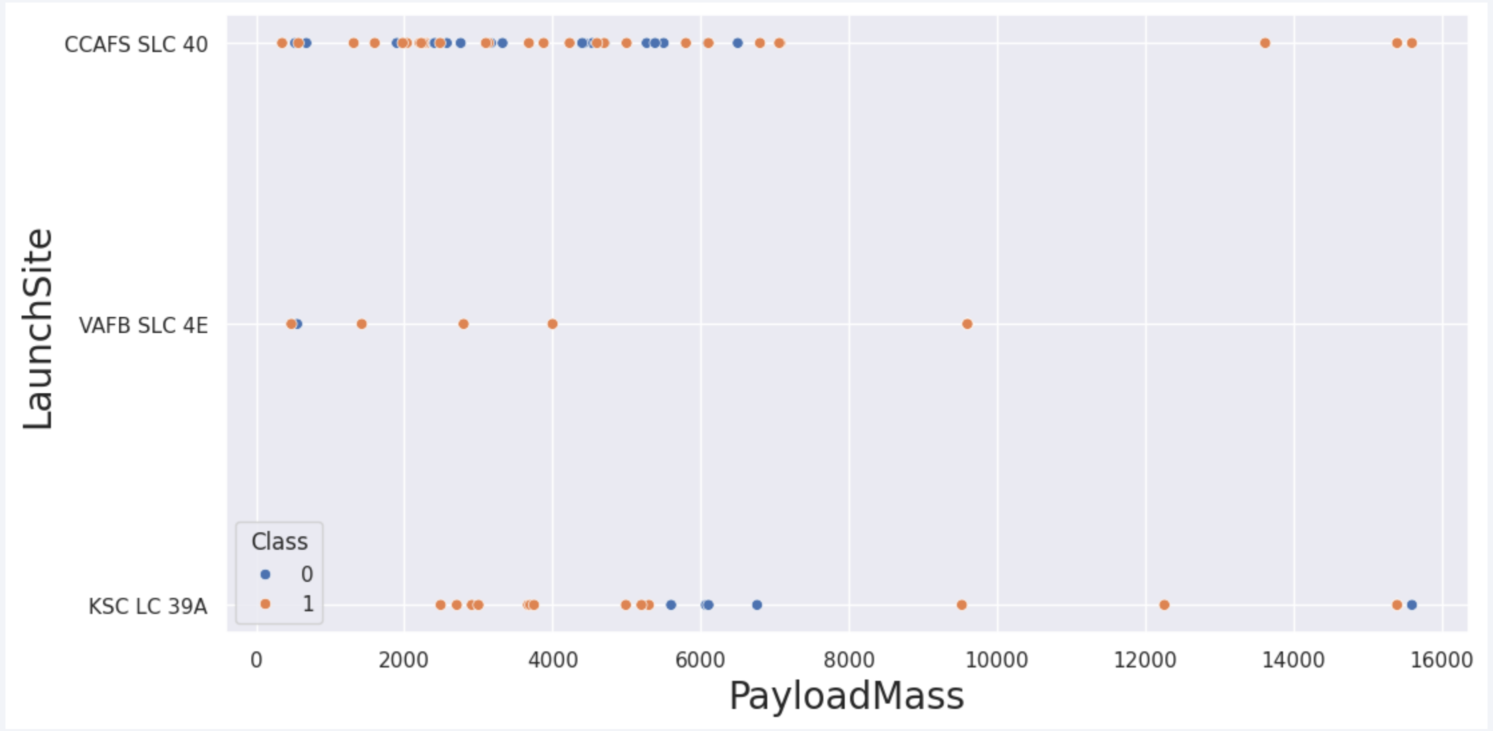
Insights drawn from EDA

Flight Number vs. Launch Site



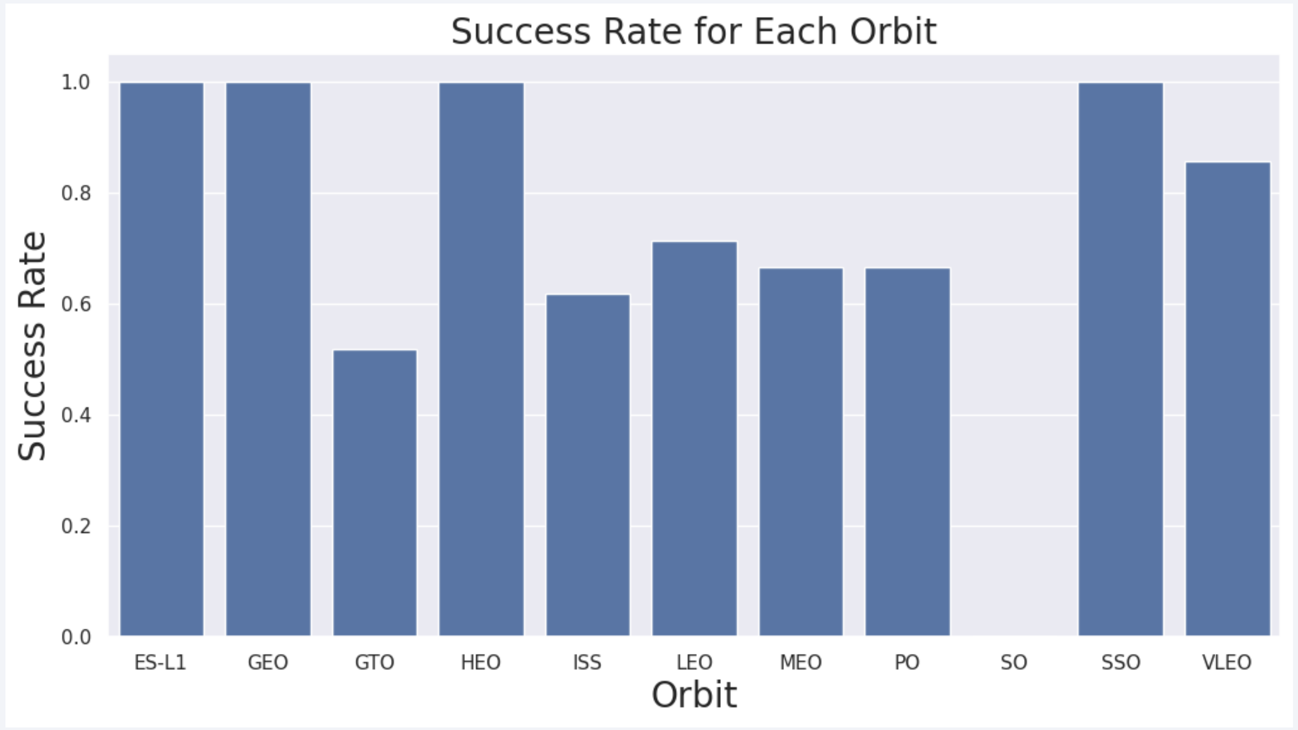
- This catterplot shows a dependency between the flight number and the launch site success rate, the larger amount of flights greater the success rate will be.
- Although, CCAFS SLC 40 doesnt show a real pattern for this variables.

Payload vs. Launch Site



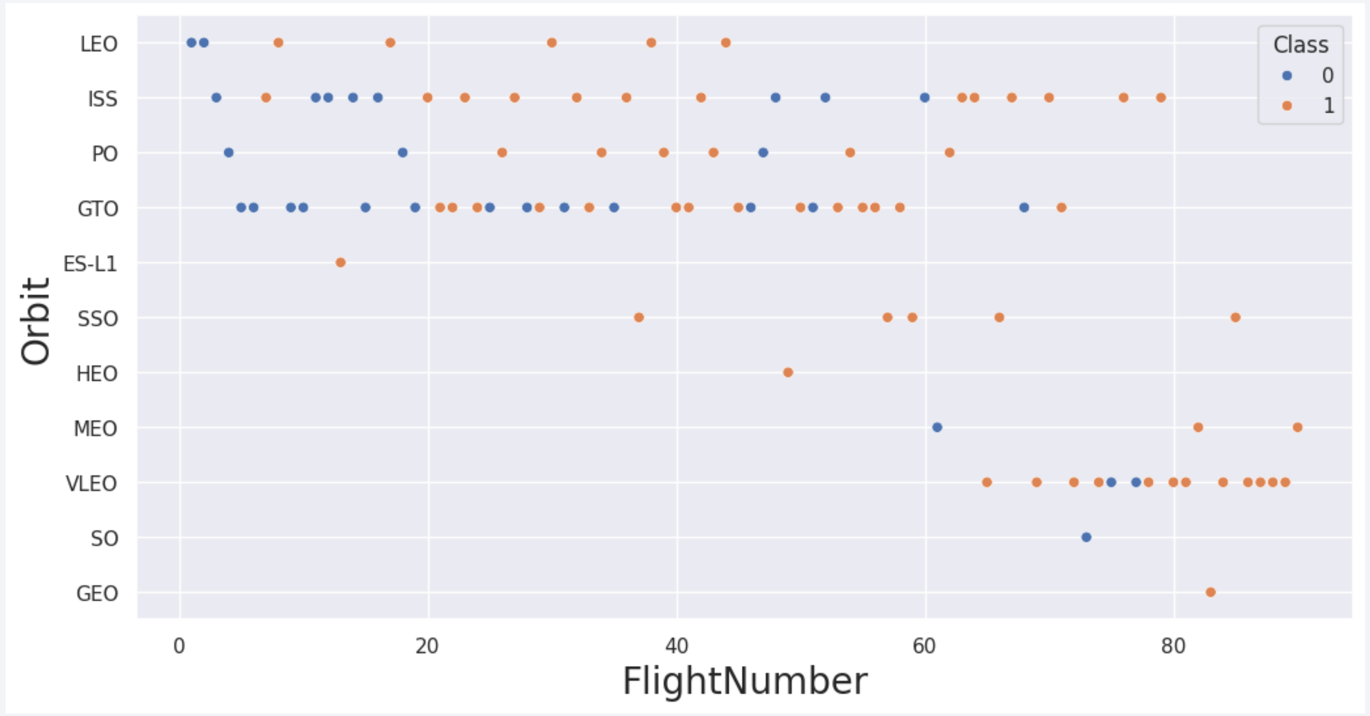
- There is no clear pattern to say the launch site is dependent to the pay load mass for the success rate.

