IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sandeep A Menon
04-Aug-2025

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
  - Data Wrangling
  - EDA with Data Visualization
  - EDA with SQL
  - Building a interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive Analysis
- Summary of all results
  - EDA results
  - Interactive Analysis
  - Predictive analysis

Introduction

- Project background and context
- Problems you want to find answers

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX open source rest api
  - Web scrapping from Wikipedia page
- Perform data wrangling
  - One-hot encoding was applied to 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

- 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.

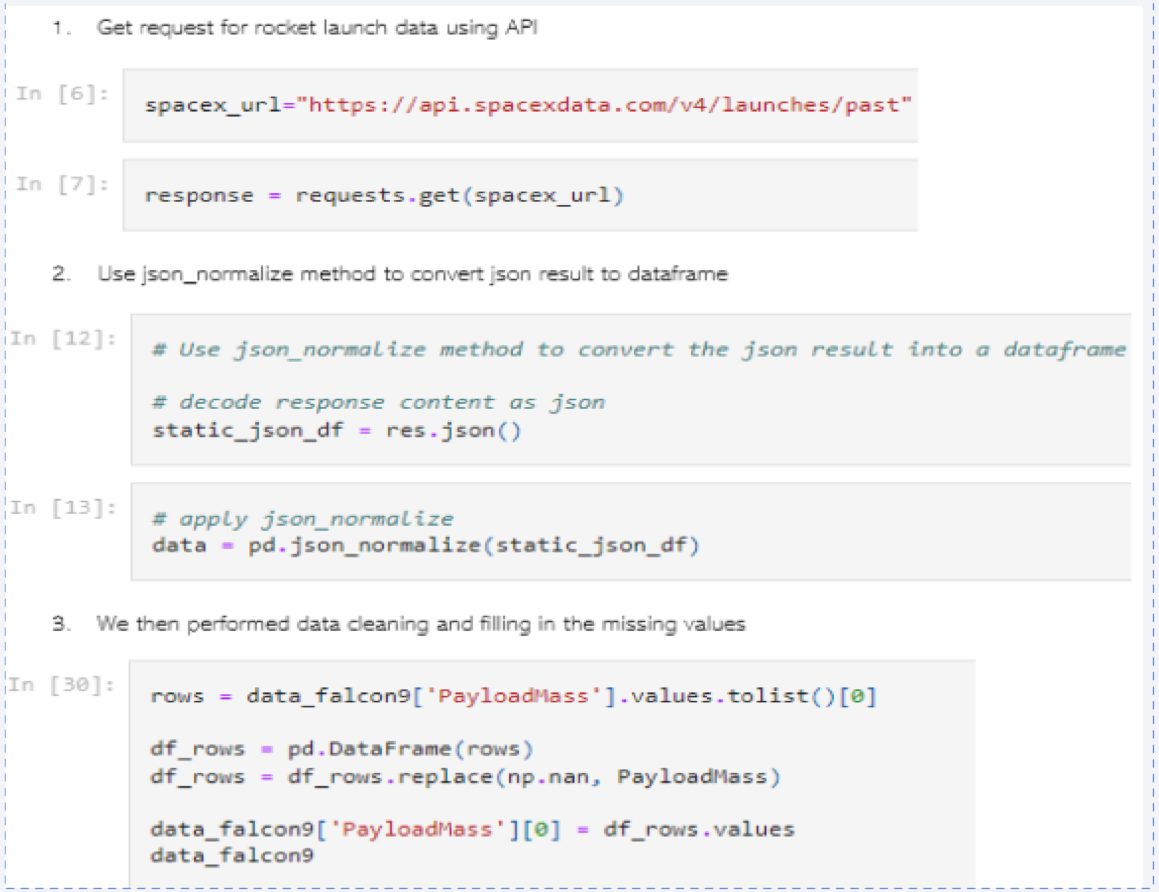
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.

  - 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 API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.

- The link to the note book is https://github.com/sandeepmenon199/IBM_Data_science_capstone_SpaceX/blob/main/spacex-data-collection-api.ipynb



1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

```
In [12]: # Use json_normalize method to convert the json result into a dataframe

         # decode response content as json
         static_json_df = res.json()
```

```
In [13]: # apply json_normalize
         data = pd.json_normalize(static_json_df)
```

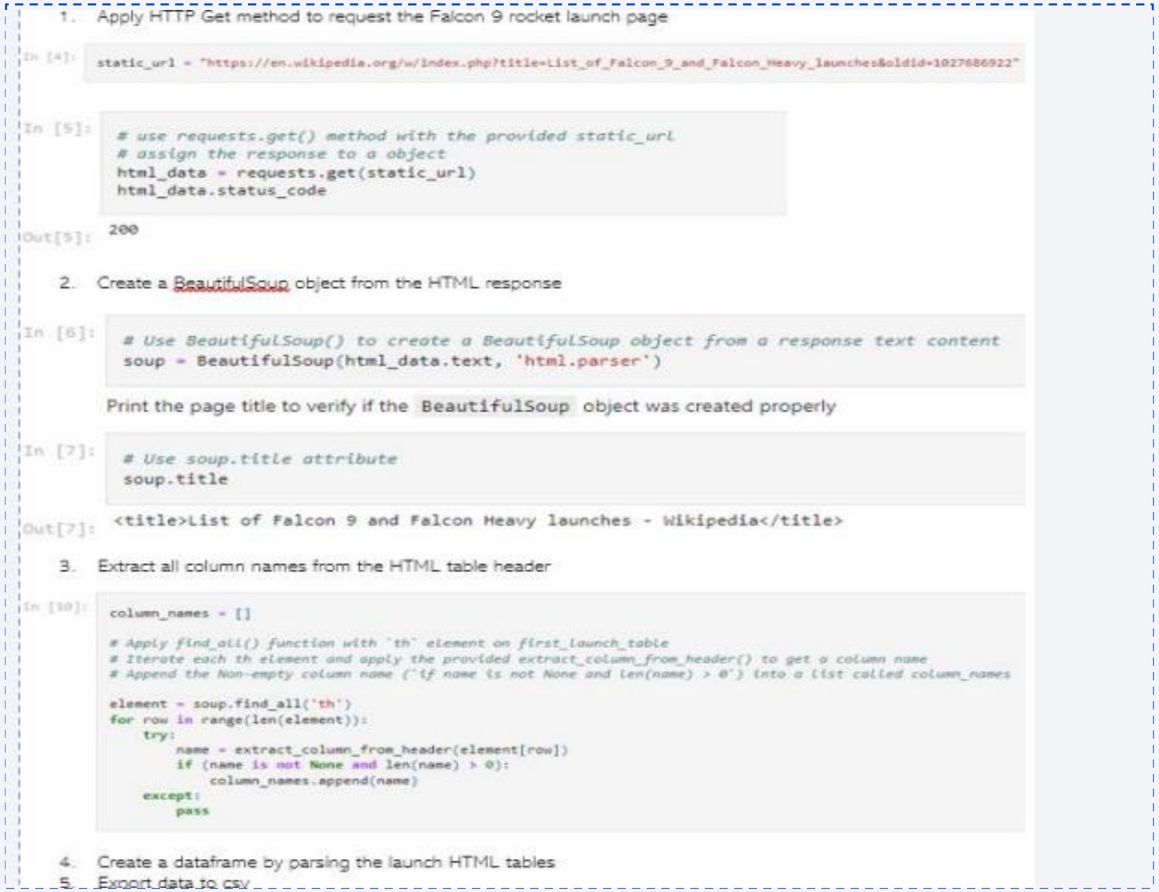
3. We then performed data cleaning and filling in the missing values

```
In [30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]

         df_rows = pd.DataFrame(rows)
         df_rows = df_rows.replace(np.nan, PayloadMass)

         data_falcon9['PayloadMass'][0] = df_rows.values
         data_falcon9
```

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- Link to the notebook

https://github.com/sandeepmenon199/IBM_Data_science_capstone_SpaceX/blob/main/spacex-data-webscraping.ipynb



1. Apply HTTP Get method to request the Falcon 9 rocket launch page

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

Out[5]:
```
200
```

2. Create a BeautifulSoup object from the HTML response

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract all column names from the HTML table header

