IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Alirio Mieres
Jun 28 2024

Outline

Executive Summary

- Summary of methodologies
  - Data collection using SpaceX API
  - Data collection with web scrapping
  - Data wrangling
  - Exploratory data analysis using SQL
  - Data visualization using Python, Pandas and Matplotlib
  - Interactive mapping with Folium and web app development with Plotly Dash
  - Machine learning prediction
- Summary of all results
  - EDA results
  - Dashboards and interactive visual analytics
  - Predictive analysis

Introduction

Project background and context

The commercial space age is here, with companies like Virgin Galactic, Rocket Lab, Blue Origin, and SpaceX leading the way. SpaceX stands out for its achievements and low costs thanks to the reuse of the first stage of its Falcon 9 rockets.

Problems

- First stage recovery: Determine if the Falcon 9 first stage will be able to land and be reused.
- Predictive model: Use machine learning to predict first-stage reuse.
- Data Analysis: Collect and analyze SpaceX launch data to establish the cost of each launch to Space Y.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - A Rest API and web scraping were used to obtain relevant data.
- Perform data wrangling
    - Data Cleaning: Removed null values and fixed inconsistent data.
    - Data transformation: Dates were converted to a standard format and relevant features were extracted.
    - Data validation: The accuracy and consistency of the prepared data was verified.
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform predictive analysis using classification models

Data Collection

Information about the SpaceX Falcon9 launches was collected in two main ways:

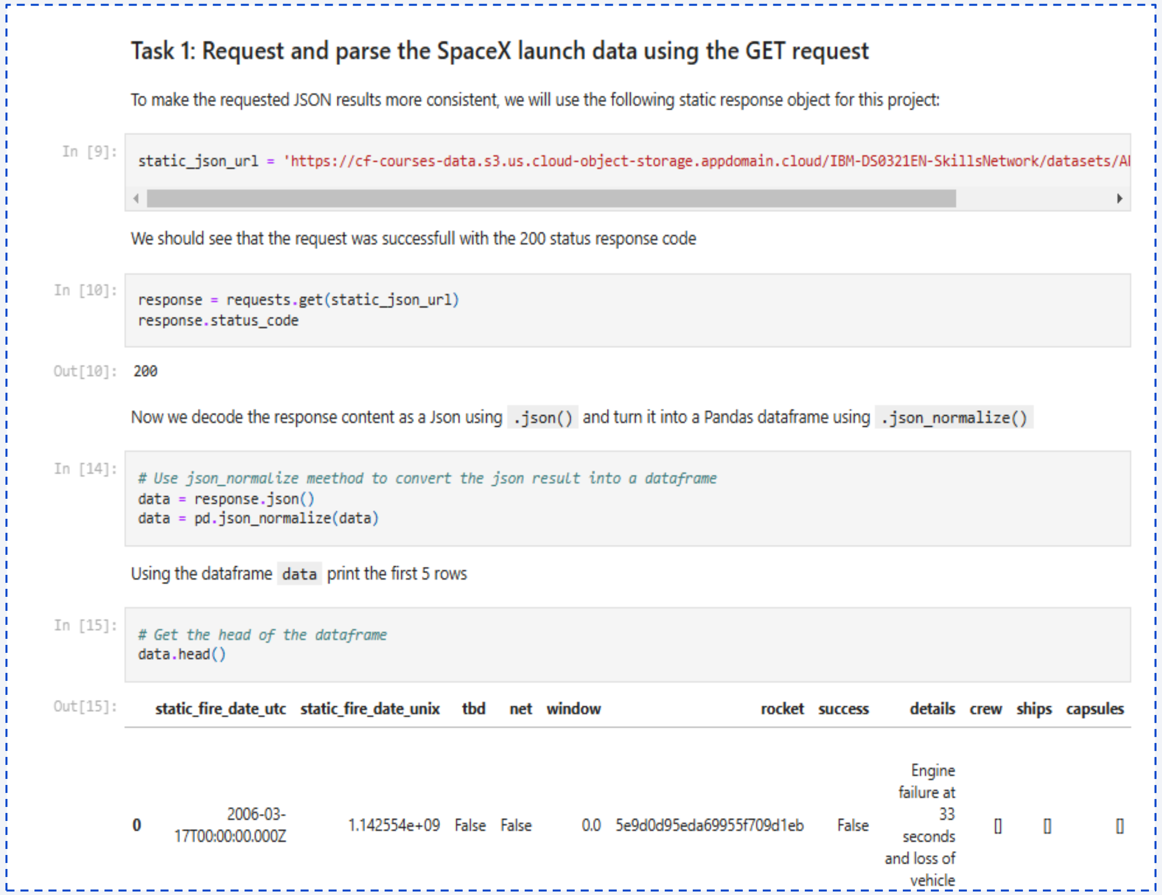
- Using the SpaceX API: First, requests were made to the SpaceX RESTful API to obtain data on rocket launches. Several helper functions were defined to make the API easier to use, using identifiers in the launch data and requesting specific information from the SpaceX API URL. The results in JSON format were requested and parsed using a GET request, and then decoded and converted to a Pandas data frame for consistency.
- Wikipedia web scraping: Web scraping was also performed to obtain historical records of Falcon 9 launches from a Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches." Using the BeautifulSoup and request libraries, the HTML table records of the Falcon 9 launches were extracted from the Wikipedia page. This data was analyzed and converted into a Pandas data frame.

Data Collection – SpaceX API

Objective: Predict the successful landing of the Falcon 9.

Process:

- Request and cleaning of data.
- Filtered by Falcon 9.
- Handling missing values.
- Exploratory analysis.
- Training and prediction.
- Importance: Reduce launch costs.
- Github url: https://github.com/Alirio-Mieres/spacex-data-science-project/blob/main/01-spacex-data-collection-api.ipynb



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

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

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

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

```
In [10]: response = requests.get(static_json_url)
         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 [14]: # 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 [15]: # Get the head of the dataframe
         data.head()
```

Out[15]:

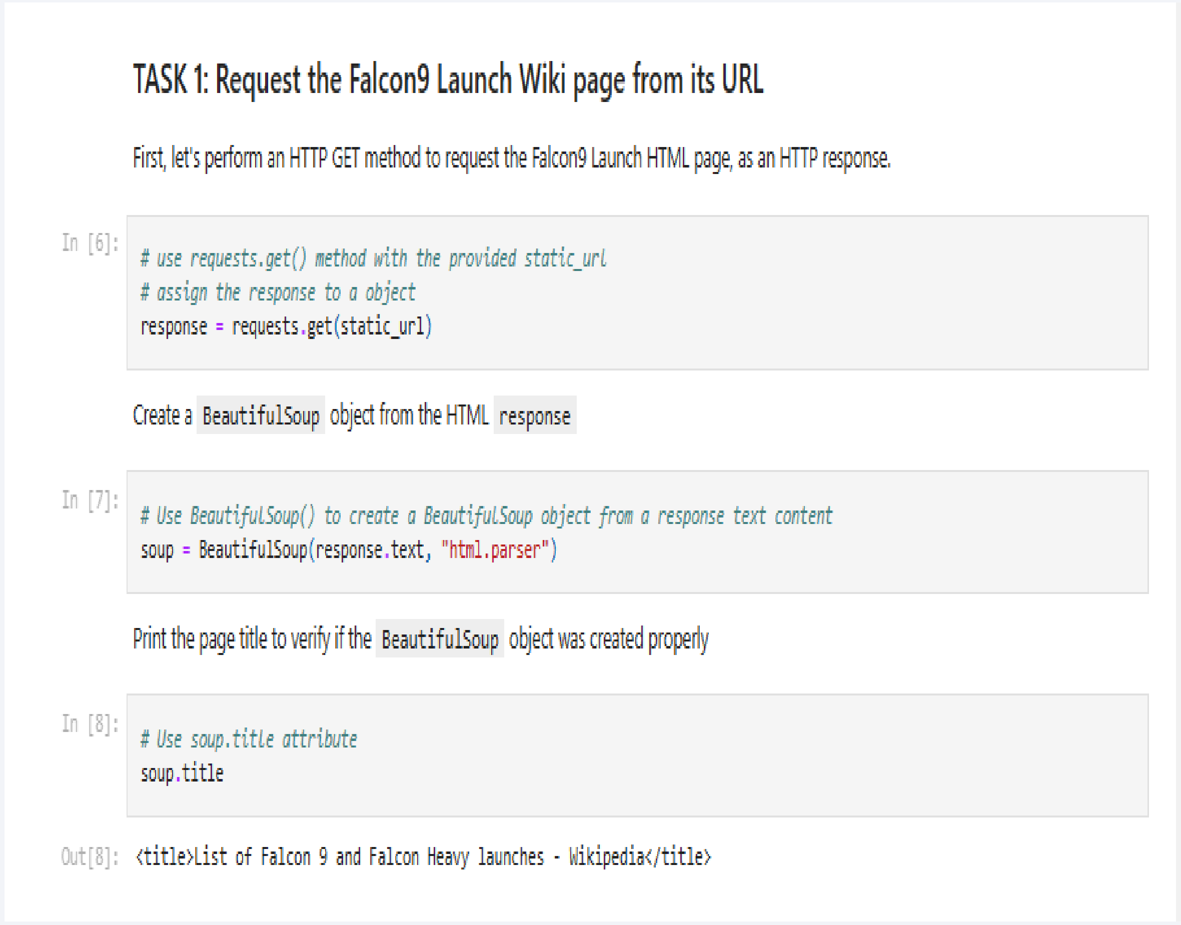
	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]

Data Collection – Web Scraping

Objective: Collect and analyze historical records of Falcon 9 launches.

Process:

- Website request.
- Extracting column names.
- Creation and filling of dictionary.
- Conversion to DataFrame.
- Data cleaning and analysis.
- Tools: BeautifulSoup, Pandas.
- Github https://github.com/Alirio-Mieres/spacex-data-science-project/blob/main/02-spacex-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 [6]:
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

```
In [7]:
# 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

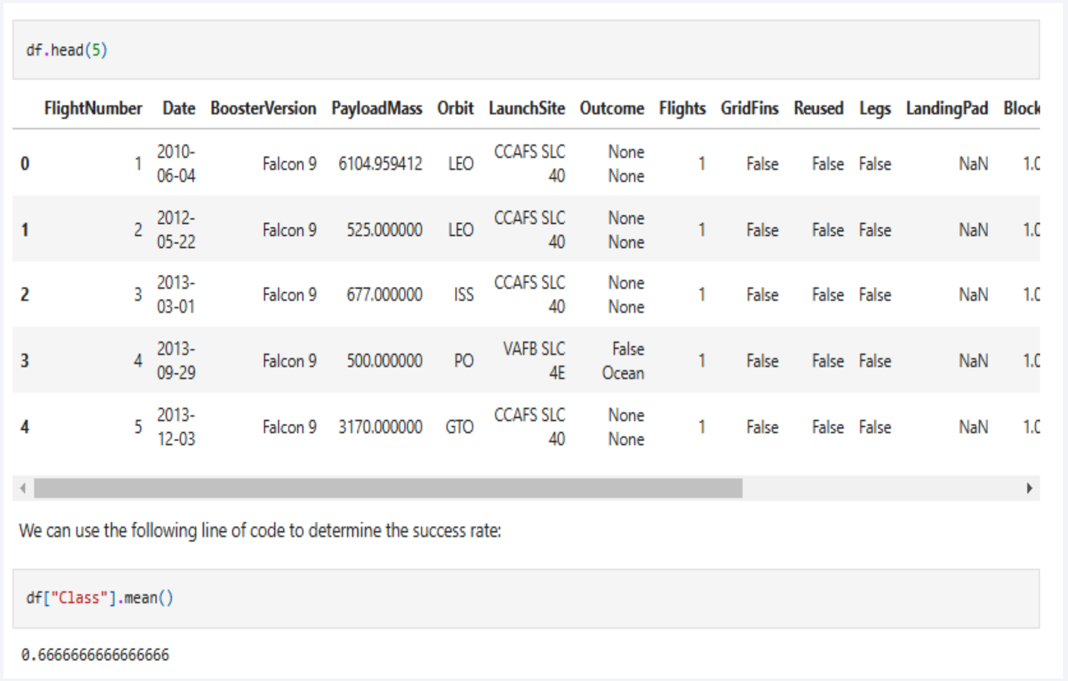
```
In [8]:
# Use soup.title attribute
soup.title

Out[8]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Data Wrangling

Objective: Predict the success of the Falcon 9 landing.