Success Rate vs. Orbit Type



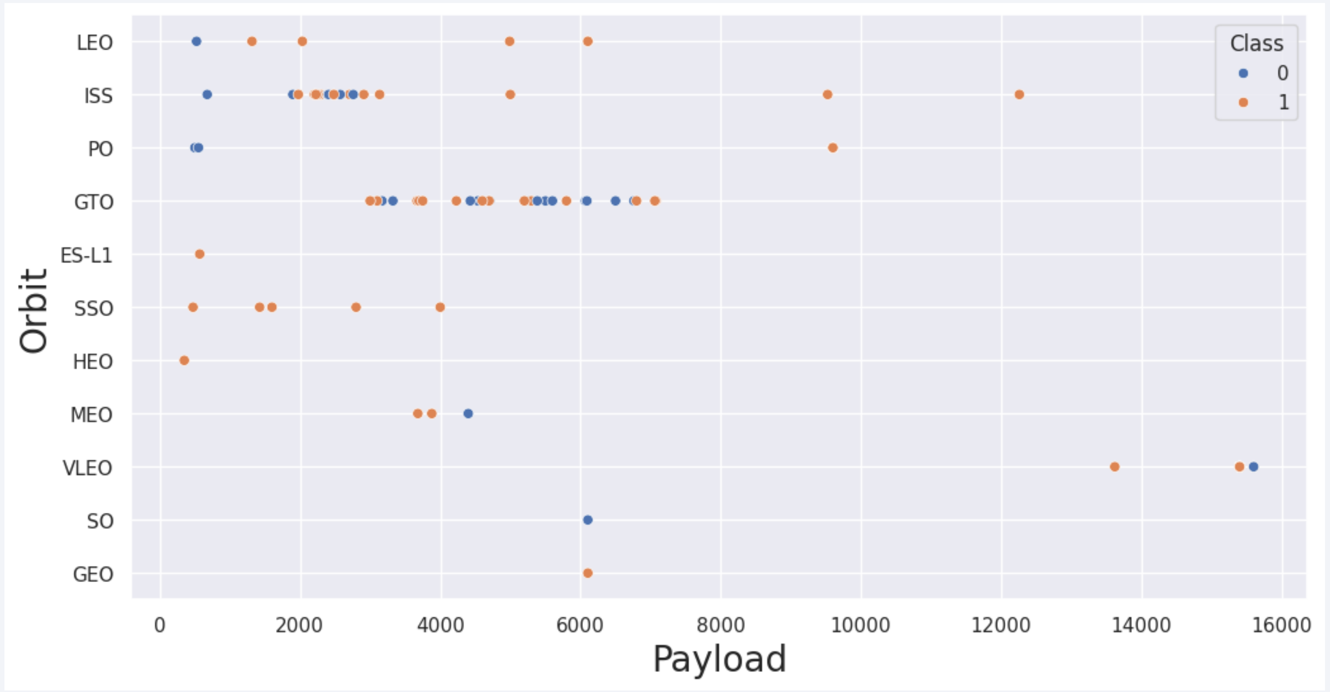
- This barchart shows the relationship between different orbits and success rate, outcoming that some orbits as ES-L1, GEO, HEO, SSO has a 1 or 100% success rate, contrary to SO orbit that shows a 0% success rate

Flight Number vs. Orbit Type



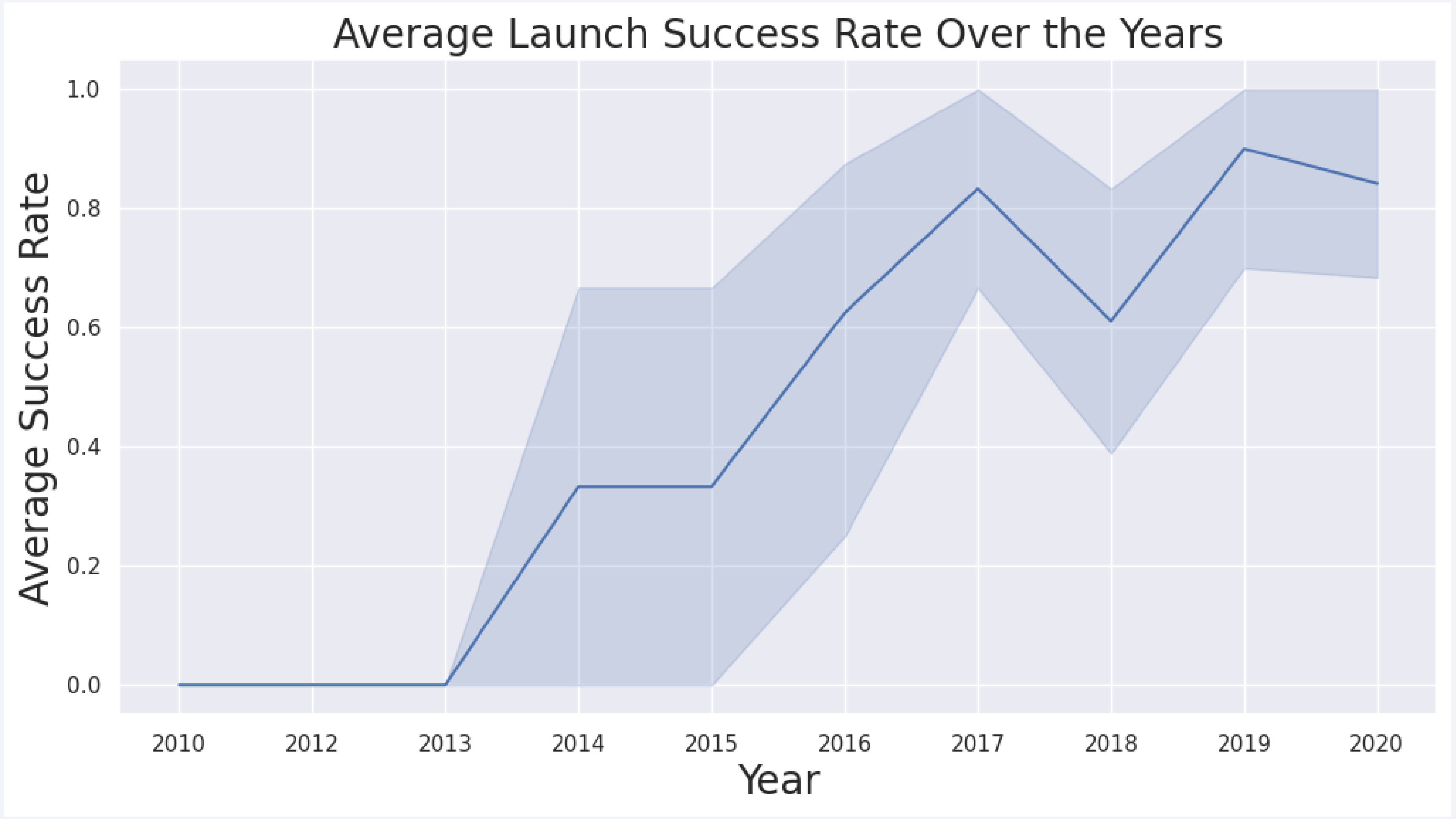
This scatter plot show a kind of dependency between each orbit and the flight number, where while flight number increases the success rate also increase. Except for the GTO orbit, this doenst show a clear pattern or dependency

Payload vs. Orbit Type



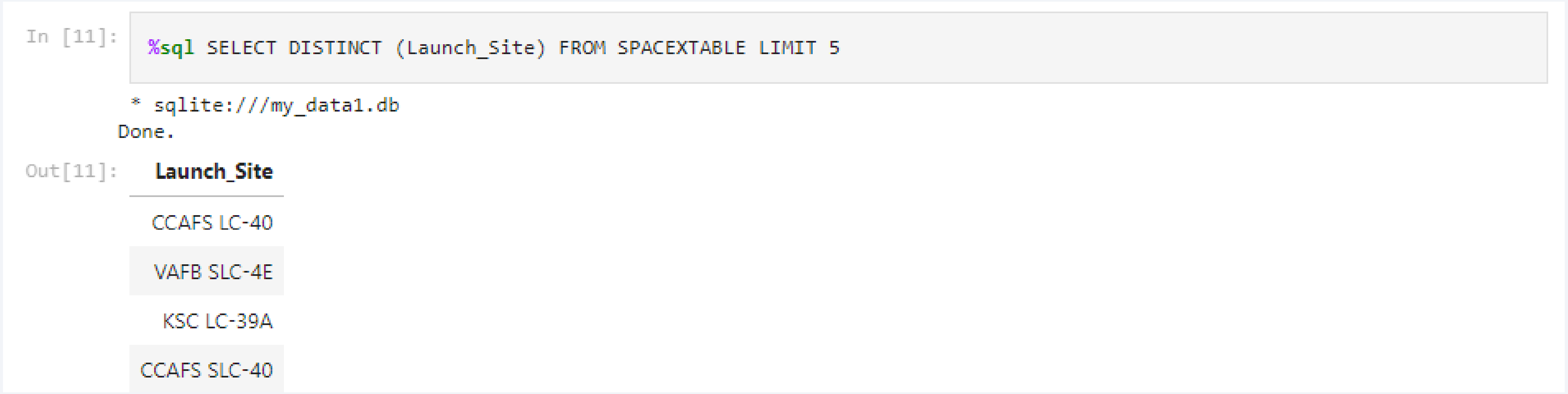
This scatterplot show that a heavier payload positive impact on most orbits. However, it has negative impact on MEO and VLEO orbit.