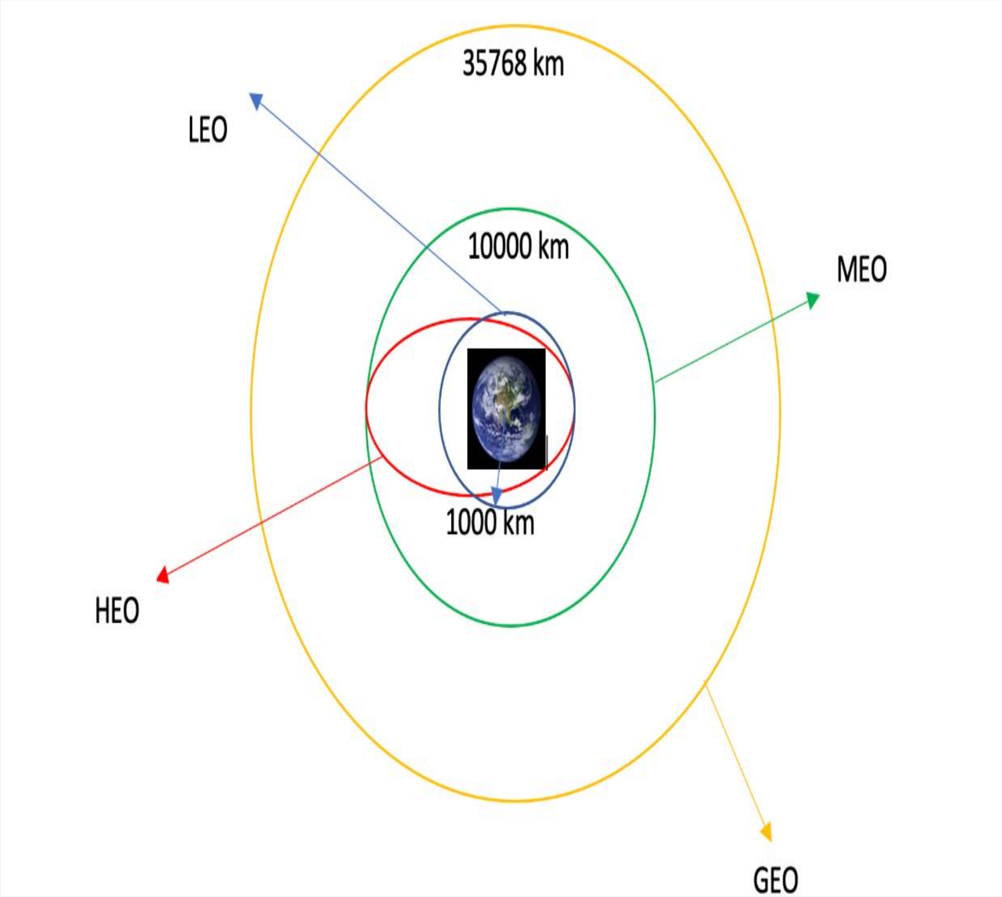
In [10]:
```
column_names = []

# 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

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

Data Wrangling

- 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 results to csv.
- The link to the notebook is

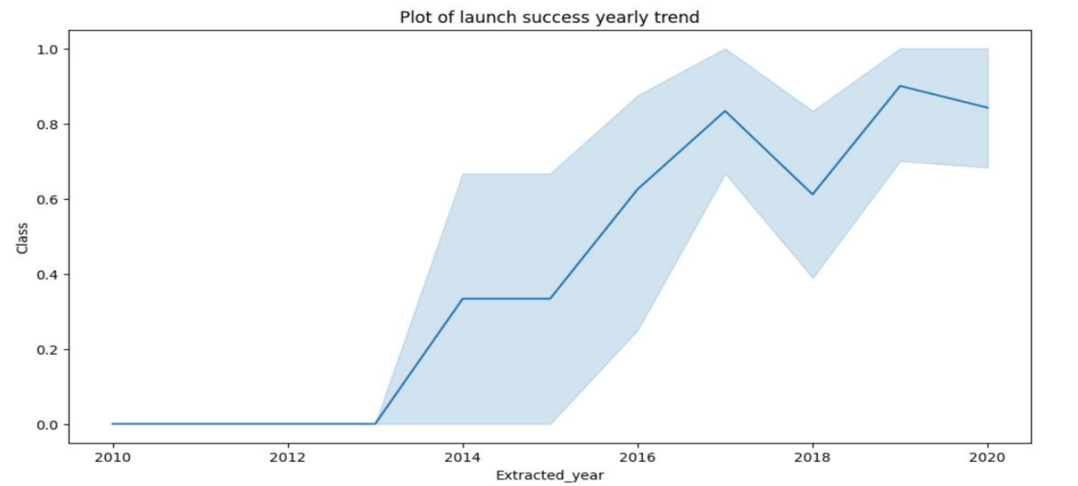
https://github.com/sandeepmenon199/IBM_Data_science_capstone_SpaceX/blob/main/spacex-Data%20wrangling.ipynb

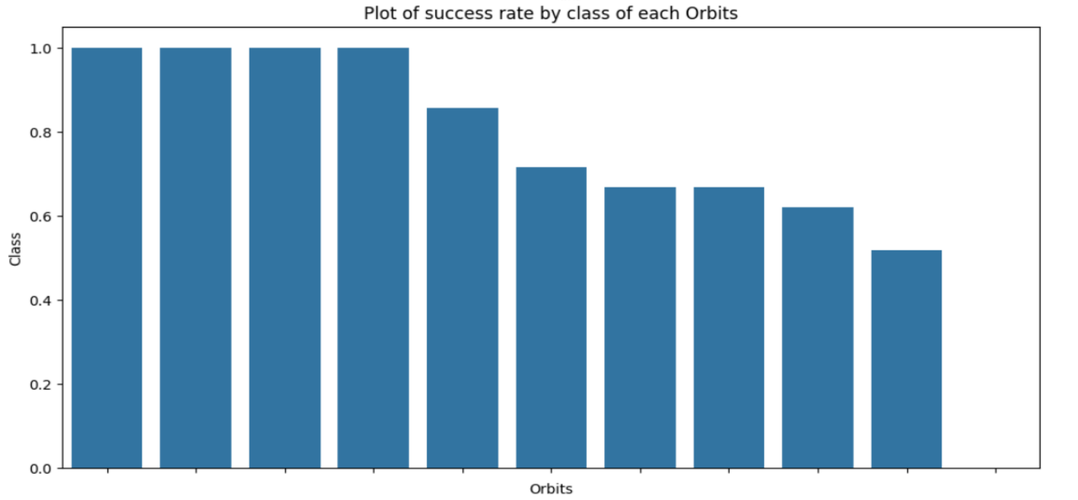


EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.
- Link to note book

https://github.com/sandeepmenon199/IBM_Data_science_capstone_SpaceX/blob/main/EDA%20with%20Data%20Visualization.ipynb





EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.

- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:

  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.

- The link to the notebook is https://github.com/sandeepmenon199/IBM_Data_science_capstone_SpaceX/blob/main/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:

- Are launch sites near railways, highways and coastlines.
- Do launch sites keep certain distance away from cities

