IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Sana Iftikhar
19/10/2025

## Outline

## Executive Summary

- Summary of methodologies
    - Data Collection through API
    - Data Collection with Web Scraping
    - Data Wrangling
    - Exploratory Data Analysis with SQL
    - Exploratory Data Analysis with Data Visualization
    - Interactive Visual Analytics with Folium
    - Machine Learning Prediction
- Summary of all results
    - Exploratory Data Analysis result
    - Interactive analytics in screenshots
    - Predictive Analytics result

## Introduction

- Project background and context

Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

- What factors determine if the rocket will land successfully?
- The interaction amongst various features that determine the success rate of a successful landing.
- What operating conditions needs to be in place to ensure a successful landing program.

Section 1

## Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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

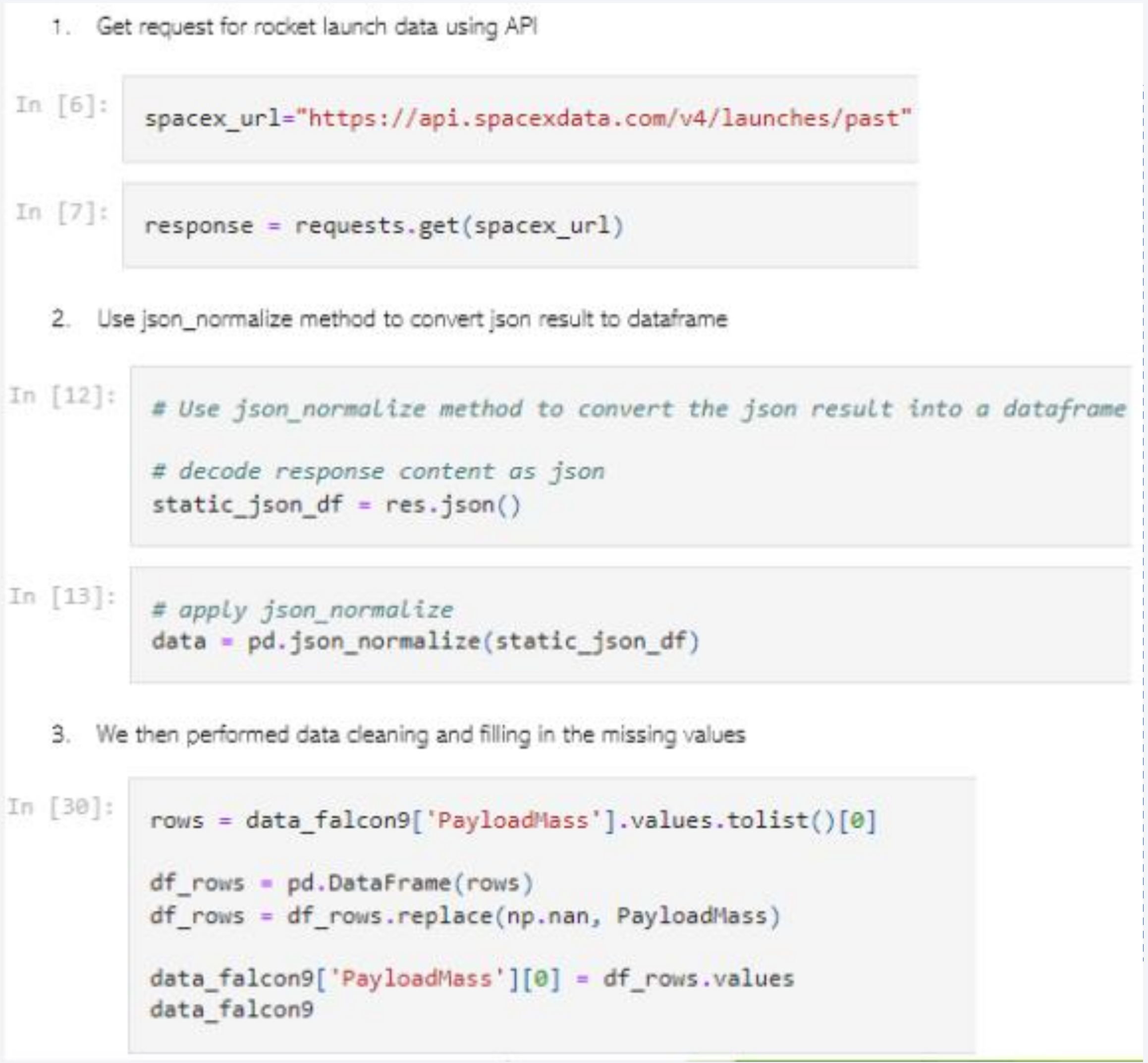
## Data Collection

- Describe how data sets were collected.
  - 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 notebook is:

https://github.com/sanaiftikharh/WinningSpaceRaceWithDataScience/blob/main/jupyter_labs_webscraping.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)
```

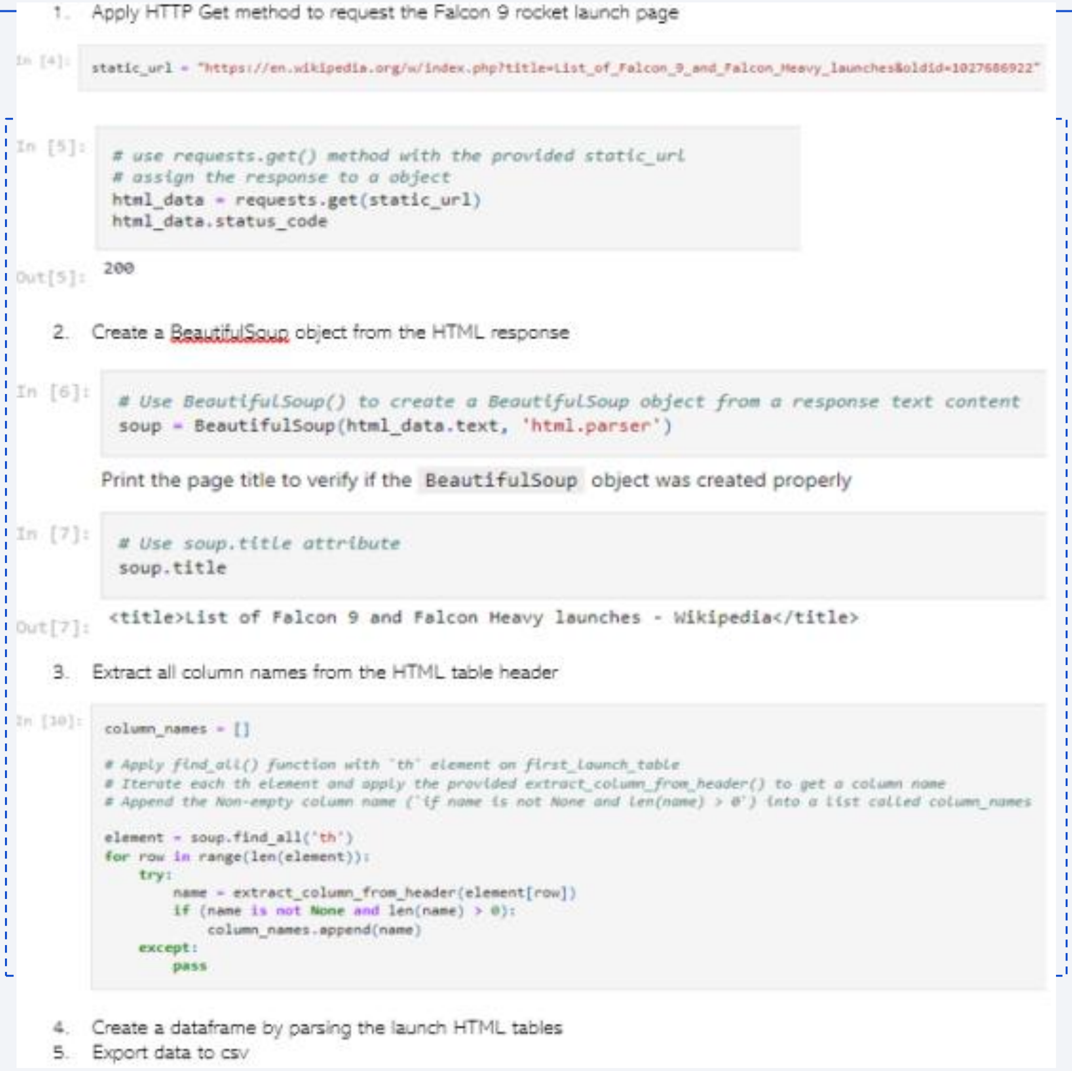
3. We then performed data cleaning and filling in the missing values

```
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.
- The link to the notebook is :

[https://github.com/sanaiftikharh/WinningSpaceRaceWithDataScience/blob/main/jupyter_labs_webscraping.ipynb](https://github.com/sanaiftikharh/WinningSpaceRaceWithDataScience/blob/main/jupyter_labs_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