Launch Success Yearly Trend



Through this line chart we can clearly observe that the success rate since 2013 kept increasing till 2020

All Launch Site Names

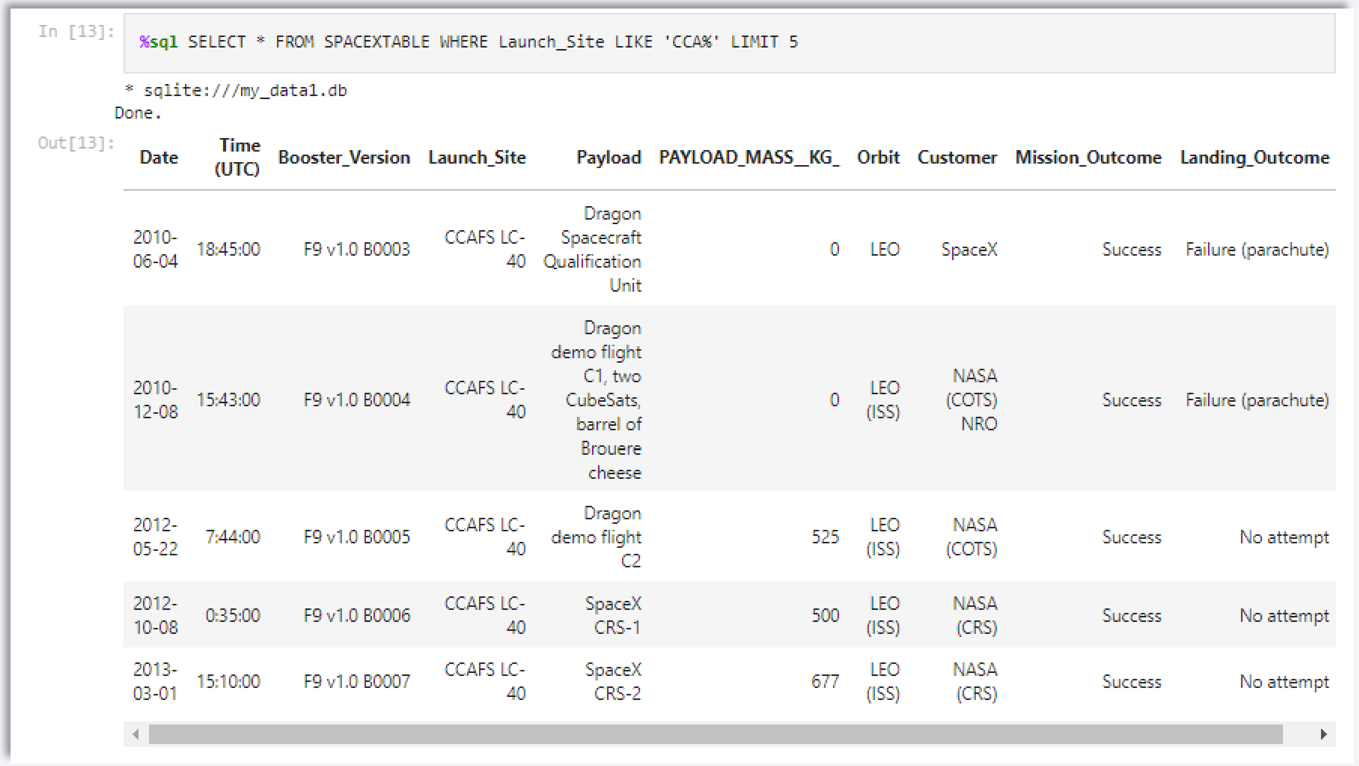


```
In [11]: %sql SELECT DISTINCT (Launch_Site) FROM SPACEXTABLE LIMIT 5

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

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

Through this **SQL** query we displayed the names of the unique launch sites in the space mission using the **SELECT statement**

Launch Site Names Begin with 'CCA'



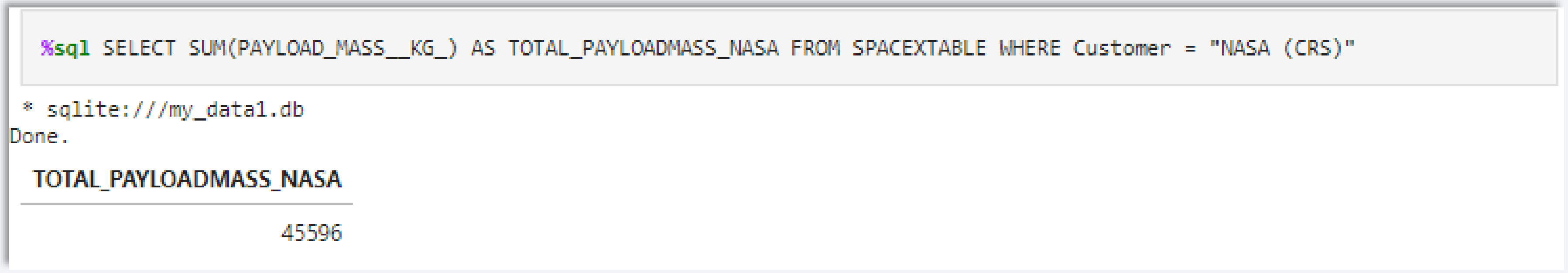
```
In [13]: %sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5

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

Out[13]:

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

For this query, we used the **LIKE operator** to filter records where the launch site starts with **'CCA'** and limits the output to 5 records.

Total Payload Mass



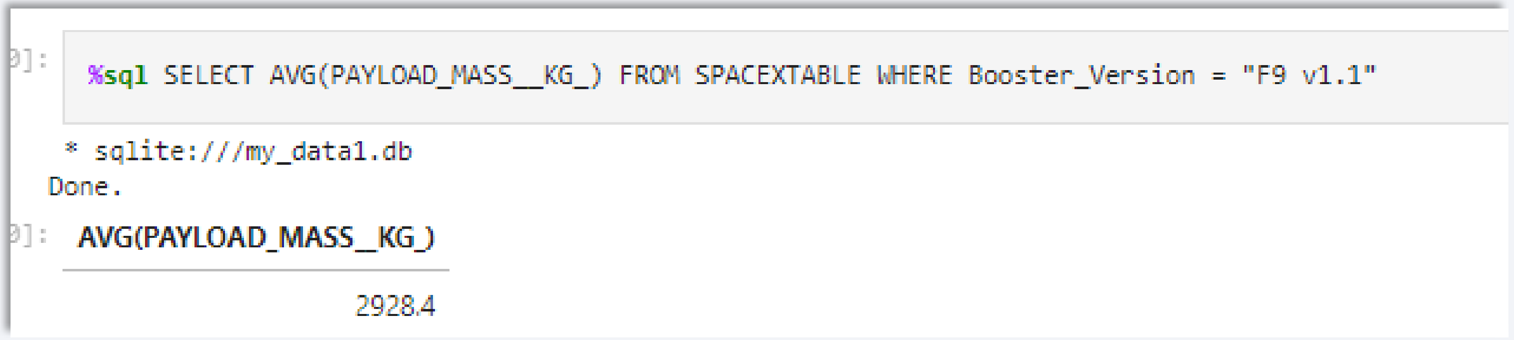
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOADMASS_NASA FROM SPACEXTABLE WHERE Customer = "NASA (CRS)"
```

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

TOTAL_PAYLOADMASS_NASA
45596

In this case, we calculated the sum of payload masses for missions from the customer "NASA (CRS)" using the SUM() statement and labels the result as "TOTAL_PAYLOADMASS_NASA."

Average Payload Mass by F9 v1.1

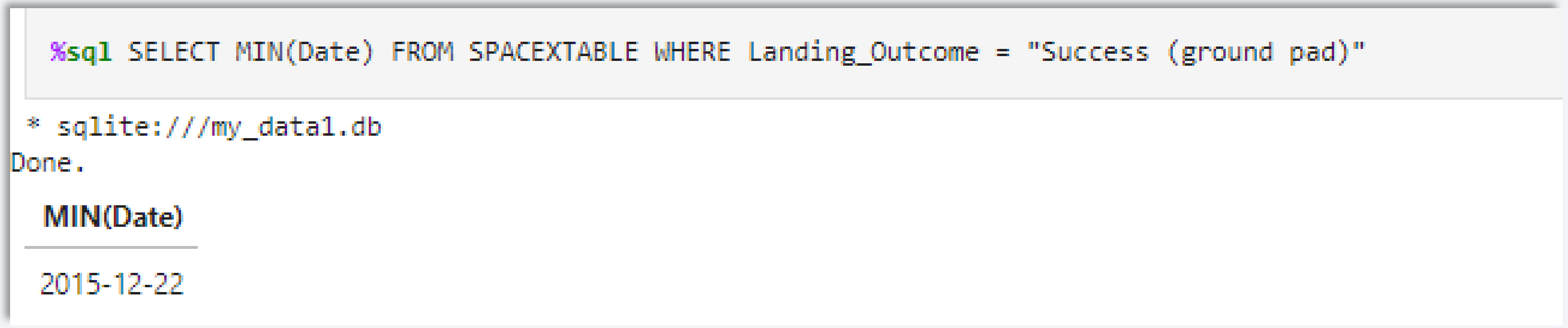


```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version = "F9 v1.1"

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

AVG(PAYLOAD_MASS__KG_)
2928.4

In this query, we calculated the average payload mass for missions using the booster version "F9 v1.1." using the AVG() statement.

