IBM Developer
SKILLS NETWORK

IBM Data Science Capstone Project

Space X Falcon 9 Landing Analysis

GitHub link: https://github.com/saudsayyad/Applied_Data_Science_Capstone.git

Table of contents

Executive Summary

Summary of Methodologies:

This project follows thee steps:

- Data Collection
- Data Wrangling
- Exploratory Data Analysis
- Interactive Visual Analytics
- Predictive Analysis (Classification)

Summary of Results:

This project produced the following outputs and visualizations:

- Exploratory Data Analysis (EDA) results
- Geospatial Analytics
- Interactive Dashboard
- Predictive Analysis of classification models

Introduction

- SpaceX launches Falcon 9 rocket at a cost of 62 million dollars. This is considerably cheaper than other providers (which usually cost upwards of $165m), and much of the savings are because SpaceX can land, and then re-use the first stage of the rocket.

- If we can make predictions on whether the first stage will land ,we can determine the cost of a launch. And use this information to asses whether or not an alternate company should bid and SpaceX for a rocket launch.

- This project will ultimately predict if the SpaceX Falcon 9 first stage will land successfully.

Methodology Summary

1. **Data Collection**
- Making GET requests to the SpaceX REST API
- Web Scrapping

2. **Data Wrangling**
- Using the *.fillna()* method to remove *NaN* values
- Using the *.value_counts()* method to determine number of occurrences of values.
- Creating a landing outcome label that shows 0 on failed landings and 1 on successful landings.

3. **Exploratory Data Analysis**
- Using SQL queries to manipulate and evaluate the SpaceX dataset
- Using Pandas and Matplotlib to visualize relationship between variables, and determine patterns.

4. **Interactive Visual Analytics**
- Geospatial Analytics using Folium
- Creating an interactive dashboard using Plotly Dash.

5. **Data Modelling and Evaluation**
- Using Scikit-Learn to:
  - Standardize the data
  - Split the data into training and testing data using *train_test_split*
  - Train different classification models
  - Find Hyperparameters using *GridSearchCV*
- Plotting confusion matrices for each classification model.
- Assessing the accuracy of each classification model.

Data Collection

The data was collected using various methods

- Data collection was done using get request to the SpaceX API.
- Next, we decoded the response content as a Json using .json() function call and turn it into a pandas DataFrame using .json_normalize().
- We then cleaned the data, checked for missing values and fill in missing values where necessary.
- In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
- The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX Rest API

Using the SpaceX API to retrieve data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.

- Make a GET response to the SpaceX REST API
- Convert the response to a .json file then to a Pandas DataFrame

- Use custom logic to clean the data (see Appendix)
- Define lists for data to be stored in
- Call custom functions (see Appendix) to retrieve data and fill the lists
- Use these lists as values in a dictionary and construct the dataset

- Create a Pandas DataFrame from the constructed dictionary dataset

- Filter the DataFrame to only include Falcon 9 launches
- Reset the FlightNumber column
- Replace missing values of PayloadMass with the mean PayloadMass value

Data Collection – Web Scraping

Web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches. Data collection was done using get request to the SpaceX API.

- Request the HTML page from the static URL
- Assign the response to an object

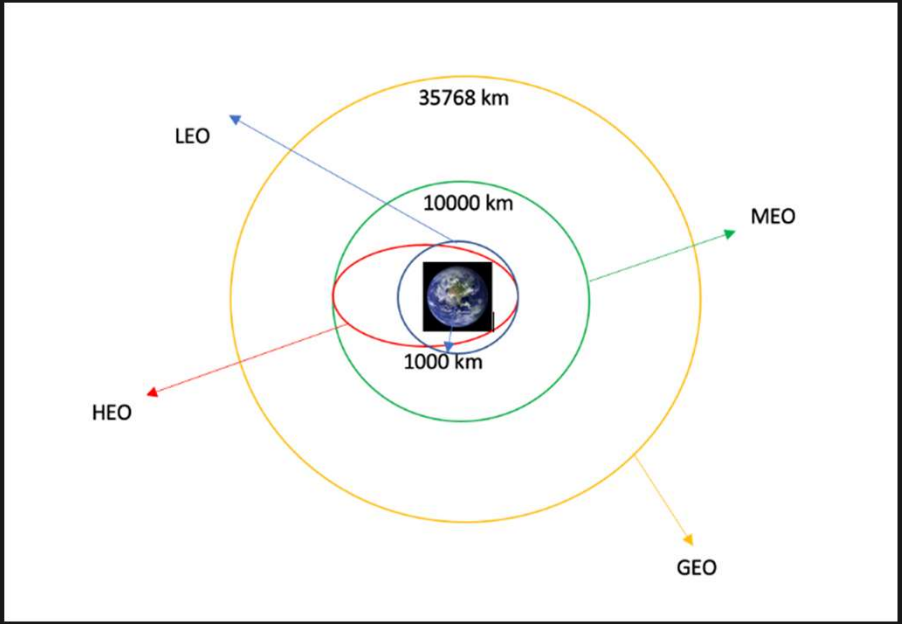
- Create a BeatifulSoup object from the response object
- Find all tables within the HTML page

- Collect all column headers from the table.

- Use the column names as keys in the dictionary

- Convert the dictionary to a Pandas DataFrame and export csv file created from DataFrame.

Data Wrangling



- The SpaceX dataset contains several launch facilities.
- Each launch aims to a dedicated orbit, and some of the common orbit types are shown in the figure below. The orbit type is in the Orbit column.
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits.
- We created landing outcome label from outcome column and exported the result to csv.

EDA with Data Visualization

Scatter Plot

Scatter Plot were produced to visualize the relationship between:

- Flight Number and Launch Site
- Payload and Launch Site
- Orbit type and Flight Number
- Payload and Orbit Type

Bar Chart

Bar Charts were produced to visualize the relationship between:

- Success Rate and Orbit Type

❖ Bar Charts are used to compare numerical value to a categorical variable.

Line Chart

Line charts were produced to visualize the relationship between:

- Success Rate and Year (Launch Success Year Trend)

❖ Line charts contain numerical values on both axes and generally used to show the change of a variable over time.

EDA with SQL

The SQL queries performed on the data set were used to:

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string ‘CCA’
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display the average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome on a ground pad was achieved
6. List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg
7. List the total number of successful and failed mission outcomes
8. List the names of the booster versions which have carried the maximum payload mass
9. List the failed landing outcomes on drone ships, their booster versions, and launch site names for 2015
10. 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

Build an Interactive Map with Folium

The following steps were taken to visualize the launch data on an interactive map:

1) Mark all launch sites on a map

- Initialize the map using a Folium Map object
- Add a folium.Circle and folium.Marker for each launch site on the launch map

2) Mark the success/failed launches for each site on a map

- As many launches have the same coordinates, it makes sense to cluster them together.
- Before clustering them, assign a marker colour of successful (class = 1) as green, and failed (class = 0) as red.
- To put the launches into clusters, for each launch, add a folium.Marker to the MarkerCluster()object.
- Create an icon as a text label, assigning the icon_color as the marker_colour determined previously.

Build a Dashboard with Plotly Dash

The following plots were added to a Plotly Dash dashboard to have an interactive visualization of the data:

1. Pie chart showing the total successful launches per site

- This makes it clear to see which sites are most successful
- The chart could also be filtered to see the success/failure ratio for an individual site.

2) Scatter graph to show the correlation between outcome (success or not) and payload mass (kg)

- This could be filtered by ranges of payload masses
- It could also be filtered by booster version

Predictive Analysis (Classification)

Model Development

- Load dataset

1) Perform necessary data transformations (standardise and pre-process)
2) Split data into training and test data sets, using train_test_split()
3) Decide which type of machine learning algorithms are most appropriate
4) Create a GridSearchCV object and a dictionary of parameters
5) Fit the object to the parameters
6) Use the training data set to train the model

Model Evaluation

- For each chosen algorithm:

1) Check the tuned hyperparameters (best_params_)
2) Check the accuracy (score and best_score_)
3) Plot and examine the Confusion Matrix

