IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Oviedo Sandoval
Esteban Orlando

14/11/2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Data Analysis with SQL
  - Visual Analytic with Folium
  - Dashboard with Dash
  - Model prediction with Machine Learning
- Summary of all results
  - EDA Results
  - Interactive Analytics graphics
  - Correlations and predictive Modeling results

Introduction

- Project background and context
  - In this project, we worked on the space rocket launches by the company SpaceX, specifically on the Falcon 9. The aim is to determine if the rocket will be launched successfully. By knowing this, we can determine the cost of the launch and future launches
- Problems you want to find answers
  - What determines a successful launch?
  - What is the cost of those launches?

Section 1

Methodology

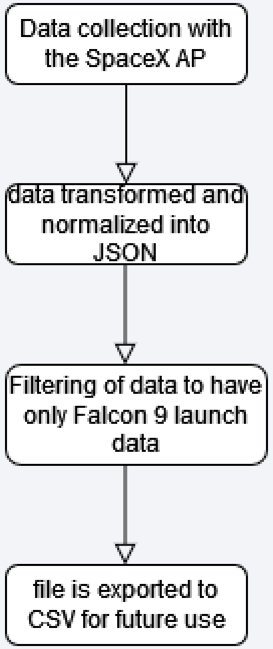
Methodology

Executive Summary

- Data collection methodology:
    - The SpaceX API was used along with web scraping
- Perform data wrangling
    - The data was collected and transformed into complex and disorganized datasets in more user-friendly formats for Machine Learning models
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - Different models, such as KNN or SVM, were created to evaluate which one provided the best results

Data Collection

The data is collected using the SpaceX API, and this data is transformed into JSON format and normalized. We filter the data to have information only on Falcon 9 rocket launches and export the file to CSV for future use



Data collection with the SpaceX AP

data transformed and normalized into JSON

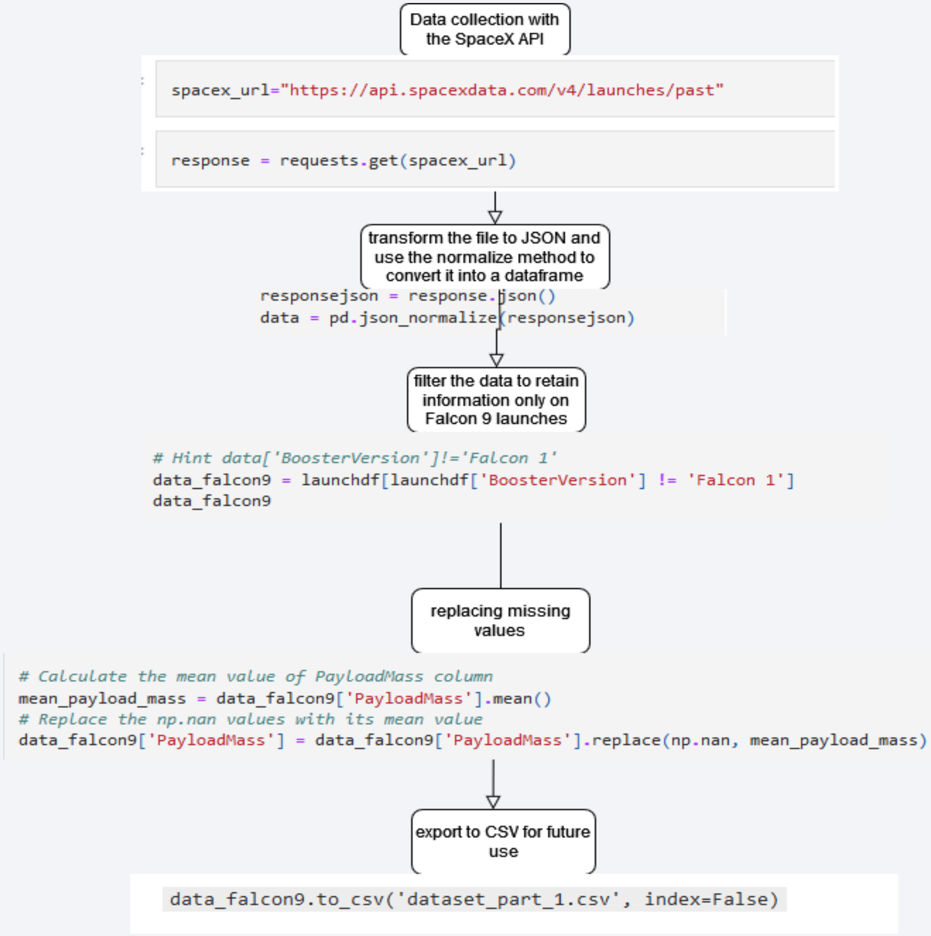
Filtering of data to have only Falcon 9 launch data

file is exported to CSV for future use

Data Collection – SpaceX API

- I require the data from the SpaceX API. We transform the file to JSON and use the normalize method to convert it into a data frame. We filter the data to retain information only on Falcon 9 launches, remove any missing values, and export to CSV for future use.

- GitHub url: https://github.com/Heleiley/Final-Assignment/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



Data collection with the SpaceX API

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

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

transform the file to JSON and use the normalize method to convert it into a dataframe

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

filter the data to retain information only on Falcon 9 launches

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launchdf[launchdf['BoosterVersion'] != 'Falcon 1']
data_falcon9
```

replacing missing values

```
# Calculate the mean value of PayloadMass column
mean_payload_mass = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, mean_payload_mass)
```

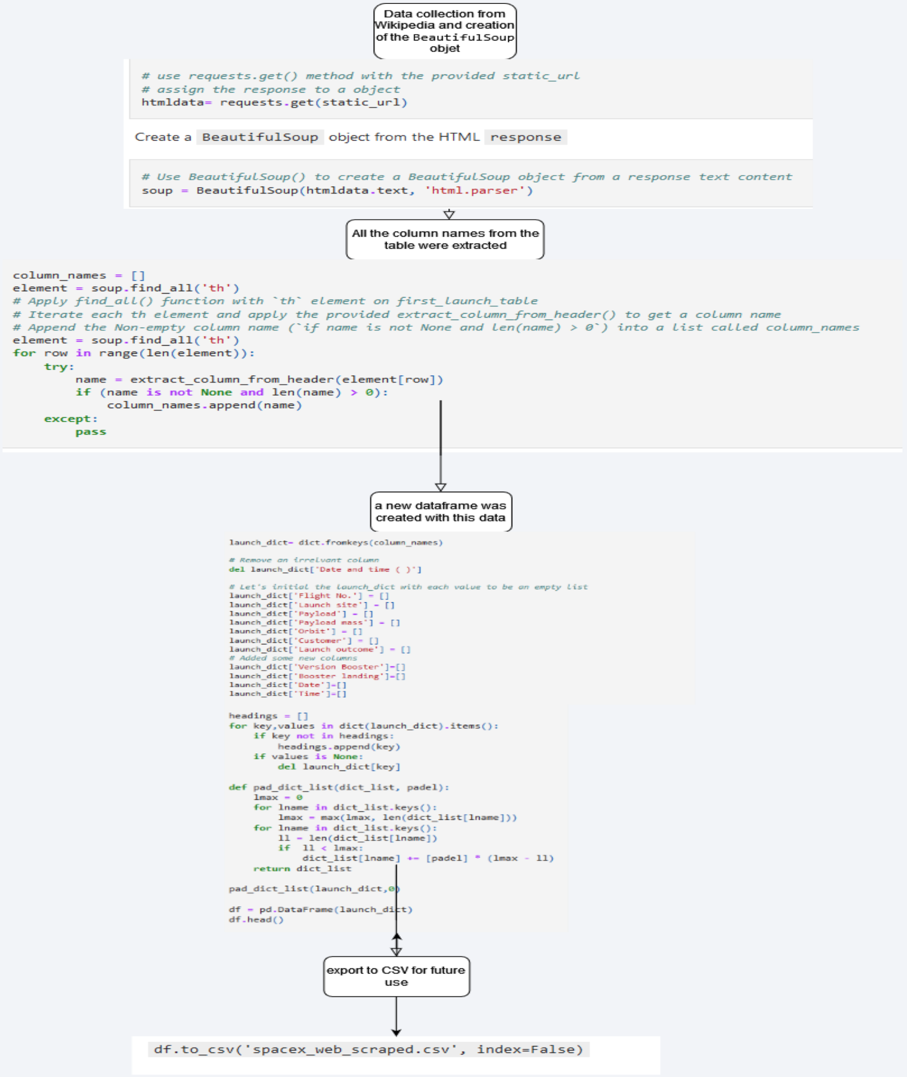
export to CSV for future use

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

- Data from Wikipedia about SpaceX's Falcon 9 launches was used. All the column names from the table were extracted, a new data frame was created with this data, and this data frame was exported to CSV for future use

- GitHub URL: https://github.com/Heleiley/Final-Assignment/blob/main/jupyter-labs-webscraping.ipynb



Data collection from Wikipedia and creation of the BeautifulSoup objet

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

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(htmldata.text, 'html.parser')
```

All the column names from the table were extracted

```
column_names = []
element = soup.find_all('th')
# 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
```

a new dataframe was created with this data

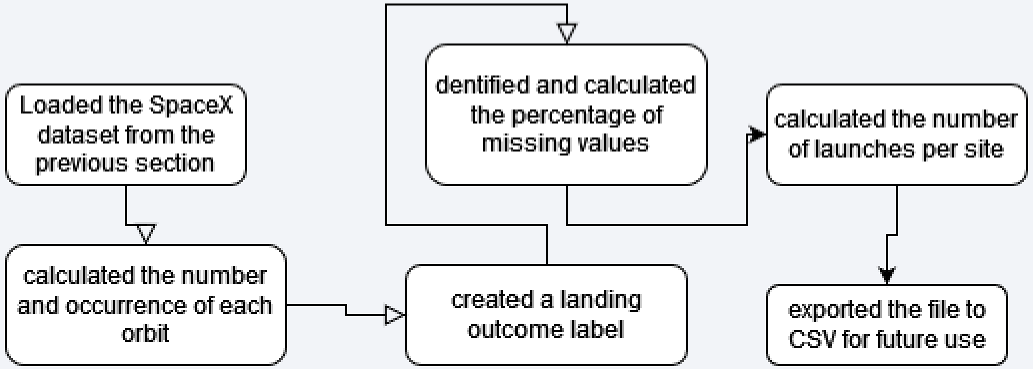
```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]

headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

df = pd.DataFrame(launch_dict)
df.head()
```

export to CSV for future use

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

- We loaded the SpaceX dataset from the previous section, identified and calculated the percentage of missing values, calculated the number of launches per site, calculated the number and occurrence of each orbit, calculated the number of missions per type of orbit, created a landing outcome label from the Outcome column, and exported the file to CSV for future use
- GitHub URL:https://github.com/Heleiley/Final-Assignment/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