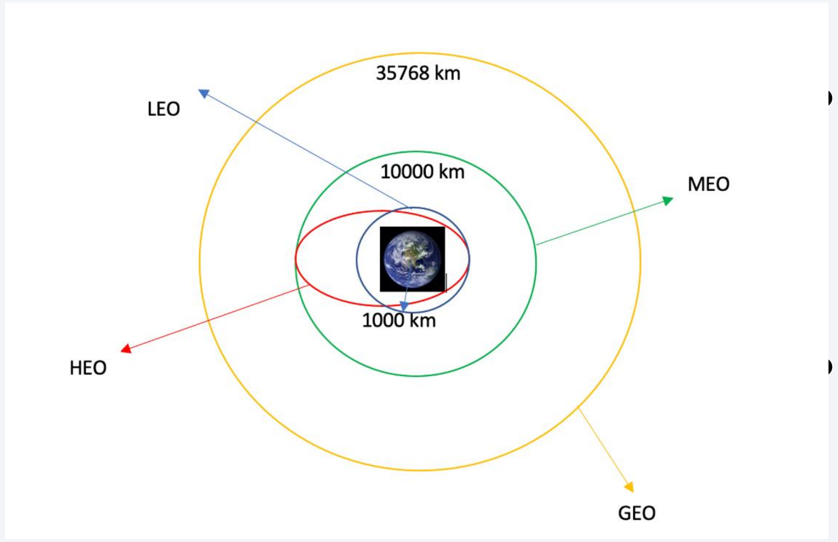
```
In [7]: # Use soup.title attribute
        soup.title

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/sanaiftikharh/WinningSpaceRaceWithDataScience/blob/main/labs_jupyter_spacex_data_wrangling_jupyterlite_jupyterlite.ipynb

## EDA with Data Visualization

- Several charts were plotted to explore launch success factors. Scatter plots showed how **flight number**, **payload mass**, and **launch site/orbit** relate to success, revealing higher success with experience and lighter payloads at some sites. A **bar chart** compared success rates across orbit types, while a **line chart** showed a steady yearly improvement from 2013 to 2020. Together, these visuals highlighted key trends and performance patterns in launch outcomes.
- Link to Notebook is :

https://github.com/sanaiftikharh/WinningSpaceRaceWithDataScience/blob/main/jupyter_labs_eda_dataviz_ipynb_jupyterlite.ipynb

## EDA with SQL

- **Query 1:** Retrieved all unique Launch_Site names using SELECT DISTINCT.
- **Query 2:** Displayed 5 launch records where Launch_Site starts with 'CCA'.
- **Query 3:** Calculated the **total payload mass** for boosters launched by **NASA (CRS)**.
- **Query 4:** Computed the **average payload mass** for booster version **F9 v1.1**.
- **Query 5:** Found the **earliest launch date** for a **successful ground pad landing**.
- **Query 6:** Listed booster versions that had **successful drone ship landings** with payload mass between **4000–6000 kg**.
- **Query 7:** Counted the **number of successful and failed missions** based on Mission_Outcome.

## EDA with SQL

•**Query 8:** Selected **booster versions** that carried the **maximum payload mass** using a subquery.
•**Query 9:** Retrieved **failed drone ship landings in 2015**, showing **month, booster version, and launch site**.
•**Query 10:** Ranked **landing outcomes** by their **occurrence frequency** between **2010-06-04** and **2017-03-20** in descending order.
•Link to my notebook is :
https://github.com/sanaiftikharh/WinningSpaceRaceWithDataScience/blob/main/jupyter_labs_eda_sql_coursera_sqllite.ipynb