First Successful Ground Landing Date

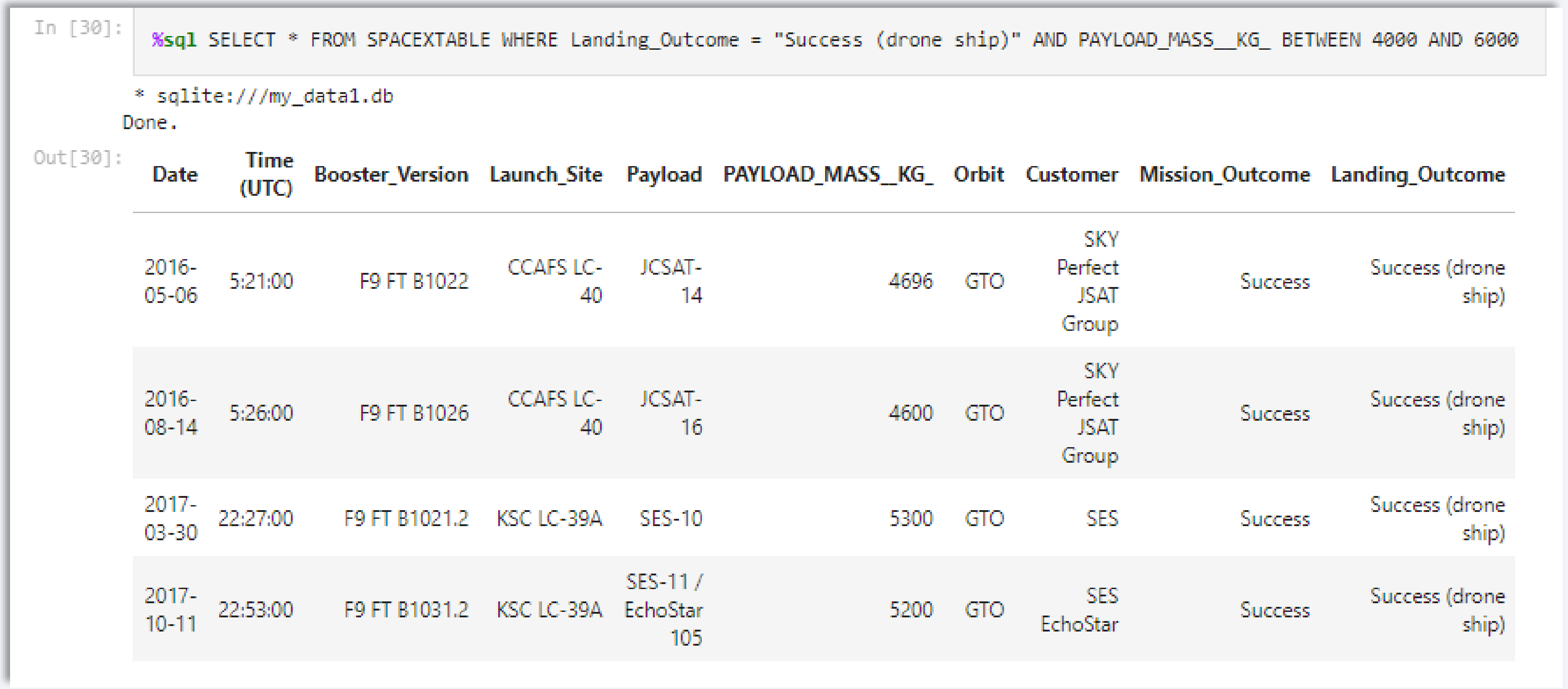


```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = "Success (ground pad)"

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

MIN(Date)
2015-12-22

In this query, we Identified and displayed the date of the earliest successful landing outcome on a ground pad. Retrieving the minimum (earliest) date where the landing outcome was "Success (ground pad)."

Successful Drone Ship Landing with Payload between 4000 and 6000



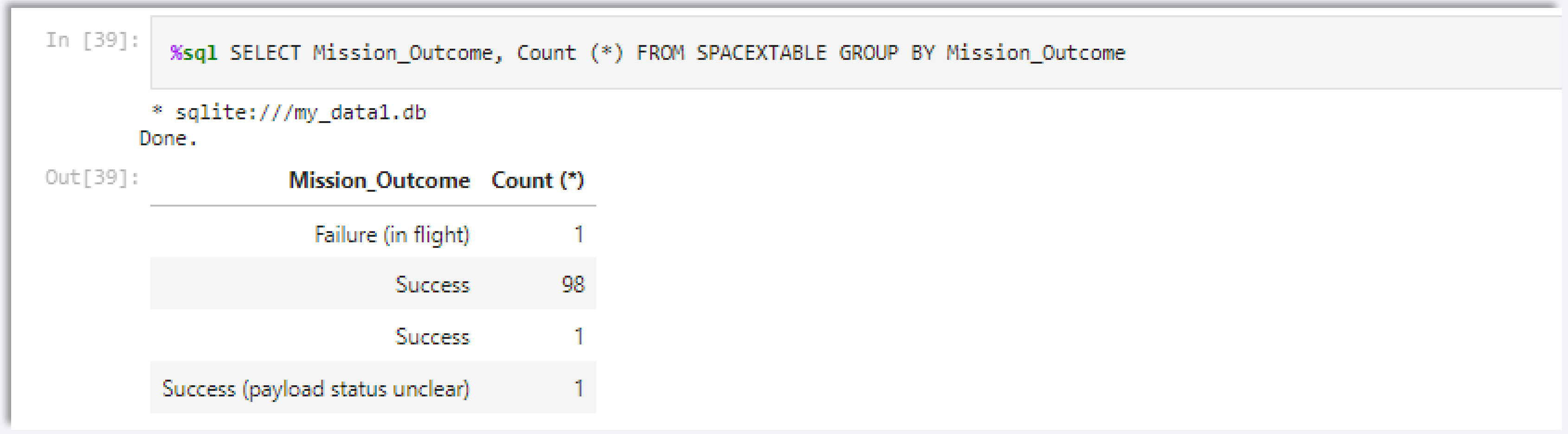
```
In [30]: %sql SELECT * FROM SPACEXTABLE WHERE Landing_Outcome = "Success (drone ship)" AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000

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

Out[30]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2016-05-06	5:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-08-14	5:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-10-11	22:53:00	F9 FT B1031.2	KSC LC-39A	SES-11 / EchoStar 105	5200	GTO	SES EchoStar	Success	Success (drone ship)

In this query, we filtered records where the landing outcome was "Success (drone ship)" and the payload mass falls within the specified range, using the **BETWEEN OPERATOR**