Loaded the SpaceX dataset from the previous section

calculated the number and occurrence of each orbit

created a landing outcome label

dentified and calculated the percentage of missing values

calculated the number of launches per site

exported the file to CSV for future use

EDA with Data Visualization

- We used the following charts: scatter, bar chart, point and line chart. These were used because they better highlight the relationships between the different variables utilized and better show comparisons
- GitHub URL: https://github.com/Heleiley/Final-Assignment/blob/main/edadataviz.ipynb

EDA with SQL

- SQL queries Performed
  - Display the names of the unique launch sites
  - Display 5 records where launch sites begin with the string 'CCA'
  - Display the total payload mass carried by boosters launched by NASA (CRS)
  - Display average payload mass carried by booster version F9 v1.1
  - List the date when the first succesful landing outcome in ground pad was acheived.
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - List the total number of successful and failure mission outcomes
  - List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
  - 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.
  - 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.
- GitHub URL: https://github.com/Heleiley/Final-Assignment/blob/main/jupyter-labs-eda-sql-coursera_sqllite(1).ipynb

Build an Interactive Map with Folium

- created and used the following objects in the Folium map
  - Markers: to create marks on the map for example, launch sites
  - Circle: used to create circles over the marks
  - cons: used to create icons on the map
  - PolyLine: created to draw lines between marks or points
  - MarkerCluster: created for those points where there are many markers with the same coordinates

- GitHub URL: https://github.com/Heleiley/Final-Assignment/blob/main/lab_jupyter_launch_site_location.ipynb

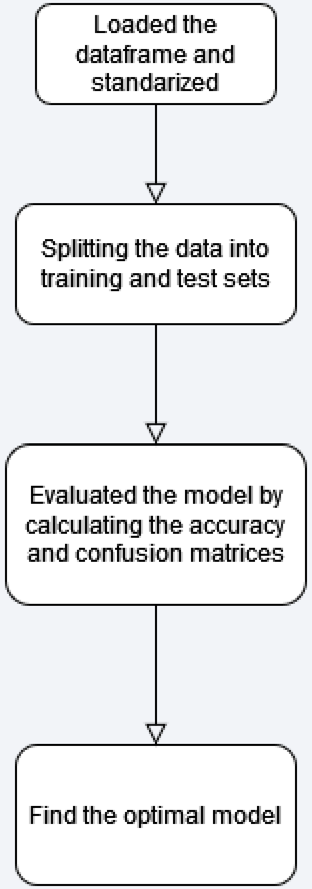
Build a Dashboard with Plotly Dash

- I created an interactive dashboard with Plotly and Dash. Some pie charts were created to show the total launches from certain sites, and a scatter plot was also created to show the relationship between payload mass and the class used

- GitHub URL: https://github.com/Heleiley/Final-Assignment/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- I created the model by creating columns for the class, standardizing the data, and splitting the data into training and test sets. We evaluated the model by calculating the accuracy and confusion matrices, and then plotted the results. Finally, we aimed to find the optimal model by searching for the best hyperparameters and comparing the models with the highest accuracy
- GitHub URL: https://github.com/Heleiley/Final-Assignment/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5(1).ipynb



Loaded the dataframe and standarized

Splitting the data into training and test sets

Evaluated the model by calculating the accuracy and confusion matrices

Find the optimal model

Results

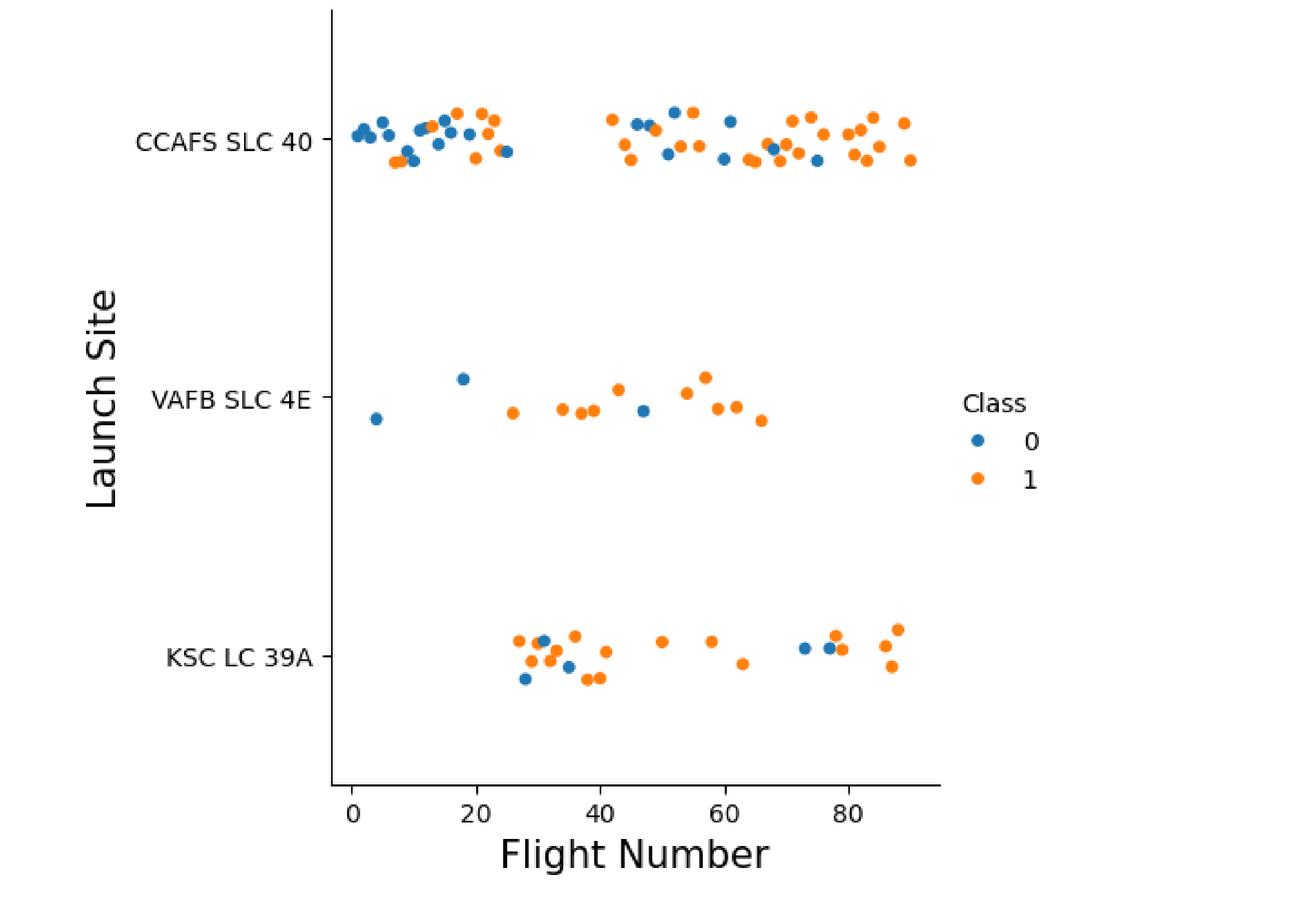
- It can be concluded that both the API and web scraping are capable of collecting and filtering SpaceX data that was later reused in subsequent labs. However, EDA with SQL was more useful for filtering the data as we could use specific functions for this purpose. Regarding the maps, we were able to see graphical data of launch sites, distances between them, and distances between these sites and different urban areas, which would have been difficult to understand without using Folium. With Dash, we were able to visualize the launch success rate by site thanks to the pie charts. Regarding the predictive analysis, we can conclude that considering the model scores, the decision tree is the best for analysis

Section 2

Insights drawn from EDA

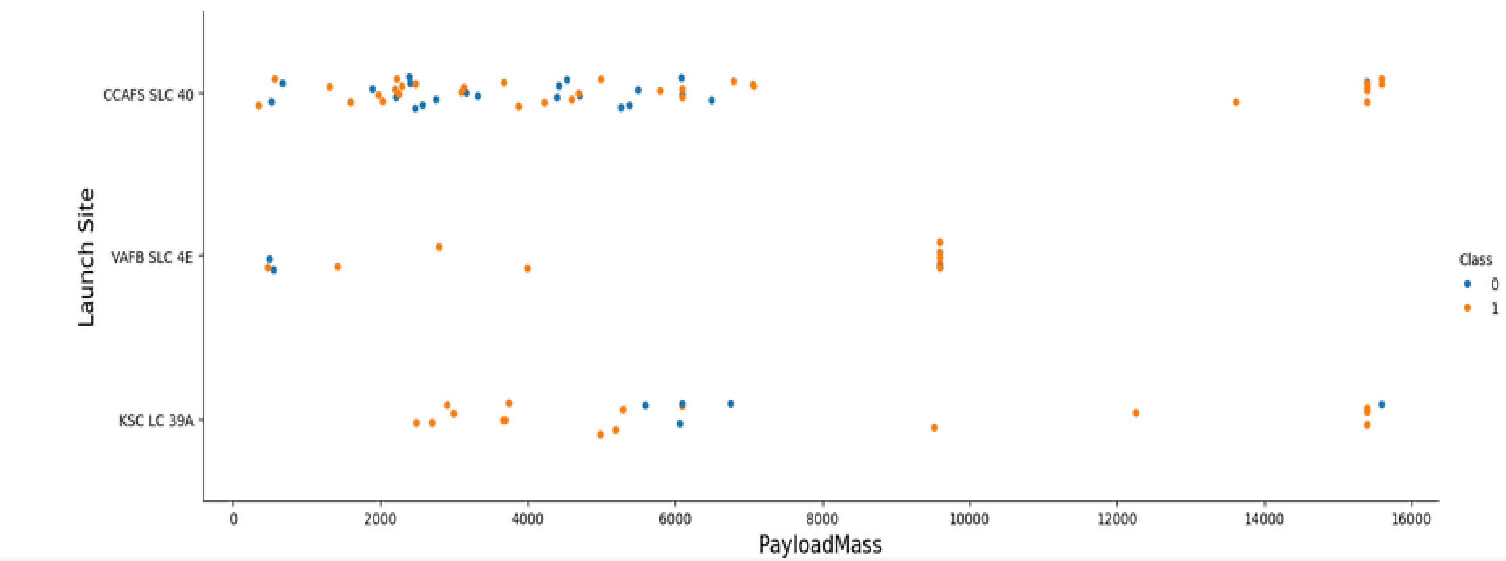
Flight Number vs. Launch Site

- We can visualize that the CCFS SLC 40 site has the highest number of successful launches, considering that Class 0 represents failed launches and Class 1 represents successful ones