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- Link to notebook https://github.com/sandeepmenon199/IBM_Data_science_capstone_SpaceX/blob/main/spacex-dash-app.py

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is https://github.com/sandeepmenon199/IBM_Data_science_capstone_SpaceX/blob/main/SpaceX_Machine%20Learning%20Prediction.ipynb

Results

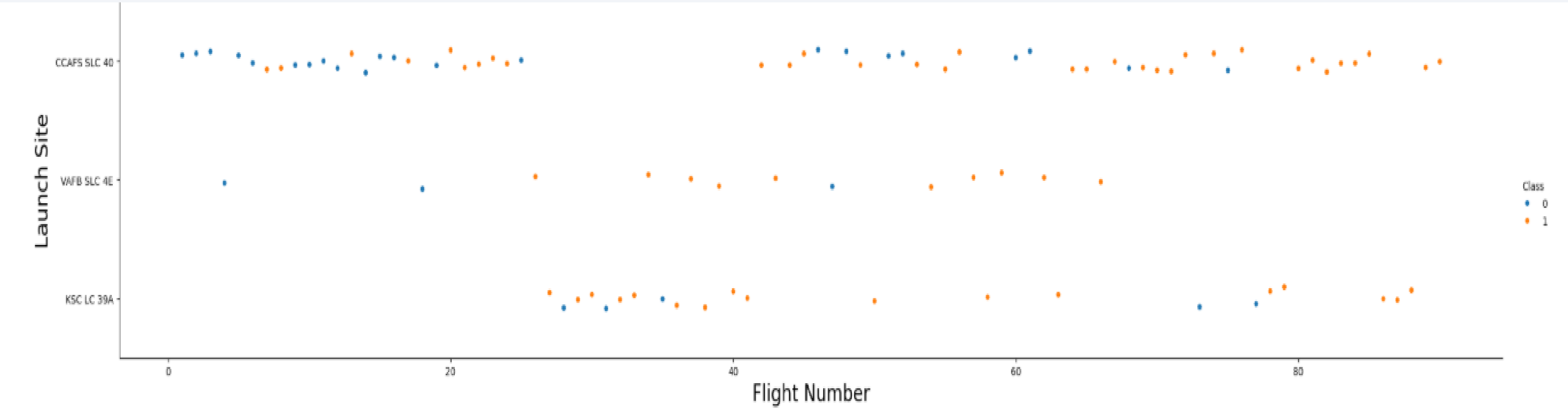
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

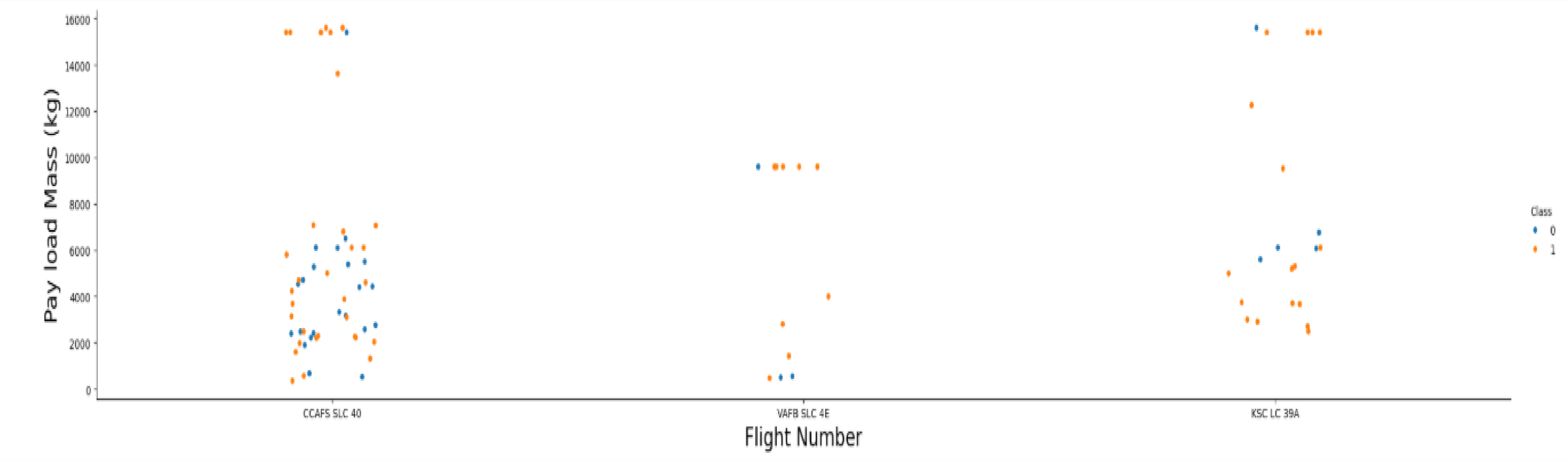
Insights drawn from EDA

Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site

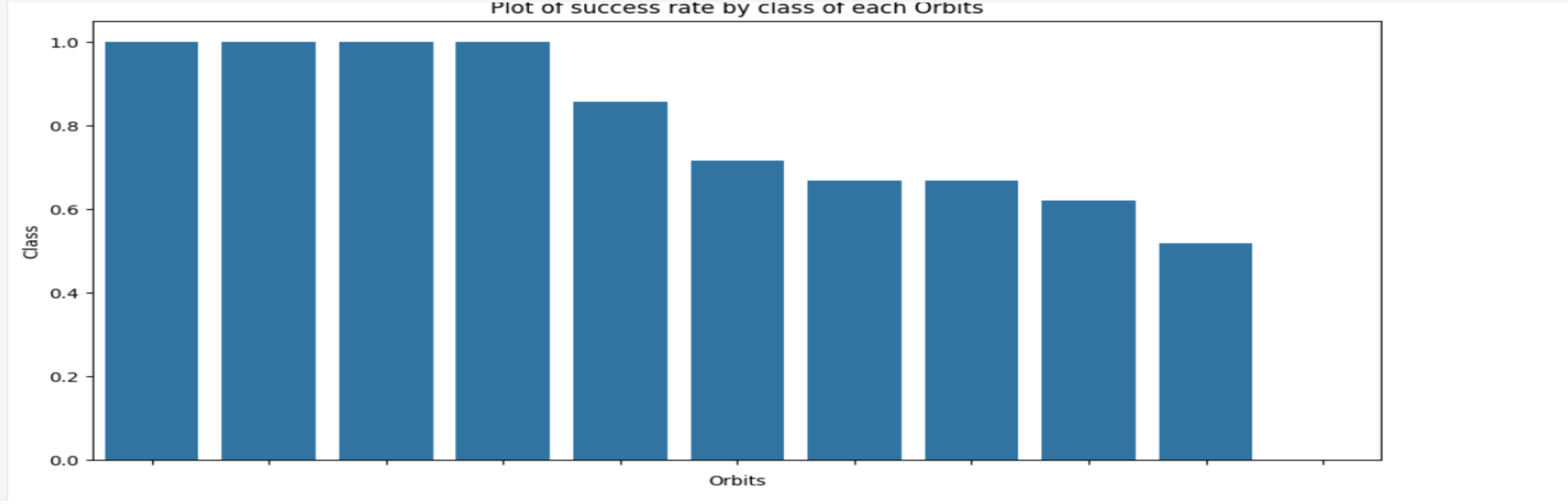


Payload vs. Launch Site



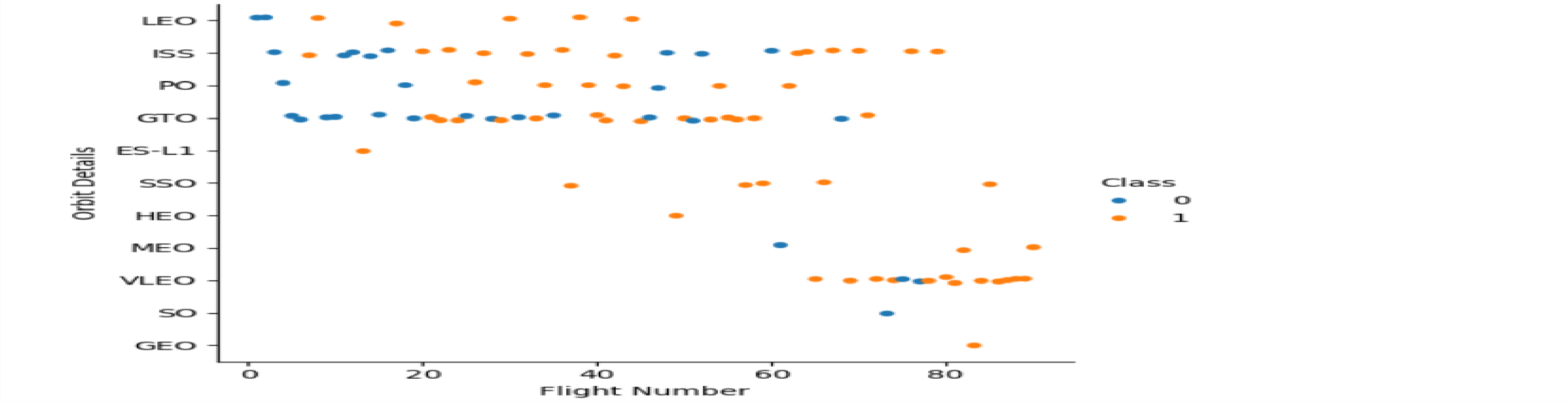
Success Rate vs. Orbit Type

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

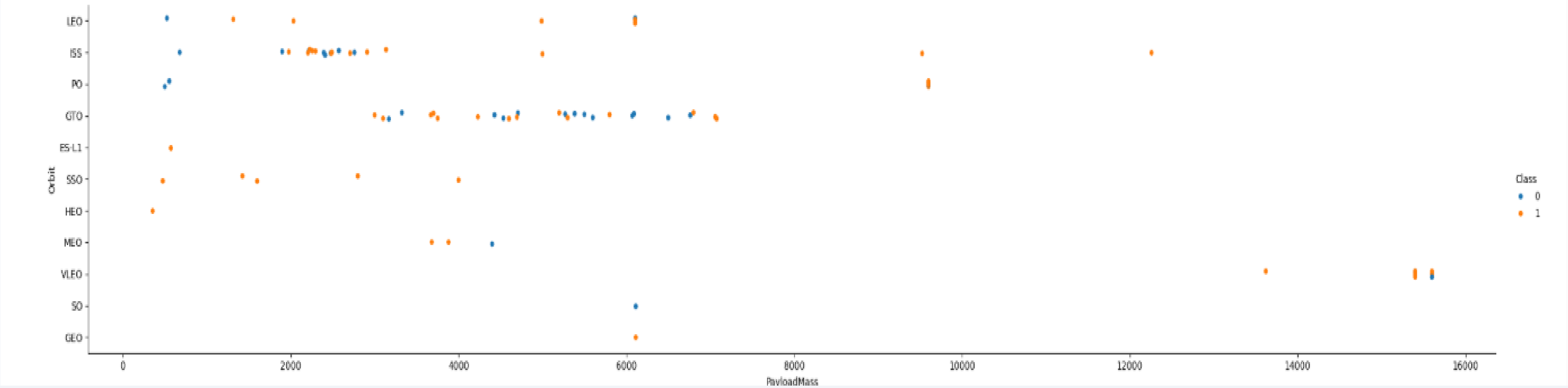


Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.

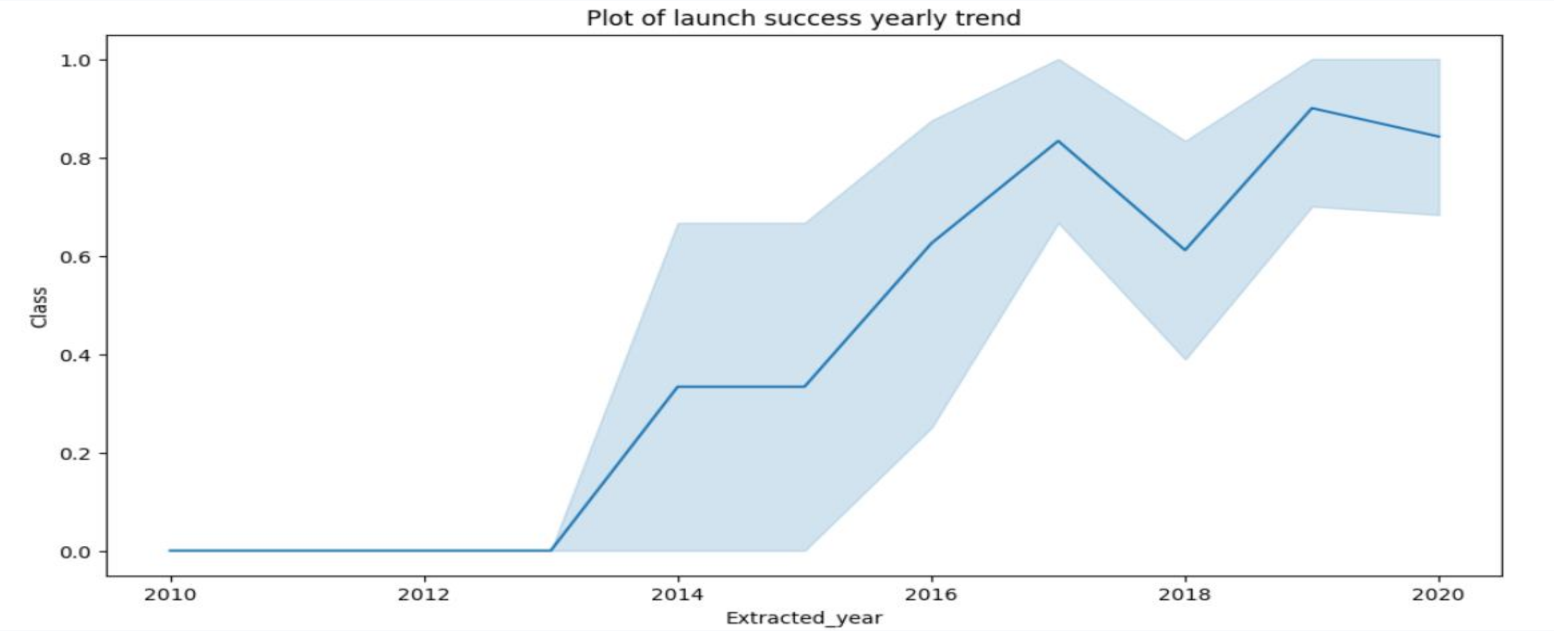


Payload vs. Orbit Type



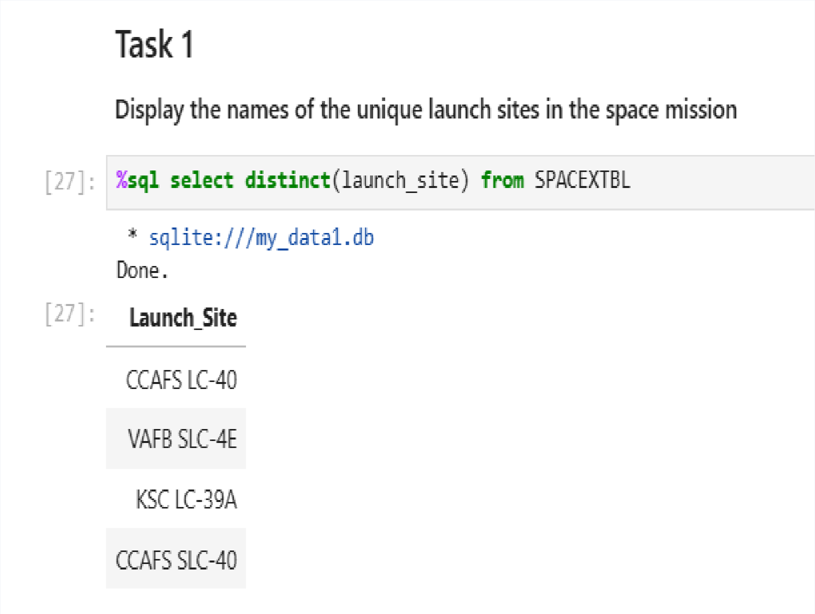
Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- We used the key word DISTINCT to show only unique launch sites from the SpaceX data.



Task 1

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

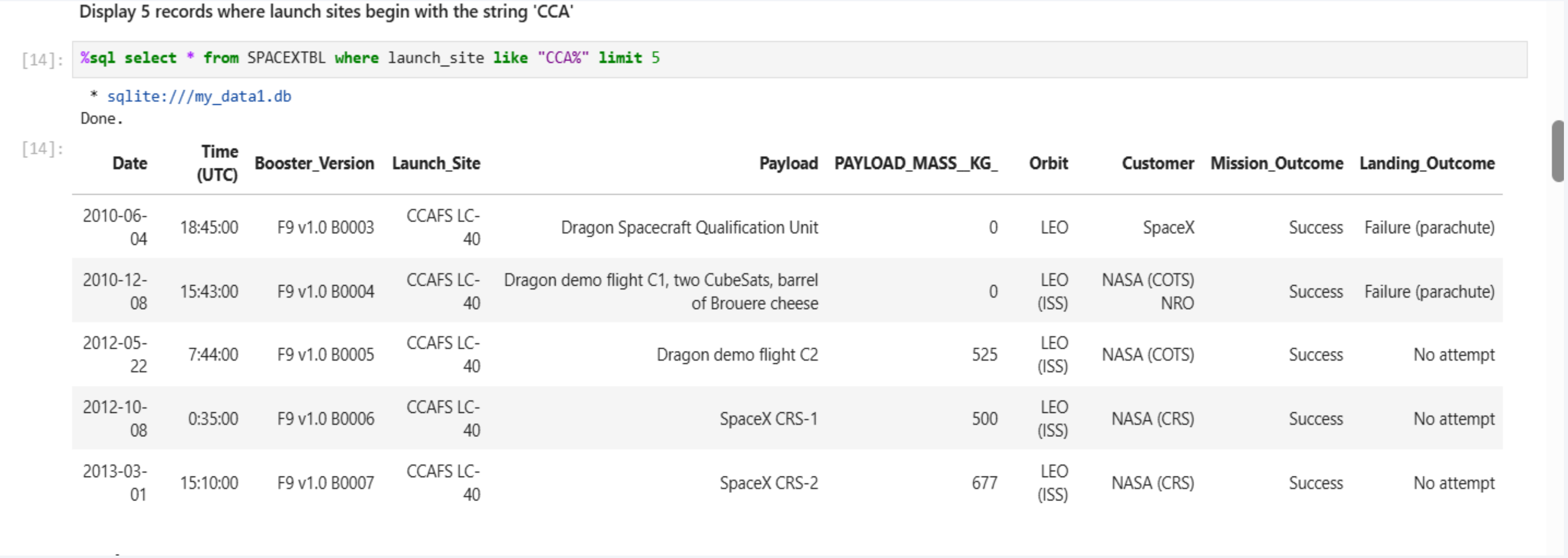
```
[27]: %sql select distinct(launch_site) from SPACEXTBL