Process:

- Exploratory Data Analysis (EDA).
- Determination of training labels.
- Calculation of launches and orbits.
- Creation of landing labels.
- Tools: Pandas, NumPy.
- https://github.com/Alirio-Mieres/spacex-data-science-project/blob/main/03-spacex-data%20wrangling.ipynb



```
df.head(5)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.(
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.(
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.(
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.(
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.(

We can use the following line of code to determine the success rate:

```
df["Class"].mean()
```

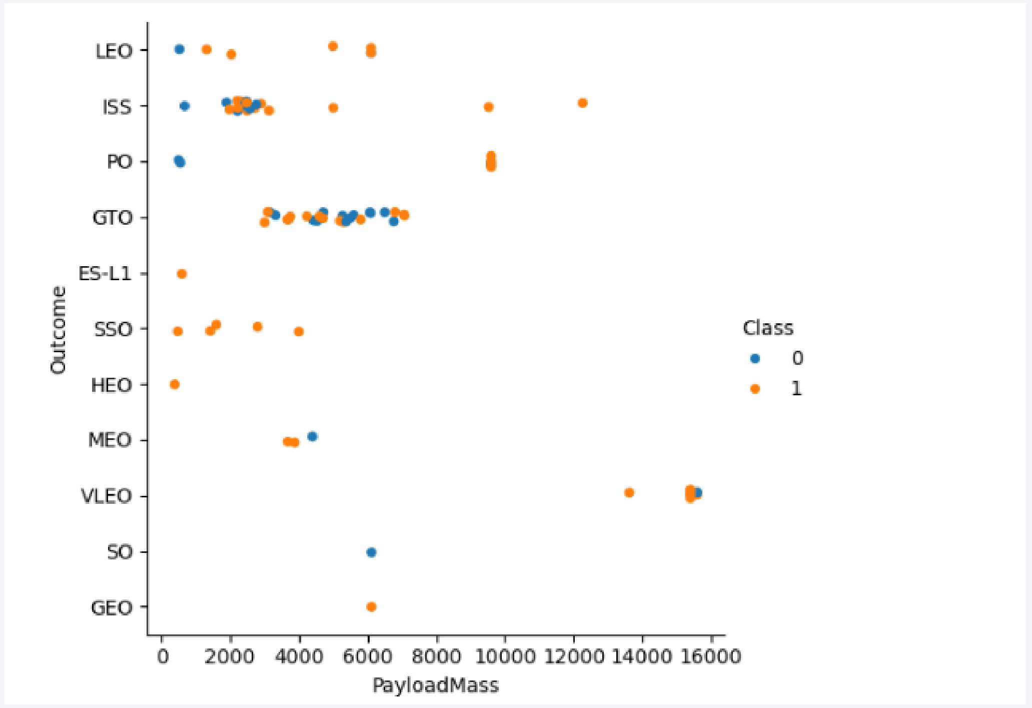
```
0.6666666666666666
```

EDA with Data Visualization

Objective: Predict the success of the Falcon 9 landing through visualization tools.

Visualize the relationship between:

- Flight Number and Launch Site.
- Payload and Launch Site
- Success rate of each orbit type
- FlightNumber and Orbit type
- Payload and Orbit type
- The launch success yearly trend
- Tools: Pandas, NumPy, Seaborn, and Matplotlib.



- https://github.com/Alirio-Mieres/spacex-data-science-project/blob/main/04-first-stage-prediction-visualization.ipynb

EDA with SQL

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

```
select distinct "Launch_Site" from SPACEXTABLE
```

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

```
select * from SPACEXTABLE where "Launch_Site" like "CCA%" limit 5
```

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

```
select sum("PAYLOAD_MASS__KG_") as total_payload_mass_kg from SPACEXTABLE where customer = "NASA (CRS)"
```

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

```
select avg("PAYLOAD_MASS__KG_") as average_payload_mass_kg from SPACEXTABLE
where "Booster_Version" = "F9 v1.1"
```

EDA with SQL

- List the date when the first successful landing outcome in ground pad was achieved

```
select MIN("Date") FROM 'SPACEXTABLE' WHERE "Landing_Outcome" = "Success (ground pad)";
```

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

```
select BOOSTER_VERSION from SPACEXTABLE where LANDING_OUTCOME =
 'Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

- List the total number of successful and failure mission outcomes

```
select "Mission_Outcome", count(*) as count_per_group from SPACEXTABLE group by "Mission_Outcome";
```

EDA with SQL

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

```
select "Booster_Version", "Payload", "PAYLOAD_MASS__KG_" from SPACEXTABLE where
"PAYLOAD_MASS__KG_" = (select max("PAYLOAD_MASS__KG_") from SPACEXTABLE )
```

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

```
select substr(Date,6,2) as Month, substr(Date, 0, 5) as Year, Landing_Outcome, Booster_Version, Launch_Site from
SPACEXTABLE where substr("Date", 0, 5) = "2015" and "Mission_Outcome" like "Failure%"
```

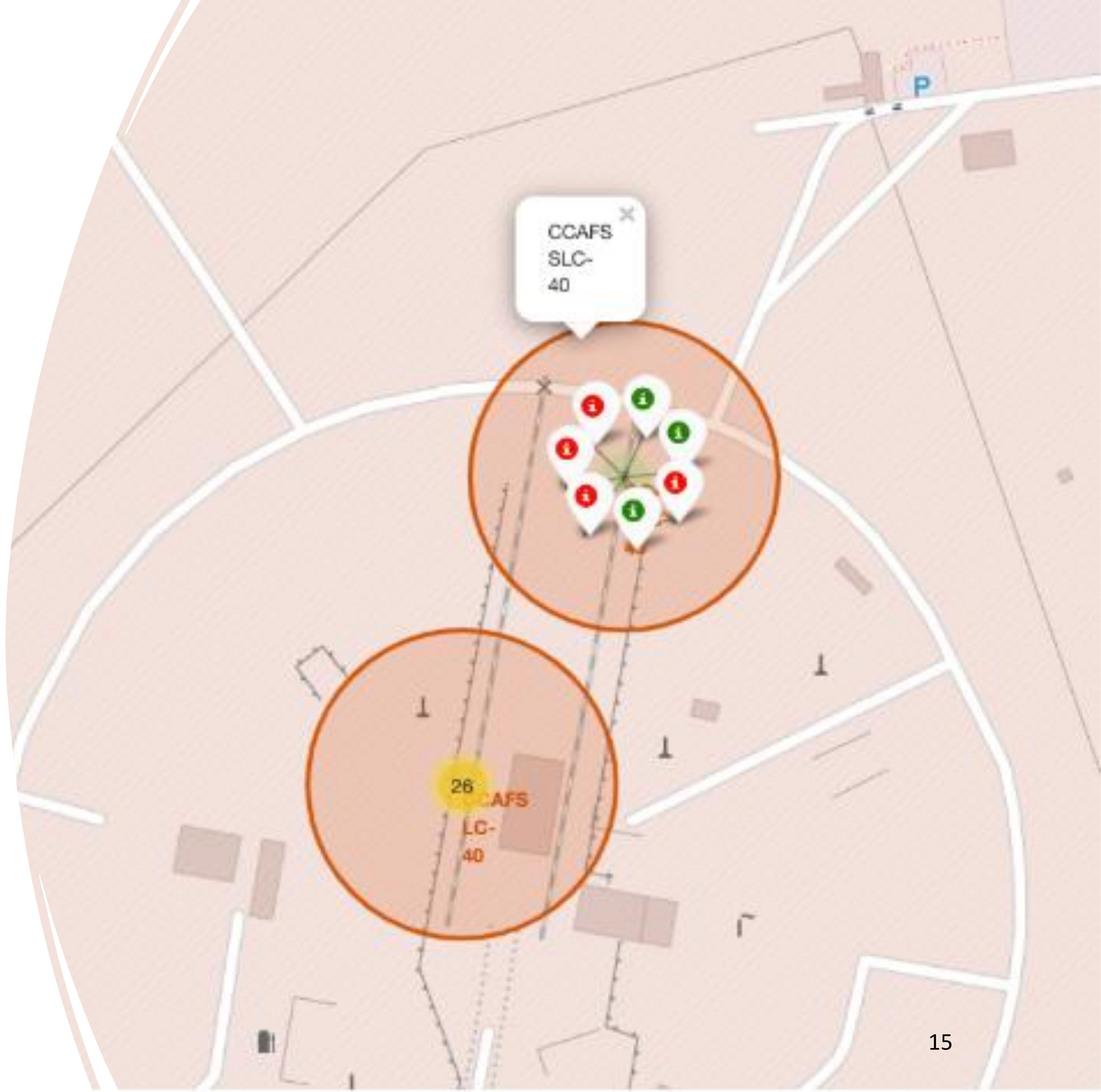
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the dates 2010-06-04 and 2017-03-20, in descending order.

```
select * FROM SPACEXTABLE WHERE "Landing _Outcome" LIKE 'Success%' AND
(Date BETWEEN '04-06-2010' AND '20-03-2017') ORDER BY Date DESC;
```

- Link: https://github.com/Alirio-Mieres/spacex-data-science-project/blob/main/05-spacex-sqllite.ipynb

Build an Interactive Map with Folium

- Objective: Analyze the influence of launch site locations on launch success rates.
- Process:
- Add bookmarks for launch sites.
- Add circles for success/failure rates.
- Calculate distances to important proximities.
- Add lines to display proximities.
- Add MousePosition functionality.
- Tools: Folium, Pandas.https://github.com/Alirio-Mieres/spacex-data-science-project/blob/main/06-spacex_site_location.ipynb



Build a Dashboard with Plotly Dash

Objective: Predict the success of SpaceX Falcon 9 launches using interactive visualizations.

Graphics:

- The pie chart shows successful launches by the site.
- Scatter plot to show the determination between payload mass and launch success.

Interactions:

- The dropdown menu to select launch sites.
- Slider to adjust the payload mass range.
- Tools: Dash, Plotly, Pandas.
- https://github.com/Alirio-Mieres/spacex-data-science-project/blob/main/07-spacex_ploty_app/spacex_dash_app.py

Predictive Analysis (Classification)

Process:

- Data Exploration and Labeling:
- Perform EDA to understand data.
- Create a success class label.
- Data Preparation:
- Standardize data.
- Split into training and test sets.
- Model Development:
- Optimize SVM, Decision Trees, and Logistic Regression using GridSearchCV.

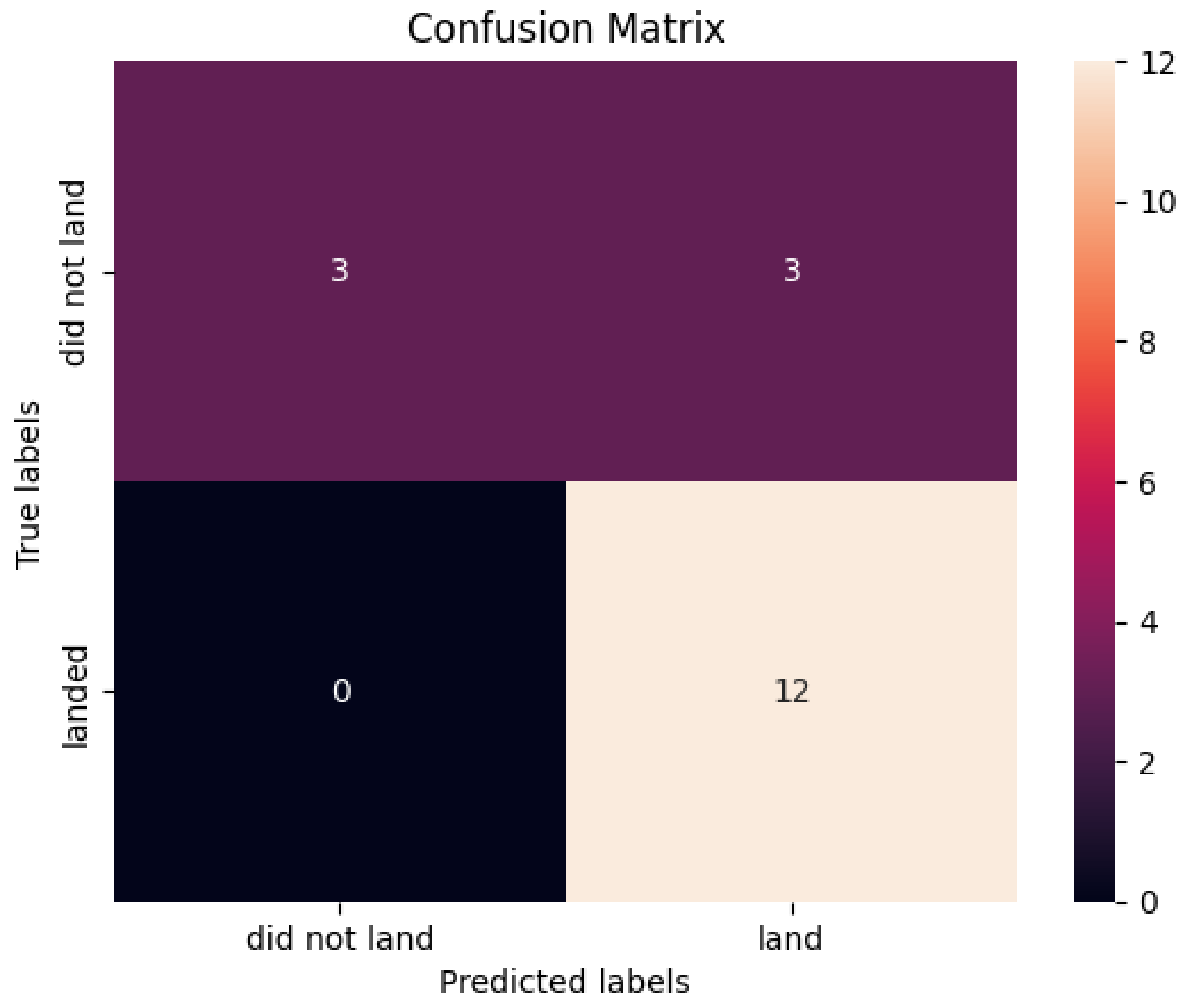
Model Evaluation:

- Compare model performances on test data to determine the best method.