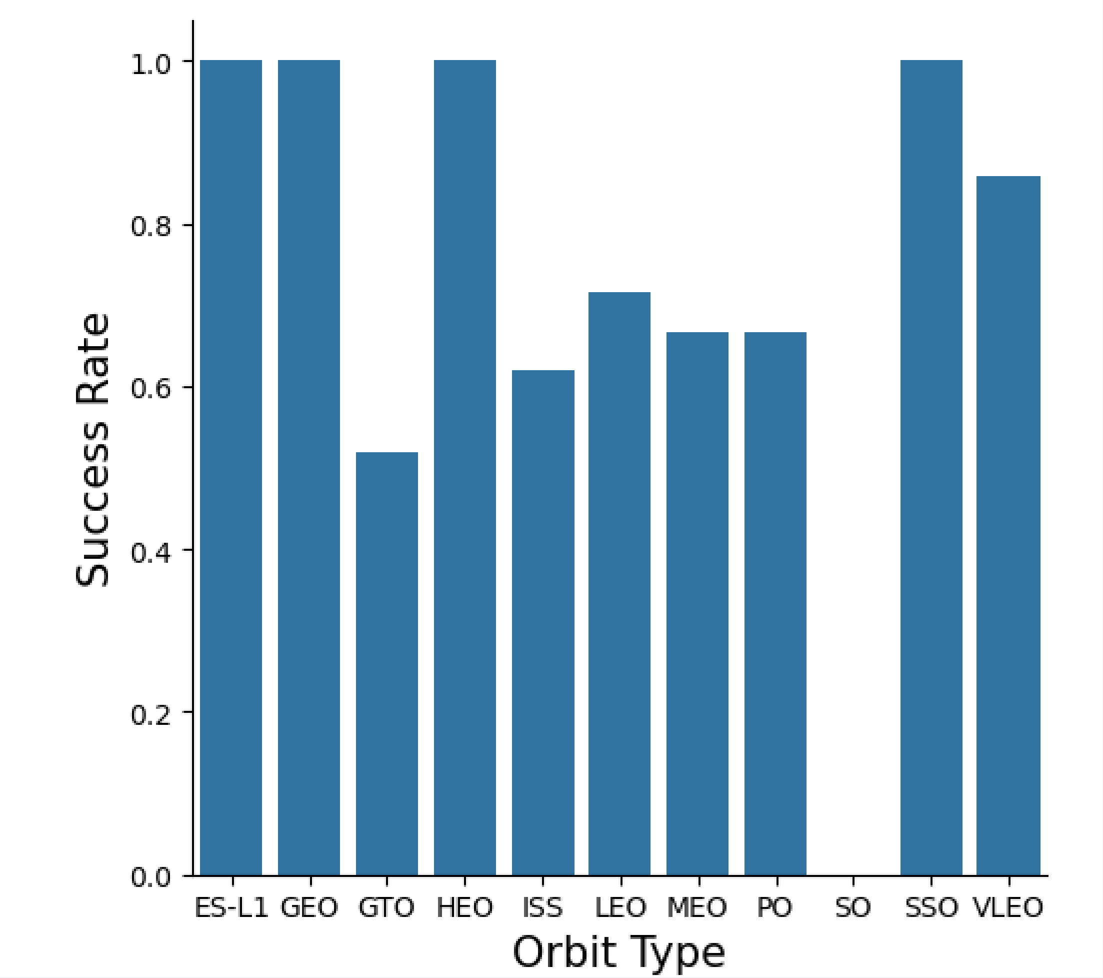
Payload vs. Launch Site

- "We can visualize that the majority of launches with low payload mass were from the CCAFS SLC 40 site. It is also the site with the most failed launches with low payload mass



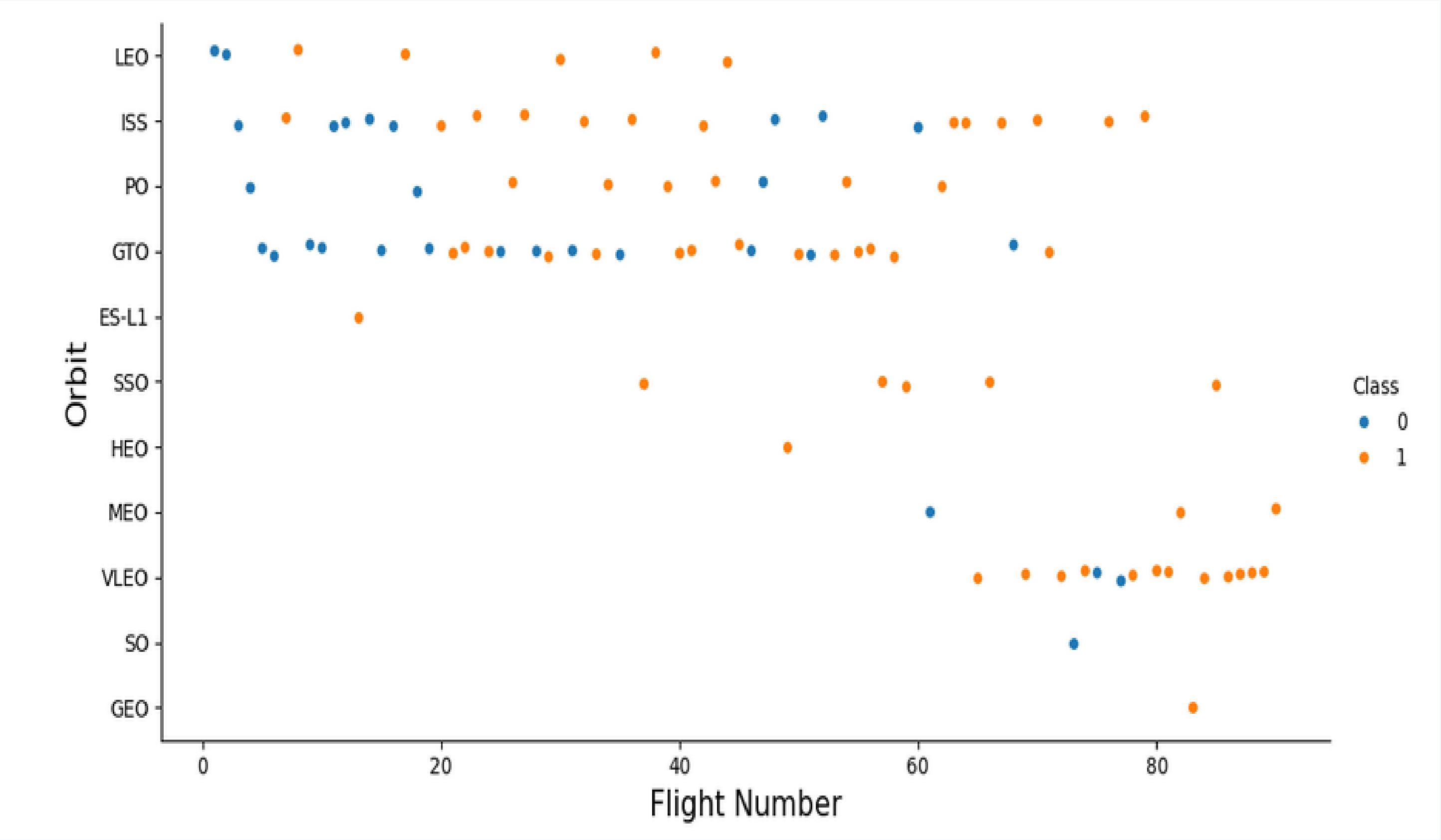
Success Rate vs. Orbit Type

- We can visualize that the ES-L1, GEO, HEO, and SSO orbits have the highest successful launch ratio, while the SO orbit did not achieve any successful launches.



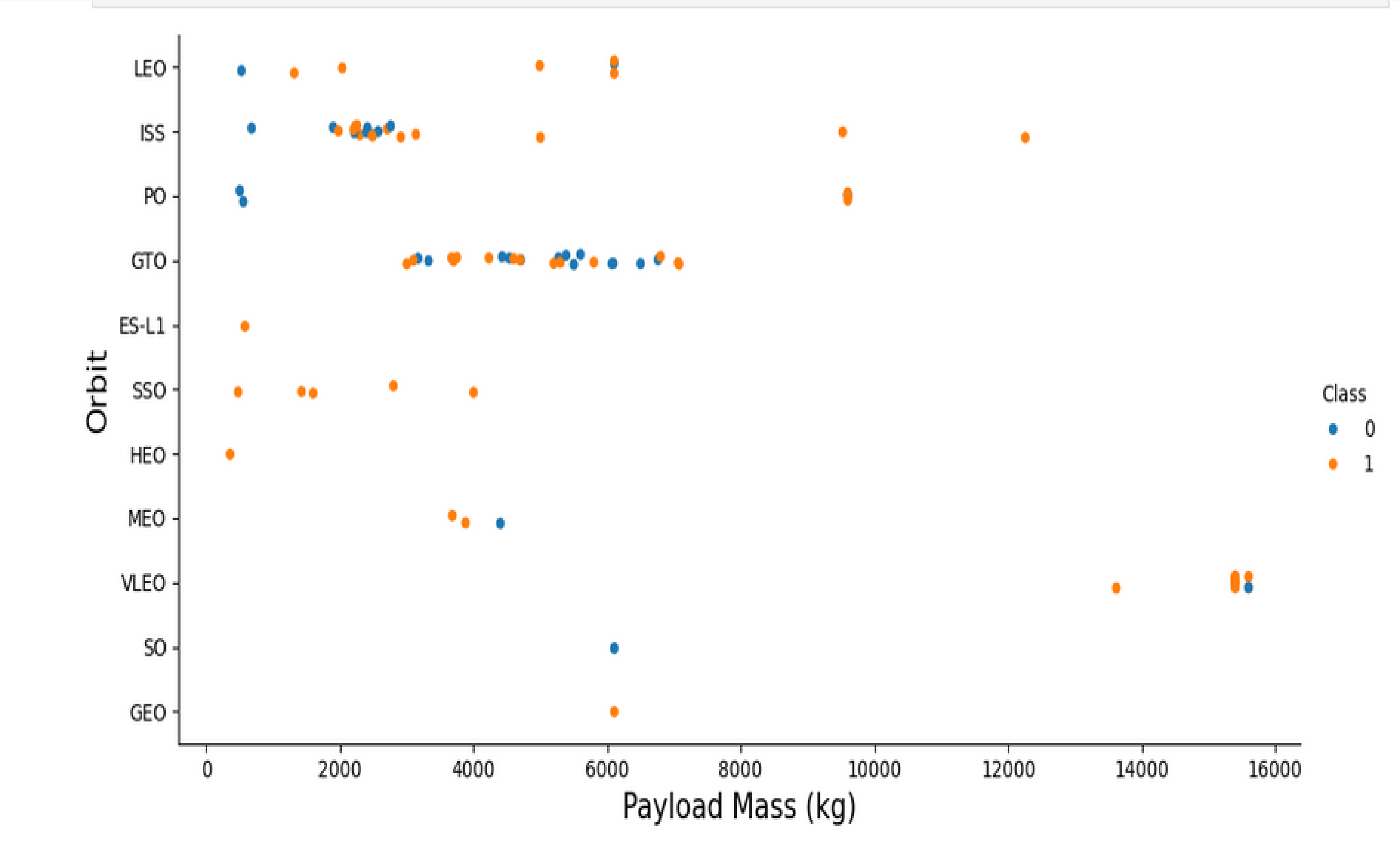
Flight Number vs. Orbit Type

- We can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.