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

[27]:

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`



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

```
[14]: %sql select * from SPACEXTBL where launch_site like "CCA%" limit 5
```

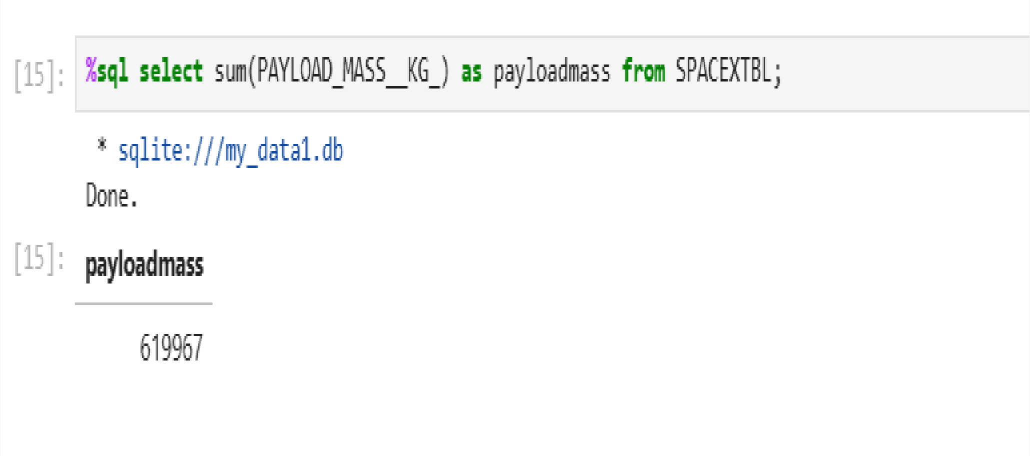
```
 * sqlite:///my_data1.db
Done.
```

[14]:

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

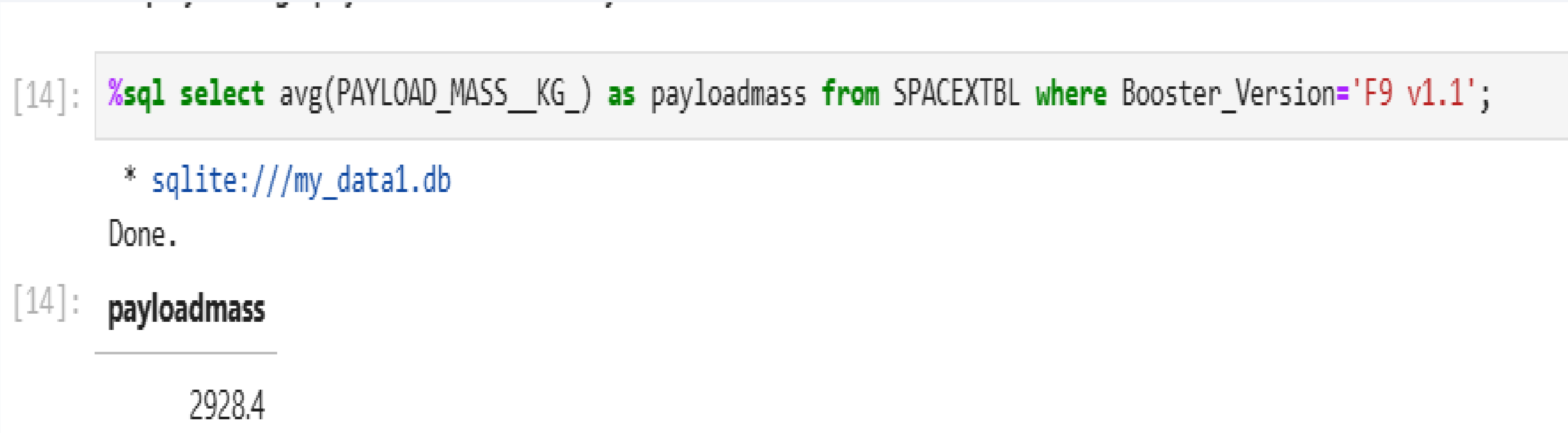
- Calculate the total payload carried by boosters from NASA



```
[15]: %sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;

 * sqlite:///my_data1.db
Done.

[15]: payloadmass

         619967
```

Average Payload Mass by F9 v1.1

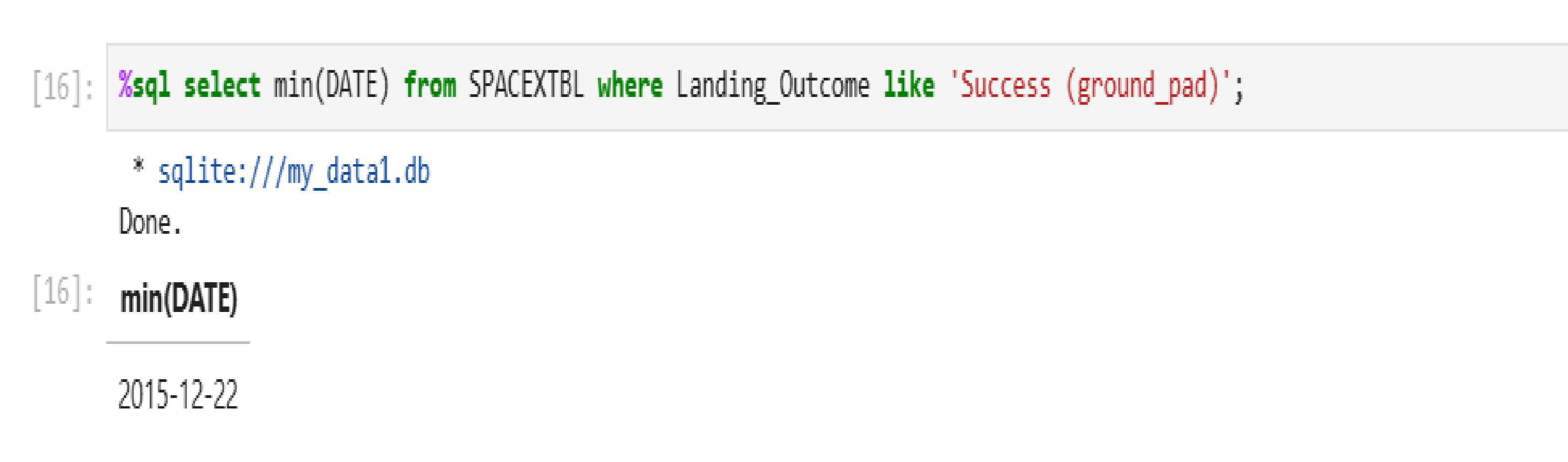
- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4



```
[14]: %sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL where Booster_Version='F9 v1.1';

 * sqlite:///my_data1.db
Done.

[14]: payloadmass
          2928.4
```

First Successful Ground Landing Date

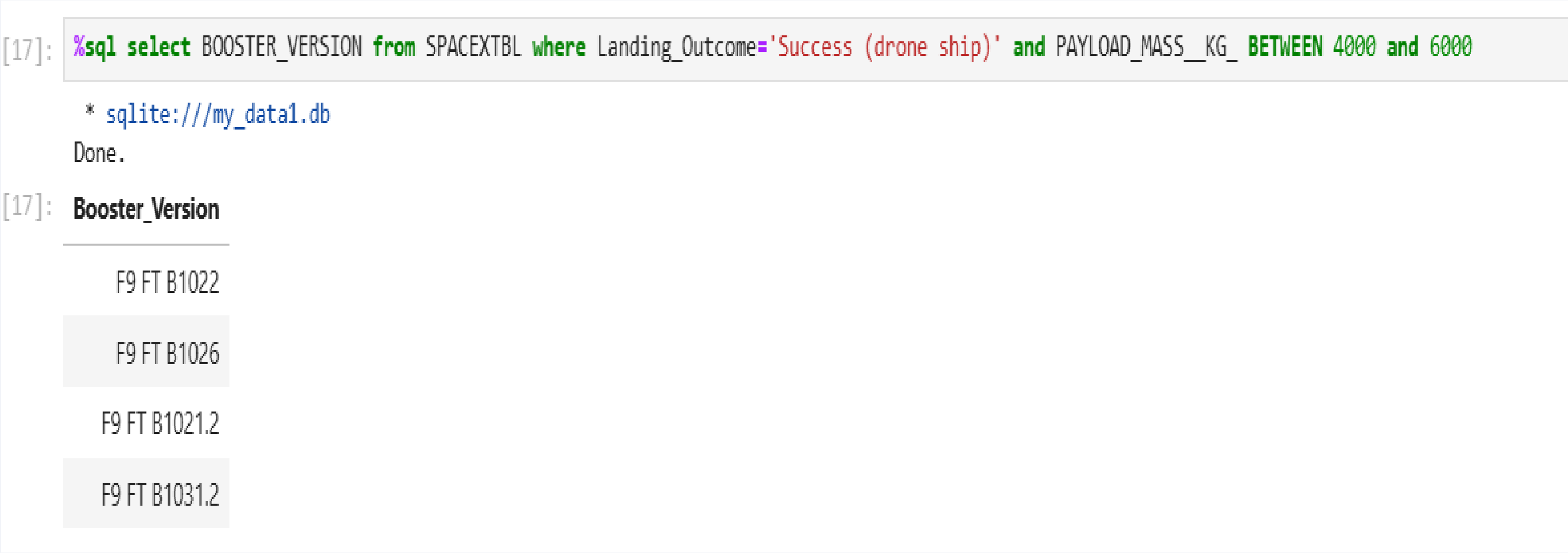
- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015



```
[16]: %sql select min(DATE) from SPACEXTBL where Landing_Outcome like 'Success (ground_pad)';

 * sqlite:///my_data1.db
Done.

[16]: min(DATE)

      2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the WHERE clause to filter for boosters which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000