Total Number of Successful and Failure Mission Outcomes

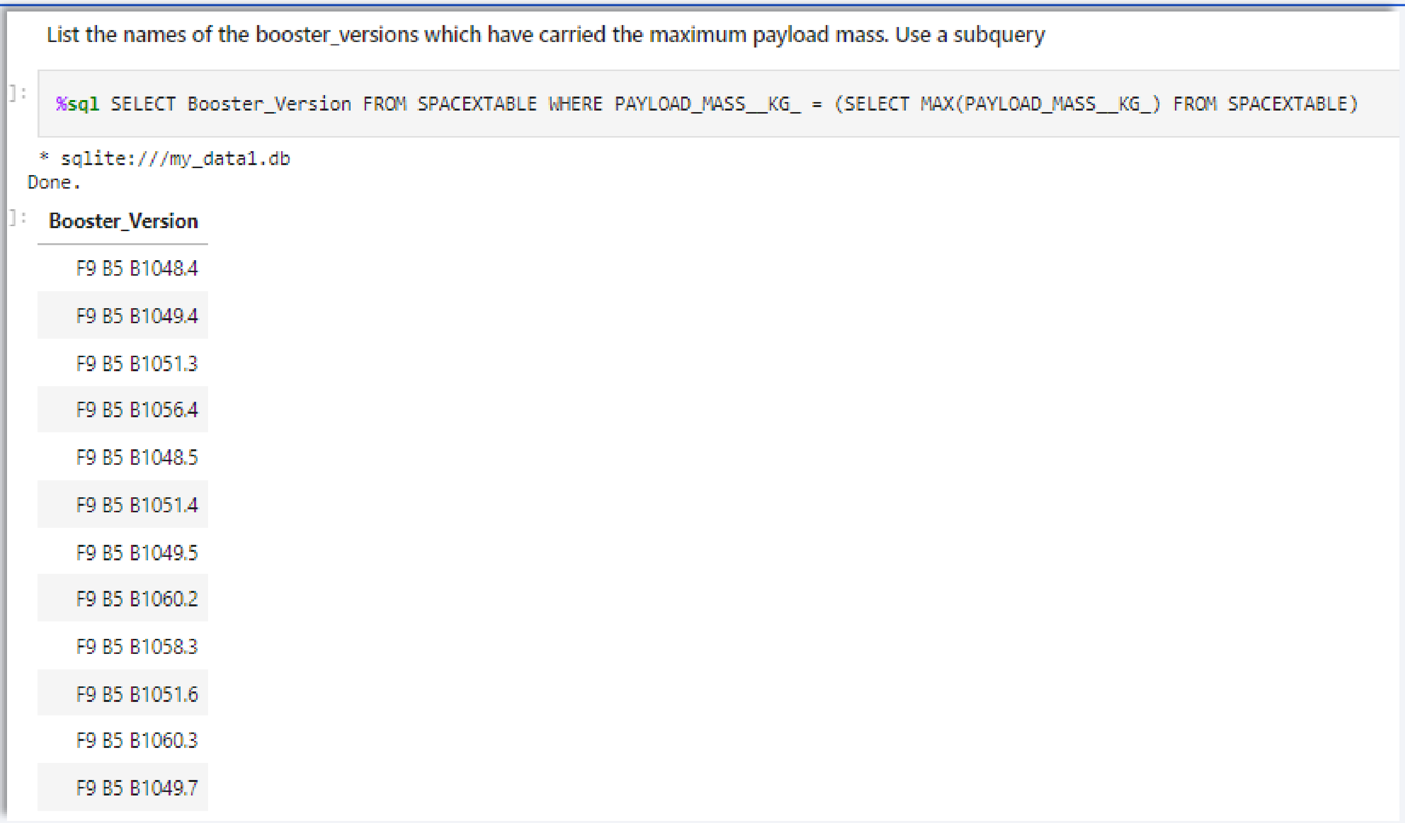


```
In [39]:  %sql SELECT Mission_Outcome, Count (*) FROM SPACEXTABLE GROUP BY Mission_Outcome

           * sqlite:///my_data1.db
          Done.
Out[39]:
```

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

In this case, we counted and grouped mission outcomes, providing the total count for each distinct mission outcome, by using the **COUNT() AND GROUP BY STATEMENT**

Boosters Carried Maximum Payload



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

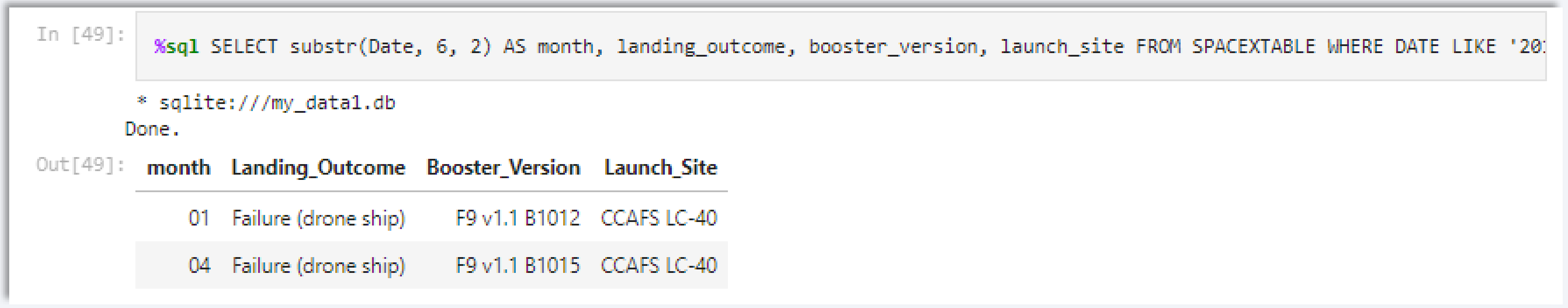
```
]: %sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

```
 * 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

In this case, we used a **subquery** to find the maximum payload mass and then retrieves the booster versions associated with that maximum payload mass.

2015 Launch Records



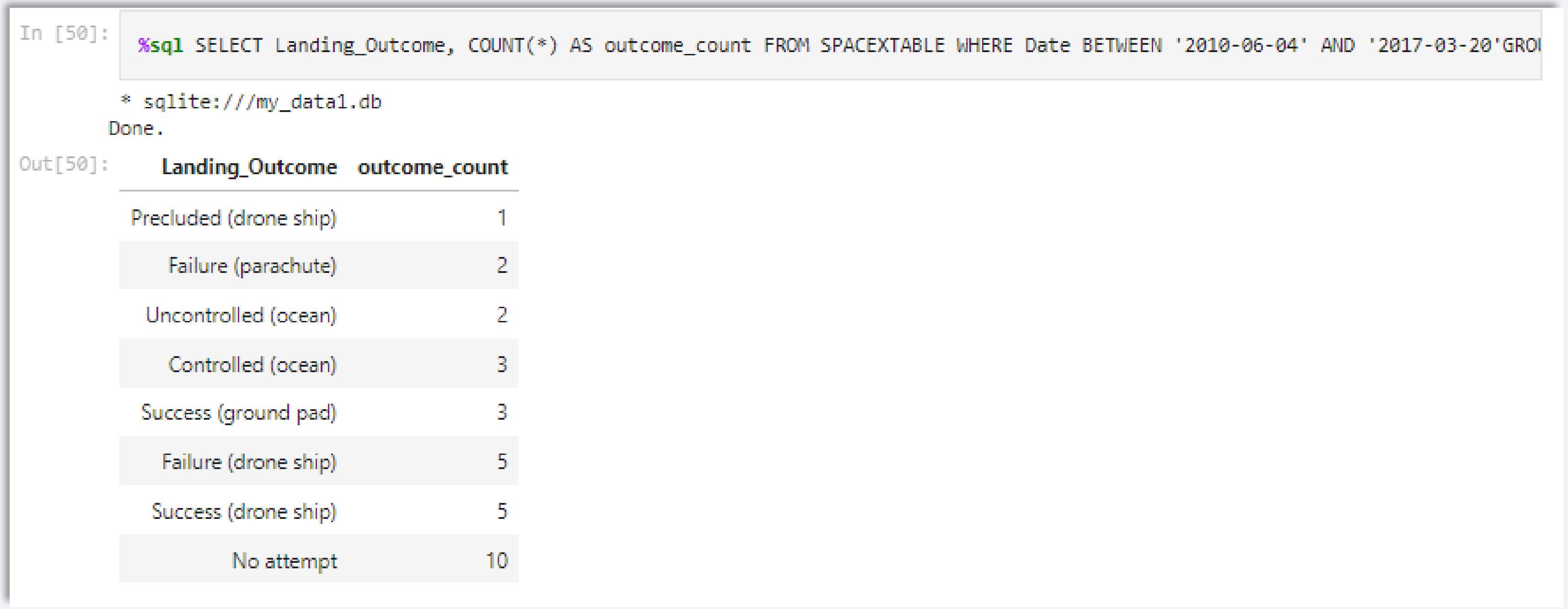
```
In [49]:  %sql SELECT substr(Date, 6, 2) AS month, landing_outcome, booster_version, launch_site FROM SPACEXTABLE WHERE DATE LIKE '20

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

Out[49]:

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

For this query, we extracted month names, landing outcomes, booster versions, and launch sites for records in the year 2015 where the landing outcome was a failure on a drone ship. Using the **SUBSTR() FUNCTION** and **LIKE OPERATOR**

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



```
In [50]:  %sql SELECT Landing_Outcome, COUNT(*) AS outcome_count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'GROI