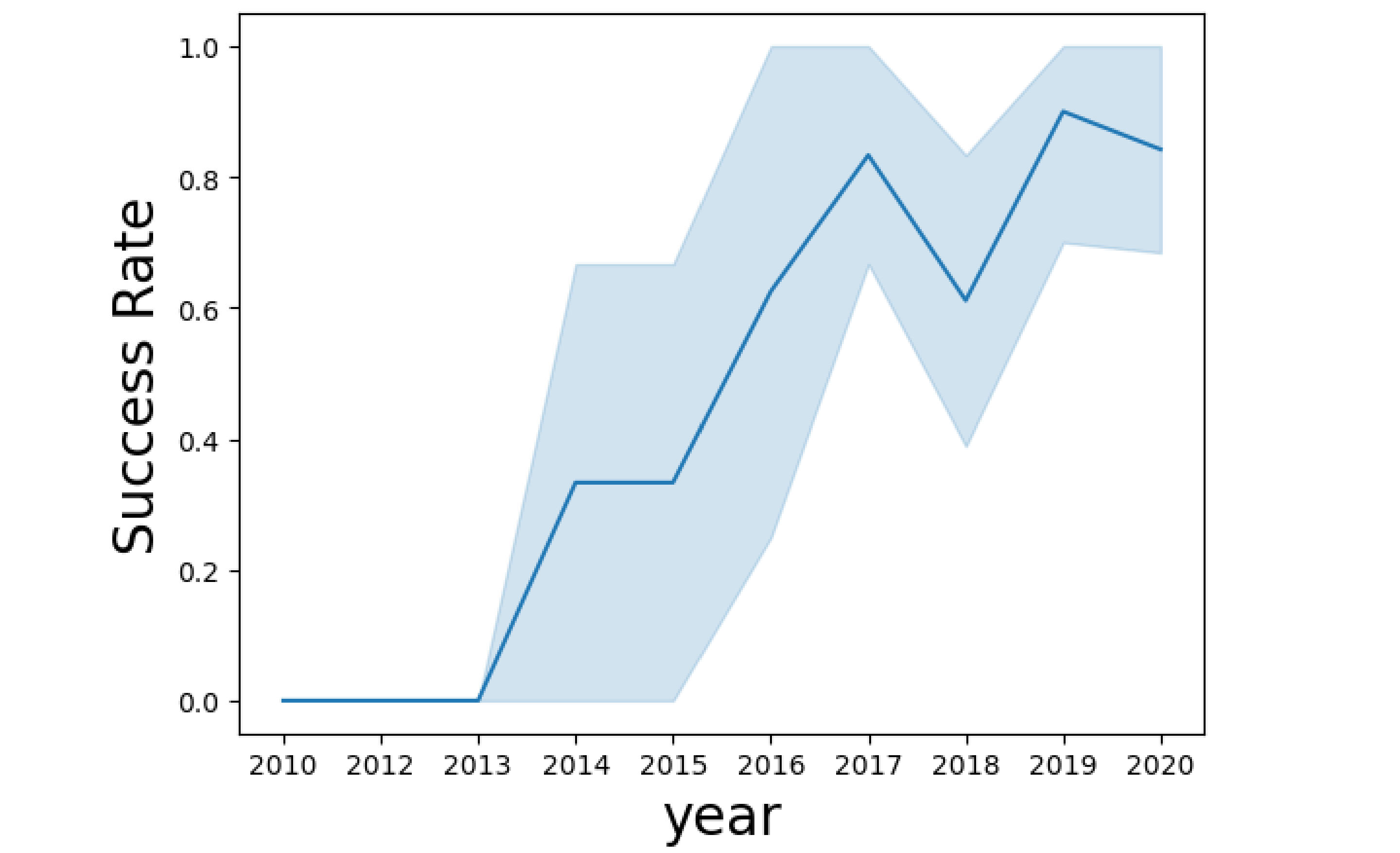
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS. However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present



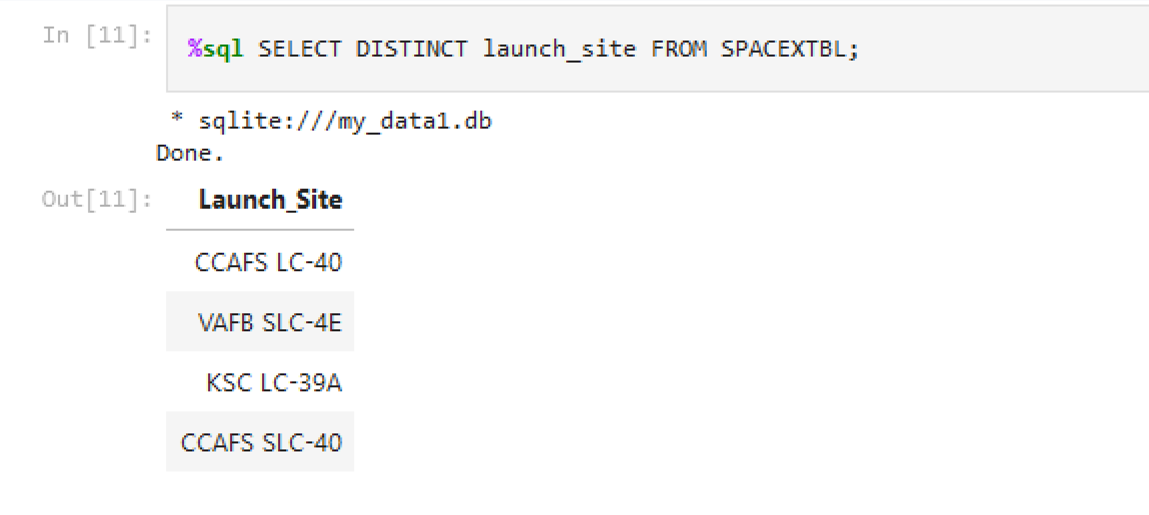
Launch Success Yearly Trend

- you can observe that the success rate since 2013 kept increasing till 2020



All Launch Site Names

- We used the 'distinct' keyword to see only the unique launch sites.



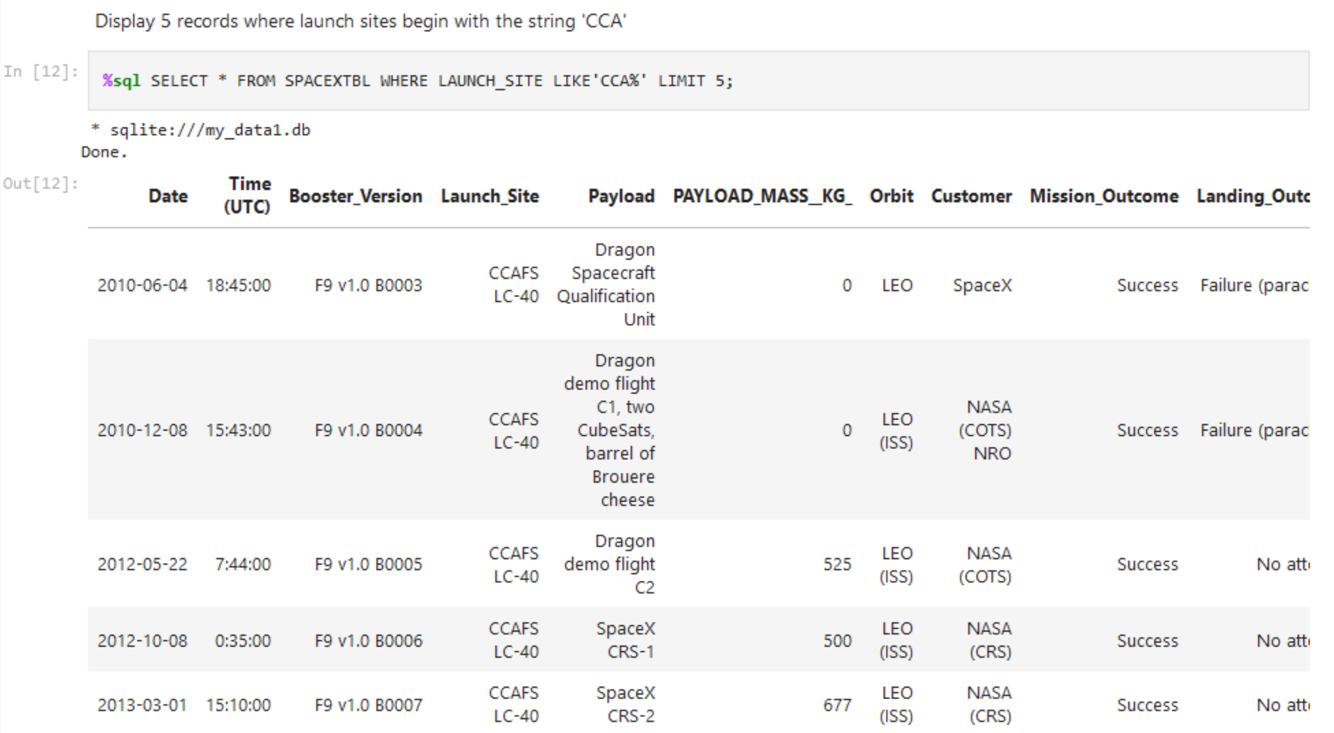
```
In [11]: %sql SELECT DISTINCT launch_site FROM SPACEXTBL;

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

Out[11]:

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

Launch Site Names Begin with 'CCA'

- We used the 'like' keyword to search within launch_site for launch sites that have CCA in their name And we limited it to 5 to show only the first 5 results.



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

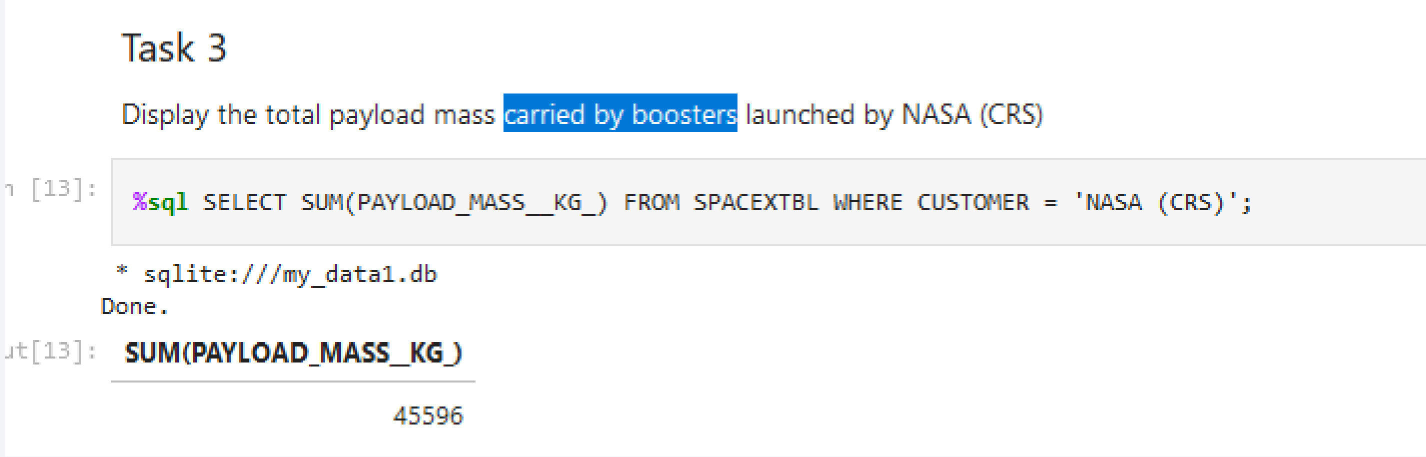
```
In [12]: %sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE'CCA%' LIMIT 5;