Tools and Libraries Used:

- Pandas, NumPy, Matplotlib, and Seaborn for data manipulation and visualization.
- Sklearn for preprocessing, model selection, and classification algorithms (Logistic Regression, SVM, Decision Trees). Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose
- https://github.com/Alirio-Mieres/spacex-data-science-project/blob/main/08-spacex_Machine%20Learning%20Prediction.ipynb

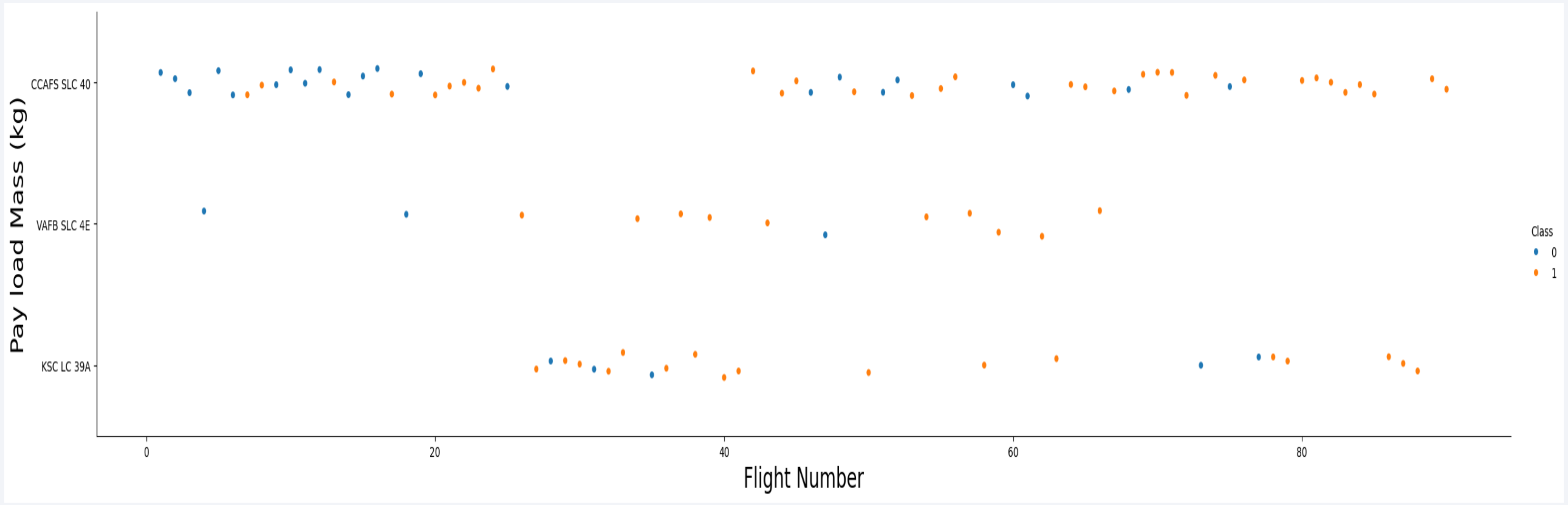
Results


Confusion Matrix
True labels
did not land
landed
Predicted labels
did not land
land
3
3
0
12
12
10
8
6
4
2
0

Section 2

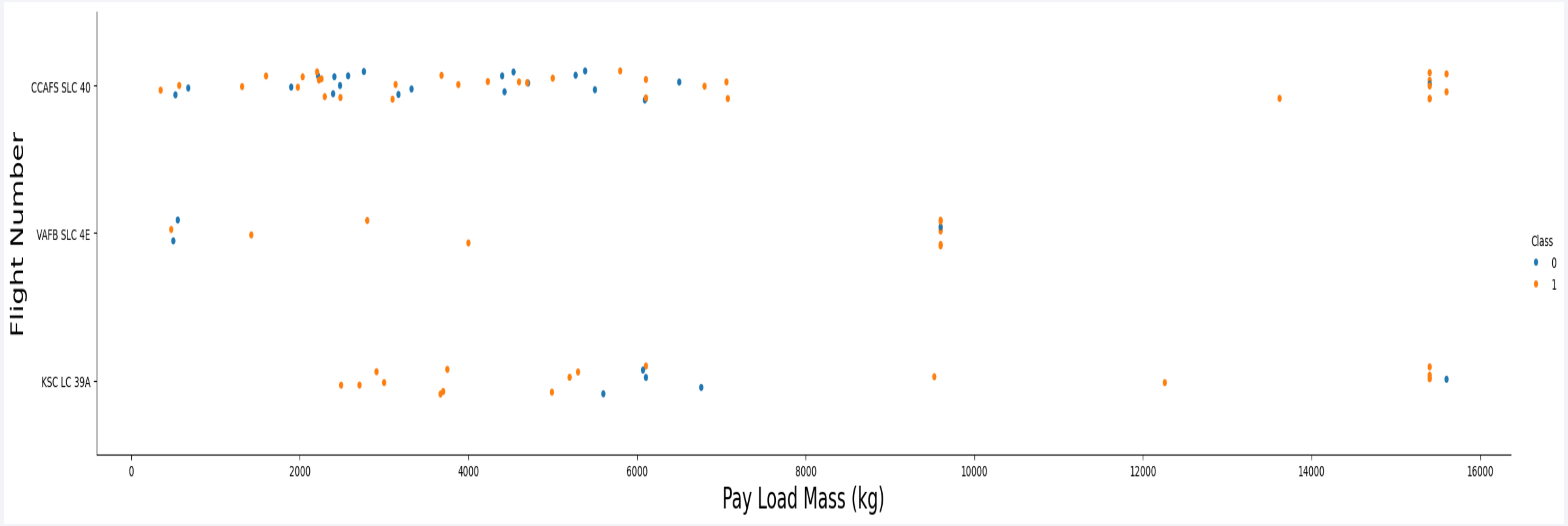
Insights drawn from EDA

Flight Number vs. Launch Site



We found that the larger the flight amount at a launch site, the greater the success rate at a launch site.

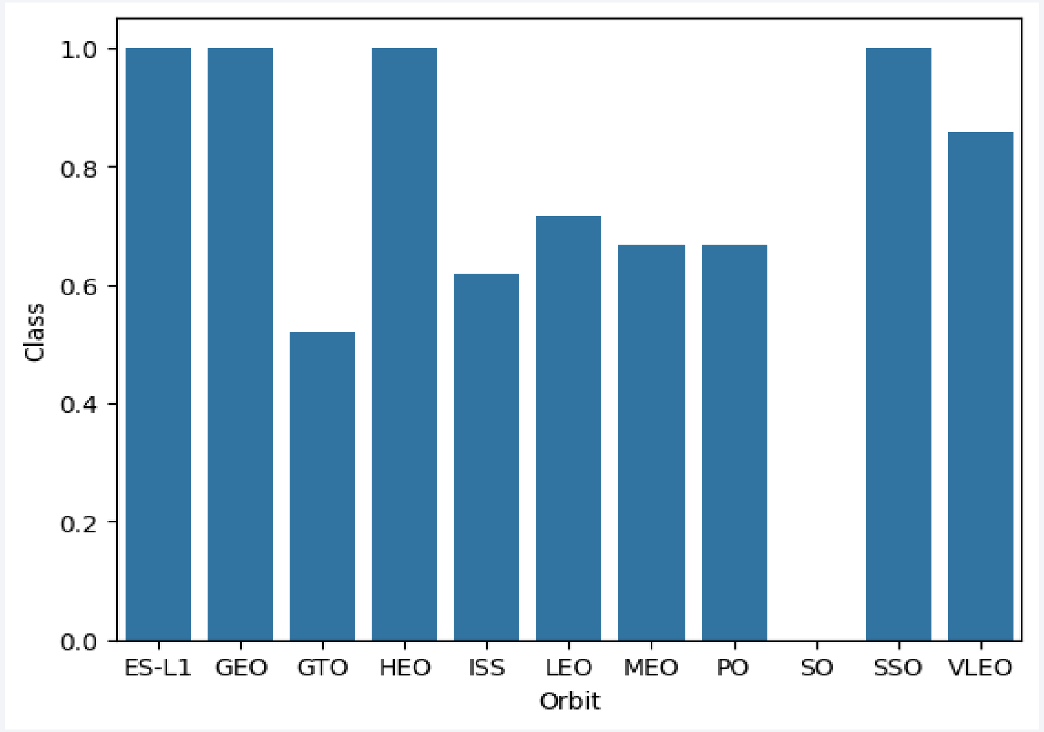
Payload vs. Launch Site



The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.

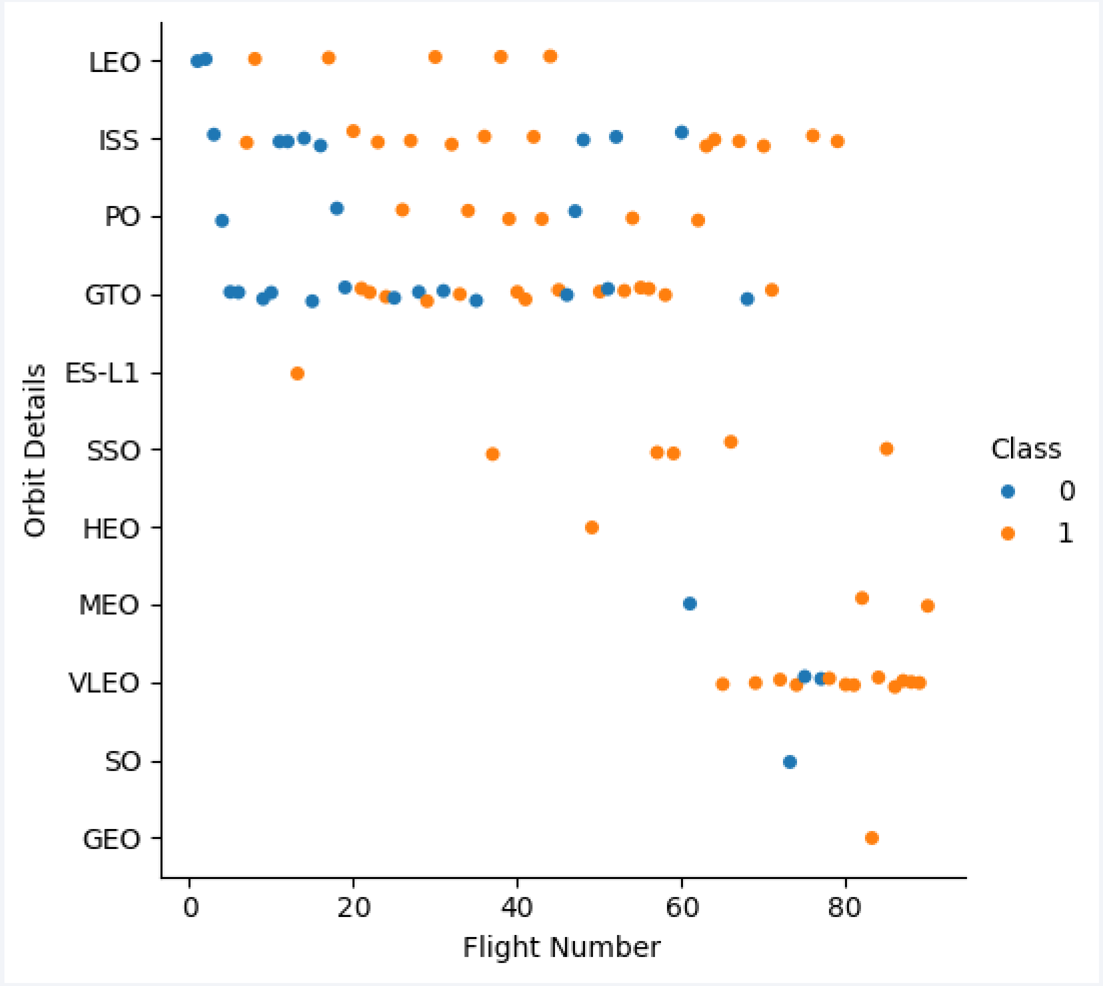
Success Rate vs. Orbit Type

From the plot, we can see that ES-L1, GEO, HEO, SSO, and VLEO had the highest success rate.



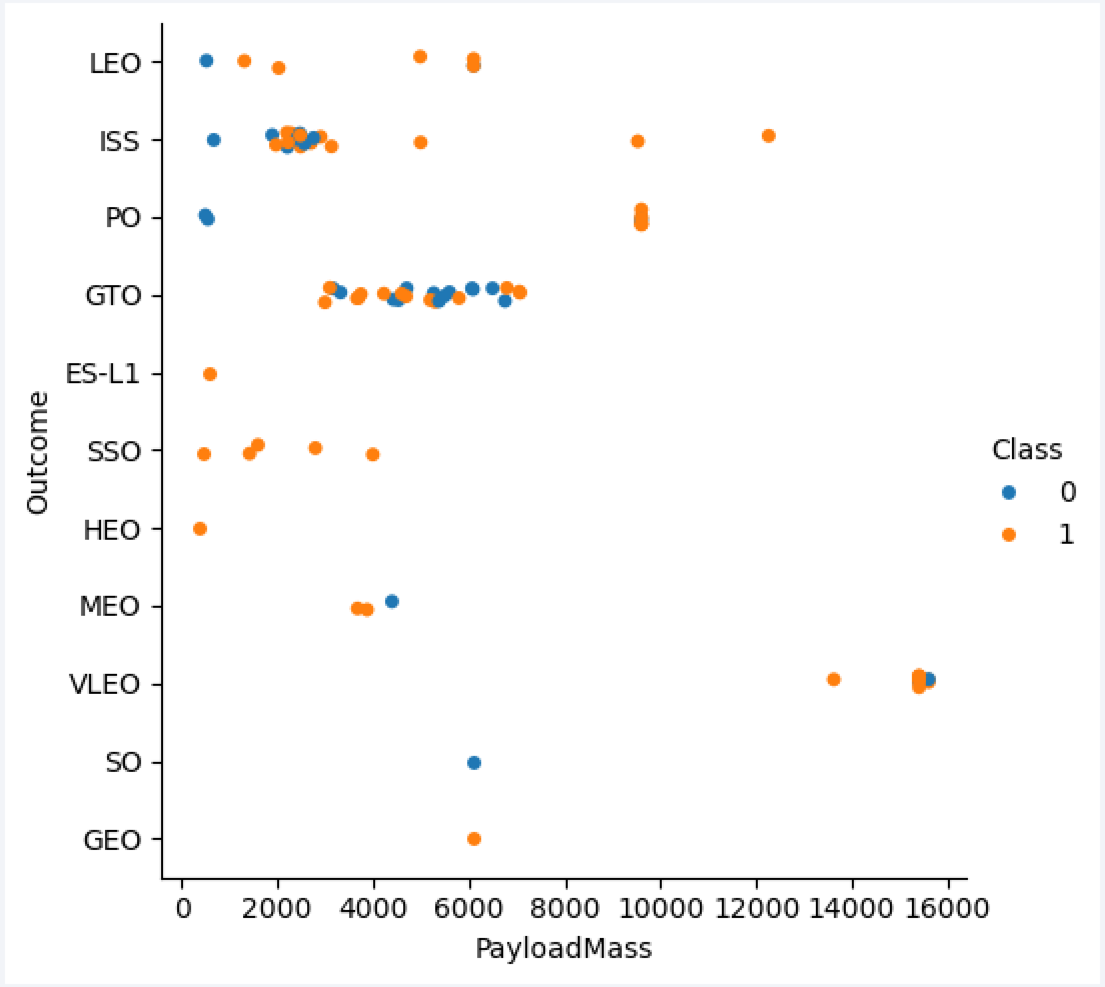
Flight Number vs. Orbit Type

We observe that in the LEO orbit, success is related to the number of flights, while in the GTO orbit, there is no relationship between the number of flights and success in orbit.