## Build an Interactive Map with Folium

- **Markers:** Indicated launch sites and outcomes (green = success, red = failure).
- **Circles:** Highlighted each launch site's location.
- **Marker Clusters:** Grouped overlapping launch markers for clarity.
- **Distance Markers:** Showed measured distances to nearby features (coastline, city, etc.).
- **Lines (Polylines):** Connected launch sites to nearby features to visualize proximity.
- **Mouse Position Tool:** Displayed live coordinates for distance mapping.

These objects were added to **visualize site locations, launch outcomes, and proximity to geographical features** for better spatial and performance insights.

- Link to my notebook is : https://github.com/sanaiftikharh/WinningSpaceRaceWithDataScience/blob/main/lab_jupyter_launch_site_location_jupyterlite.ipynb

## 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.

## 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 and improved it using feature engineering and algorithm tuning.
- We found the best performing classification model.
- Link to Notebook is : https://github.com/sanaiftikharh/WinningSpaceRaceWithDataScience/blob/main/SpaceX_Machine_Learning_Prediction_Part_5_v1.ipynb

## Results

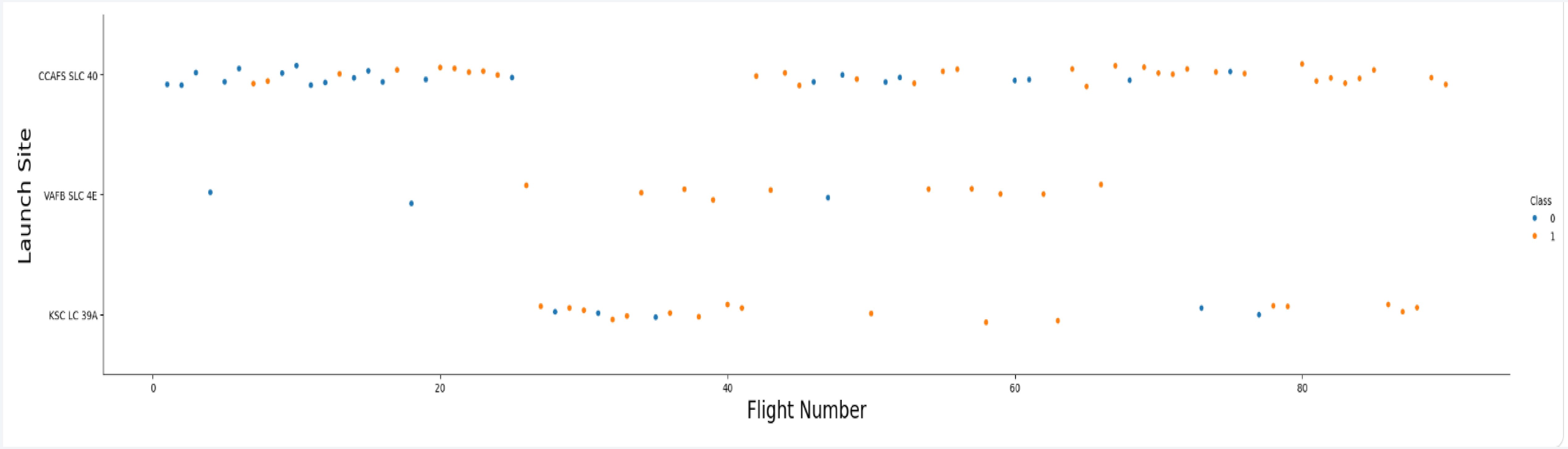
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

## Insights drawn from EDA

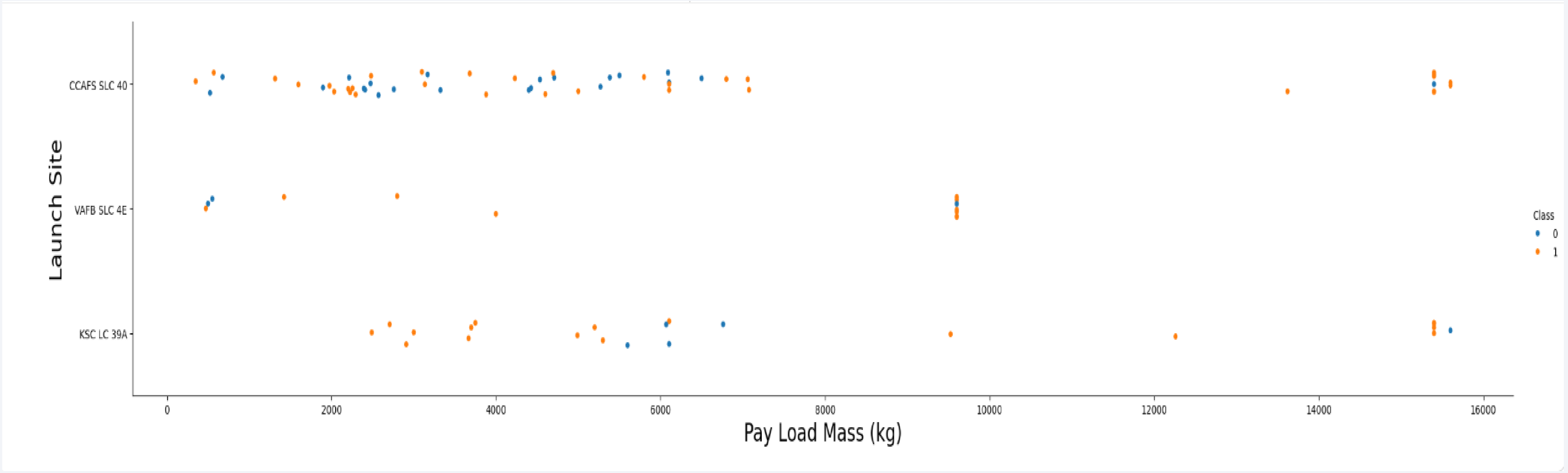
## Flight Number vs. Launch Site

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



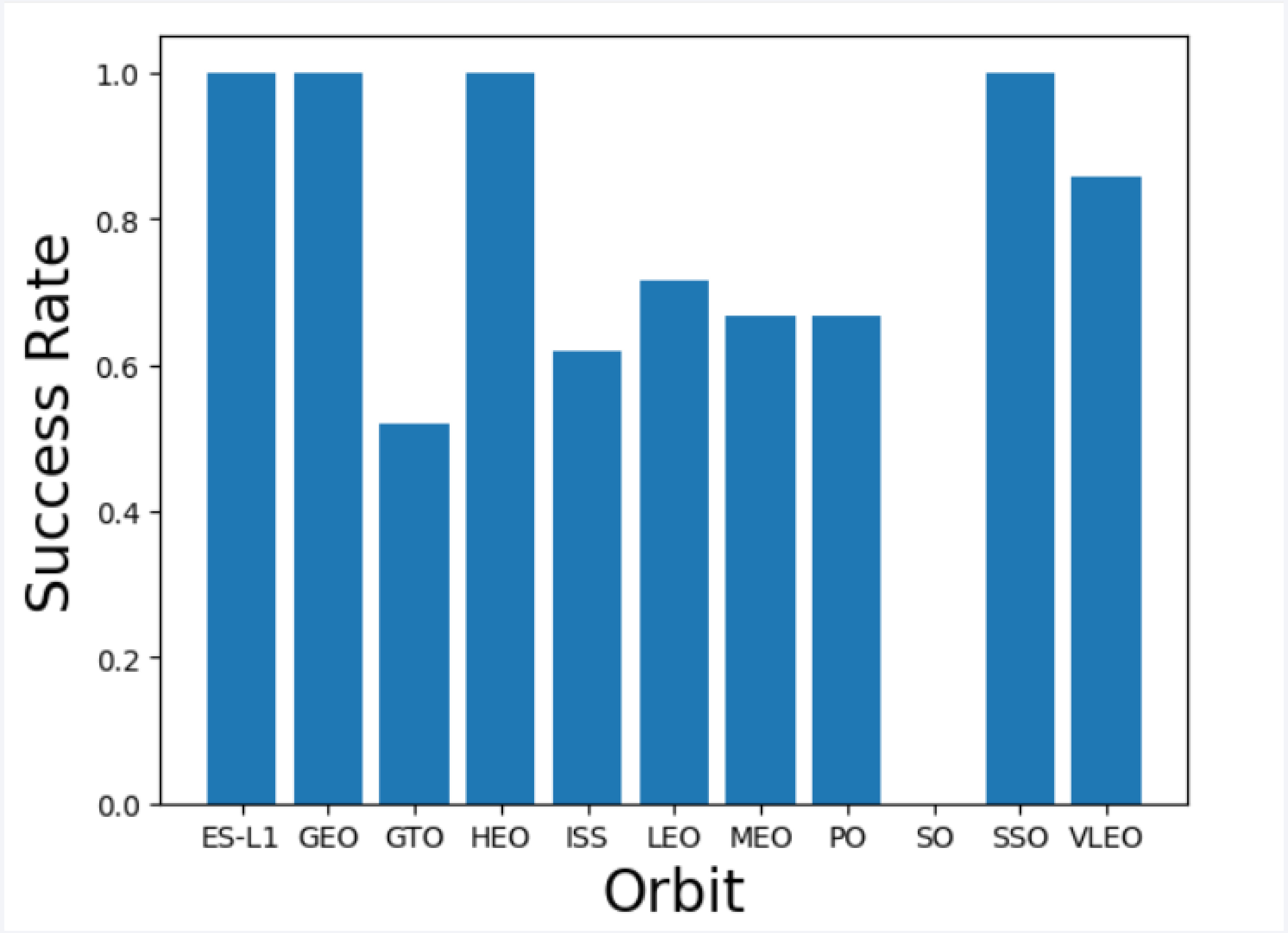
# Payload vs. Launch Site

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



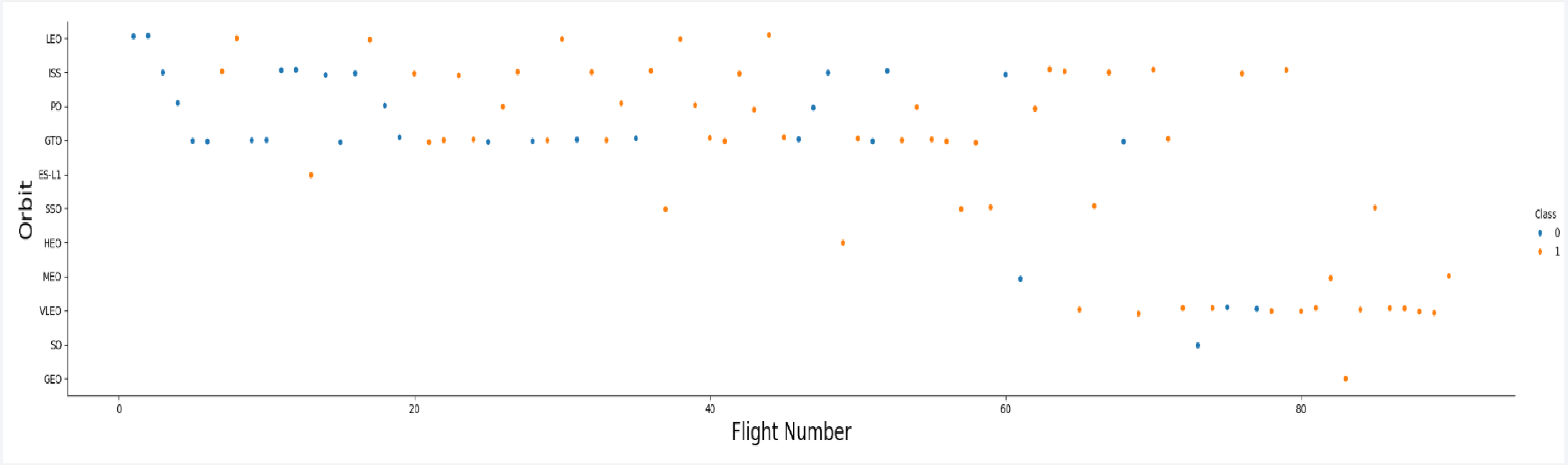
## Success Rate vs. Orbit Type

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



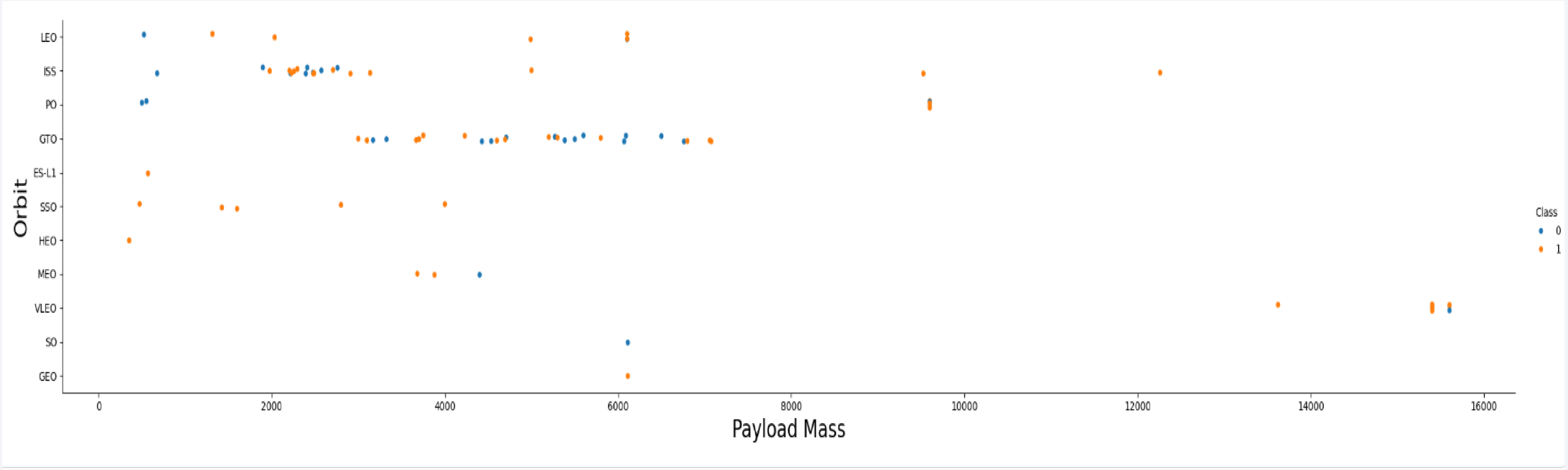
## Flight Number vs. Orbit Type

- 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.



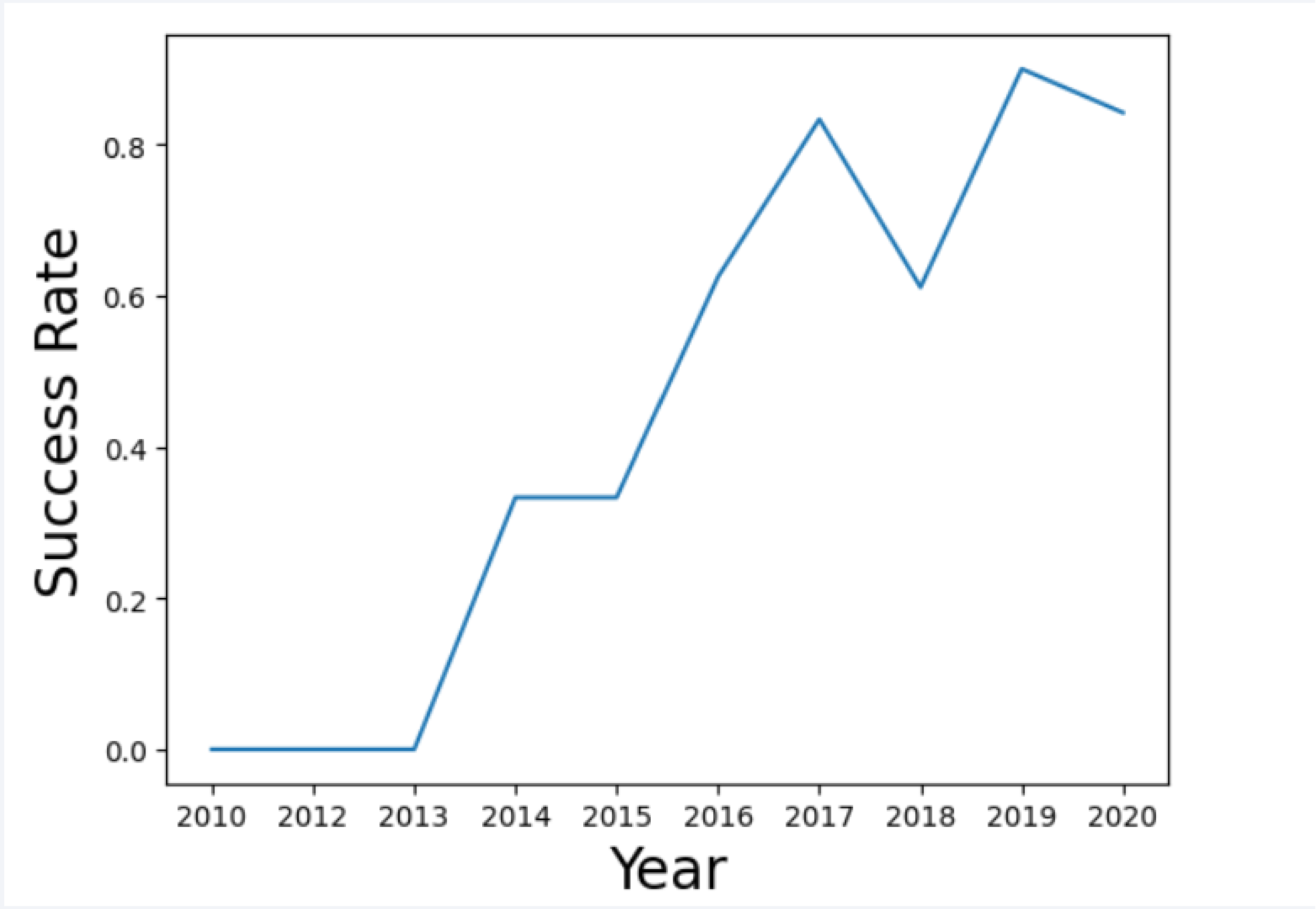
# Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



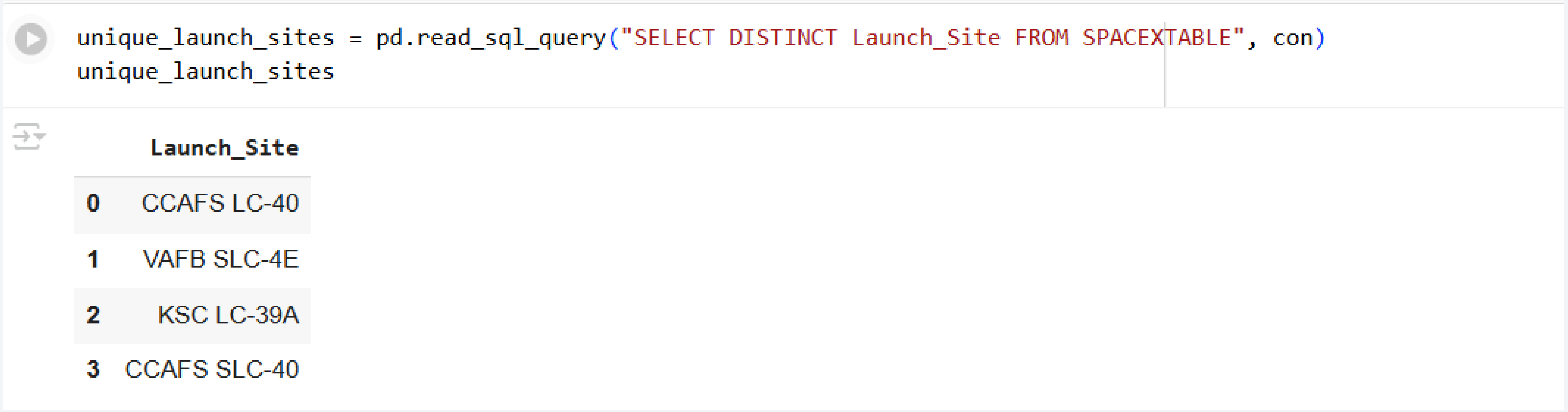
## Launch Success Yearly Trend

- 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.

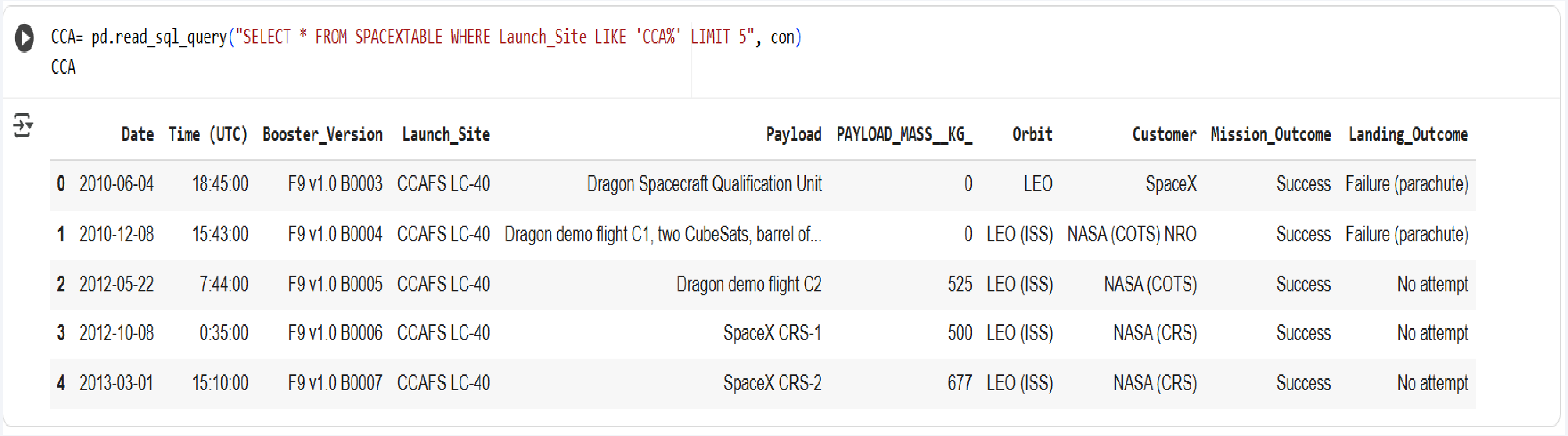