Results

Exploratory Data Analysis

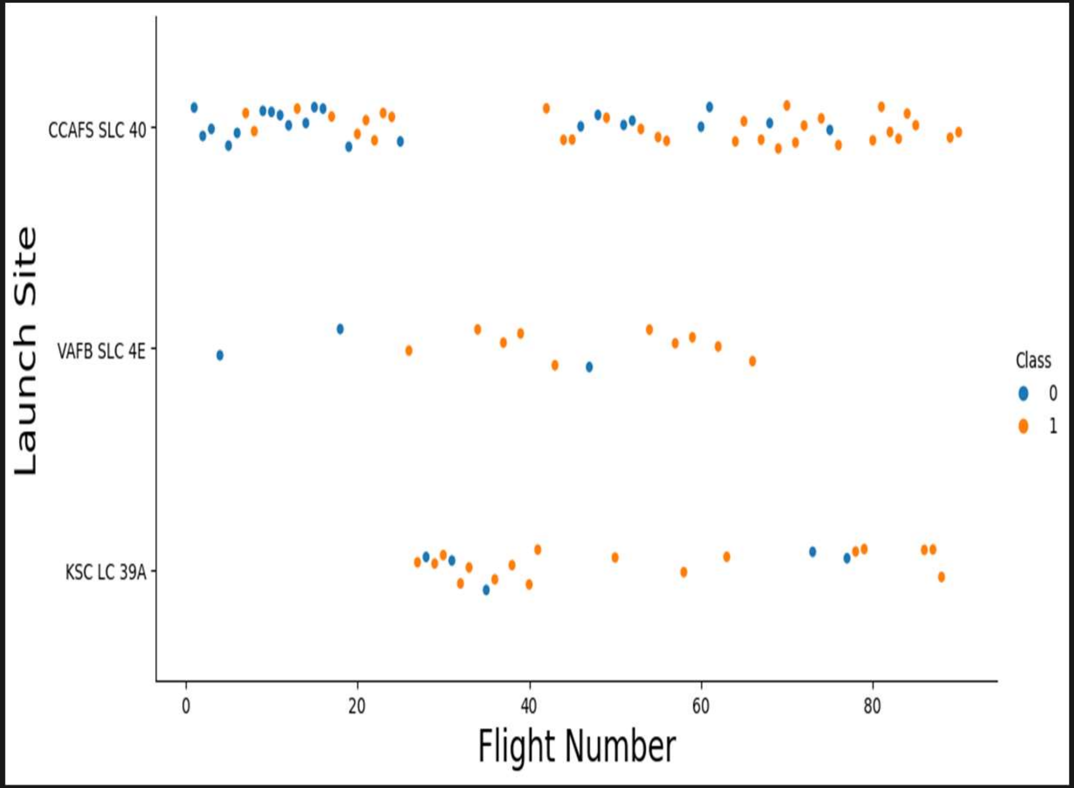
Interactive Analytics

Predictive Analysis

EDA – With Visualization

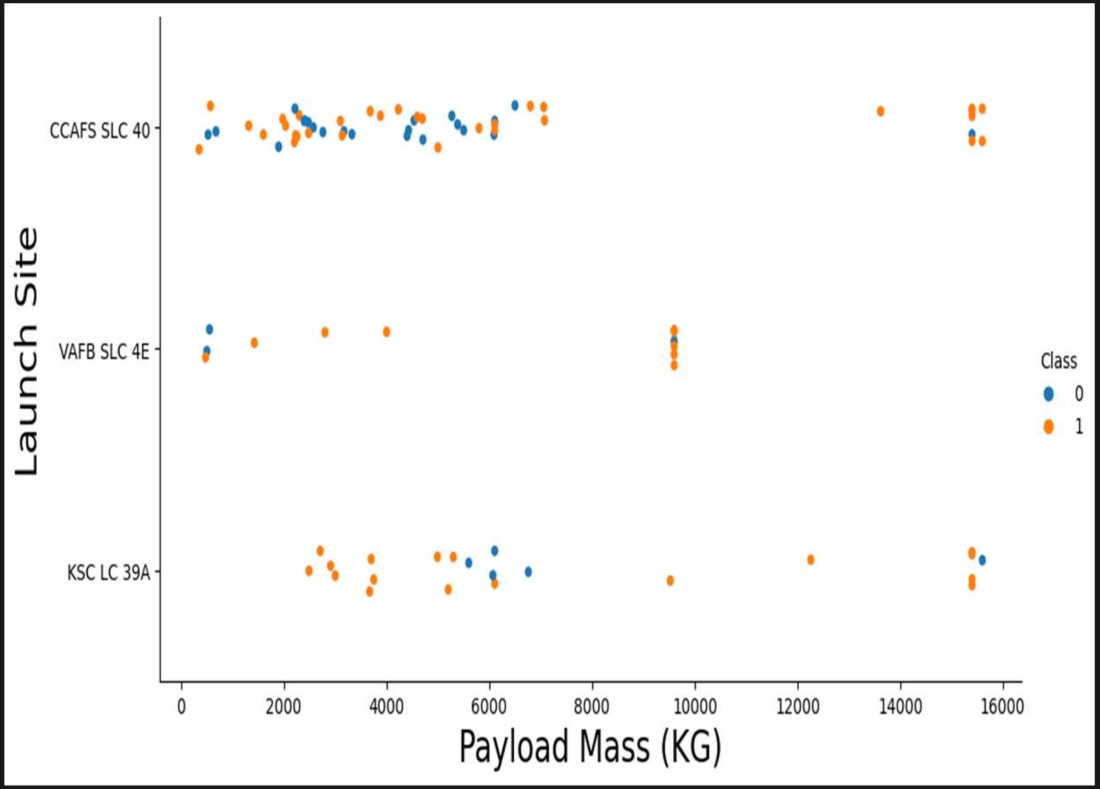
Flight Number vs. Launch Site

- As the number of flights increases, the rate of success at a launch site increases.
- Most of the early flights (flight numbers < 30) were launched from CCAFS SLC 40, and were generally unsuccessful.
- The flights from VAFB SLC 4E also show this trend, that earlier flights were less successful.
- No early flights were launched from KSC LC 39A, so the launches from this site are more successful.
- Above a flight number of around 30, there are significantly more successful landings (Class = 1).



Payload vs. Launch Site

- Above the payload mass of around 7000 kg, there are very few unsuccessful landings, but there is also far less data for these heavier launches.
- There is no clear correlation between payload mass and success rate for a given launch site.
- All sites launched a variety of payload masses, with most of the launches from CCAFS SLC 40 being comparatively lighter payloads (with some outliers).



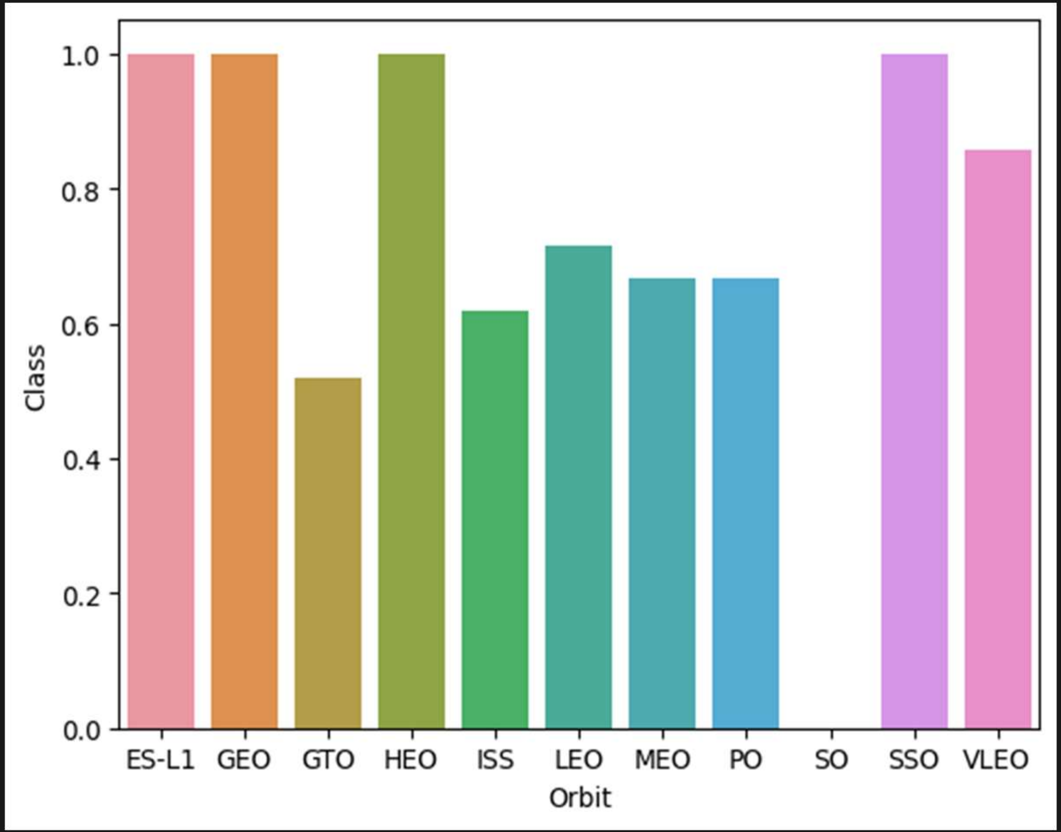
Success Rate vs. Orbit Type

The bar chart of Success Rate vs. Orbit Type shows that the following orbits have the highest (100%) success rate:

- ES-L1(Earth-Sun First Lagrangian Point)
- GEO (Geostationary Orbit)
- HEO (High Earth Orbit)
- SSO (Sun-synchronous Orbit)

The orbit with the lowest (0%) success rate is:

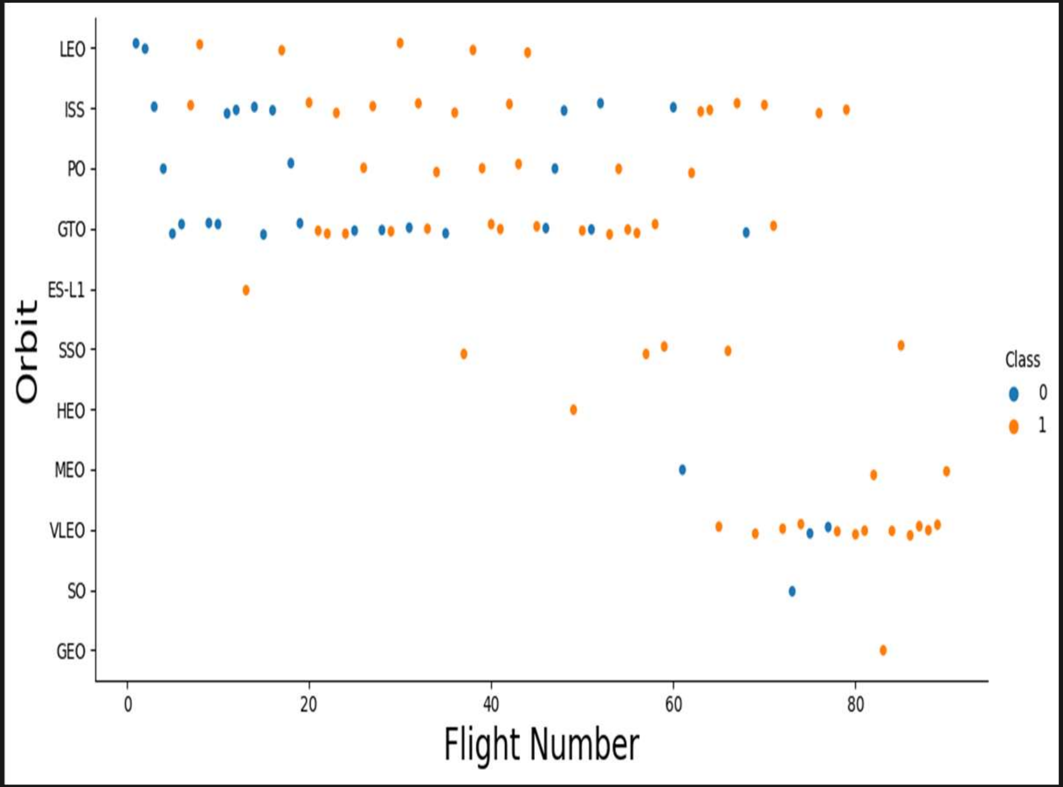
- SO (Heliocentric Orbit)



Flight Number vs. Orbit Type

This scatter plot of Orbit Type vs. Flight number shows a few useful things that the previous plots did not, such as:

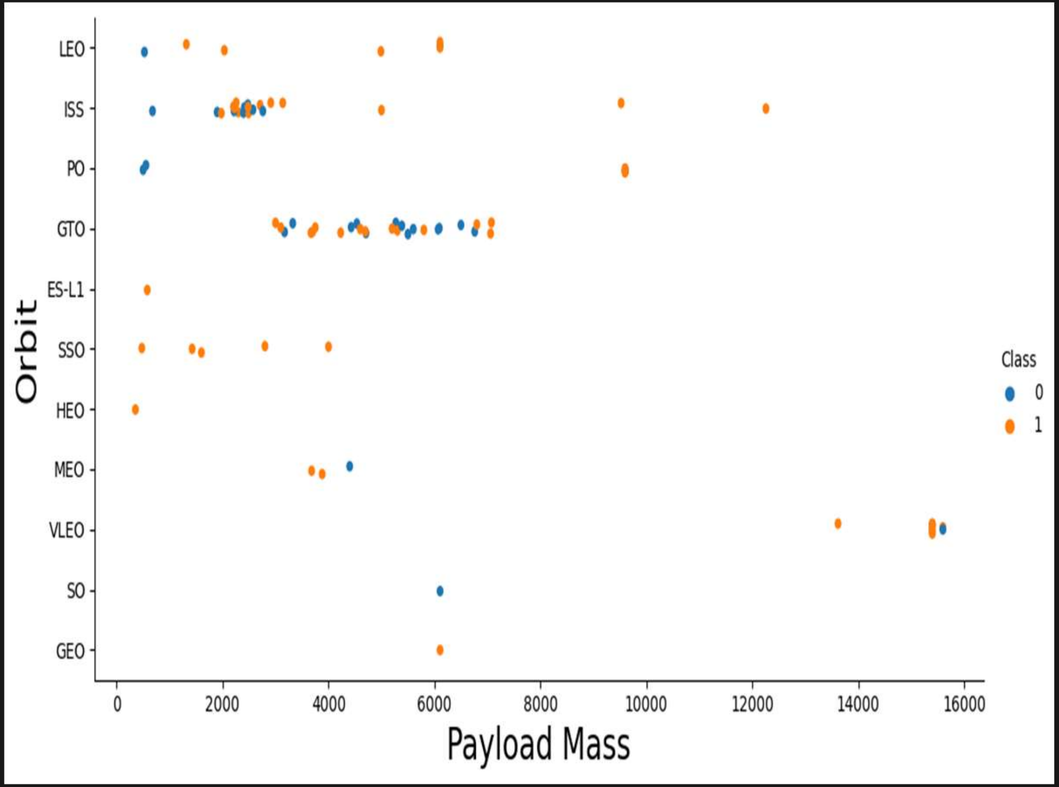
- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
- The 100% success rate in SSO is more impressive, with 5 successful flights.
- Generally, as Flight Number increases, the success rate increases. This is most extreme for LEO, where unsuccessful landings only occurred for the low flight numbers (early launches).