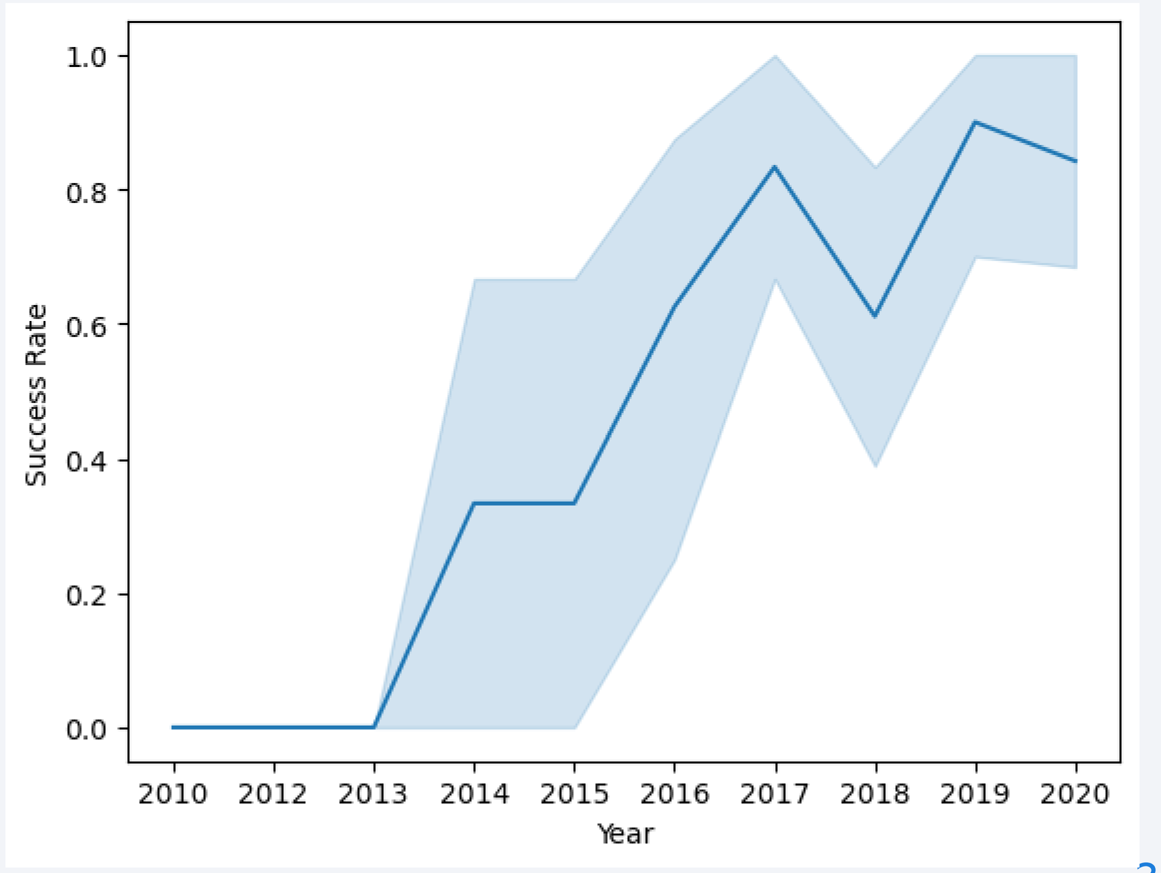
Payload vs. Orbit Type

We can observe that with heavy loads, successful landings are more frequent in PO, LEO, and ISS orbits.



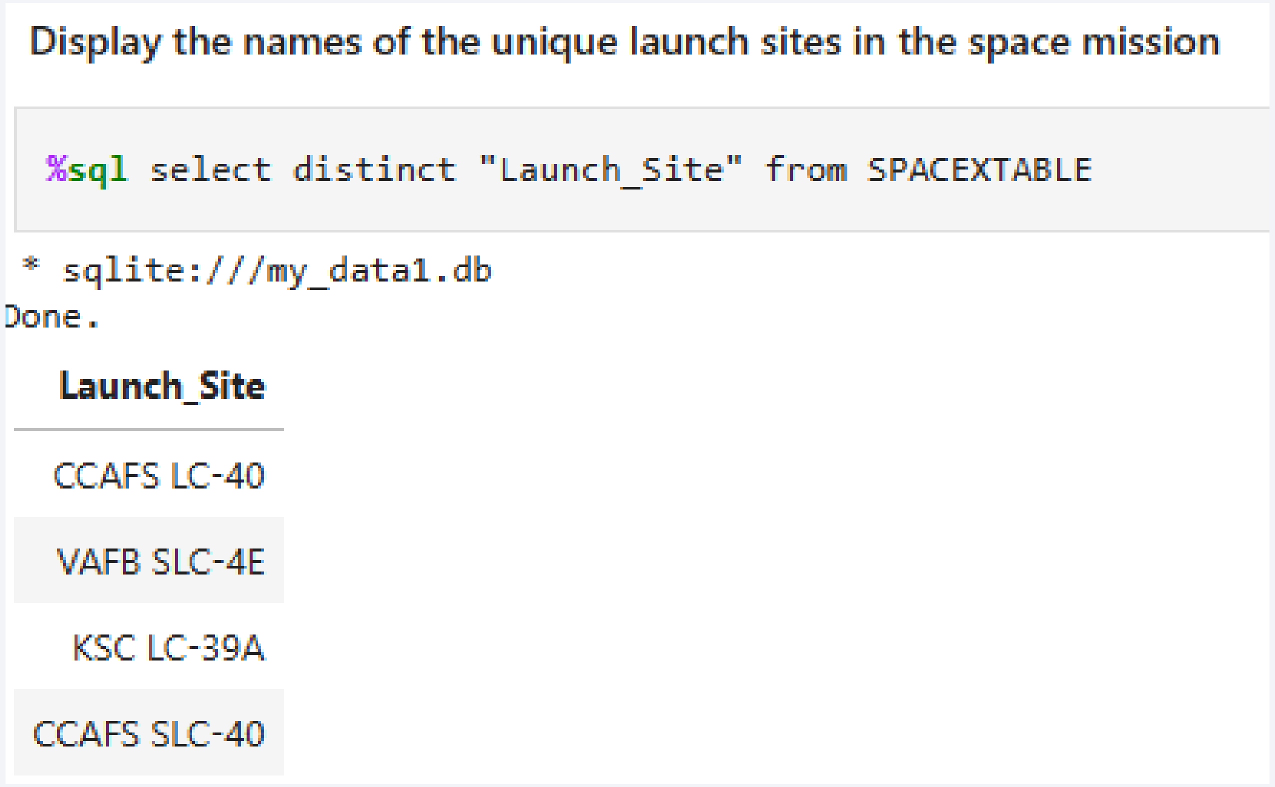
Launch Success Yearly Trend

From the graph, we can see that the success rate has been increasing continuously from 2013 to 2020.



All Launch Site Names

The SQL query select distinct "Launch_Site" from SPACEXTABLE retrieves unique values from the "Launch_Site" column of the SPACEXTABLE. The distinct keyword ensures that only unique values are returned, eliminating any duplicates. This query specifically identifies and lists all unique launch sites recorded in the SPACEXTABLE dataset.



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

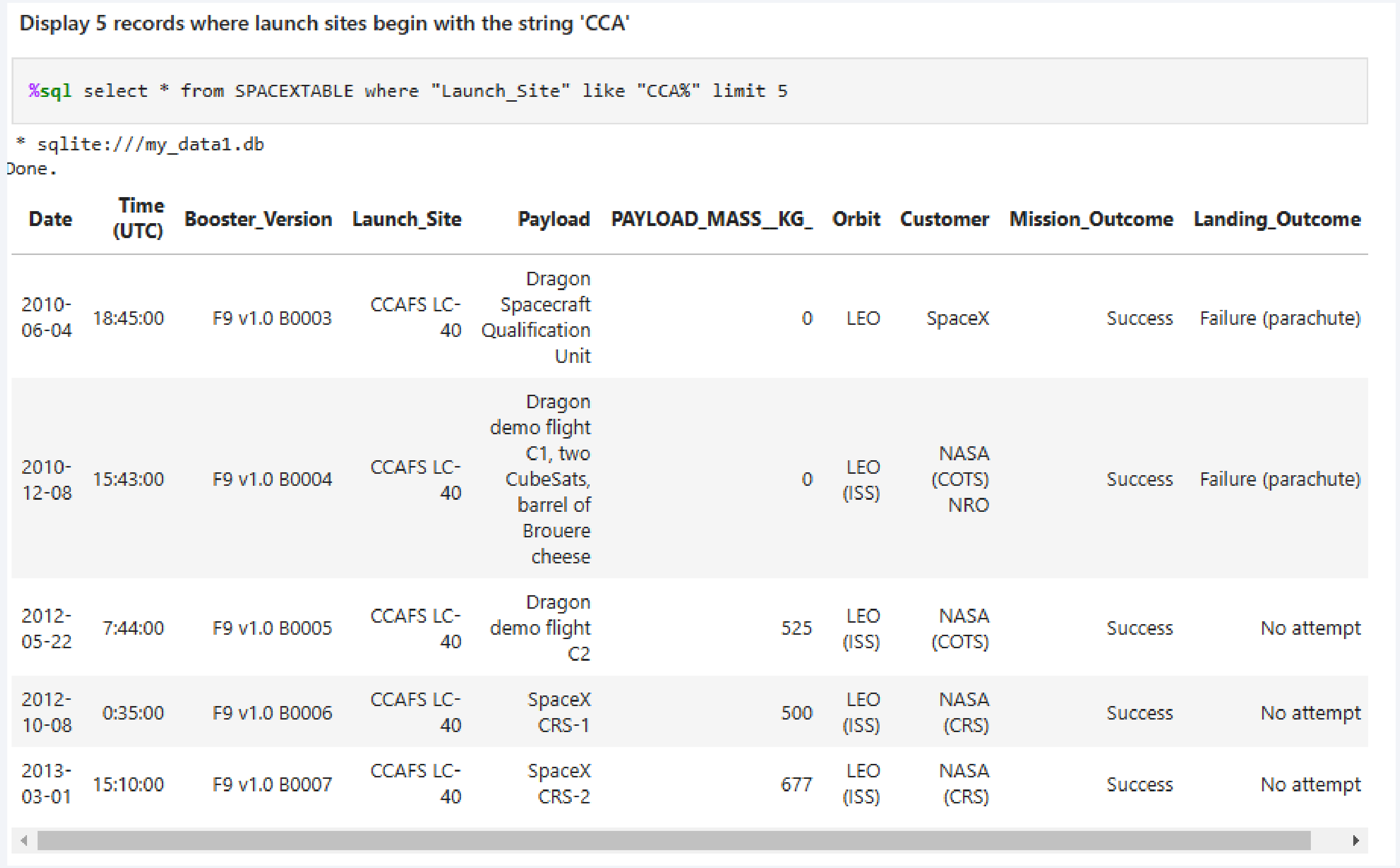
```
%sql select distinct "Launch_Site" from SPACEXTABLE
```

```
 * 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'

This SQL query selects and displays up to 5 rows from the SPACEXTABLE where the "Launch_Site" column starts with "CCA". The % symbol in like "CCA%" acts as a wildcard, matching any sequence of characters that begins with "CCA". This query allows us to quickly retrieve and examine specific records related to launch sites that match this pattern.



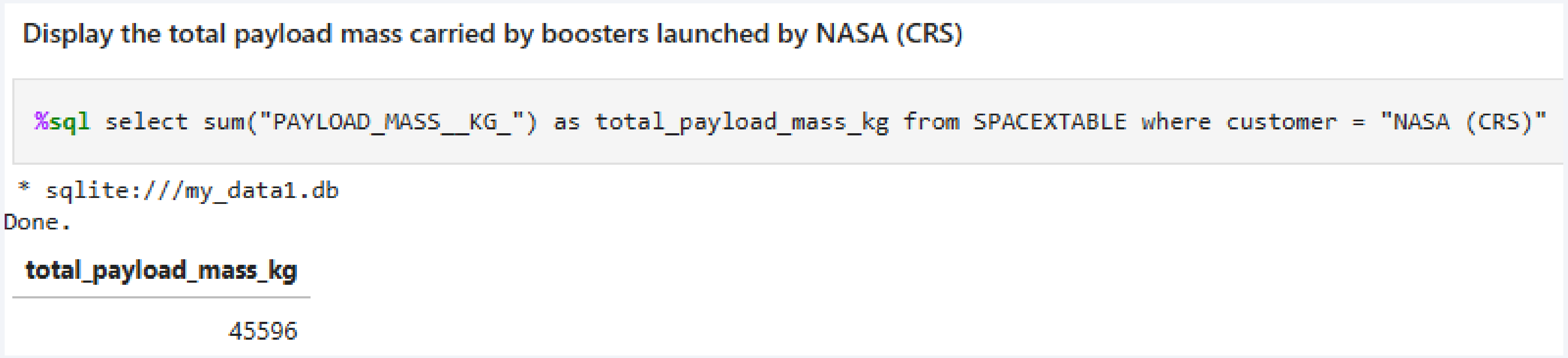
Display 5 records where launch sites begin with the string 'CCA'