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

Out[12]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parac
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 (parac
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No att
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No att
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No att

Total Payload Mass

- we used the SUM() function to find out the total payload mass carried by boosters



Task 3

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

```
n [13]: %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
```

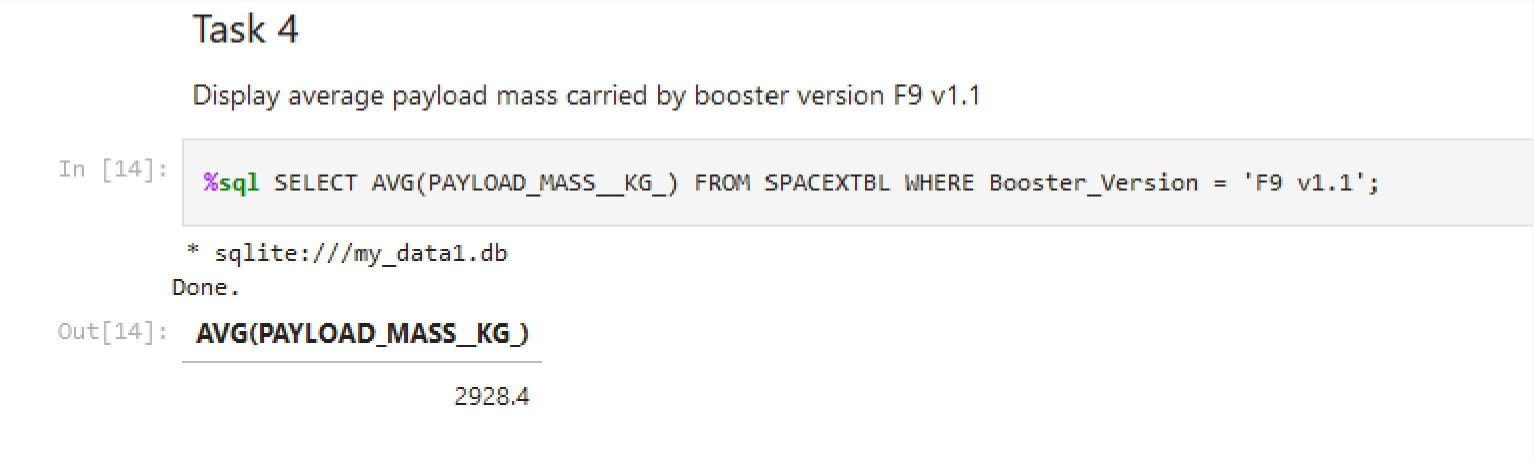
```
 * sqlite:///my_data1.db
Done.
```

ut[13]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- We used the AVG() function to calculate the amount of mass carried by the booster version F9 v1.1 using the 'where booster_Version = F9 V1.1' clause



Task 4

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

In [14]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version = 'F9 v1.1';
```

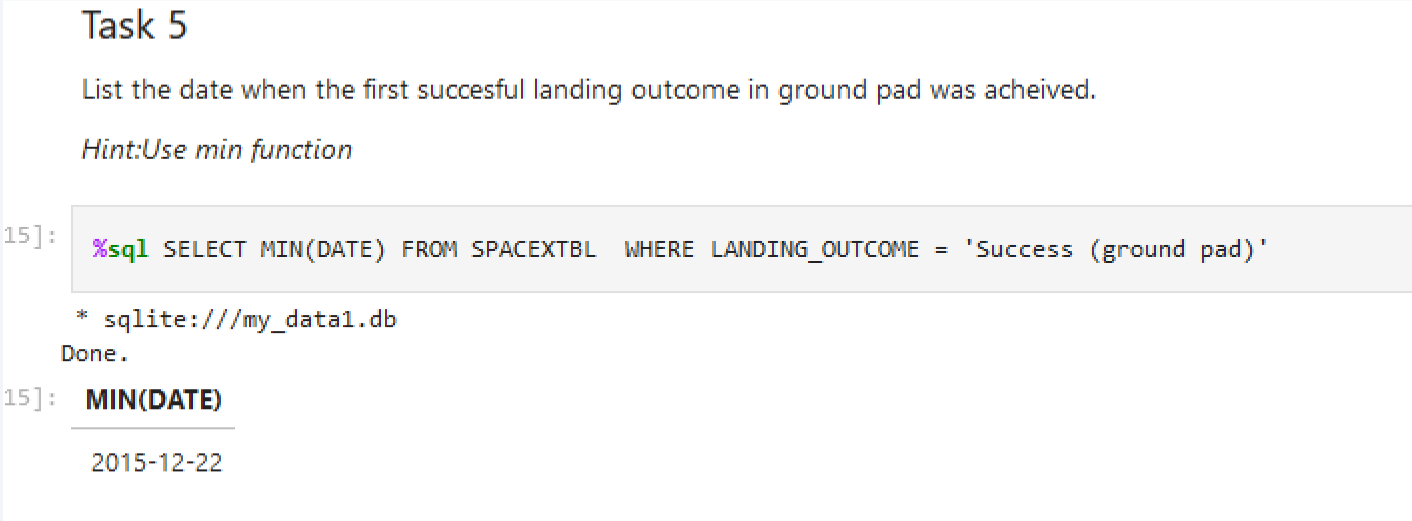
```
 * sqlite:///my_data1.db
Done.
```

Out[14]:

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- With MIN(DATE), we can find the date of the first successful landing



Task 5

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

Hint:Use min function

```
15]:  %sql SELECT MIN(DATE) FROM SPACEXTBL  WHERE LANDING_OUTCOME = 'Success (ground pad)'
```

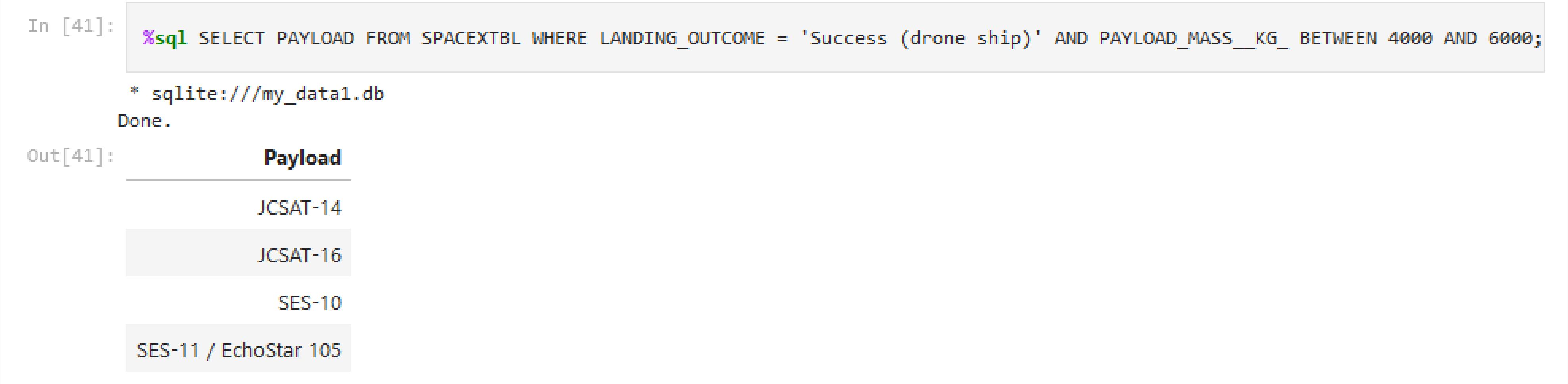
```
 * sqlite:///my_data1.db
Done.
```

15]:

MIN(DATE)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We searched only for successful drone ship landings using the 'Where' clause and restricted the payload mass to be greater than 4000 but less than 6000

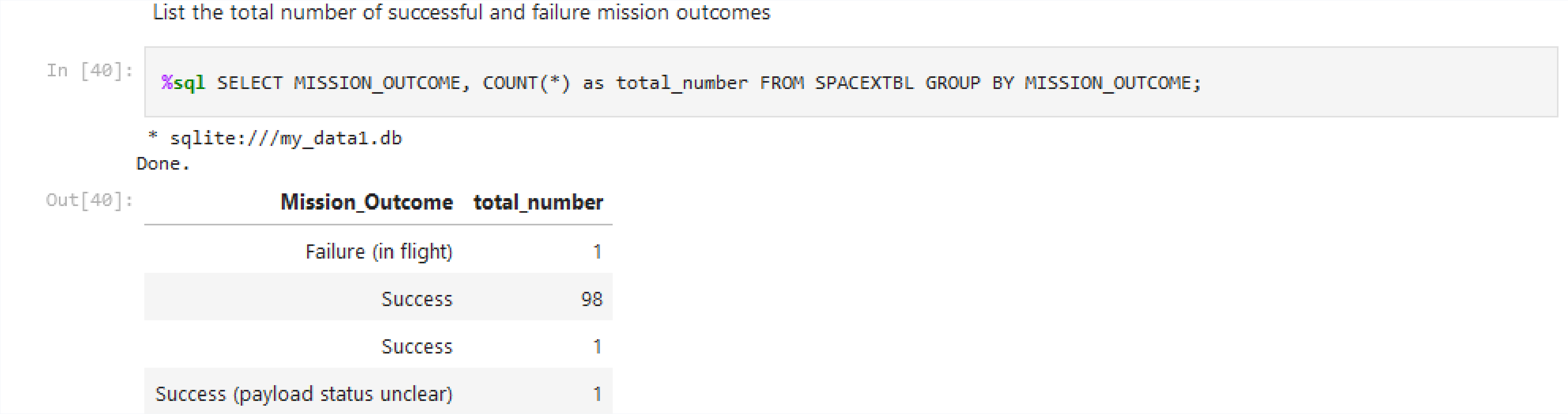


```
In [41]:  %sql SELECT PAYLOAD FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;

           * sqlite:///my_data1.db
          Done.
Out[41]:
```

Payload
JCSAT-14
JCSAT-16
SES-10
SES-11 / EchoStar 105

Total Number of Successful and Failure Mission Outcomes

- The query counts the number of failed and successful missions outcome



List the total number of successful and failure mission outcomes

In [40]:

```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

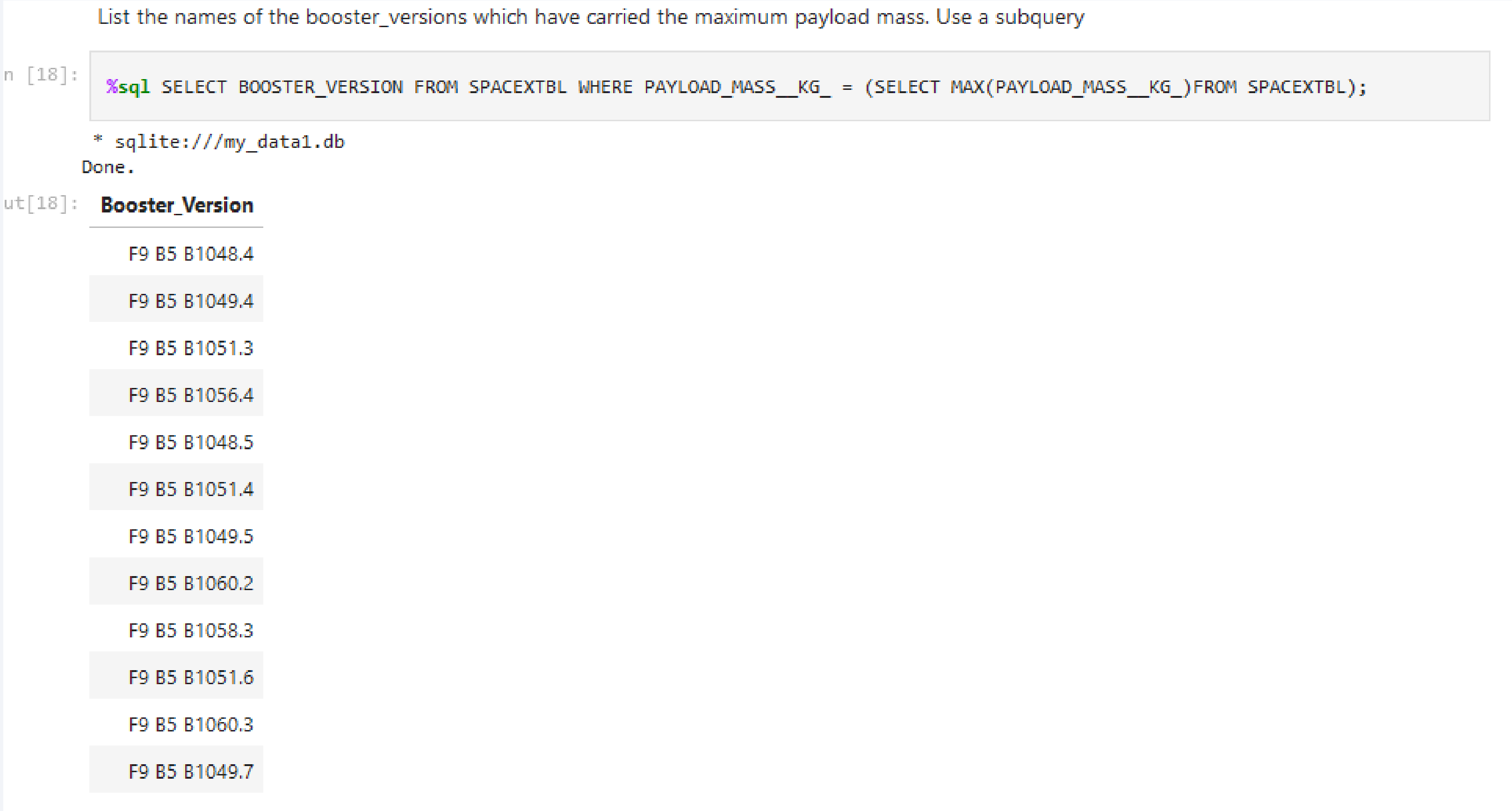
```
 * sqlite:///my_data1.db
Done.
```

Out[40]:

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

Boosters Carried Maximum Payload

- We can find out which booster versions had the maximum payload mass using the SELECT MAX(payload_mass_kg) statement



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

```
n [18]:  %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_)FROM SPACEXTBL);
```

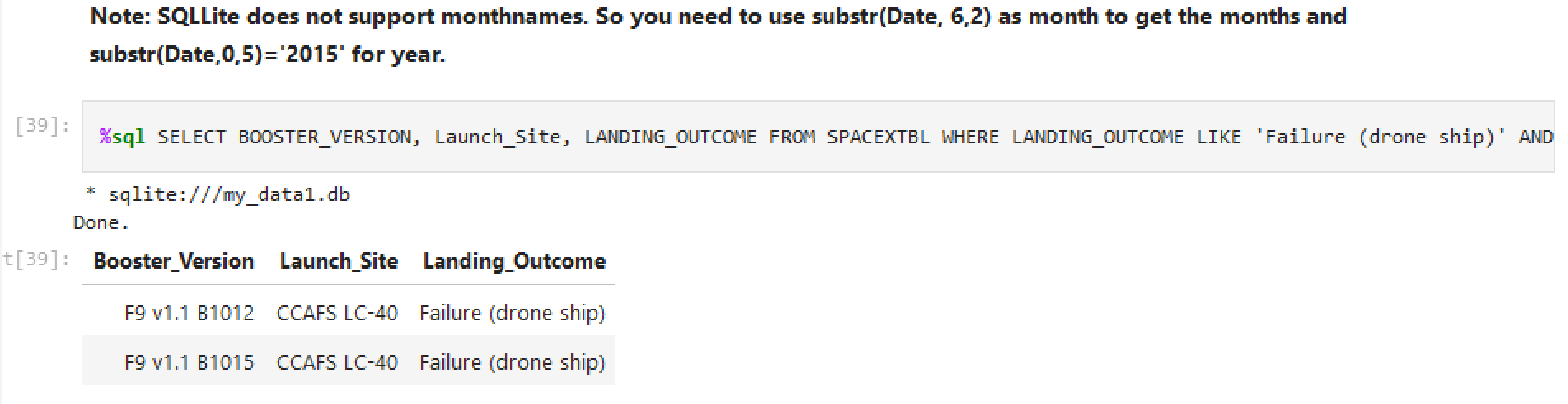
```
 * sqlite:///my_data1.db
Done.
```

ut[18]:

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

2015 Launch Records

- We used combinations of where, like, and to filter the failed landings on drone ships, their booster versions, and the launch site.



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.

```
[39]: %sql SELECT BOOSTER_VERSION, Launch_Site, LANDING_OUTCOME FROM SPACEXTBL WHERE LANDING_OUTCOME LIKE 'Failure (drone ship)' AND
```

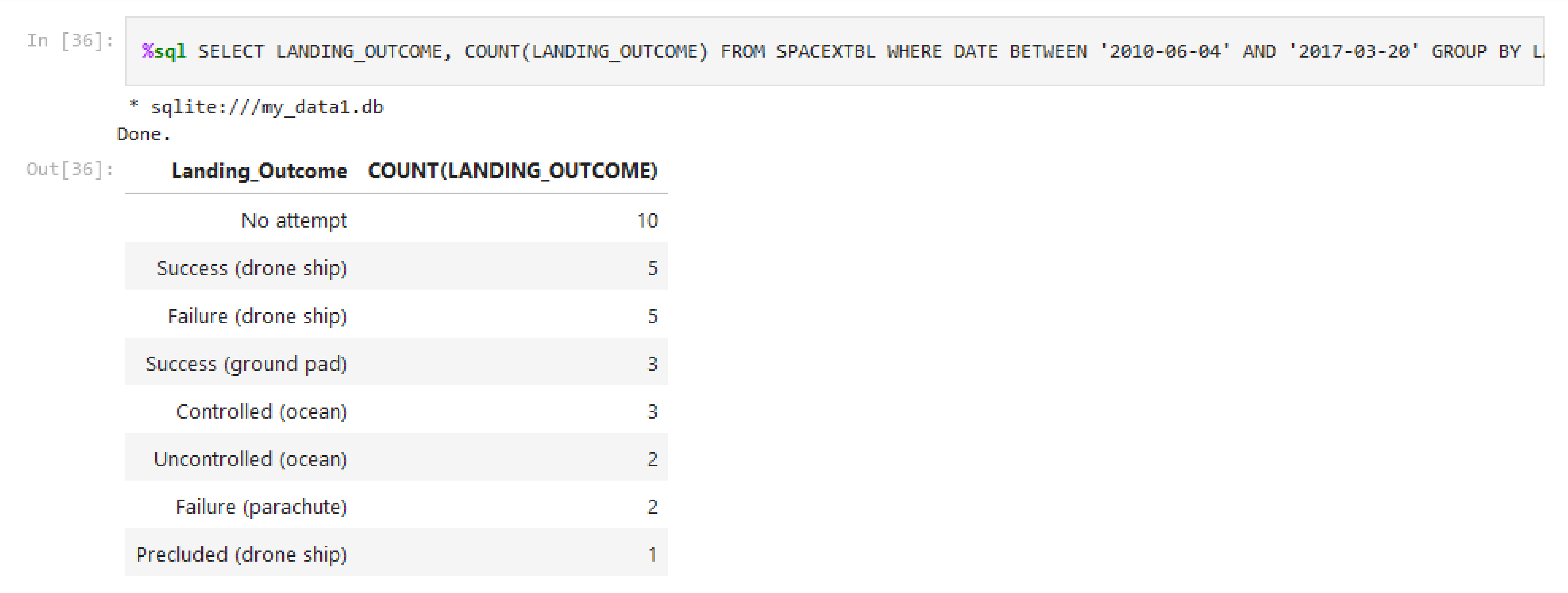
```
 * sqlite:///my_data1.db
Done.
```

t[39]:

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

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

With where we restricted the search to be between 2010-06-04 and 2017-03-20. Then, we used group by for landing_outcome and sorted it in descending order with DESC



```
In [36]: %sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY L
```

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

Out[36]:

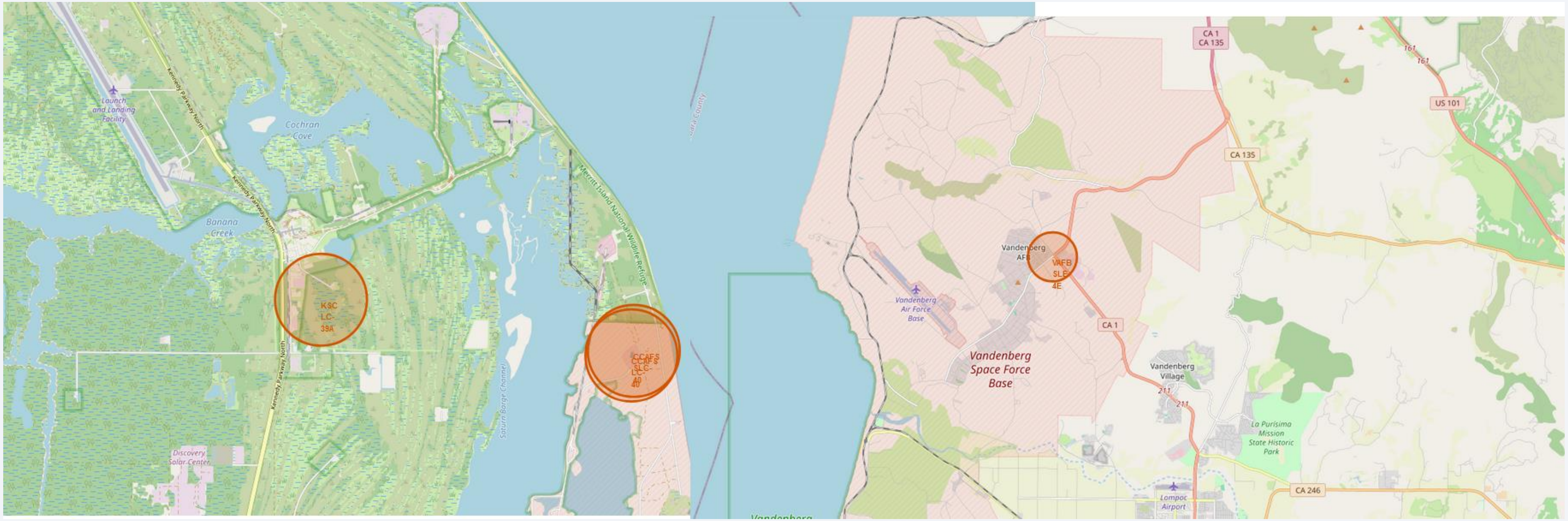
Landing_Outcome	COUNT(LANDING_OUTCOME)
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