```
[17]: %sql select BOOSTER_VERSION from SPACEXTBL where Landing_Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000
```

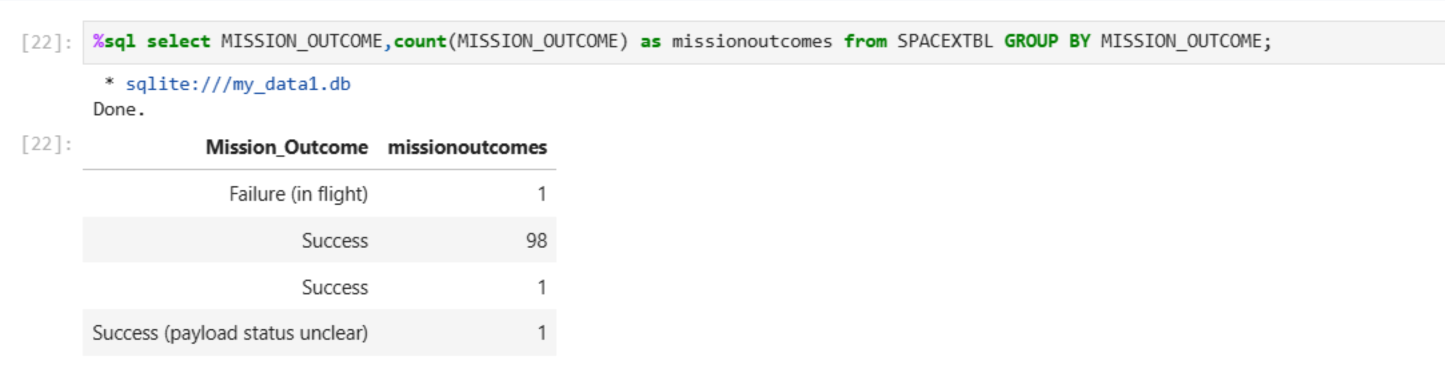
```
 * sqlite:///my_data1.db
Done.
```

[17]:

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

Total Number of Successful and Failure Mission Outcomes

- We used the group by to get success and failure

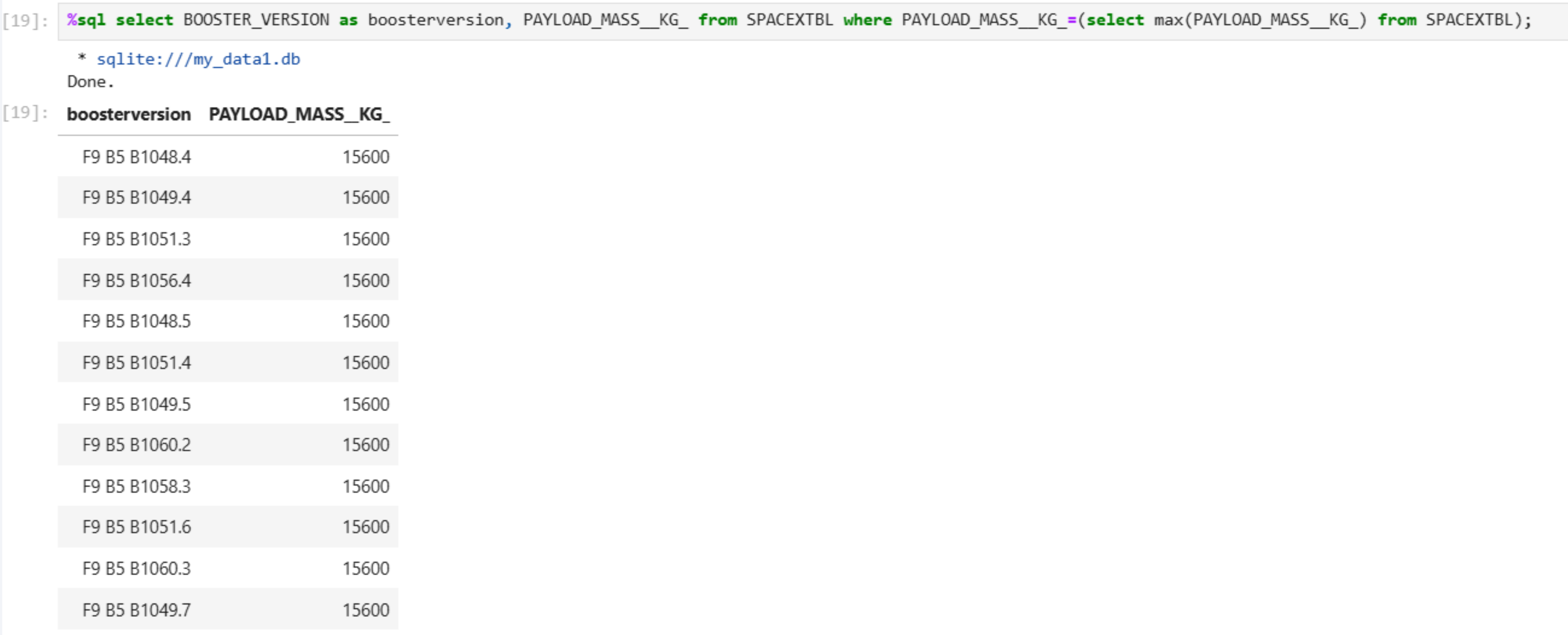


```
[22]: %sql select MISSION_OUTCOME,count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;

 * sqlite:///my_data1.db
Done.
[22]:
```

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

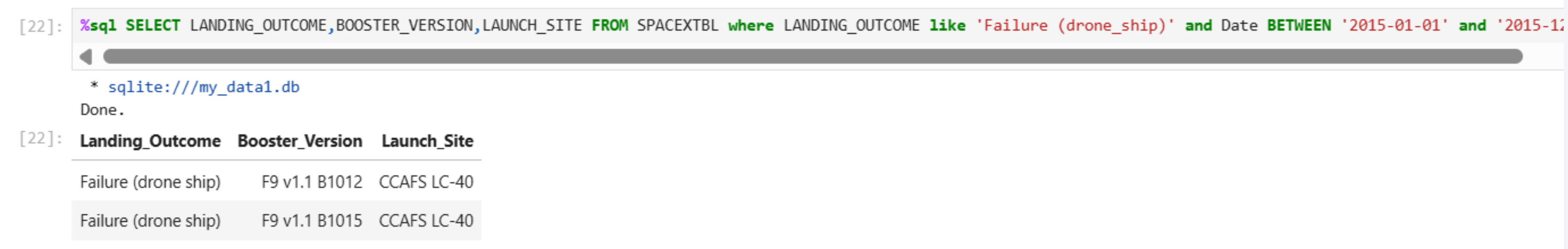


```
[19]: %sql select BOOSTER_VERSION as boosterversion, PAYLOAD_MASS__KG_ from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);

       * sqlite:///my_data1.db
      Done.
[19]:
```

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

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015



```
[22]: %sql SELECT LANDING_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where LANDING_OUTCOME like 'Failure (drone_ship)' and Date BETWEEN '2015-01-01' and '2015-1
```

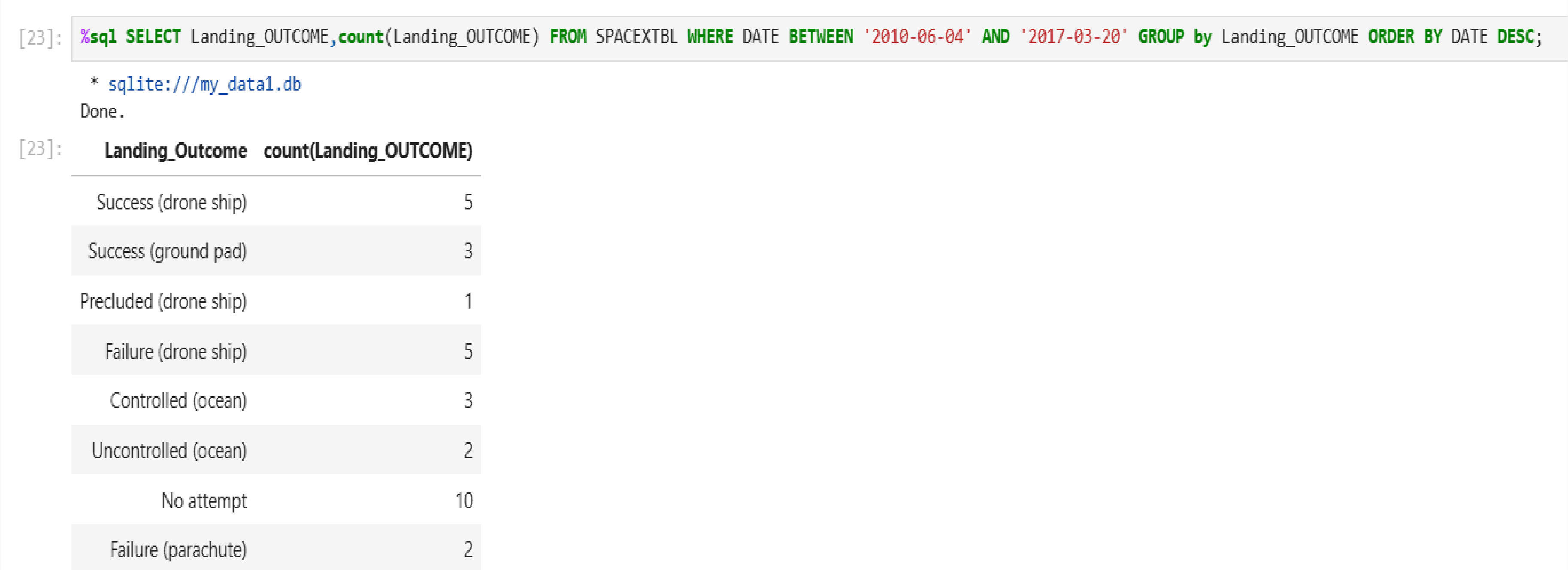
```
 * sqlite:///my_data1.db
Done.
```

[22]:

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