```
%sql select * from SPACEXTABLE 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

Total Payload Mass


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

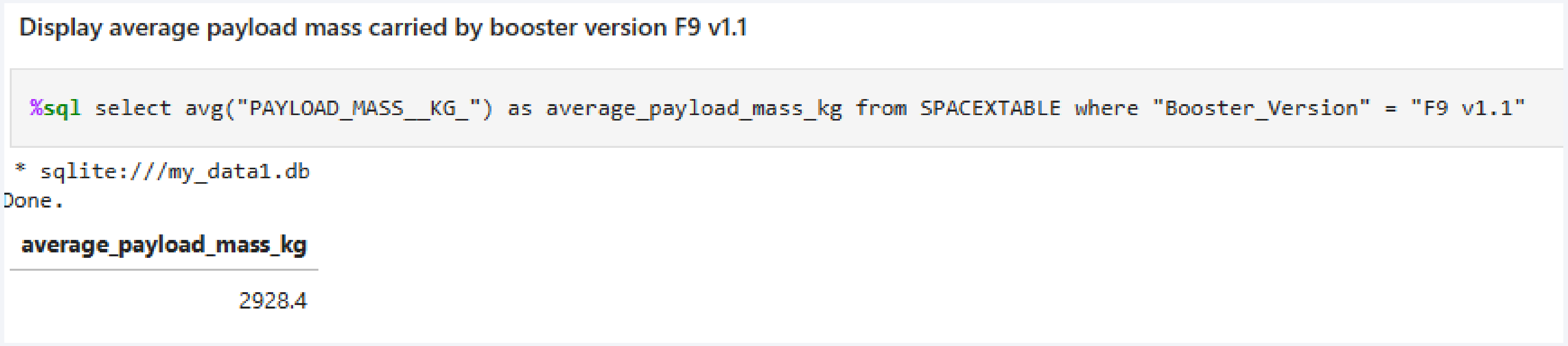
```
%sql select sum("PAYLOAD_MASS__KG_") as total_payload_mass_kg from SPACEXTABLE where customer = "NASA (CRS)"
```

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

total_payload_mass_kg
45596

The query retrieves the total payload mass (in kilograms) for missions where the customer is "NASA (CRS)" from the SPACEXTABLE. It uses sum("PAYLOAD_MASS__KG_") as total_payload_mass_kg to calculate and alias the sum of payload masses under this specific customer category.

Average Payload Mass by F9 v1.1



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

```
%sql select avg("PAYLOAD_MASS__KG_") as average_payload_mass_kg from SPACEXTABLE where "Booster_Version" = "F9 v1.1"
```

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

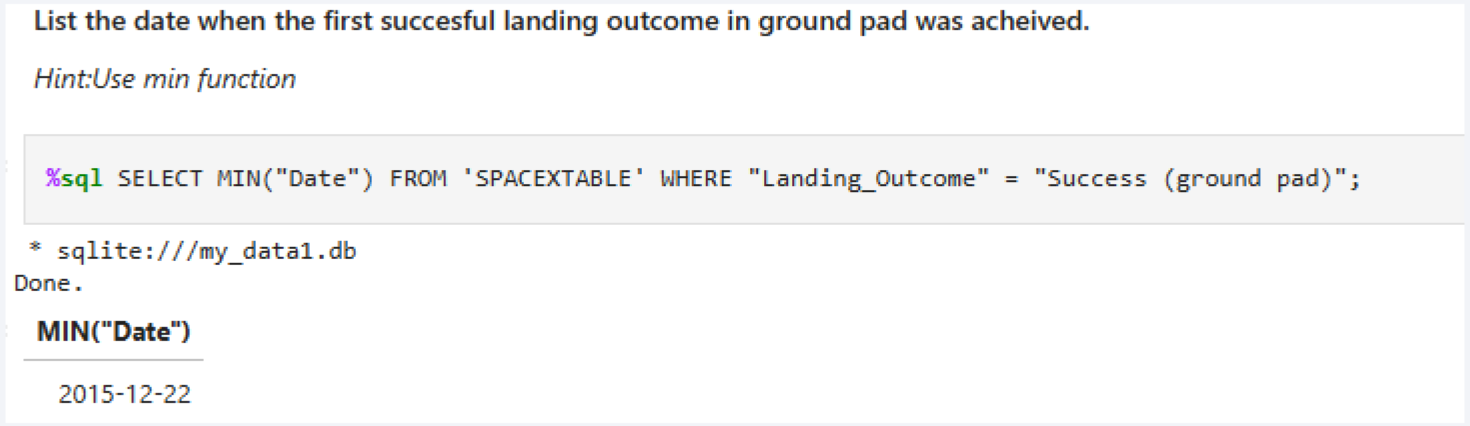
average_payload_mass_kg
2928.4

Calculates the average payload mass (in kilograms) for missions where the booster version is specifically "F9 v1.1". It computes this average using the avg("PAYLOAD_MASS__KG_") function and aliases the result as average_payload_mass_kg.

This query selects the earliest date (MIN("Date")) from the SPACEXTABLE where the landing outcome is specifically "Success (ground pad)".

First Successful Ground Landing Date

This query selects the earliest date (MIN("Date")) from the SPACEXTABLE where the landing outcome is specifically "Success (ground pad)".


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

Hint:Use min function

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

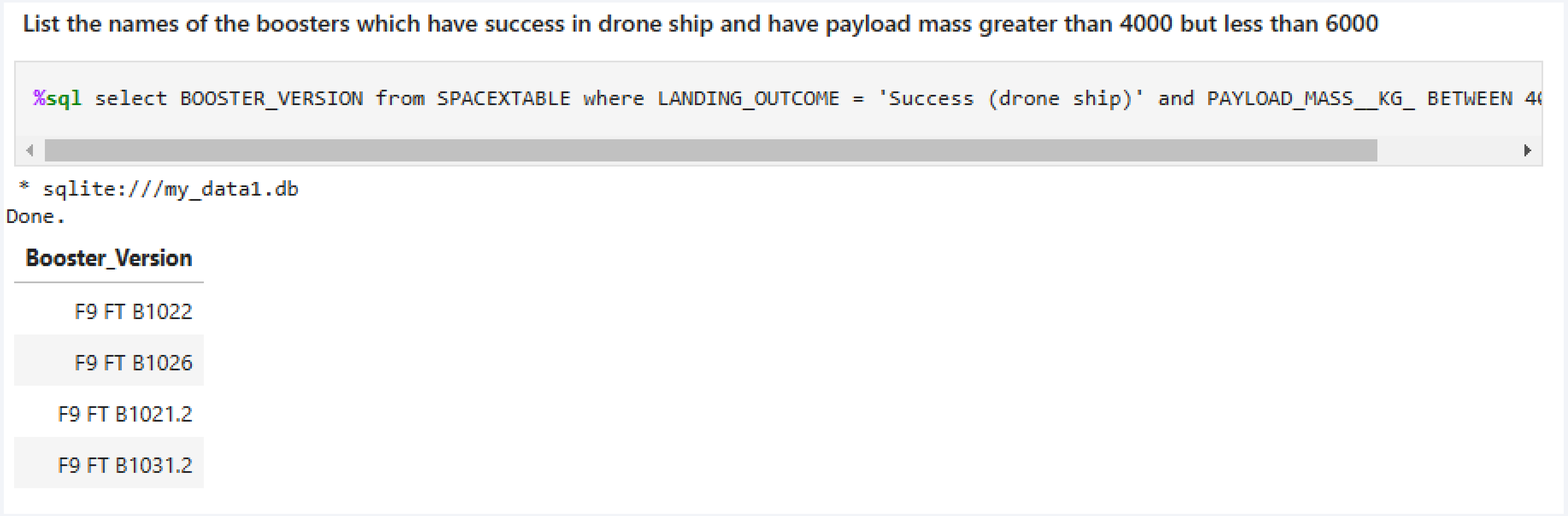
```
 * sqlite:///my_data1.db
Done.
```

MIN("Date")
2015-12-22

This query retrieves the BOOSTER_VERSION from the SPACEXTABLE where the LANDING_OUTCOME is 'Success (drone ship)' and the PAYLOAD_MASS__KG_ falls between 4000 and 6000 kilograms.

Successful Drone Ship Landing with Payload between 4000 and 6000

This query retrieves the BOOSTER_VERSION from the SPACEXTABLE where the LANDING_OUTCOME is 'Success (drone ship)' and the PAYLOAD_MASS__KG_ falls between 4000 and 6000 kilograms.



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

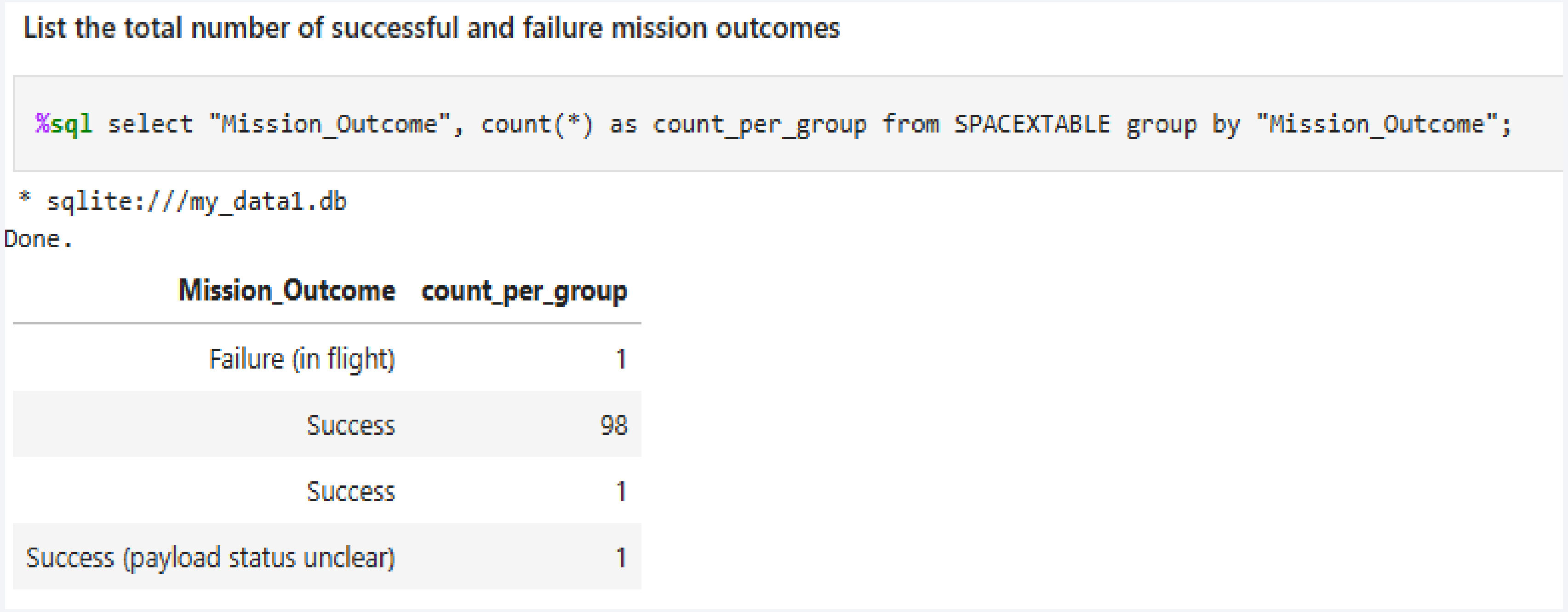
```
%sql select BOOSTER_VERSION from SPACEXTABLE where LANDING_OUTCOME = 'Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 40
```

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

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

Total Number of Successful and Failure Mission Outcomes

This query calculates the count of records (count(*)) grouped by the values in the "Mission_Outcome" column (GROUP BY "Mission_Outcome"). It summarizes how many records belong to each unique mission outcome category in the SPACEXTABLE dataset.