           * sqlite:///my_data1.db
          Done.
Out[50]:
```

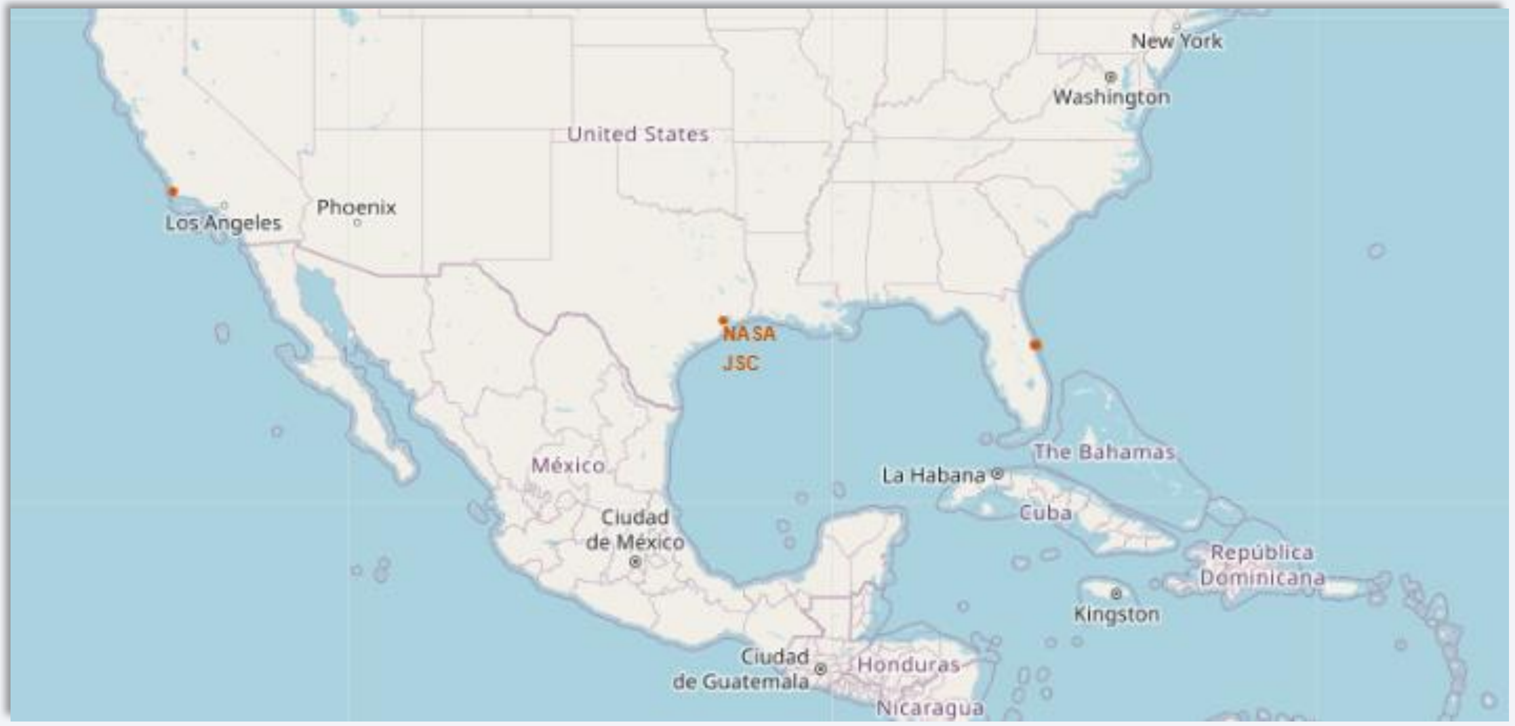
Landing_Outcome	outcome_count
Precluded (drone ship)	1
Failure (parachute)	2
Uncontrolled (ocean)	2
Controlled (ocean)	3
Success (ground pad)	3
Failure (drone ship)	5
Success (drone ship)	5
No attempt	10

For this query, we ranked the count of landing outcomes in descending order for records within the specified date range, by using the **COUNT () STATEMENT** AND **BETWEEN OPERATOR**

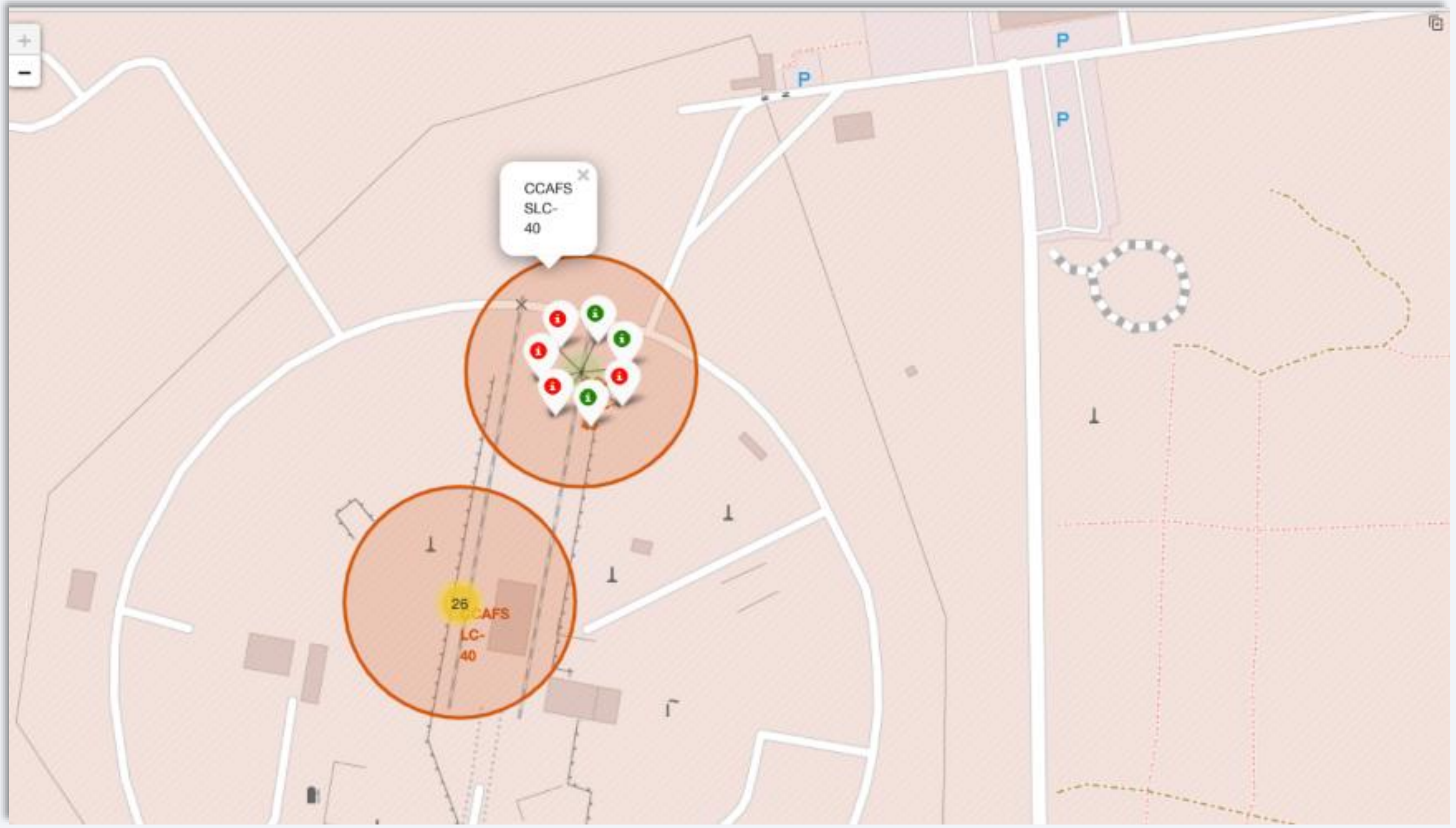
Section 3

Launch Sites Proximities Analysis

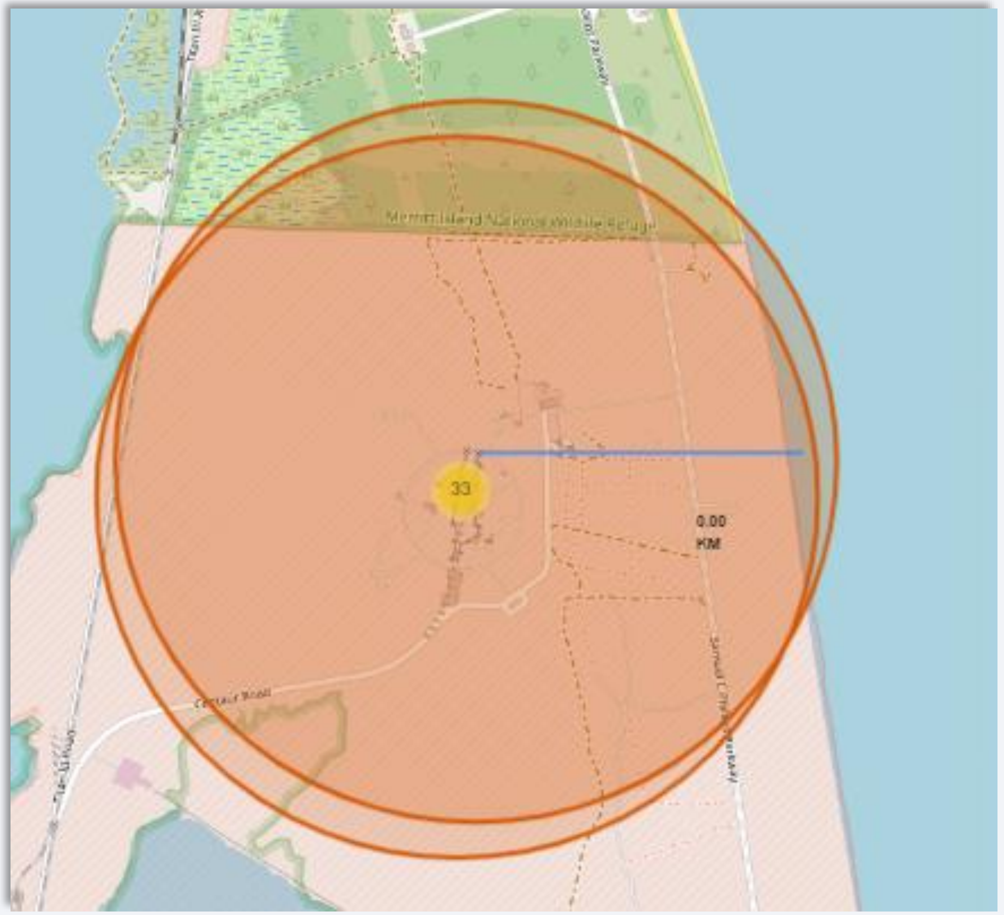
Location of Launch Sites

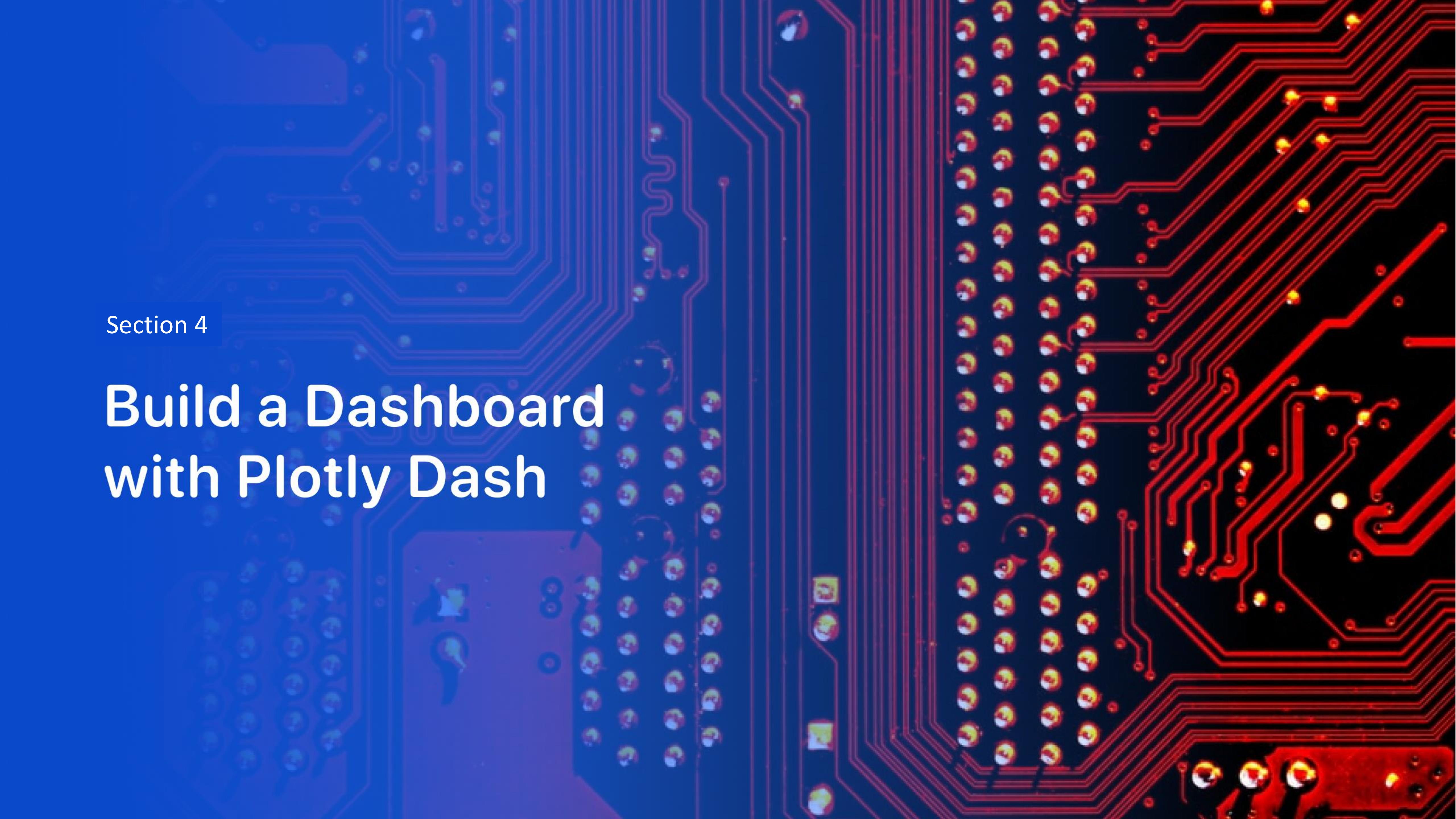


Markers showing launch sites with color labels



Launch Sites Distance to Landmarks

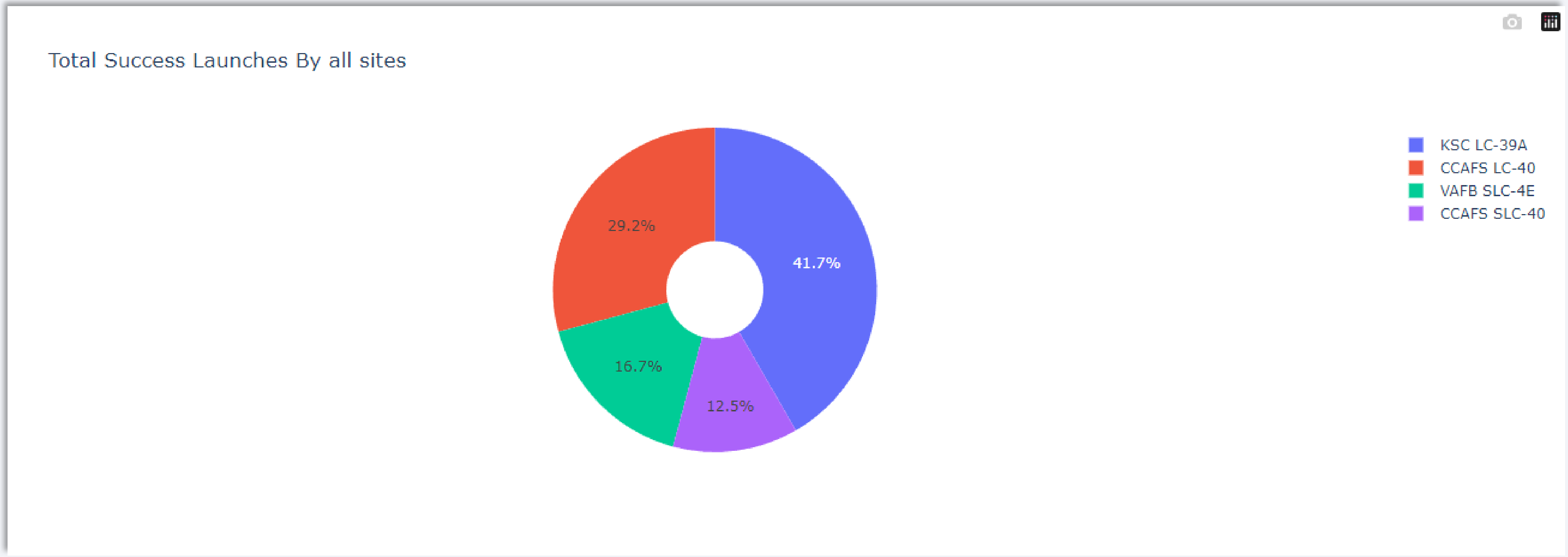