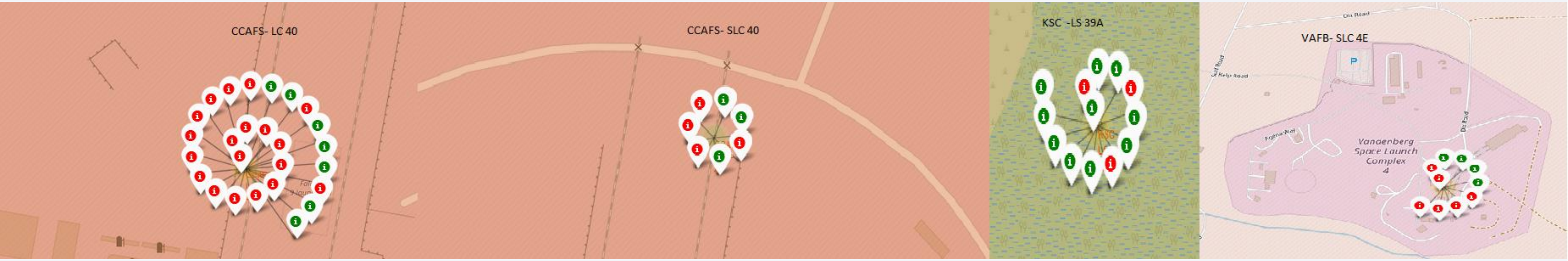
Launch sites locations

- We can see that in most cases, the launch sites are near coastal areas



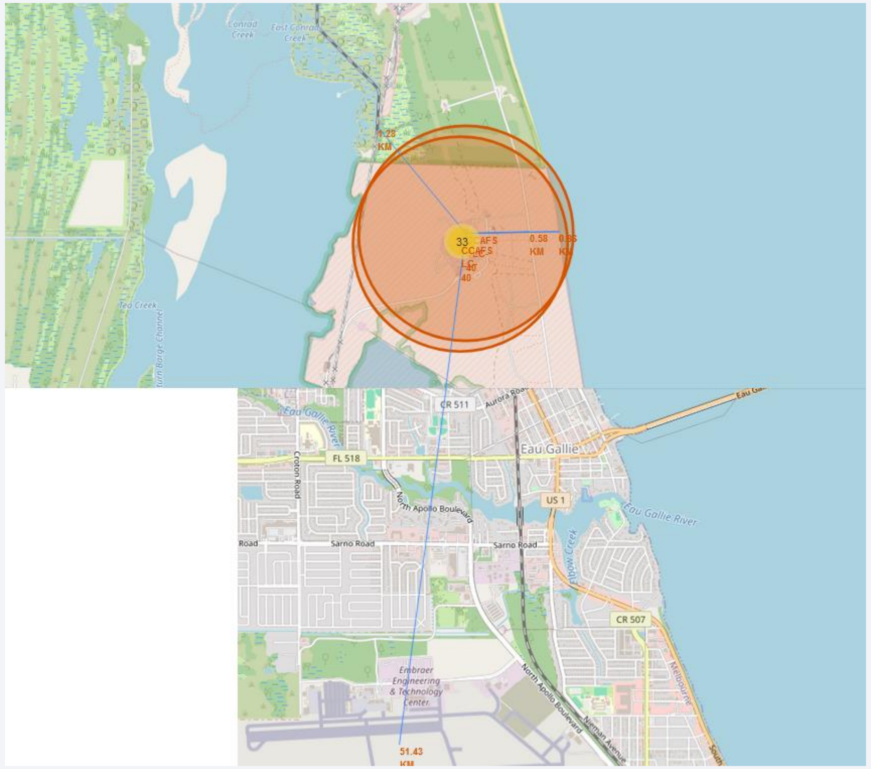
Launch sites, failed launches, and successful launches

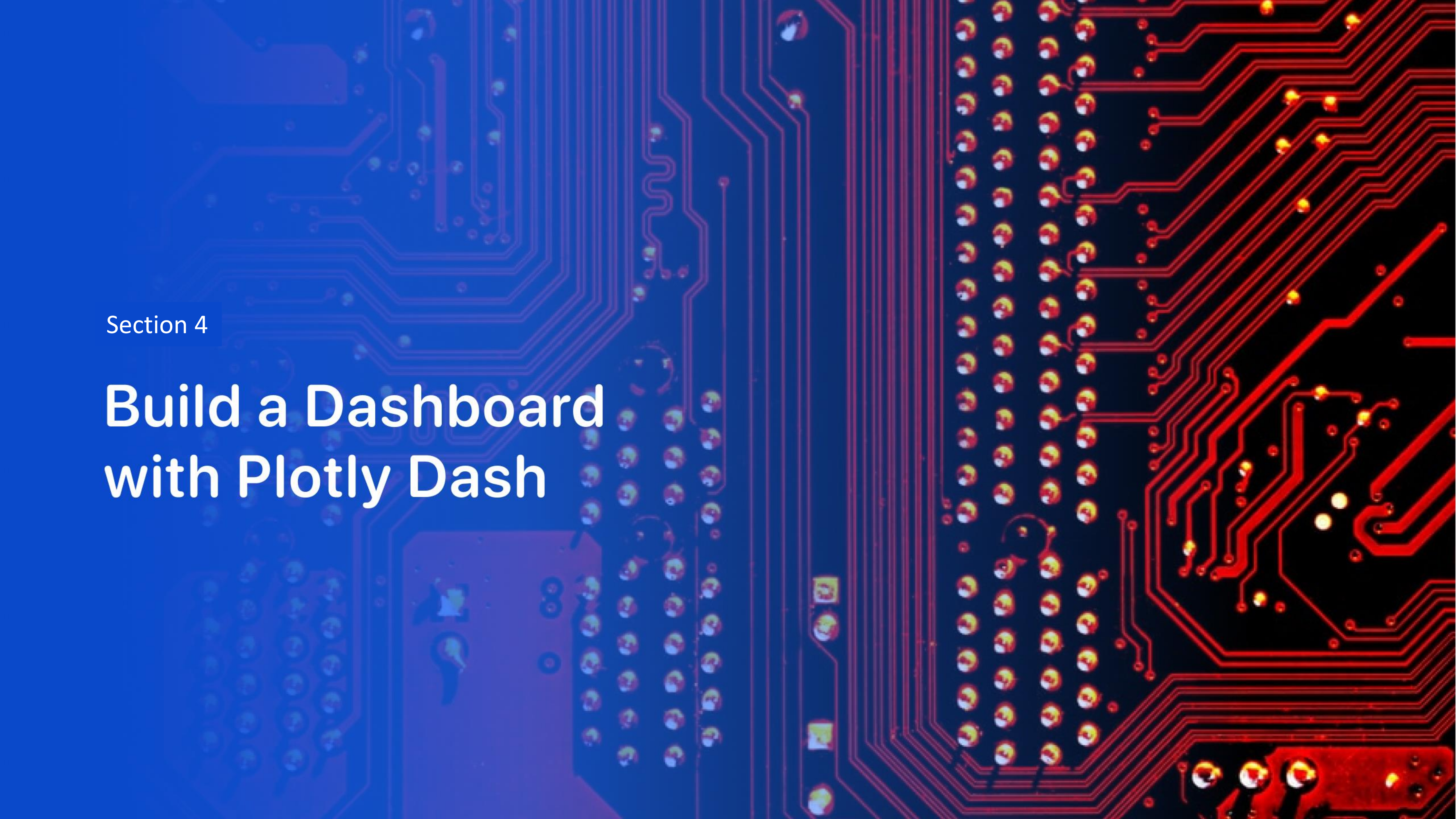
- We can see the number of launches, which ones were successful and which ones were not, among the different launch sites. For example, we can see that the site with the most launches was CCAFS LC40, but it also has the highest number of failed launches (color red mark a failed launch).



Launch Site proximity

- I created a line on the map to see the distance between the launch site and a train station, a city, and a coastal area For example, the nearest city is 51.43 km away