List the total number of successful and failure mission outcomes

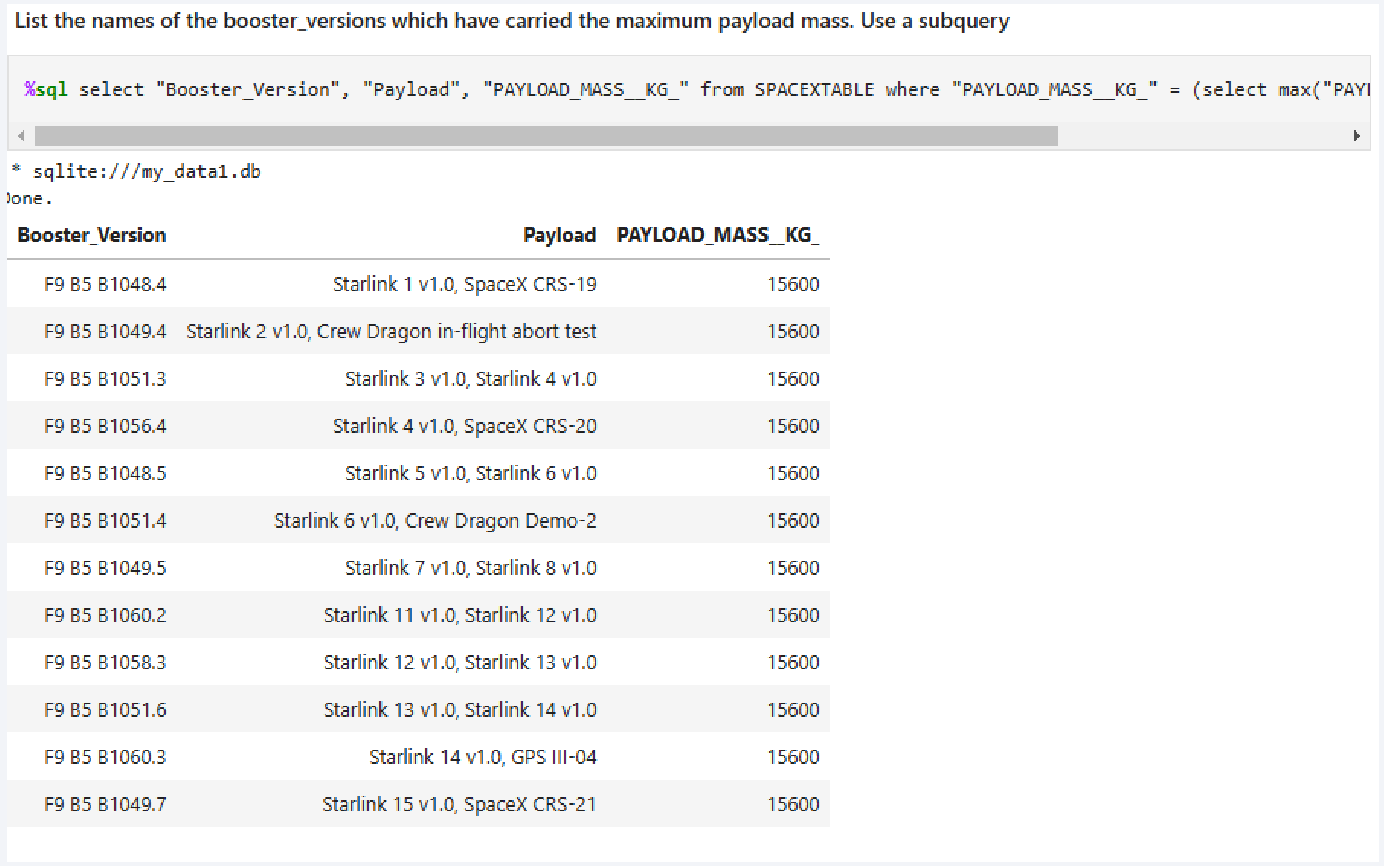
```
%sql select "Mission_Outcome", count(*) as count_per_group from SPACEXTABLE group by "Mission_Outcome";
```

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

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

Boosters Carried Maximum Payload

This query retrieves the Booster_Version, Payload, and PAYLOAD_MASS__KG_ from the SPACEXTABLE where the payload mass is equal to the maximum payload mass in the table. The subquery (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTABLE) identifies the maximum payload mass, and the main query filters the records to find those matching this maximum value.



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

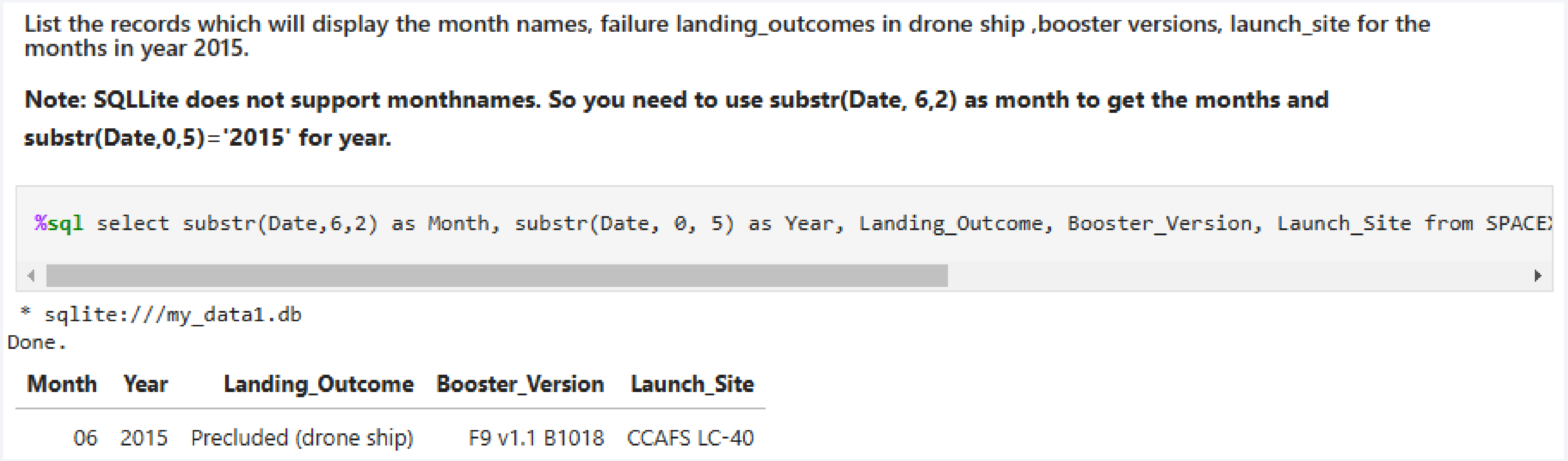
```
%sql select "Booster_Version", "Payload", "PAYLOAD_MASS__KG_" from SPACEXTABLE where "PAYLOAD_MASS__KG_" = (select max("PAYI
```

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

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

2015 Launch Records

This query extracts the month and year from the Date column, filters the results to include only records from the year 2015 where the mission outcome starts with "Failure", and selects the Landing_Outcome, Booster_Version, and Launch_Site. The substr(Date,6,2) function extracts the month, and substr(Date, 0, 5) extracts the year from the Date column.