Payload vs. Orbit Type

This scatter plot of Orbit Type vs. Payload Mass shows that:

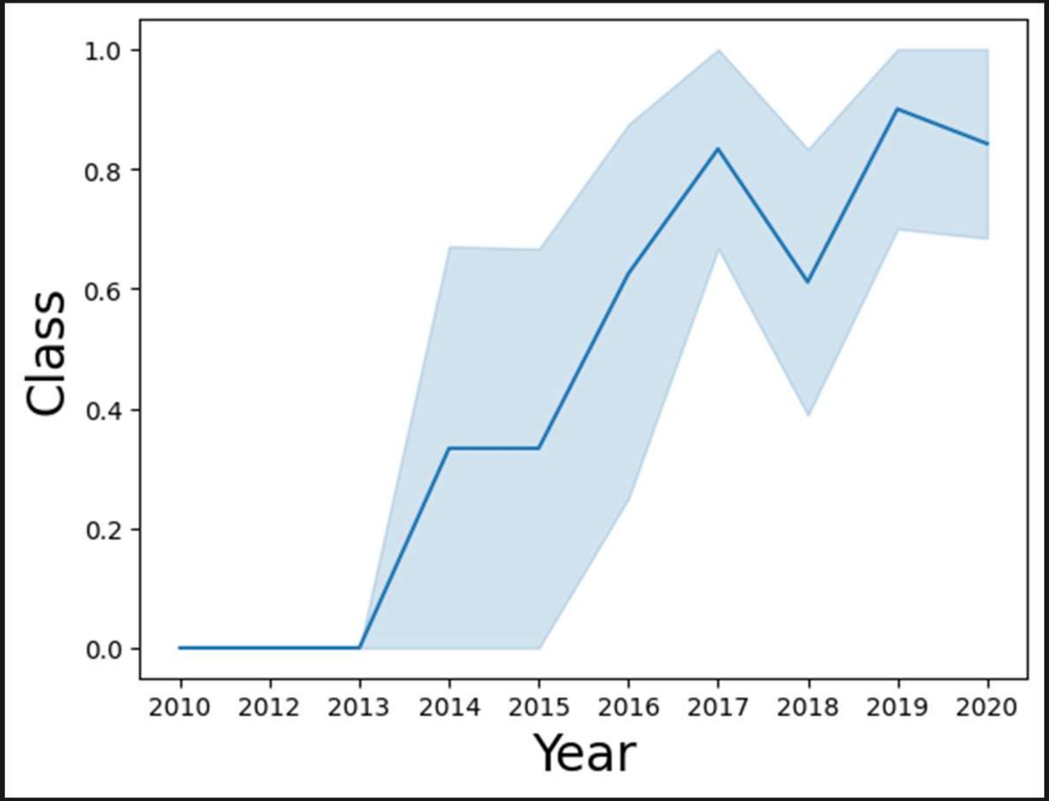
- The following orbit types have more success with heavy payloads:
    - PO
    - ISS
    - LEO
- For GTO, the relationship between payload mass and success rate is unclear.
- VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.



Launch Success Yearly Trend

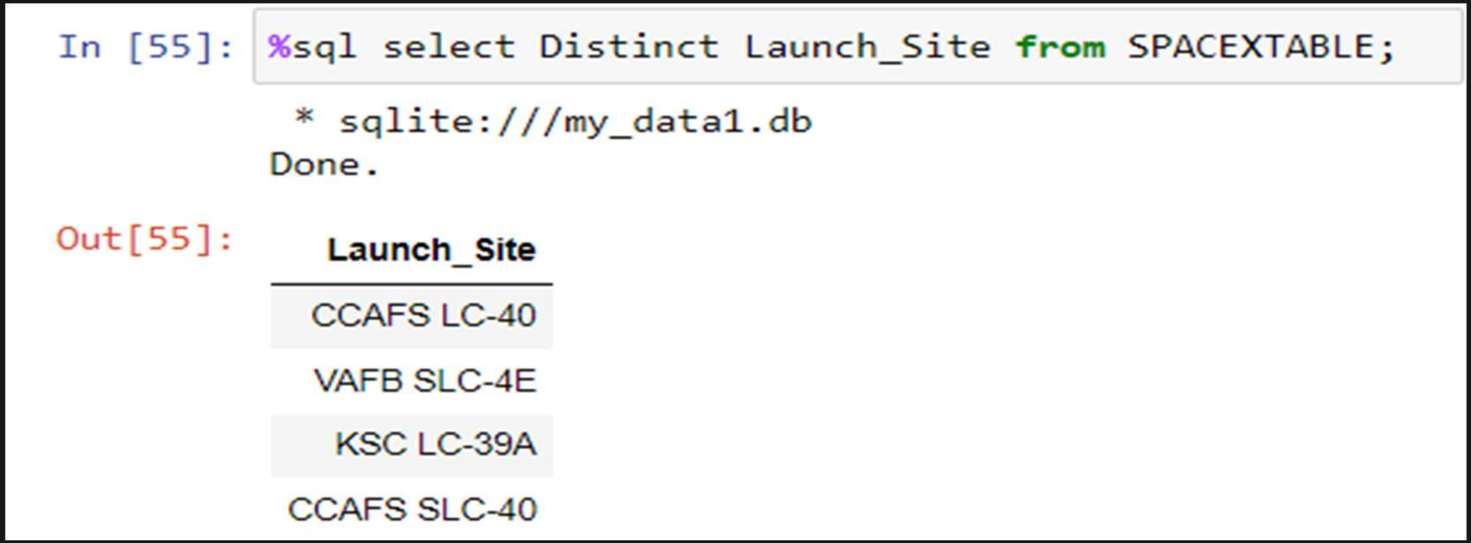
The line chart of yearly average success rate shows that:

- Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
- After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
- After 2016, there was always a greater than 50% chance of success.



EDA – With SQL

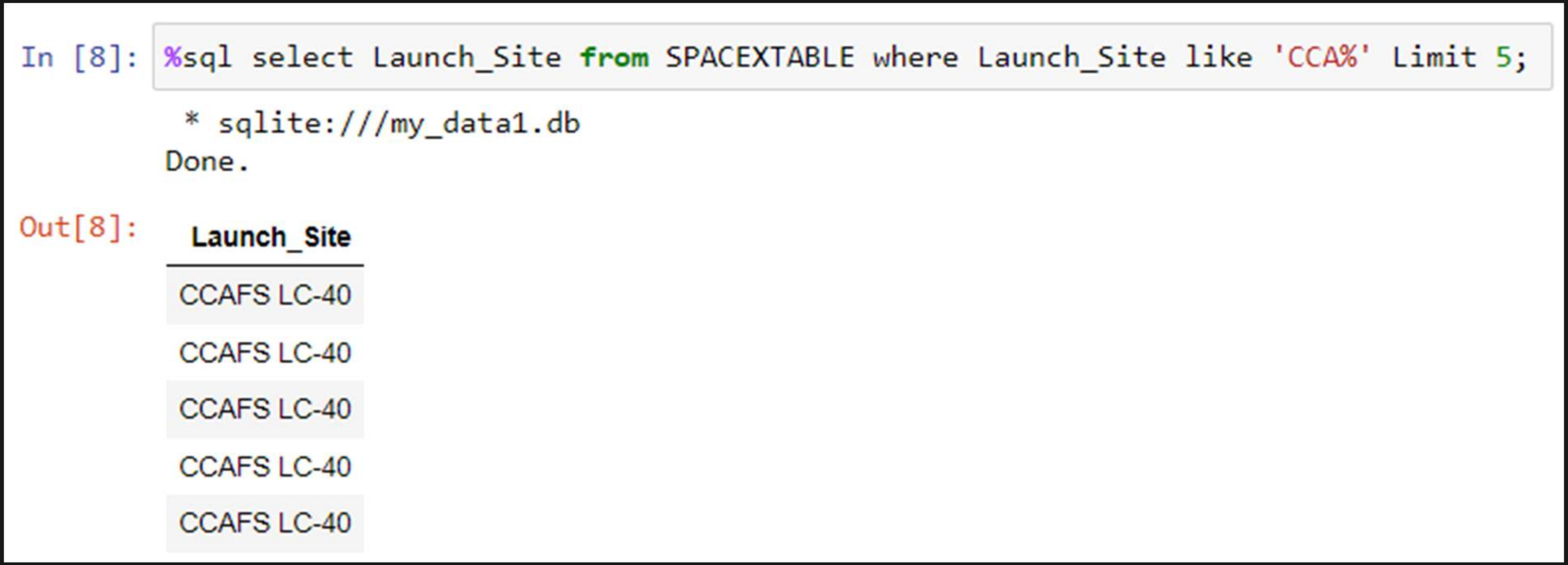
All Launch Site Names



```
In [55]: %sql select Distinct Launch_Site from SPACEXTABLE;

          * sqlite:///my_data1.db
         Done.

Out[55]:
              Launch_Site
             CCAFS LC-40
             VAFB SLC-4E
              KSC LC-39A
            CCAFS SLC-40
```

The keyword Distinct returns all unique/Distinct values from Launch Site column.