```
[23]: %sql SELECT Landing_OUTCOME,count(Landing_OUTCOME) FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP by Landing_OUTCOME ORDER BY DATE DESC;

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

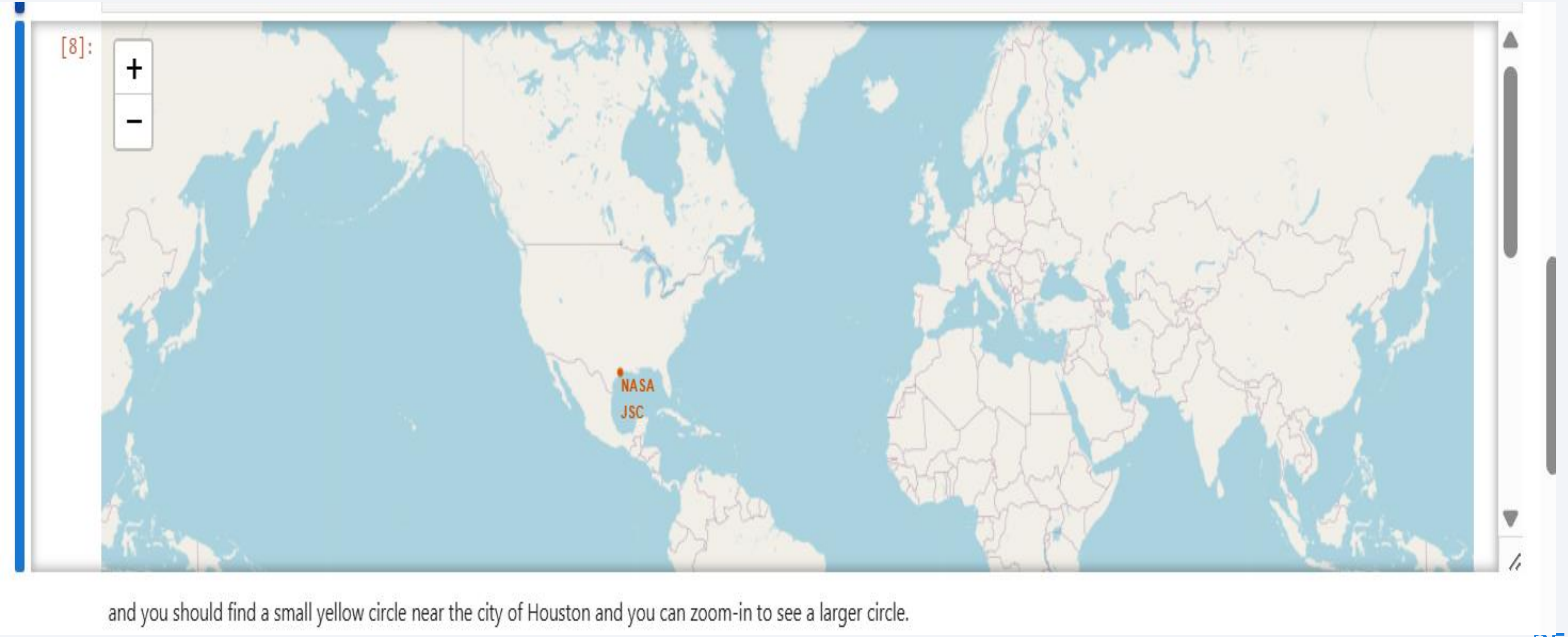
[23]:

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

Section 3

Launch Sites Proximities Analysis

All launch sites global map markers



and you should find a small yellow circle near the city of Houston and you can zoom-in to see a larger circle.

<Folium Map Screenshot 2>

- Replace <Folium map screenshot 2> title with an appropriate title

- Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map

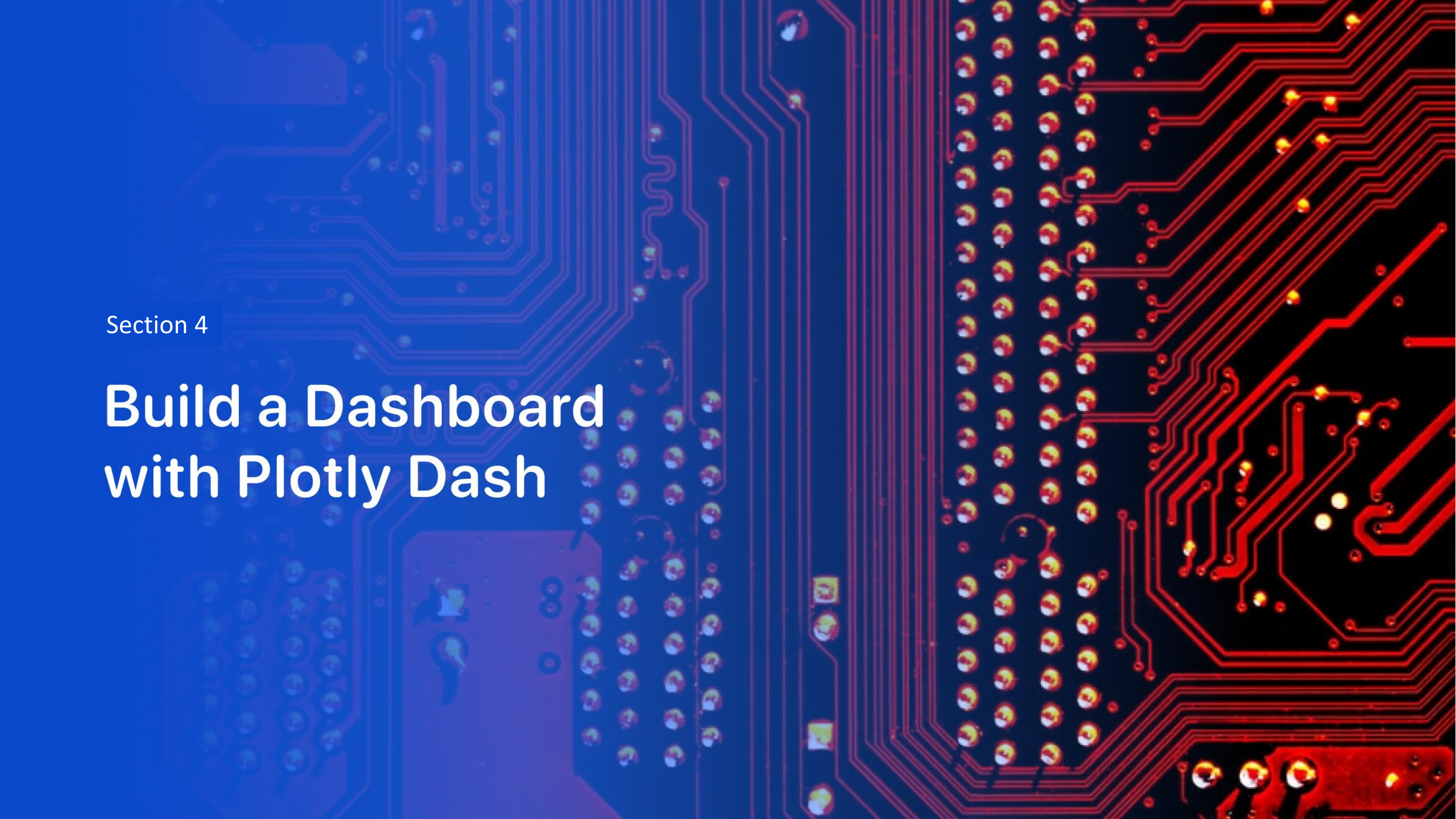
- Explain the important elements and findings on the screenshot