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

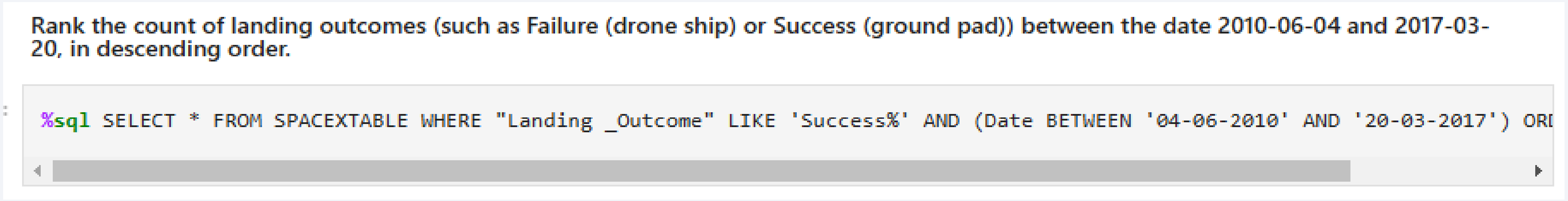
Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%sql select substr(Date,6,2) as Month, substr(Date, 0, 5) as Year, Landing_Outcome, Booster_Version, Launch_Site from SPACEX
```

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

Month	Year	Landing_Outcome	Booster_Version	Launch_Site
06	2015	Precluded (drone ship)	F9 v1.1 B1018	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.

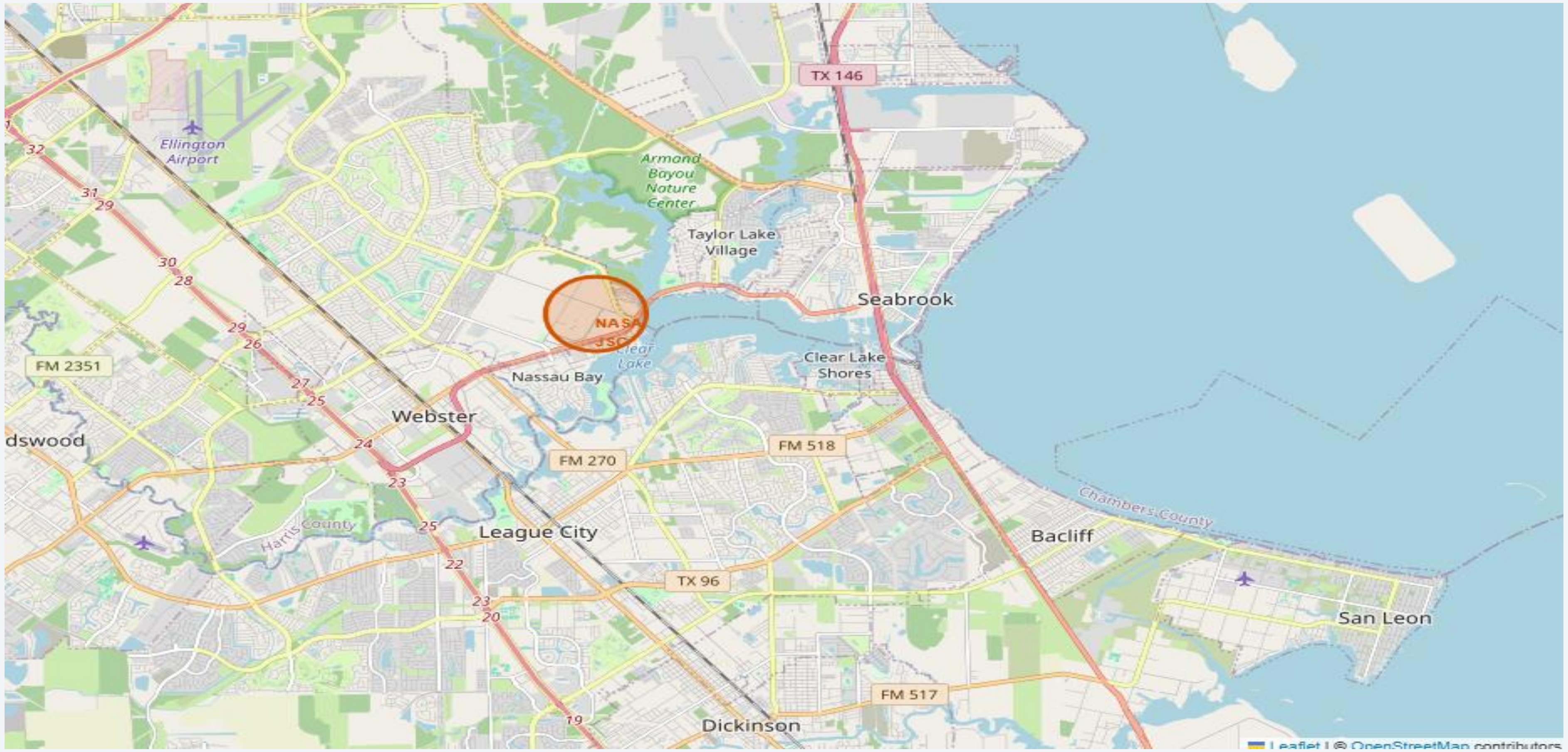
```
%sql SELECT * FROM SPACEXTABLE WHERE "Landing _Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017') ORI
```

This query selects all columns from the SPACEXTABLE where the Landing_Outcome starts with 'Success' and the Date is between '2010-06-04' and '2017-03-20', inclusive. The results are ordered by Date in descending order.

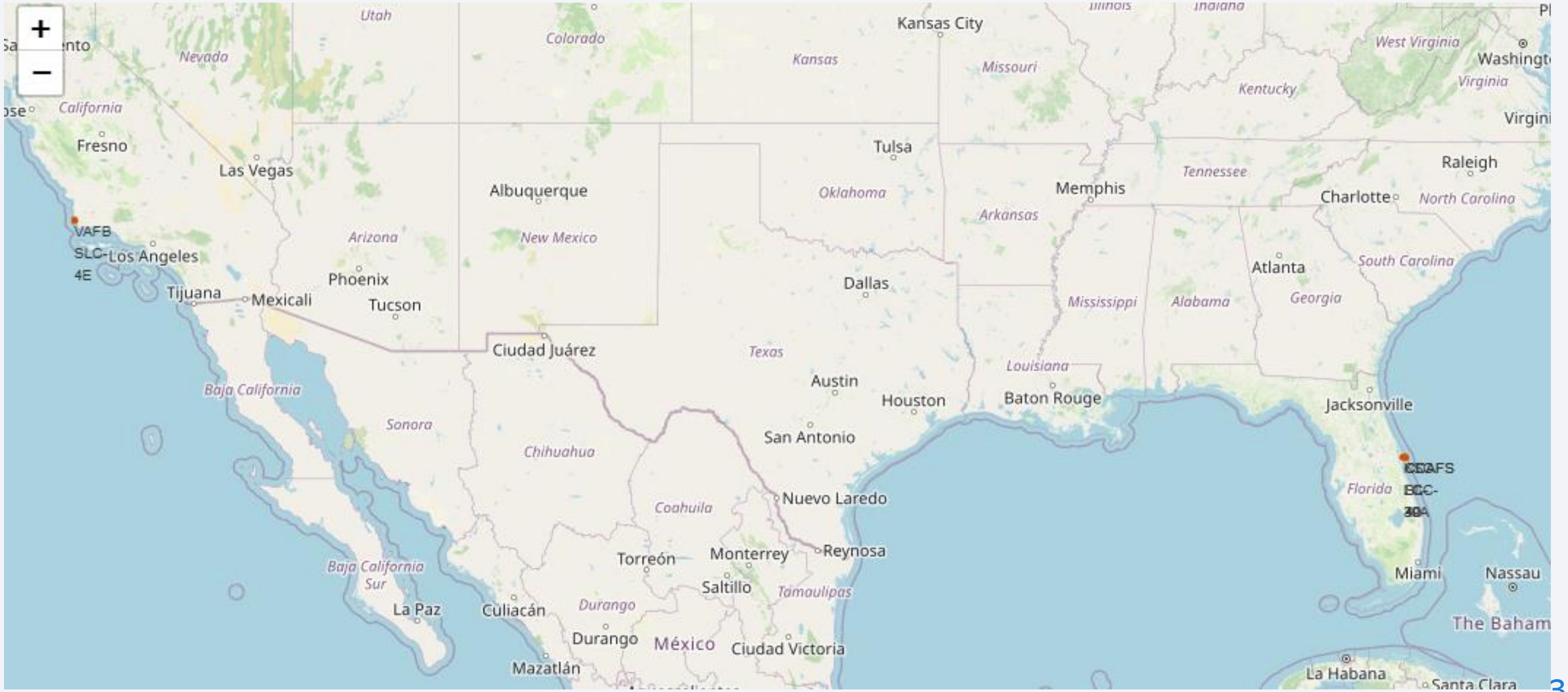
Section 3

Launch Sites Proximities Analysis

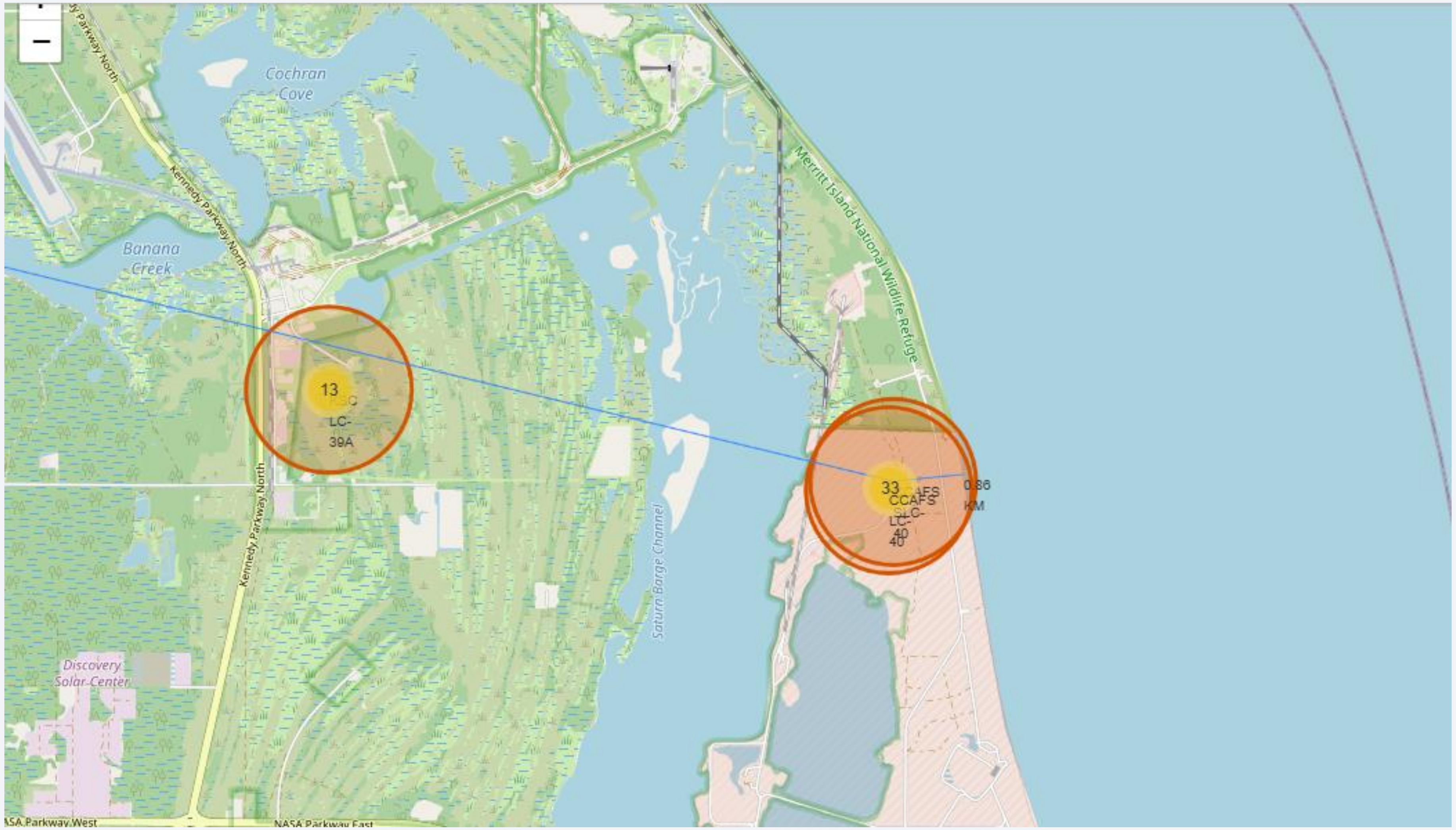
Mark all launch sites on a map

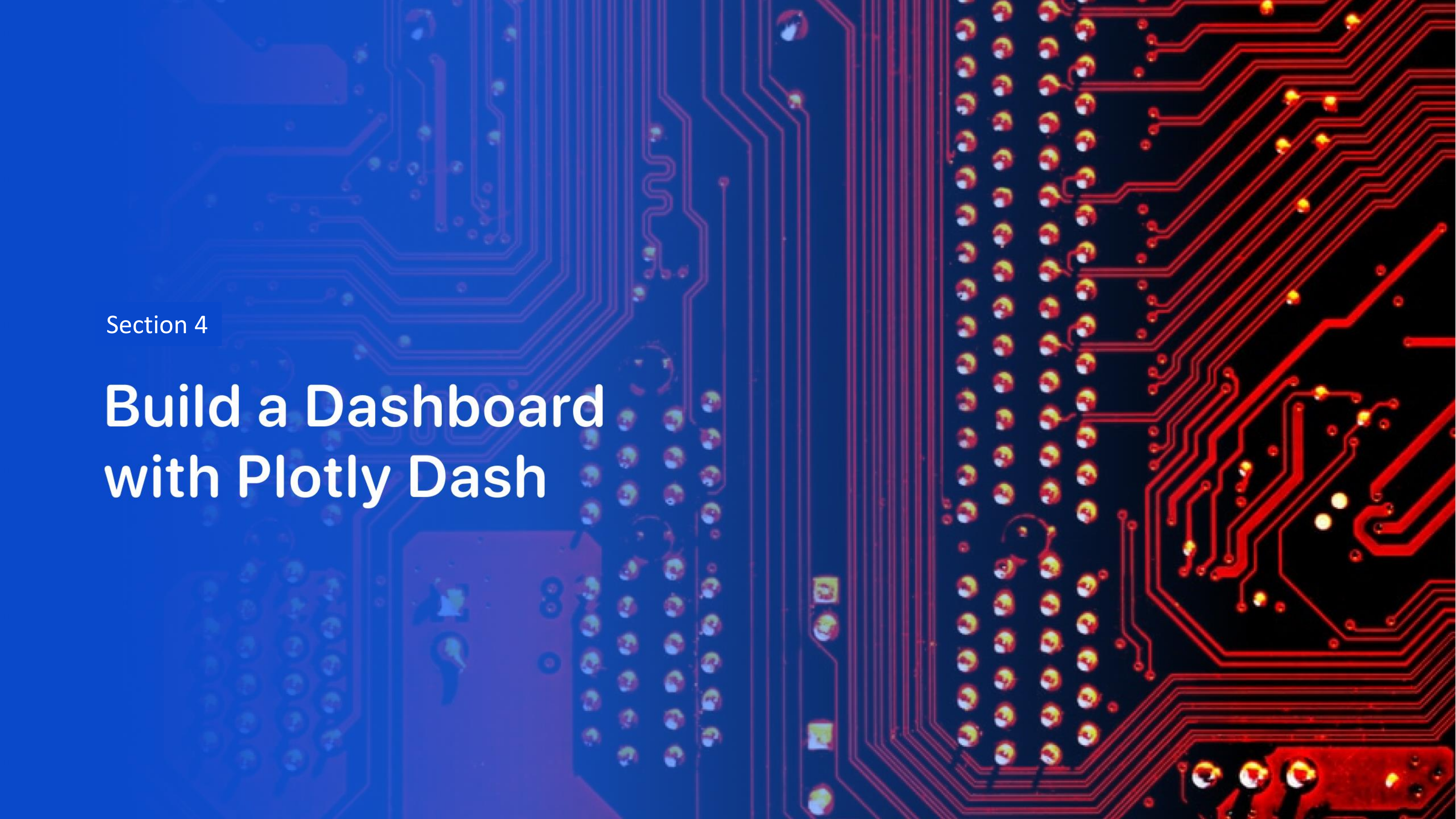


Mark the success/failed launches for each site on the map



Calculate the distances between a launch site to its proximities

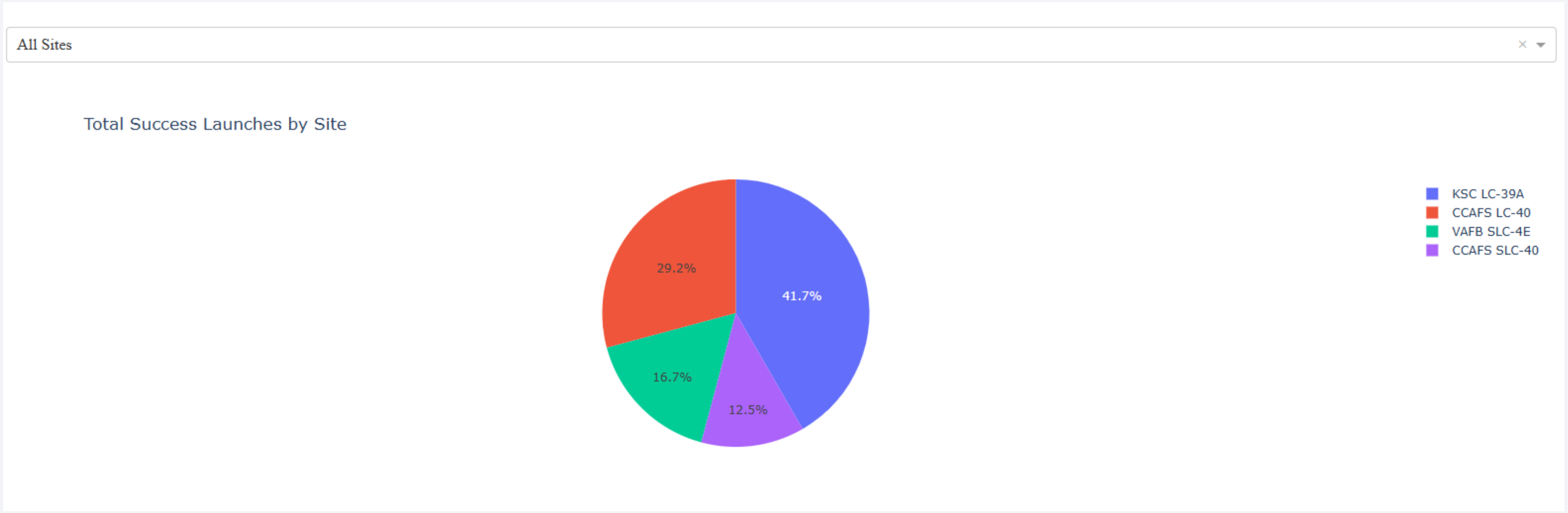




Section 4

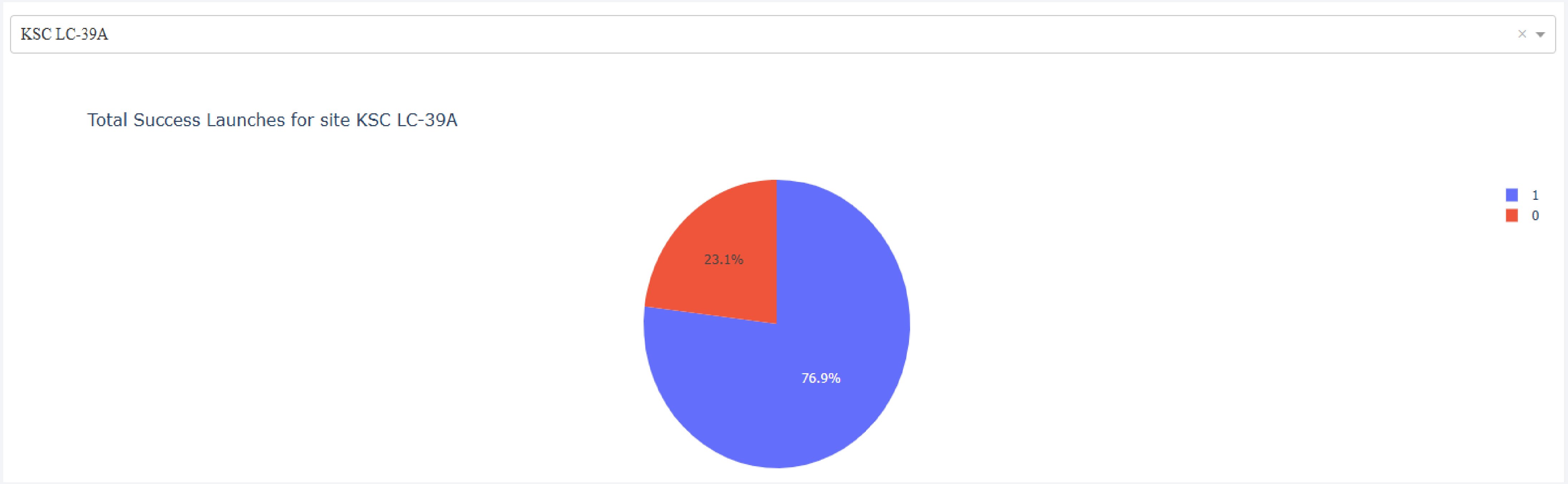
Build a Dashboard with Plotly Dash

SpaceX Launch Records Dashboard



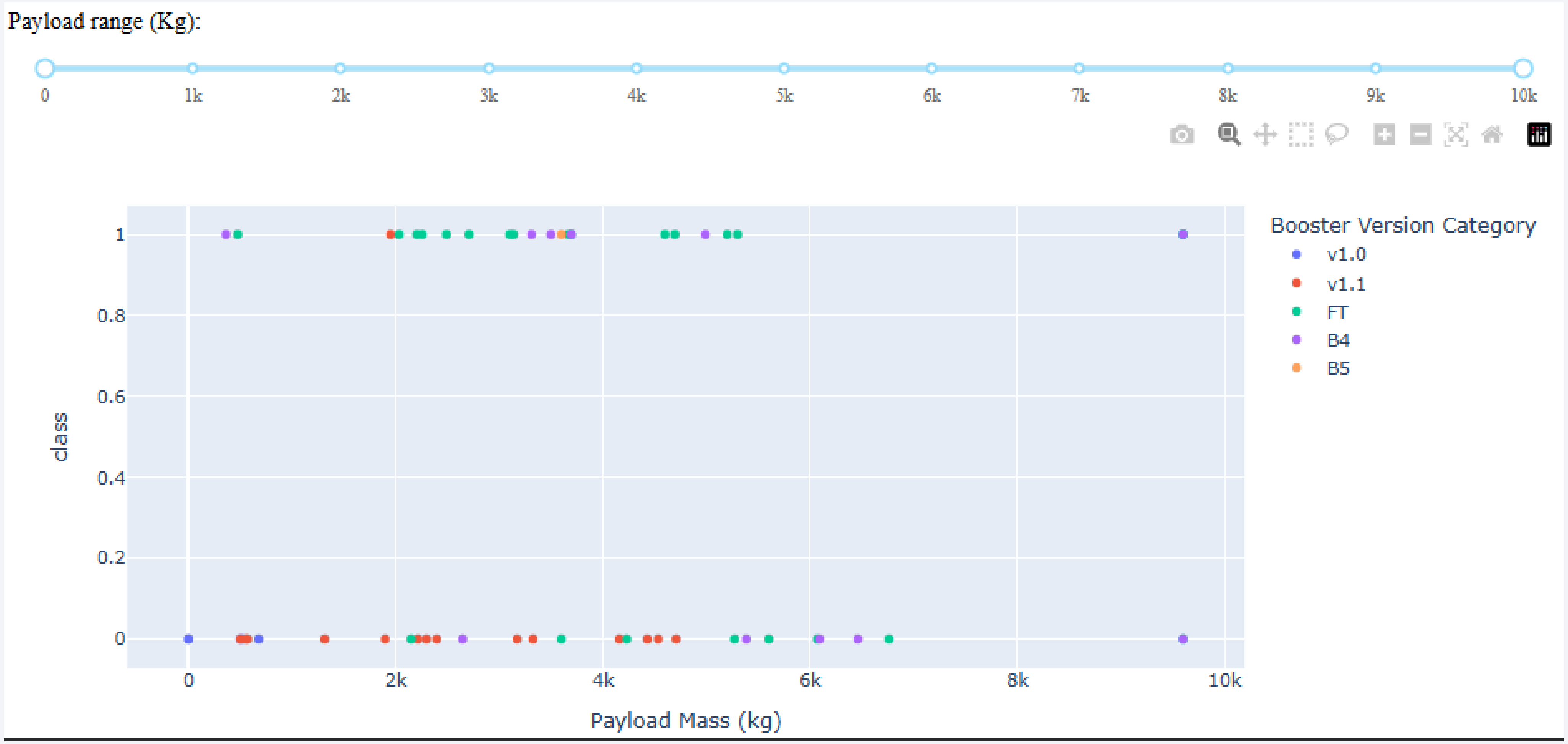