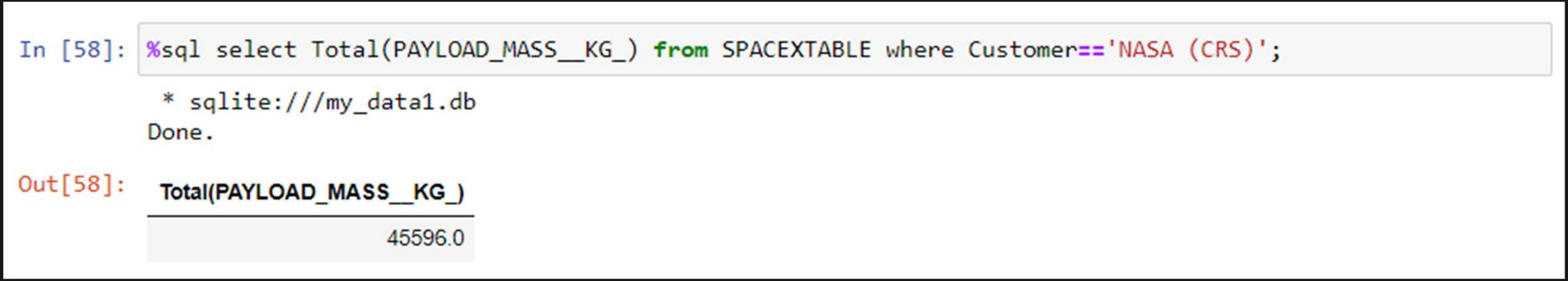
Launch Site Names Begin with ‘CCA’



```
In [8]: %sql select Launch_Site from SPACEXTABLE where Launch_Site like 'CCA%' Limit 5;

         * sqlite:///my_data1.db
        Done.

Out[8]:
         Launch_Site
         CCAFS LC-40
         CCAFS LC-40
         CCAFS LC-40
         CCAFS LC-40
         CCAFS LC-40
```

Where clause use to add condition in queries while LIMIT 5 fetches only 5 record and Like is used with a wildcard to retrieve values starting with ‘CCA’

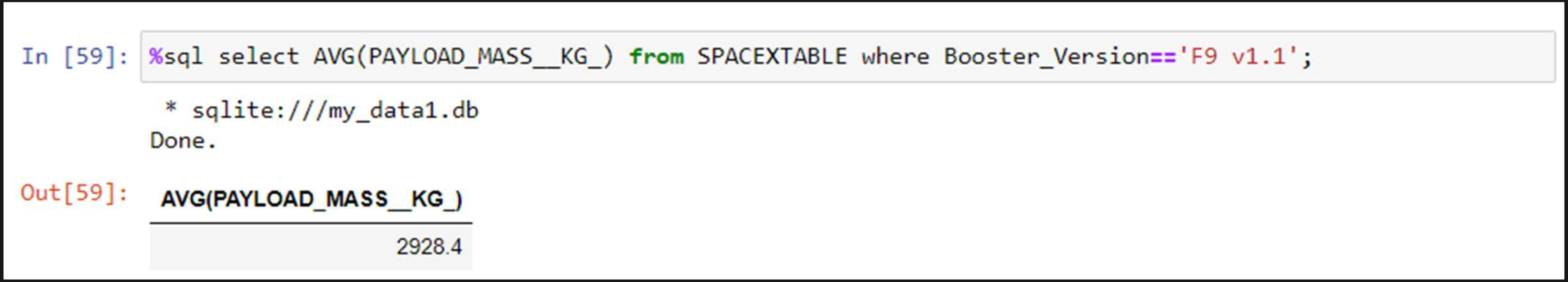
Total Payload Mass



```
In [58]: %sql select Total(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer=='NASA (CRS)';

          * sqlite:///my_data1.db
         Done.

Out[58]:  Total(PAYLOAD_MASS__KG_)
                           45596.0
```

The Total is used to calculate the total of Payload Mass. While where used to filter the results.

Average Payload Mass by F9 v1. 1

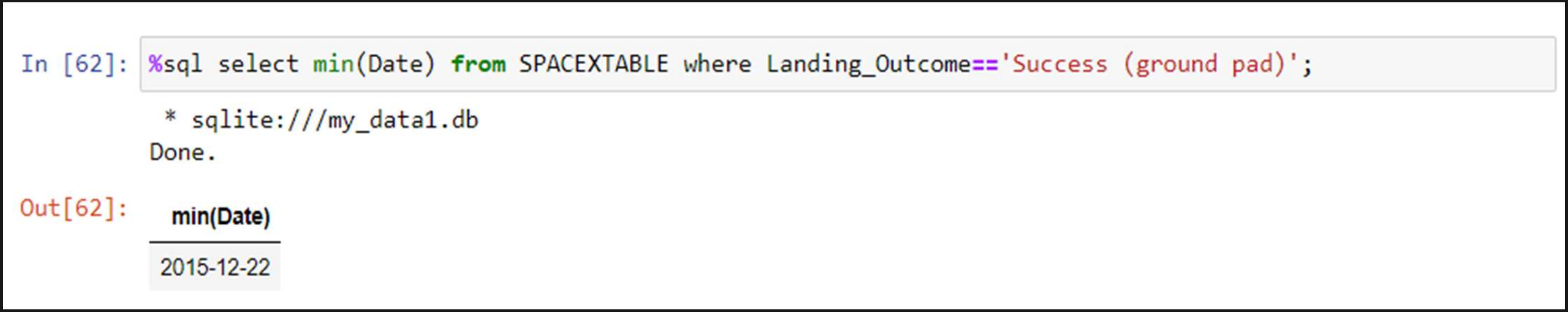


```
In [59]: %sql select AVG(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version=='F9 v1.1';

          * sqlite:///my_data1.db
         Done.

Out[59]:
```

AVG(PAYLOAD_MASS__KG_)
2928.4

The keyword AVG is used to calculate the average of the Payload_Mass_KG_ column and WHERE is used to filter results for Booster Version F9 v1.1.

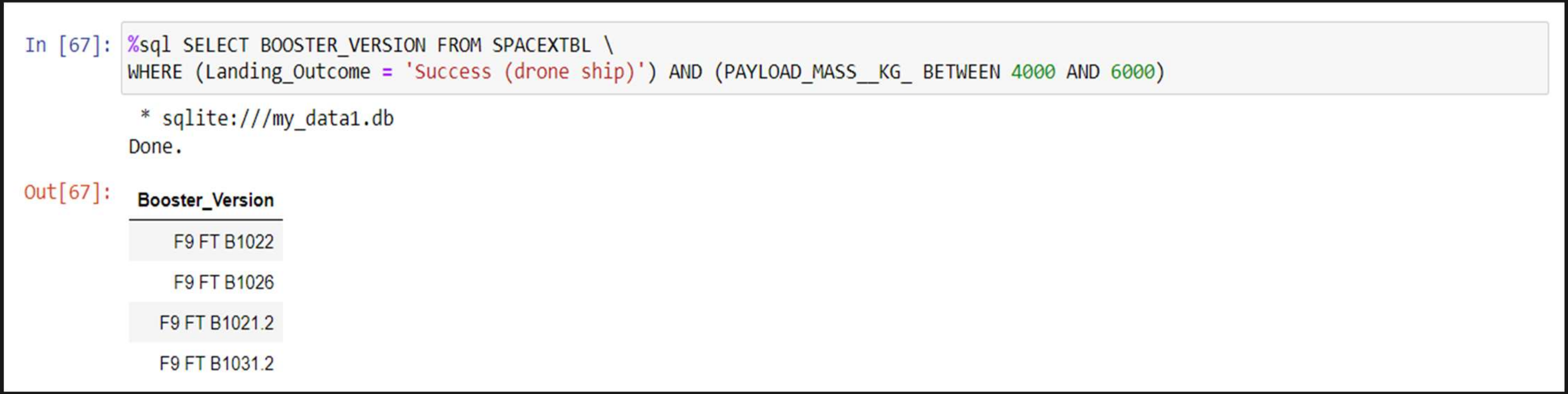
First Successful Ground Landing Date



```
In [62]: %sql select min(Date) from SPACEXTABLE where Landing_Outcome=='Success (ground pad)';

          * sqlite:///my_data1.db
         Done.

Out[62]:  min(Date)
          2015-12-22
```

The MIN keyword is used to calculate the minimum of the DATE column, i.e. the first date, and the WHERE keyword (and the associated condition) filters the results to only the successful ground pad landings.

Successful Drone Ship Landing with Payload between 4000 and 6000

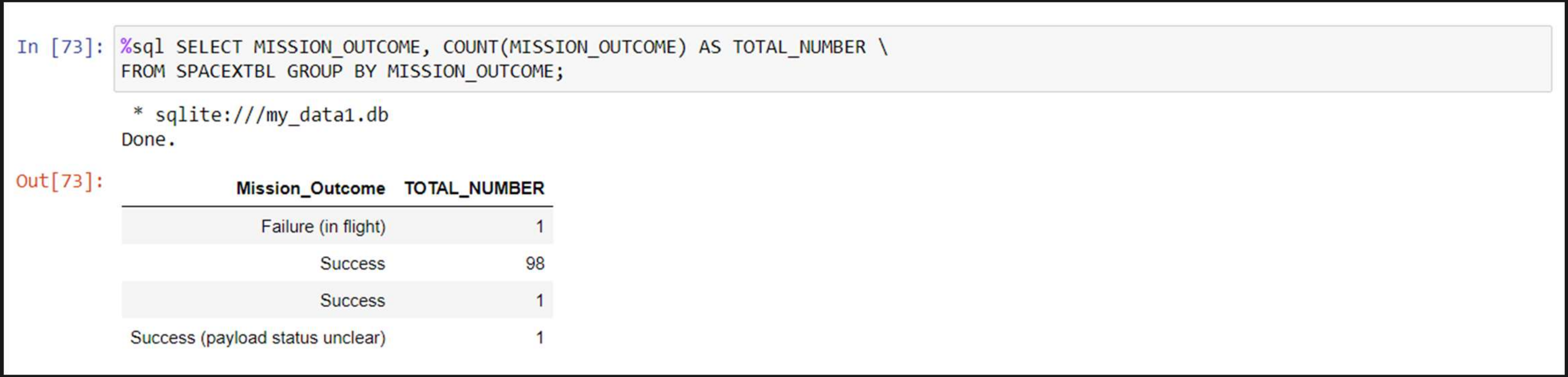


```
In [67]: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL \
         WHERE (Landing_Outcome = 'Success (drone ship)') AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)

          * sqlite:///my_data1.db
         Done.

Out[67]:
```

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

The WHERE keyword is used to filter the results to include only those that satisfy both conditions in the brackets (as the AND keyword is also used). The BETWEEN keyword allows for 4000 < x < 6000 values to be selected.

Total Number of Successful and Failure Mission Outcomes

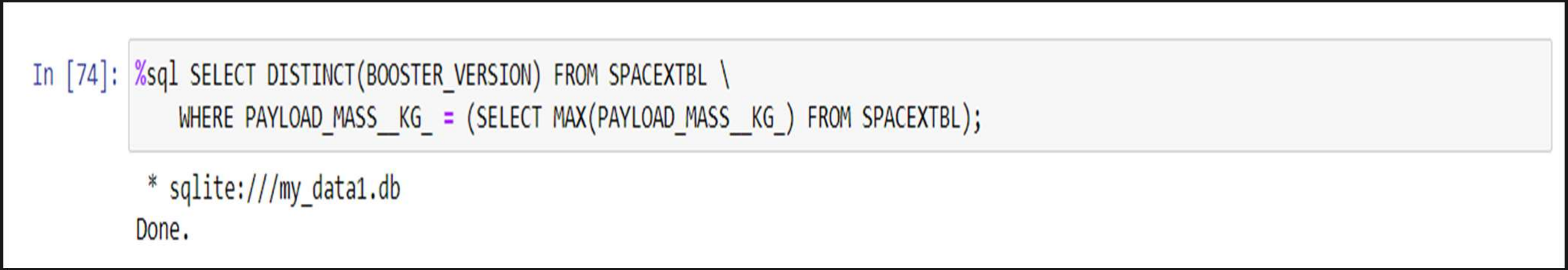


```
In [73]: %sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER \
         FROM SPACEXTBL GROUP BY MISSION_OUTCOME;

          * sqlite:///my_data1.db
         Done.

Out[73]:
```

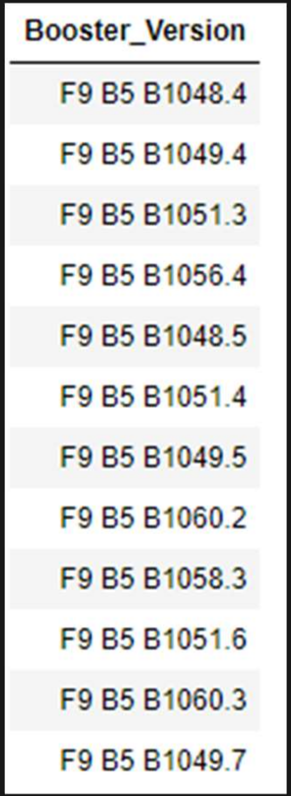
Mission_Outcome	TOTAL_NUMBER
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

The COUNT keyword is used to calculate the total number of mission outcomes, and the GROUPBY keyword is also used to group these results by the type of mission outcome.