<Folium Map Screenshot 3>

- Replace <Folium map screenshot 3> title with an appropriate title

- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed

- Explain the important elements and findings on the screenshot

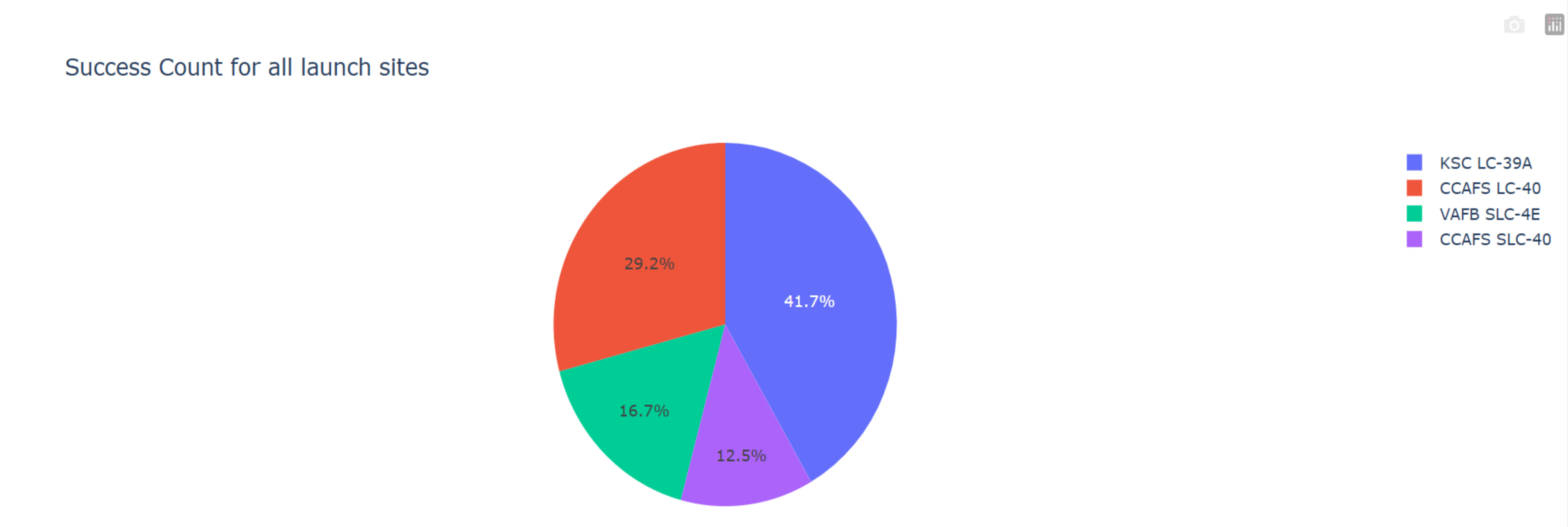


Section 4

Build a Dashboard with Plotly Dash

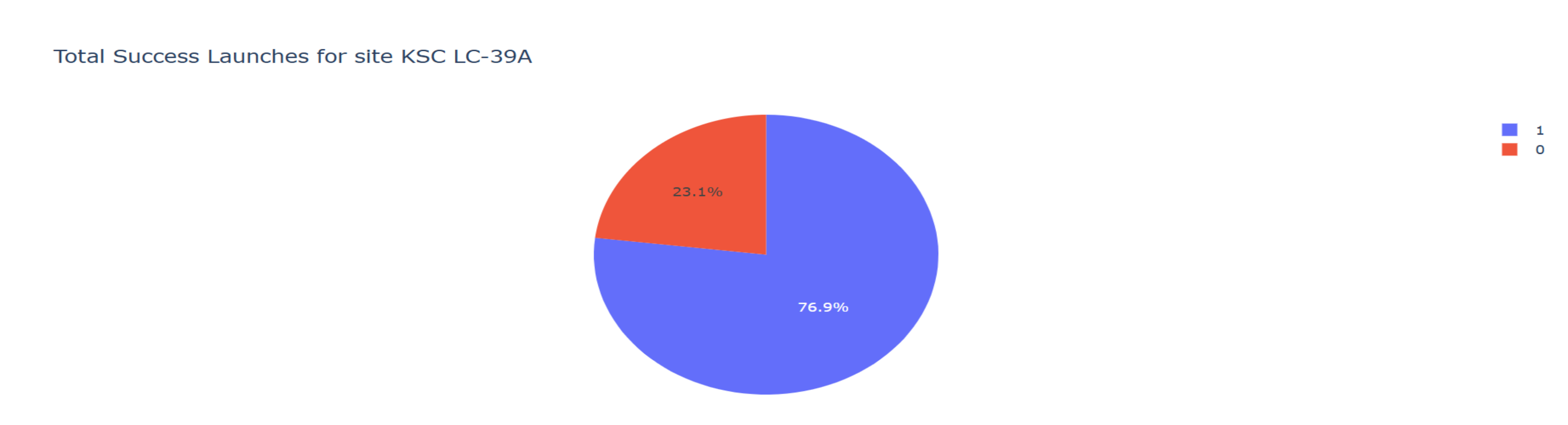
Pie chart showing the success percentage achieved by each launch site

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

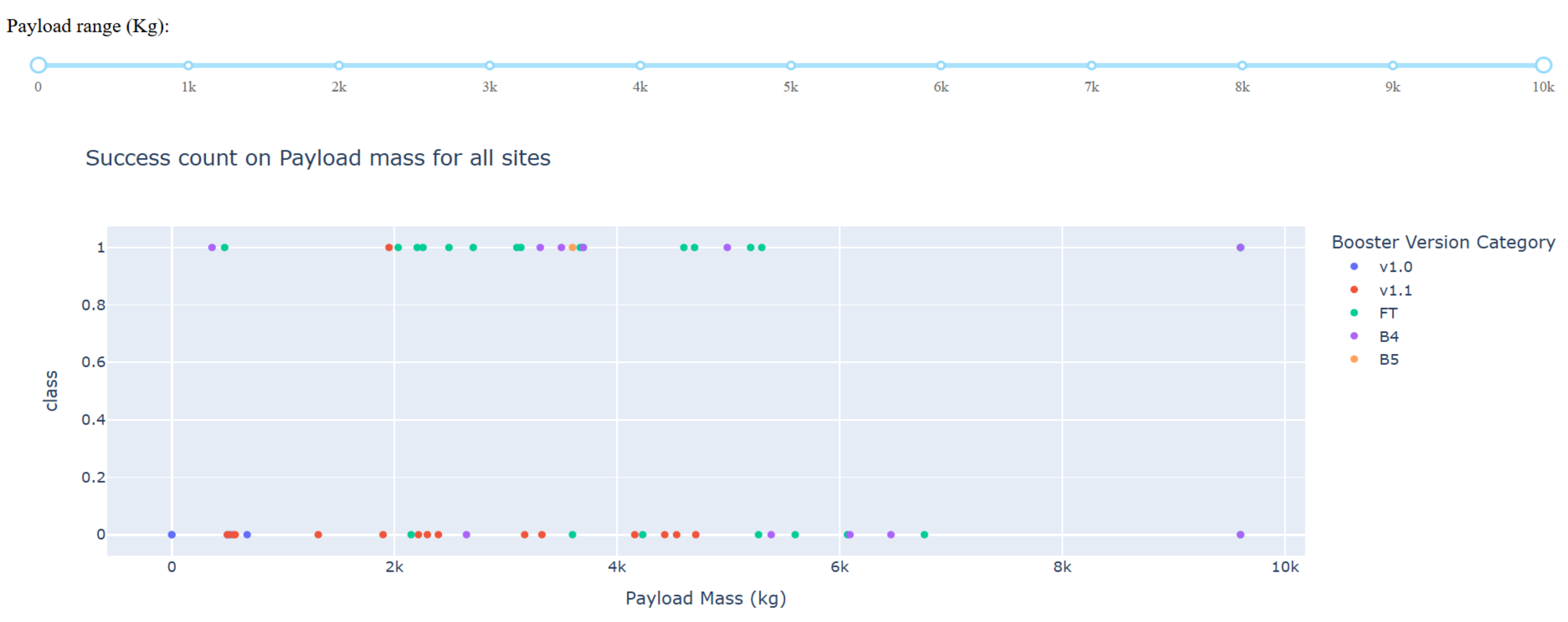


Pie chart showing the Launch site with the highest launch success ratio

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



Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



Payload range (Kg):

Section 5

Predictive Analysis (Classification)

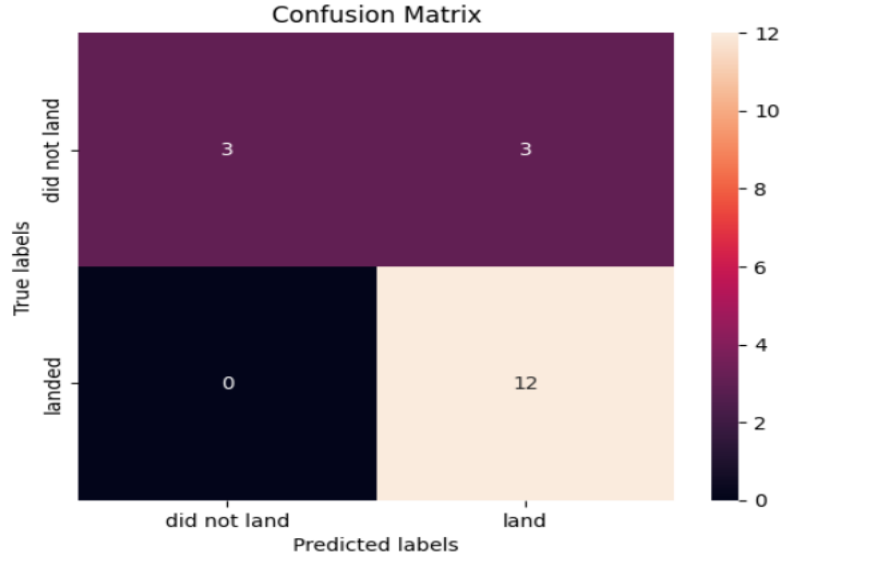
Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart

- Find which model has the highest classification accuracy

Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!