In comparison of all the sites KSC LC-39A is the most successful.

Most successful site


KSC LC-39A

The site with the highest success rate is KSC LC-39A.

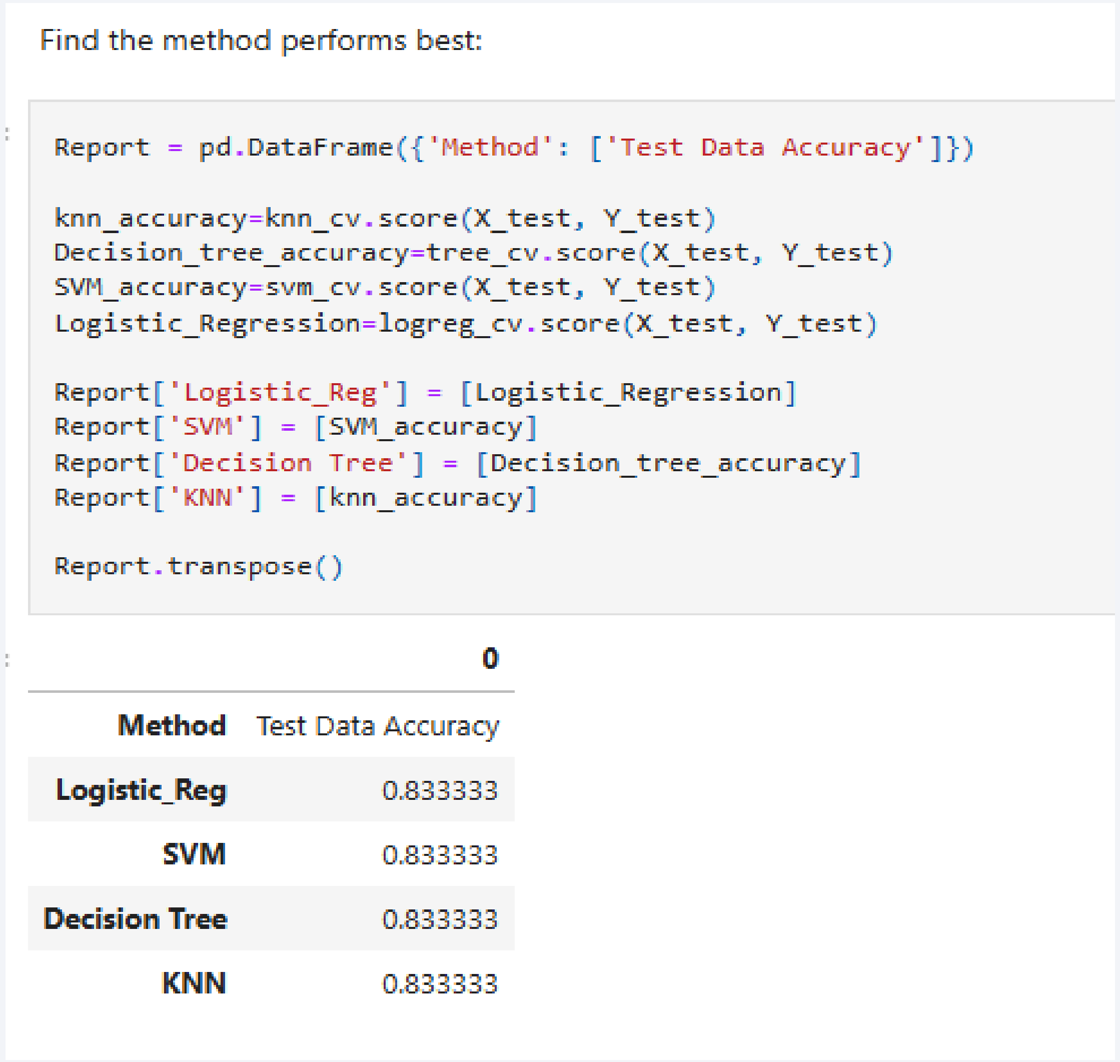
Booster version category and payload mass



Section 5

Predictive Analysis (Classification)

Classification Accuracy



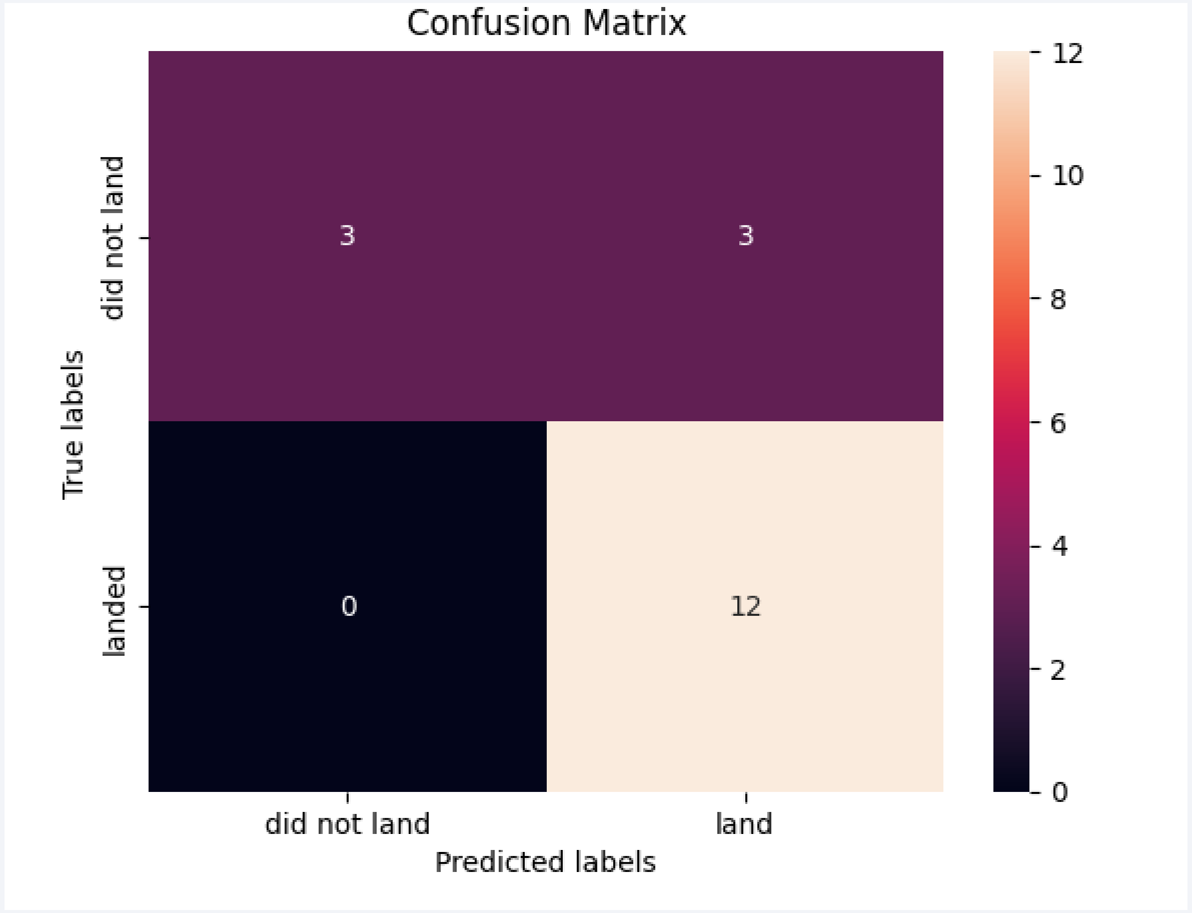
Find the method performs best:

```
Report = pd.DataFrame({'Method': ['Test Data Accuracy']})

knn_accuracy=knn_cv.score(X_test, Y_test)
Decision_tree_accuracy=tree_cv.score(X_test, Y_test)
SVM_accuracy=svm_cv.score(X_test, Y_test)
Logistic_Regression=logreg_cv.score(X_test, Y_test)

Report['Logistic_Reg'] = [Logistic_Regression]
Report['SVM'] = [SVM_accuracy]
Report['Decision Tree'] = [Decision_tree_accuracy]
Report['KNN'] = [knn_accuracy]

Report.transpose()
```

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

Confusion Matrix

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The main problem is false positives, that is, unsuccessful landings that the classifier marks as successful.



Conclusions

- The launch success rate began to increase from 2013 to 2020.
- ES-L1, GEO, HEO, SSO, and VLEO orbits had the highest success rates.
- KSC LC-39A had the highest number of successful launches among all sites.
- The decision tree classifier is the best machine-learning algorithm for this task.

Thank you!