Merritt Island National Wildlife Refuge
33
0.00 KM
Samuel C Phillips Parkway
Contact Road



Section 4

Build a Dashboard with Plotly Dash

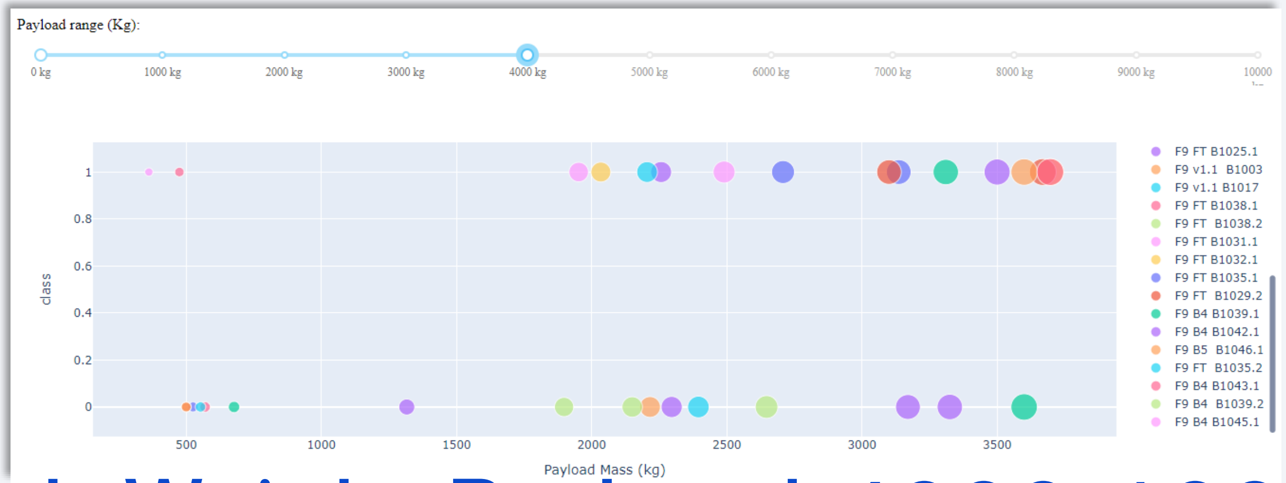
Total Success Launches by all sites



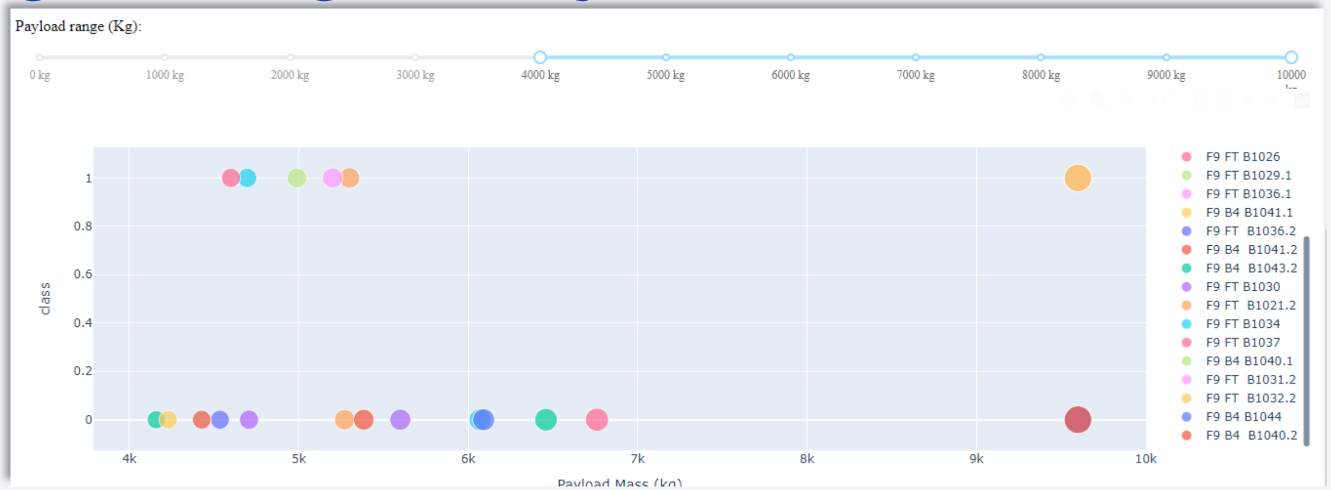
This PIE CHART show how KSC LC-39A had the most successful launches out of all the sites

Low Weight PayLoad 0-4000kg

We can see how lighter payload flights had more successful trips



High Weight PayLoad 4000-10000kg

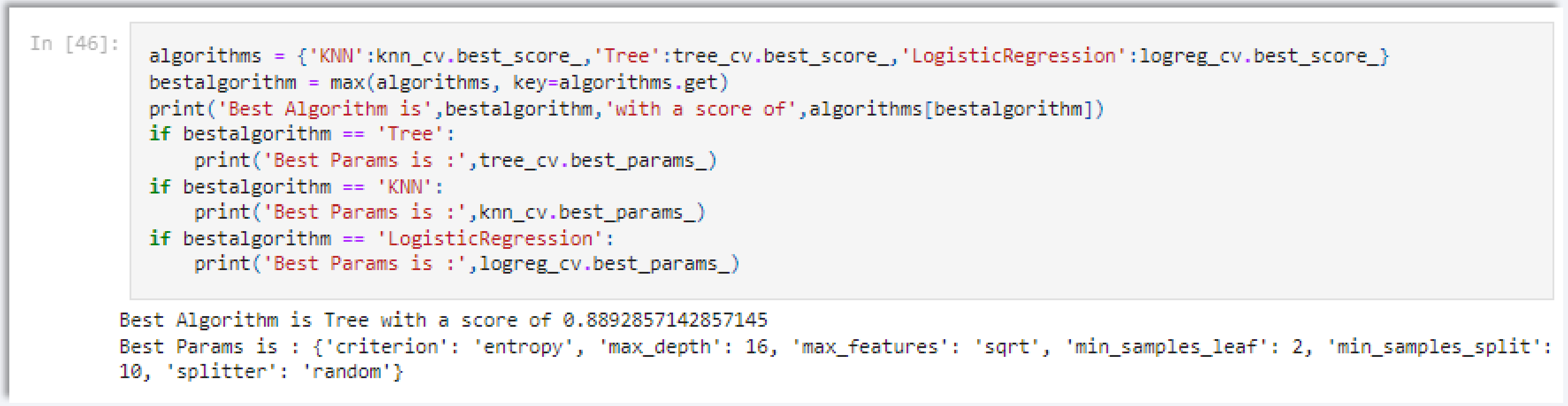


Section 5

Predictive Analysis (Classification)

In your specific case, the best-performing algorithm is a Decision Tree ('Tree') with a score of **0.8892857142857145**, and the corresponding best hyperparameters

Classification Accuracy

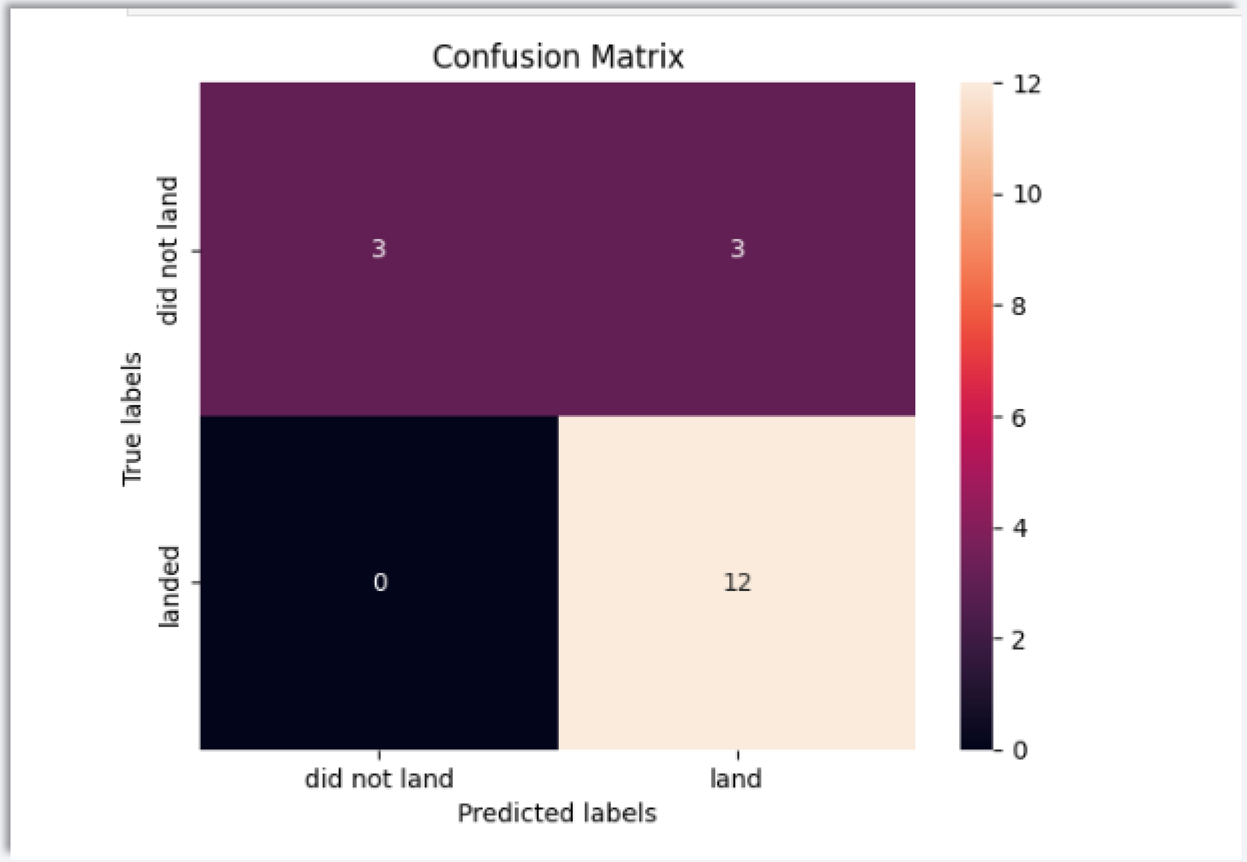