```
unique_launch_sites = pd.read_sql_query("SELECT DISTINCT Launch_Site FROM SPACEXTABLE", con)
unique_launch_sites
```

|   | Launch_Site |
|---|---|
| 0 | CCAFS LC-40 |
| 1 | VAFB SLC-4E |
| 2 | KSC LC-39A |
| 3 | CCAFS SLC-40 |

## Launch Site Names Begin with 'CCA'

- We used the query above to display 5 records where launch sites begin with `CCA`

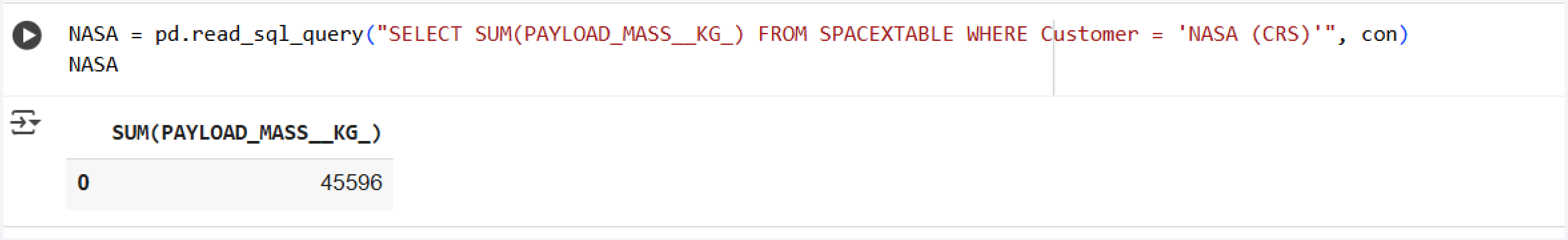