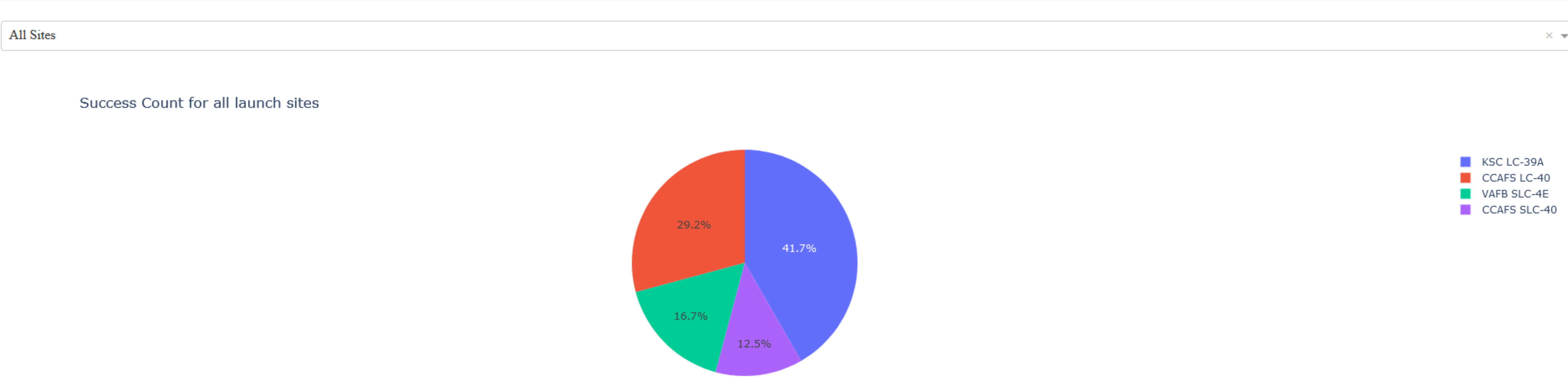


Section 4

Build a Dashboard with Plotly Dash

Success Count for all sites

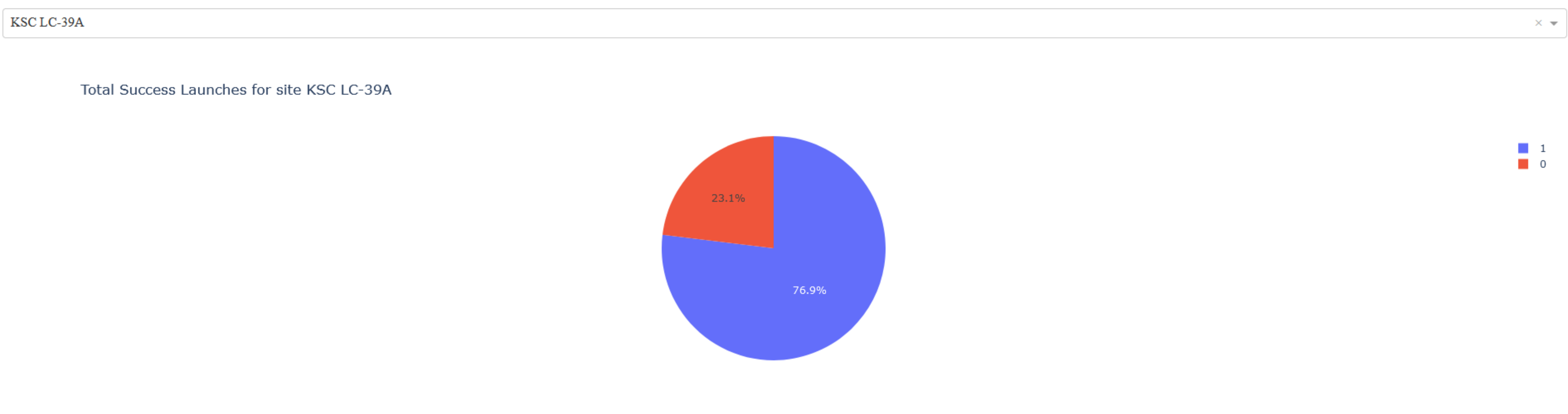
- In this pie chart, we can visualize the success ratio of launches from all the launch sites. We can see that among the sites, the one with the highest chance of a successful launch is KSC LC-39A



All Sites

<Dashboard Screenshot 2>

- We can see that indeed the KSC LC 39A site has the highest launch success rate, with 76.9%

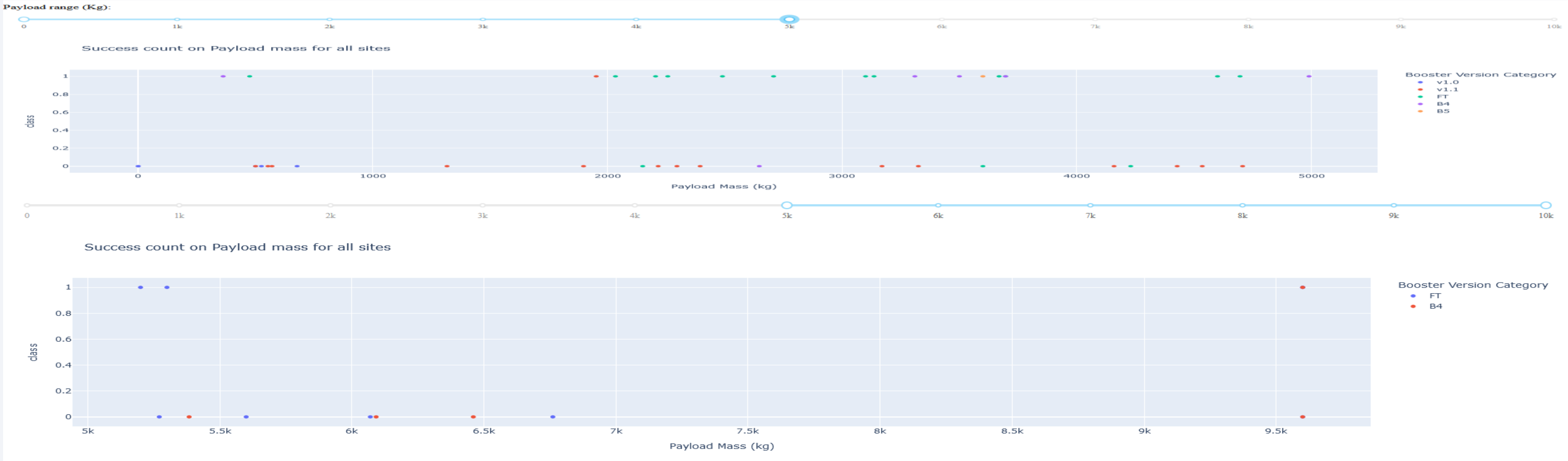


KSC LC-39A

Total Success Launches for site KSC LC-39A

<Dashboard Screenshot 3>

- We can see that there is a significant difference in success between light and heavy payload masses

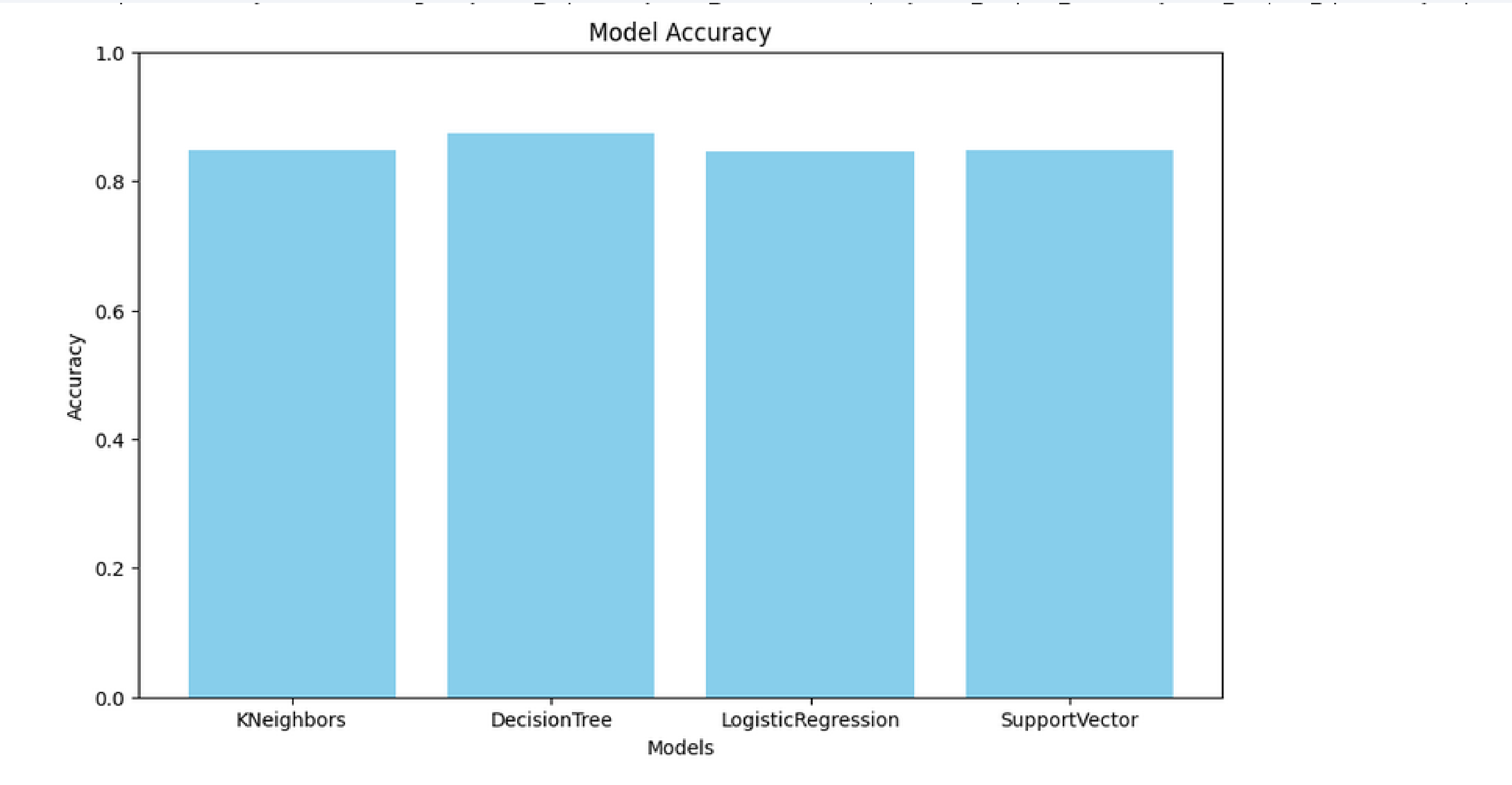


Section 5

Predictive Analysis (Classification)

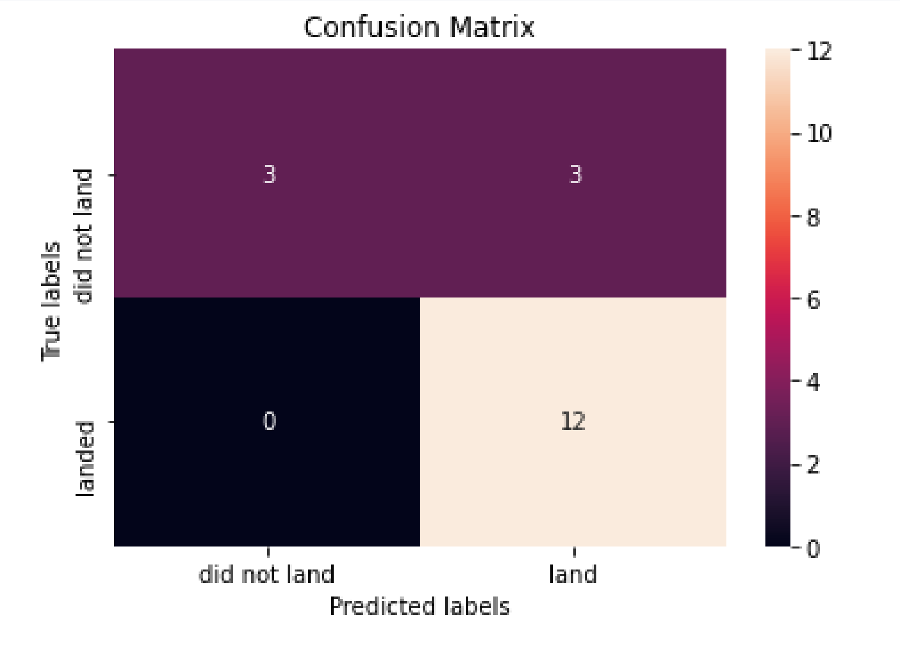
Classification Accuracy

- According to the graph, we can see that the Decision Tree has the highest score



Confusion Matrix

- The confusion matrix for the decision tree shows that the classifier can distinguish between the different classes. The major problem is the false positives



Conclusions

- 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 and SSO 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!