Boosters Carried Maximum Payload



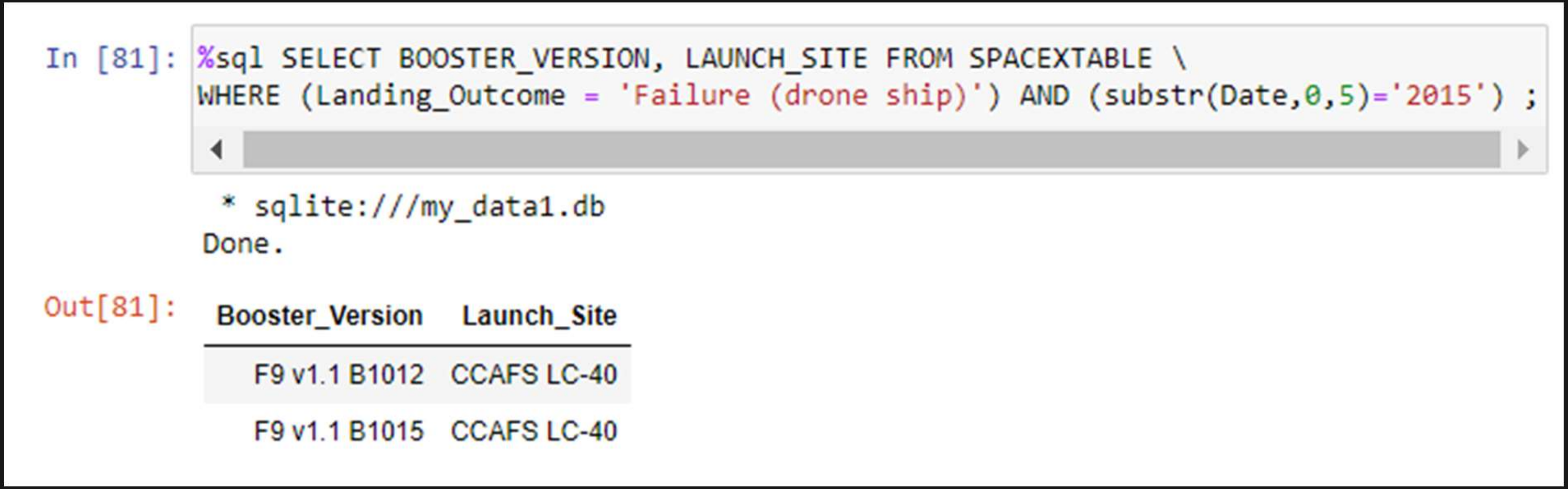
```
In [74]: %sql SELECT DISTINCT(BOOSTER_VERSION) FROM SPACEXTBL \
             WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);

          * 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

A subquery is used here. The SELECT statement within the brackets finds the maximum payload, and this value is used in the WHERE condition. The DISTINCT keyword is then used to retrieve only distinct /unique booster versions.

2015 Launch Records



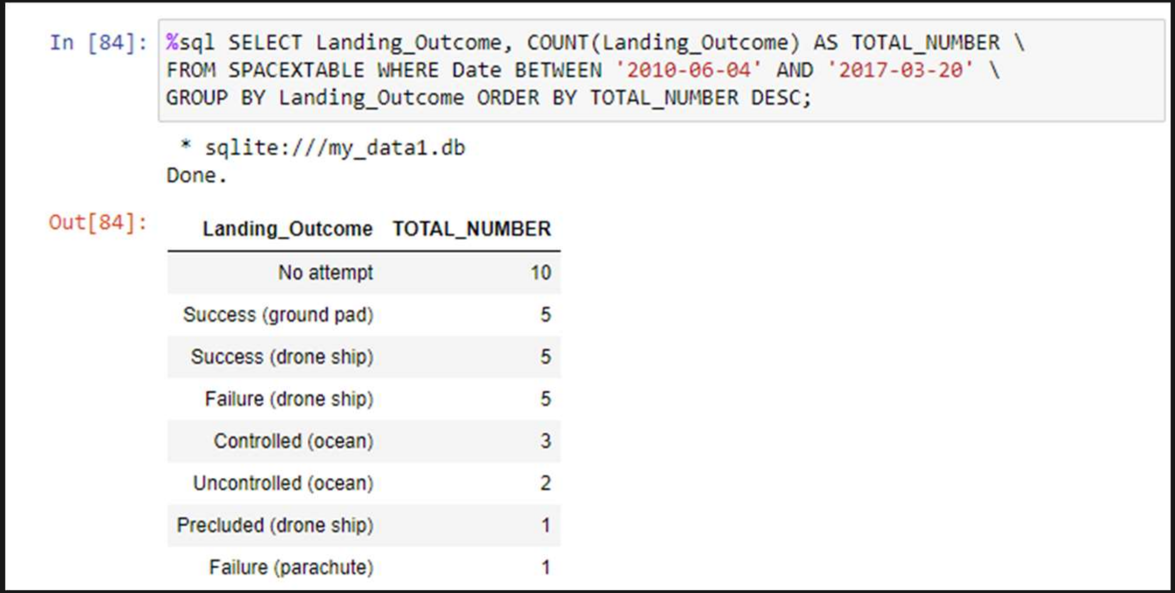
```
In [81]: %sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTABLE \
         WHERE (Landing_Outcome = 'Failure (drone ship)') AND (substr(Date,0,5)='2015') ;

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

Out[81]:

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

The WHERE keyword is used to filter the results to include only failed landing outcomes, AND is used to combine two conditions.