```
CCA= pd.read_sql_query("SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5", con)
CCA
```

| | Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 1 | 2010-12-08 | 15:43:00 | F9 v1.0 B0004 | CCAFS LC-40 | Dragon demo flight C1, two CubeSats, barrel of... | 0 | LEO (ISS) | NASA (COTS) NRO | Success | Failure (parachute) |
| 2 | 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 |
| 3 | 2012-10-08 | 0:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 4 | 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

- We calculated the total payload carried by boosters from NASA as 45596 using the query below

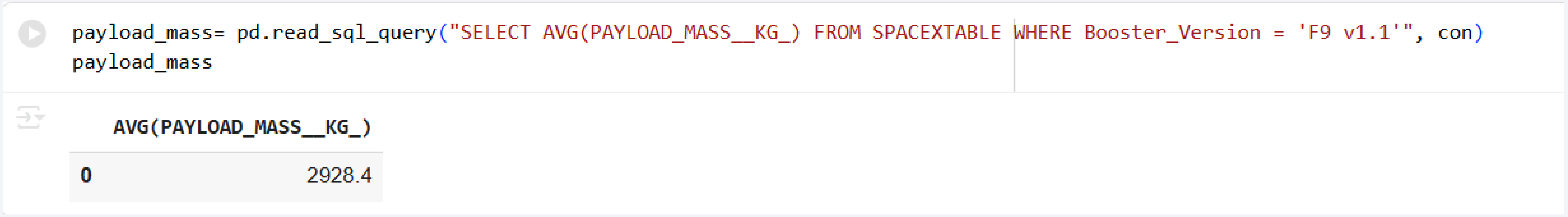