```
In [46]: algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
         bestalgorithm = max(algorithms, key=algorithms.get)
         print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
         if bestalgorithm == 'Tree':
             print('Best Params is :',tree_cv.best_params_)
         if bestalgorithm == 'KNN':
             print('Best Params is :',knn_cv.best_params_)
         if bestalgorithm == 'LogisticRegression':
             print('Best Params is :',logreg_cv.best_params_)

         Best Algorithm is Tree with a score of 0.8892857142857145
         Best Params is : {'criterion': 'entropy', 'max_depth': 16, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split':
         10, 'splitter': 'random'}
```

For this code, we evaluated different algorithms (KNN, Decision Tree, Logistic Regression) using cross-validation, where the best-performing algorithm is selected along with its corresponding hyperparameters.

In this specific case, the best-performing algorithm is a **Decision Tree ('Tree')** with a score of **0.8892857142857145**, and the corresponding best hyperparameters are printed.

Confusion Matrix



Confusion matrix is a table used in machine learning to evaluate the performance of a classification algorithm. It provides a summary of the predictions made by a classification model compared to the actual ground truth. The matrix has four entries, each corresponding to one of the four possible outcomes of a binary classification problem:

Conclusions

- The decision tree algorithm (Tree) appears to be the best-performing algorithm based on the provided script. It achieved the highest score among the evaluated algorithms (KNN, Tree, Logistic Regression).
- KSC LC-39A had the most successful launches out of all the sites
- Lighter payload flights had more successful trips
- The success rate of the flights since 2013 kept increasing till 2020…

Thank you!