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

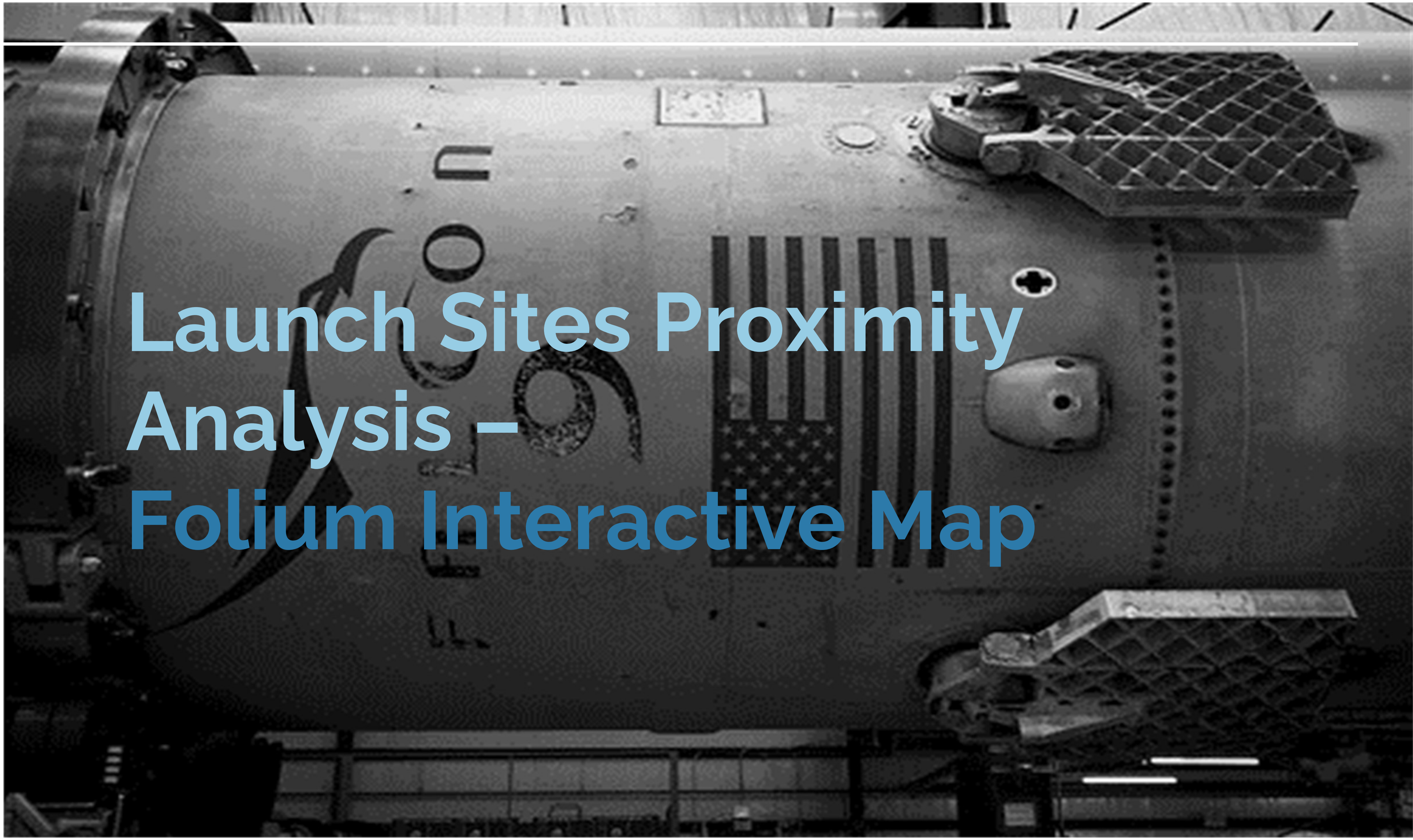


```
In [84]: %sql SELECT Landing_Outcome, COUNT(Landing_Outcome) AS TOTAL_NUMBER \
         FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' \
         GROUP BY Landing_Outcome ORDER BY TOTAL_NUMBER DESC;

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

Out[84]:

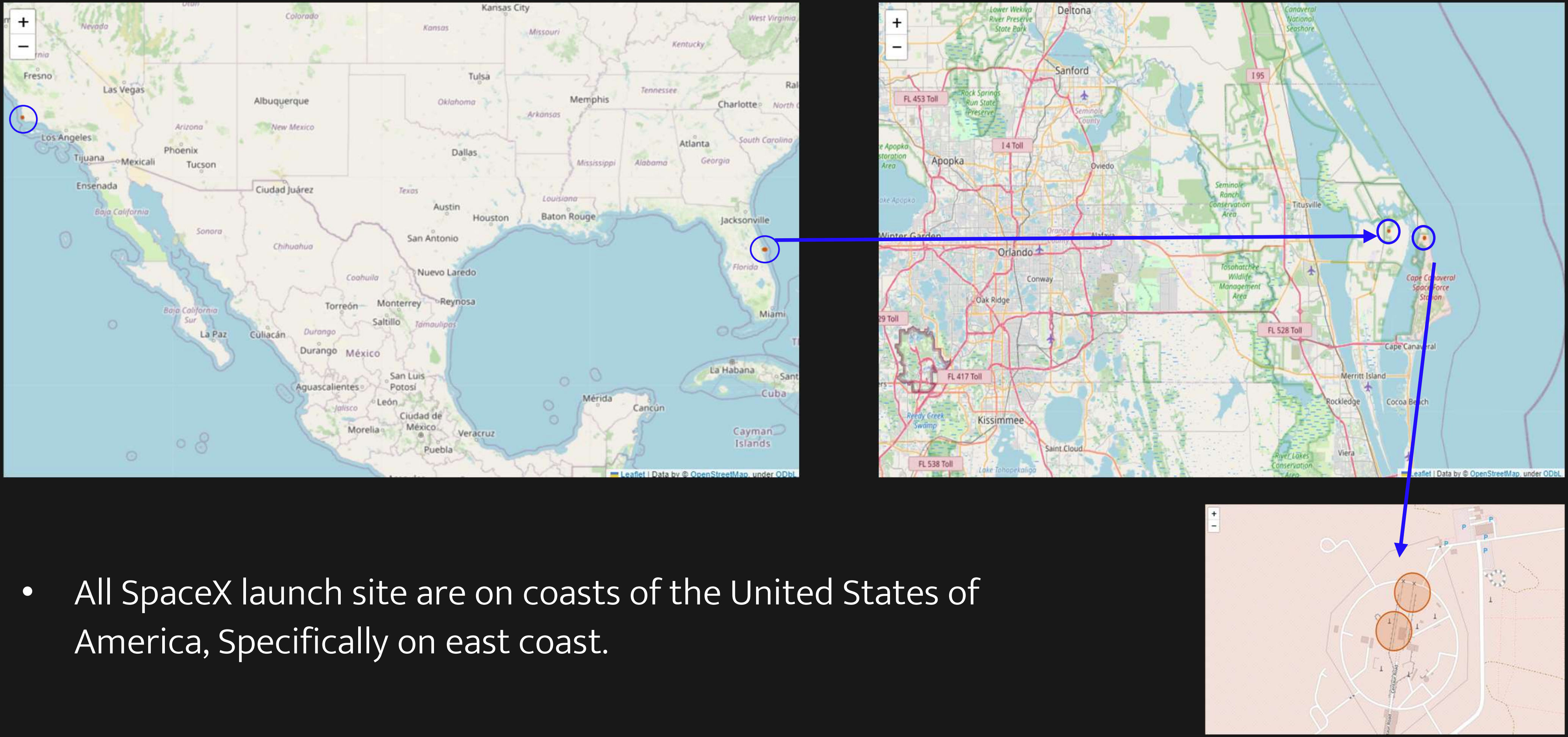
Landing_Outcome	TOTAL_NUMBER
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

The WHERE keyword is used with the BETWEEN keyword to filter the results to dates only within those specified. The results are then grouped and ordered, using the keywords GROUP BY and ORDER BY, respectively, where DESC is used to specify the descending order.



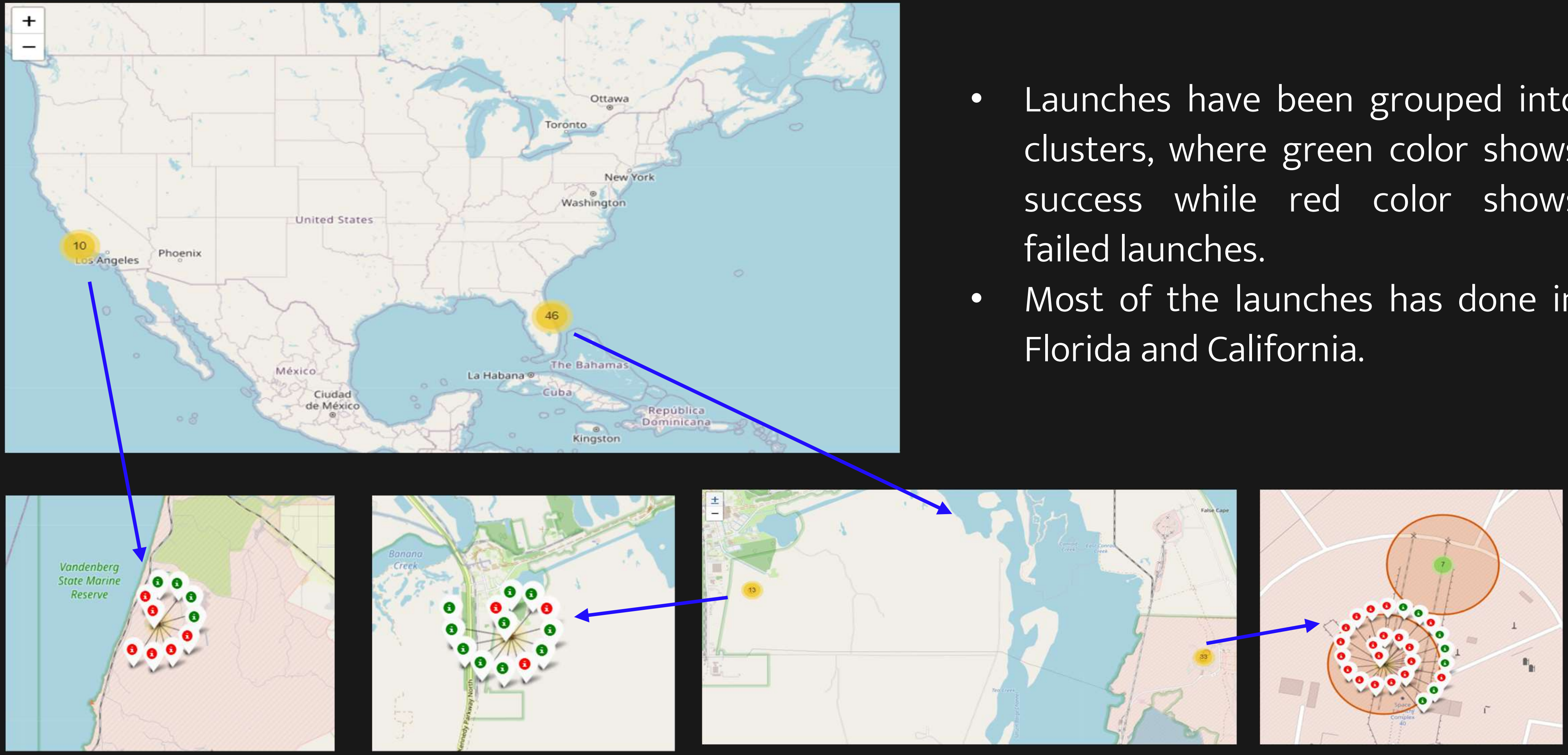
Launch Sites Proximity Analysis – Folium Interactive Map

Marking All Launch Sites on Map



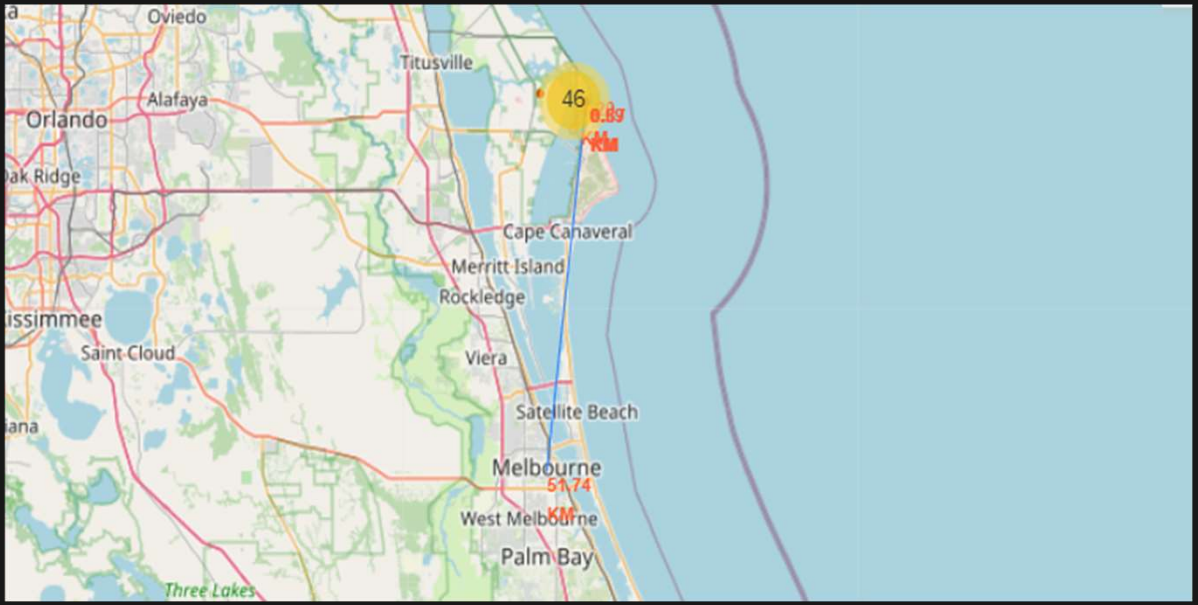
- All SpaceX launch site are on coasts of the United States of America, Specifically on east coast.

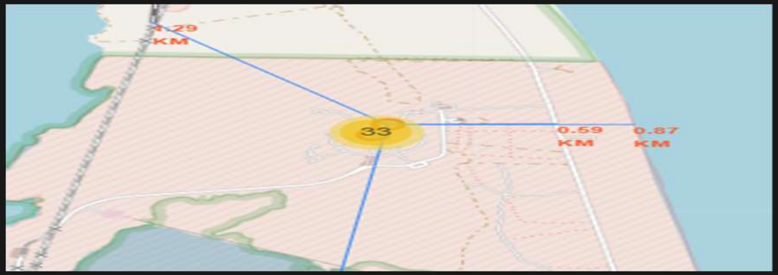
Success/Failed Launch on each site



- Launches have been grouped into clusters, where green color shows success while red color shows failed launches.
- Most of the launches has done in Florida and California.

Launch Site Distance to Proximities





- We used CCAFS SLC-40 launch site as example from where we can understand of launch sites.

- Are launch sites in close proximity to railways?

YES. The coast line is only 0.87km due East.

- Are launch sites in close proximity to highways?

YES. The nearest highway is only 0.59km away.

- Are launch sites in close proximity to railways?

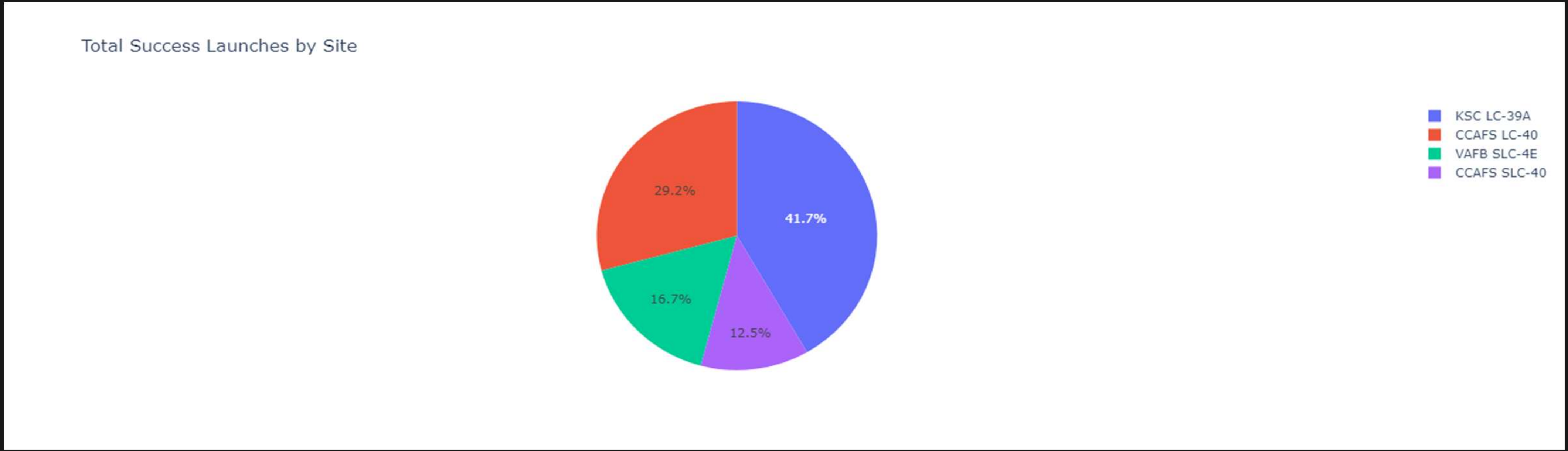