```
NASA = pd.read_sql_query("SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)'", con)
NASA
```

```
   SUM(PAYLOAD_MASS__KG_)
0                   45596
```

# Average Payload Mass by F9 v1.1

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

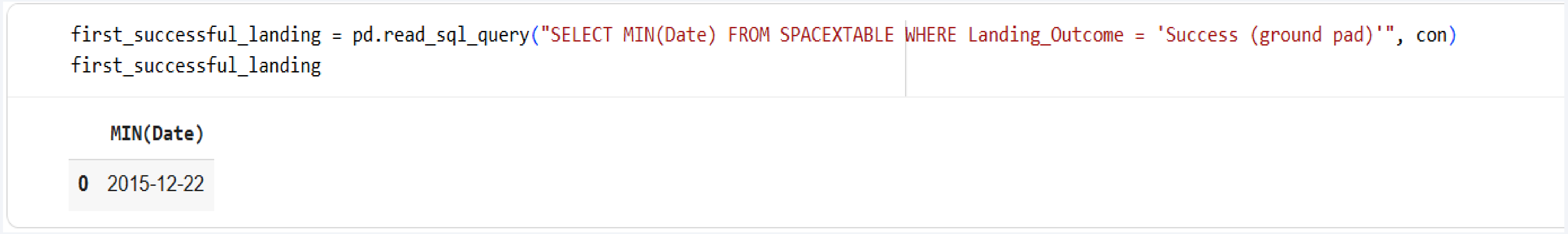