YES. The nearest railway is only 1.29 km away.

- Do launch sites keep certain distance away from cities?

YES. The nearest city is 51.74 km away.

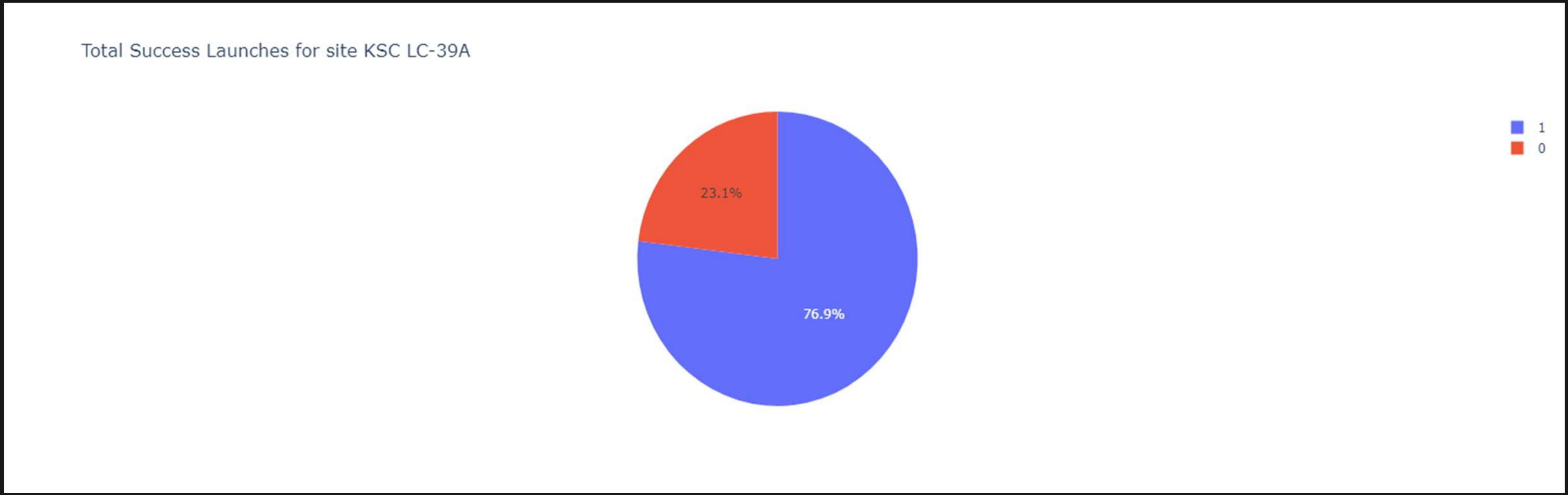
Interactive Dashboard – Plotly Dash

Total Success launches by Site



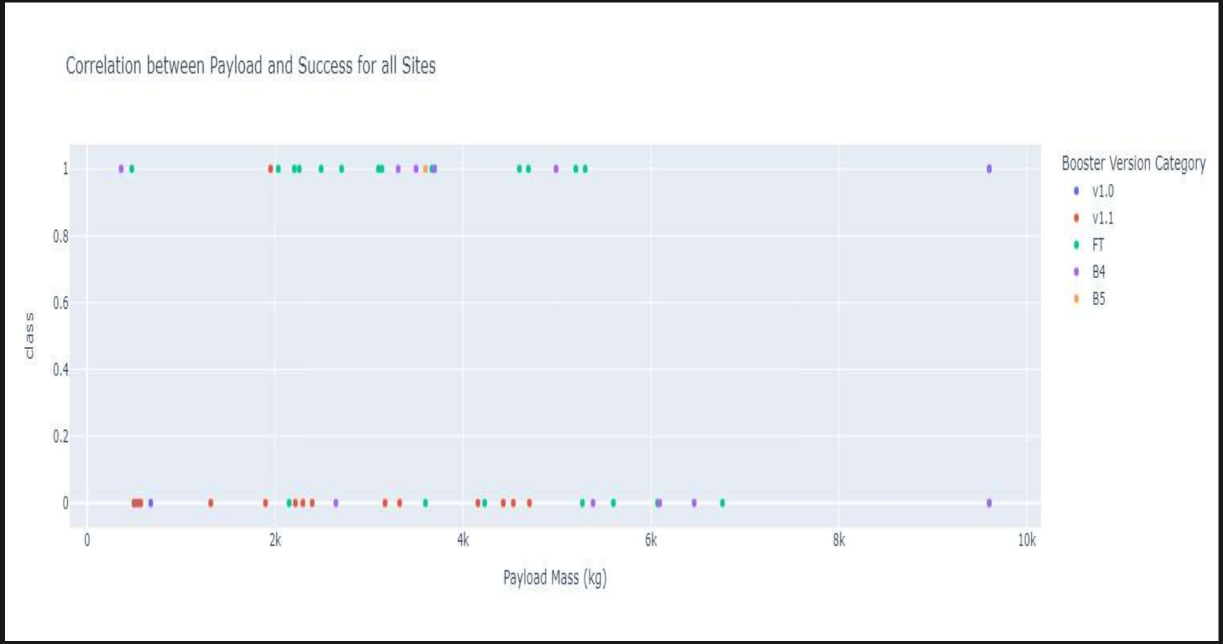
- The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches.
- While launch site CCAFS SLC-40 had the lowest success rate, with 12.5%.

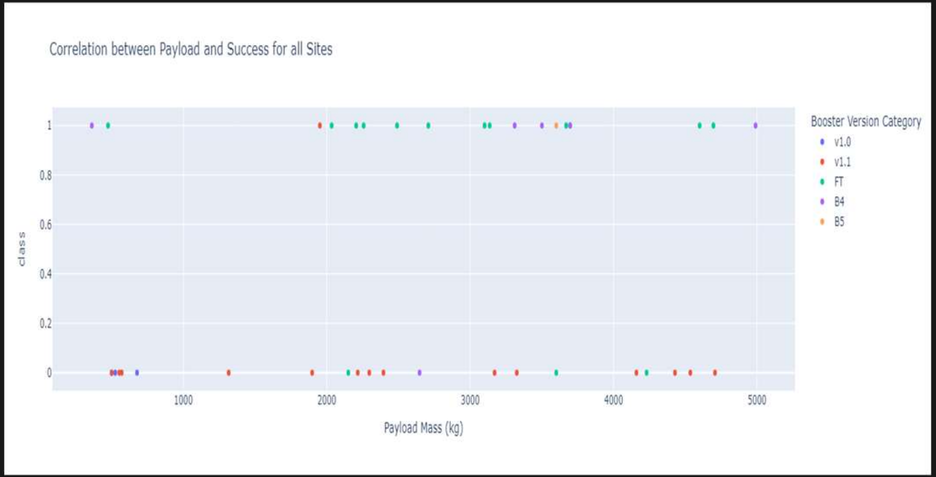
Launch Site with Highest Launch Success Rate

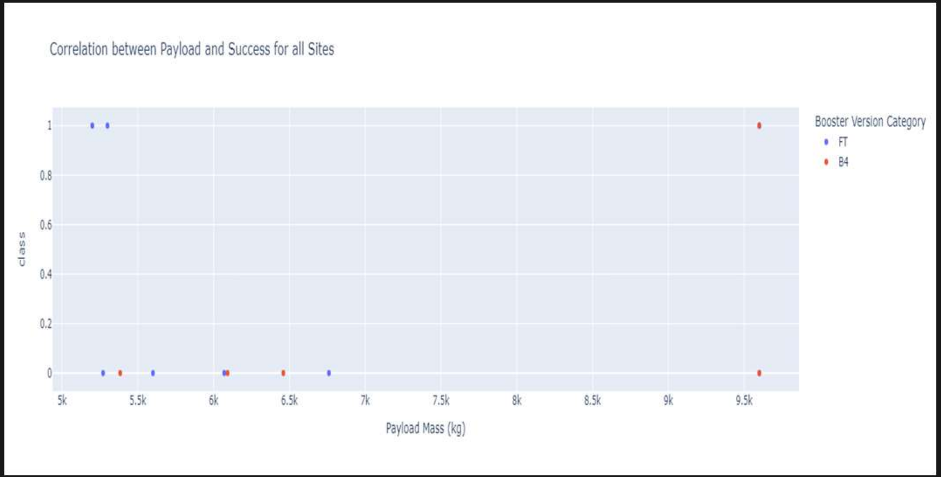


The launch site KSC LC-39 A had the highest rate of successful launches, with a 76.9% success rate.

Correlation Between Payload and Success







- Plotting the Scatter plot for Payload Mass and Success for different boosters.
- Payload can be split into 2 categories

  0 – 4000 kg (low payload)

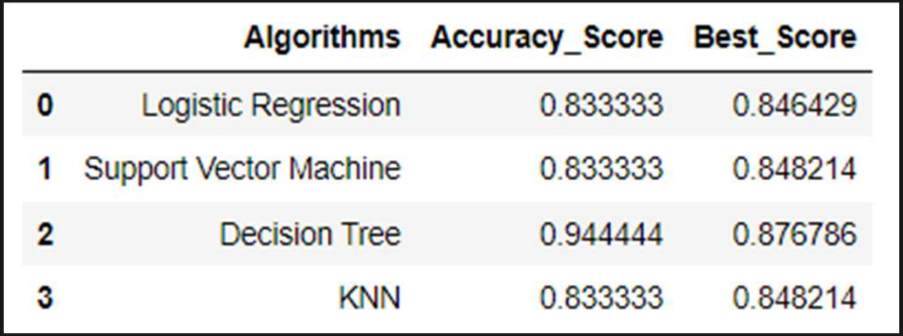
  4000 – 10000 kg (Massive Payload)

Predictive Analysis – Classification

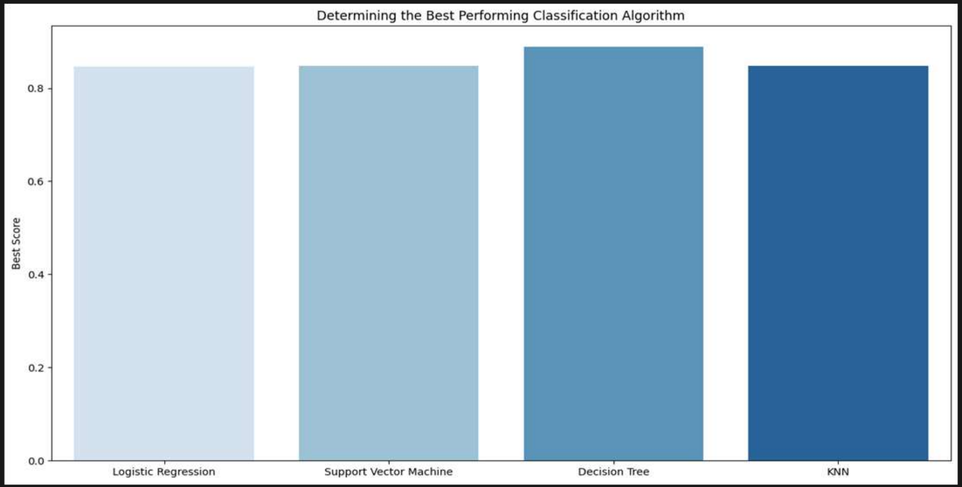
Classification Accuracy

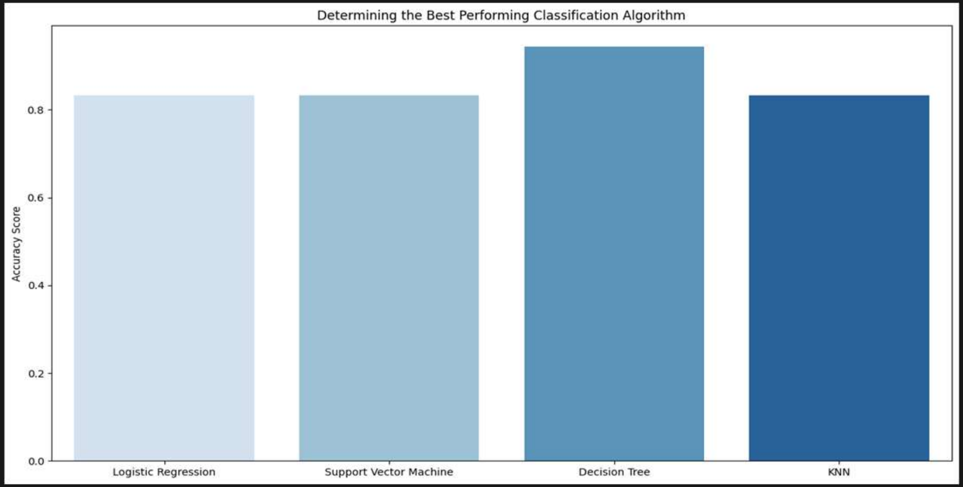
Plotting the Accuracy score and Best Score for each classification algorithm produces the following results:

- The Decision Tree model has the highest accuracy score



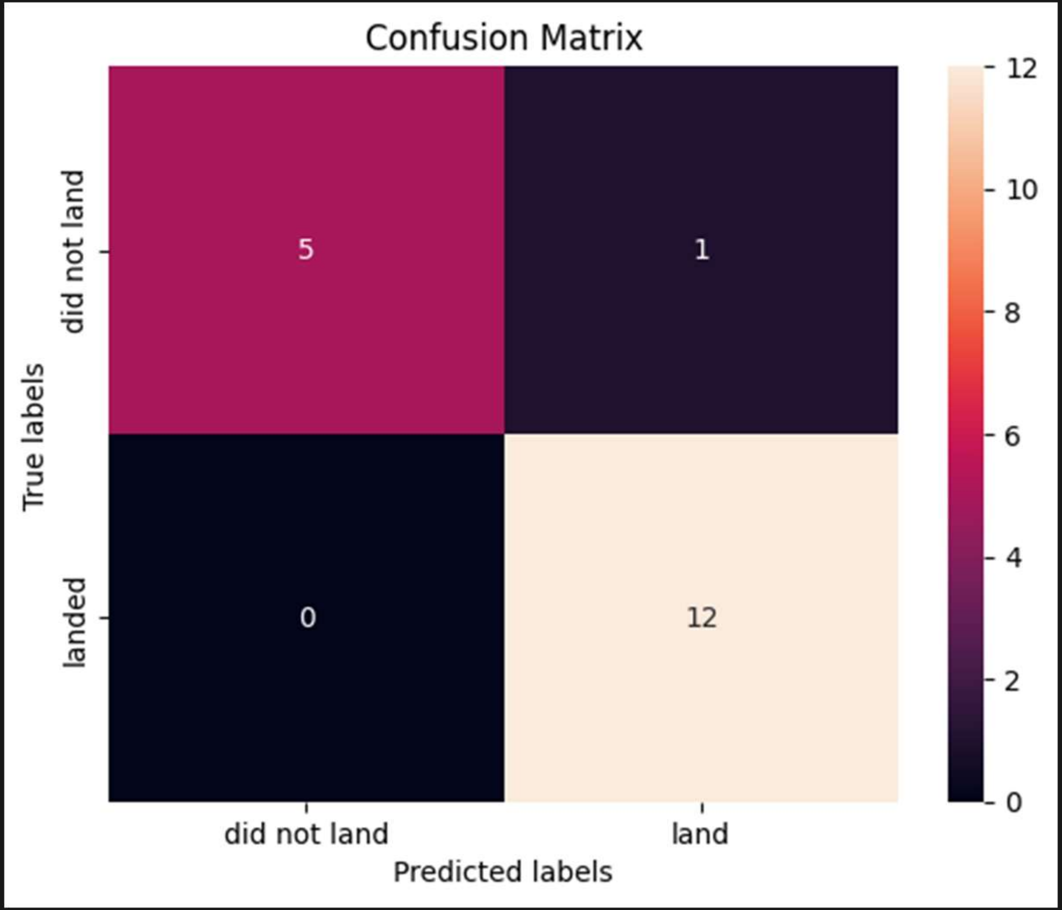
	Algorithms	Accuracy_Score	Best_Score
0	Logistic Regression	0.833333	0.846429
1	Support Vector Machine	0.833333	0.848214
2	Decision Tree	0.944444	0.876786
3	KNN	0.833333	0.848214





Confusion Matrix

- As shown in previous slide, best Performing classification model is the Decision Tree model, with an accuracy of 94.44%.
- This is explained by this confusion matrix, which shows only 1 out of 18 results classified incorrectly.
- While other 17 results are correctly classified in which 12 did land successfully and 5 did not land.



Conclusion

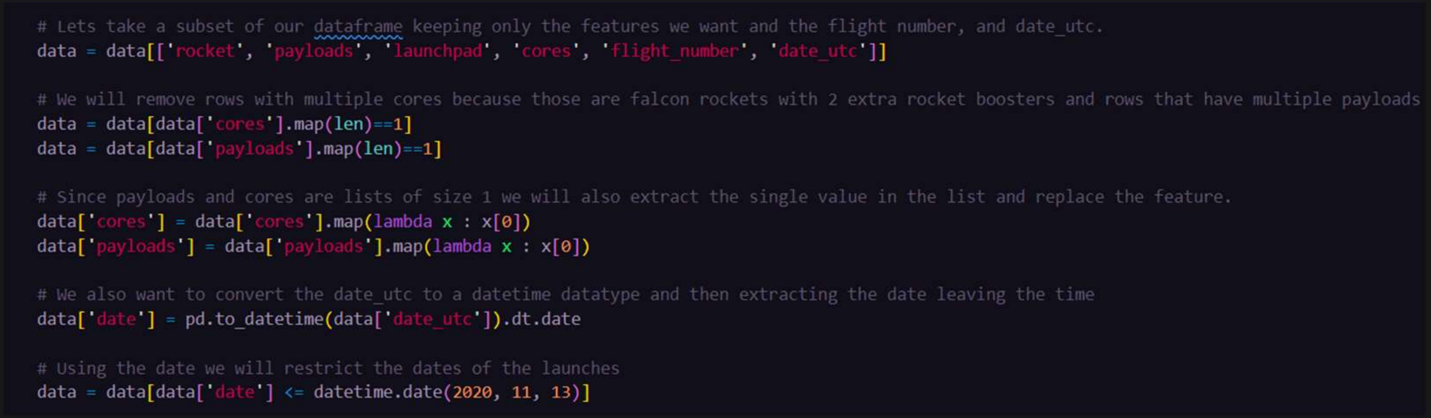
Conclusion

- As the number flight increases, the rate of success is also increases, with most early flights being unsuccessful. i.e. with more experience, the success rate increases.
- Between 2010 and 2013, all landing were unsuccessful (0% success rate). After 2013, the success rate gradually increased despite small dips in 2018 and 2020.
- After 2016 success rate increased to more than 50%.
- The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches, while launch site CCAFS SLC-40 had the lowest success rate, with 12.5%.
- The success for massive payloads (over 4000kg) is lower than that for low payloads.
- The Best performing classification model is the Decision Tree Model, with an accuracy of 94.4%.

Appendix

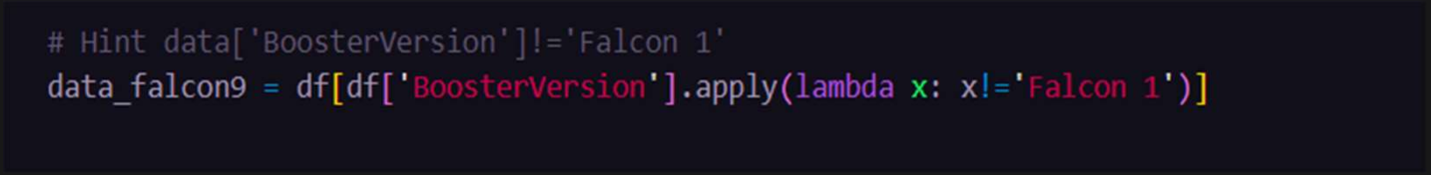
Data Collection - SpaceX REST API

We used the API to get information about the launches using the IDs given for each launch. Specifically used columns rocket, payloads, launchpad, and cores.



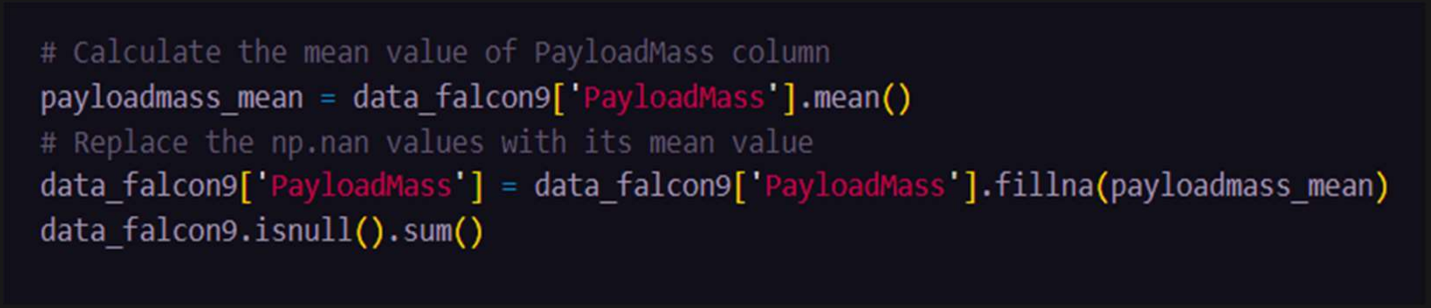
```python
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads
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)]
```

Filter the dataframe to only include `Falcon 9` launches



```python
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df[df['BoosterVersion'].apply(lambda x: x!='Falcon 1')]
```

Dealing with missing values by using mean and filling missing values with fillna() method



```python
# Calculate the mean value of PayloadMass column
payloadmass_mean = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(payloadmass_mean)
data_falcon9.isnull().sum()
```

THANK YOU !!!