```
payload_mass= pd.read_sql_query("SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1'", con)
payload_mass
```

```
   AVG(PAYLOAD_MASS__KG_)
0                  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

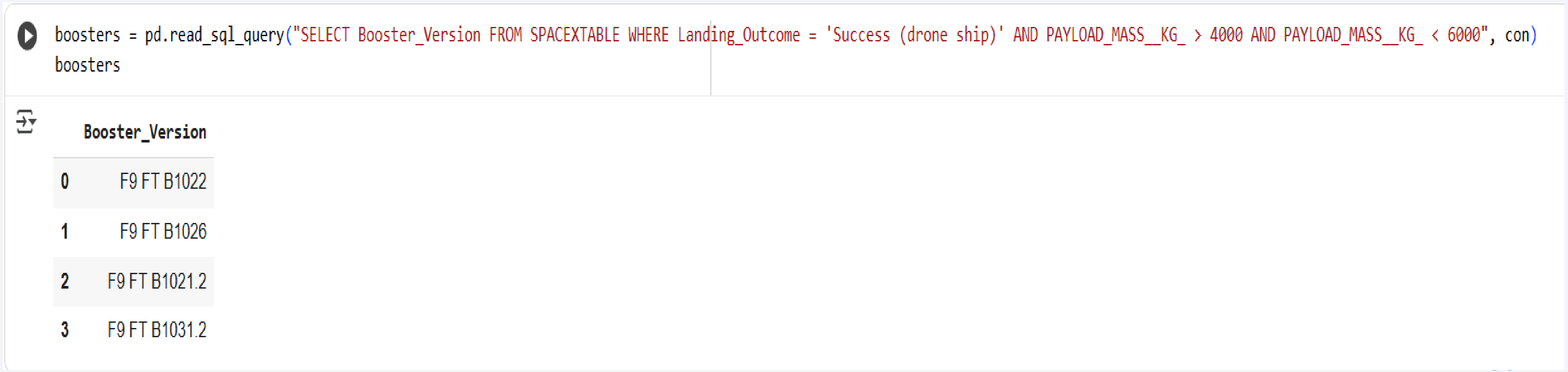


```
first_successful_landing = pd.read_sql_query("SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)'", con)
first_successful_landing
```

```
     MIN(Date)
0   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

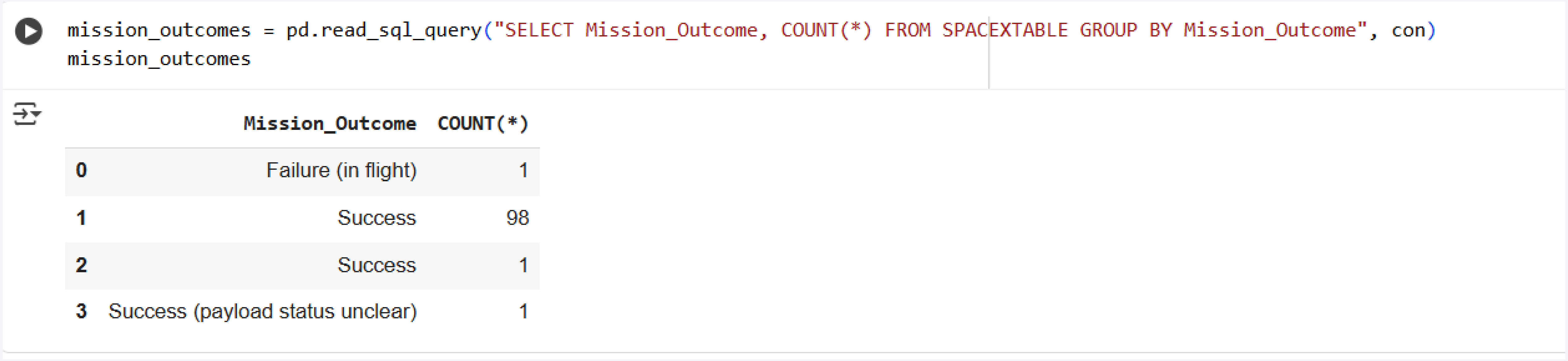


```
boosters = pd.read_sql_query("SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000", con)
boosters
```

|   | Booster_Version |
|---|---|
| 0 | F9 FT B1022 |
| 1 | F9 FT B1026 |
| 2 | F9 FT B1021.2 |
| 3 | F9 FT B1031.2 |

## Total Number of Successful and Failure Mission Outcomes

- We used wildcard like ‘%’ to filter for WHERE MissionOutcome was a success or a failure.

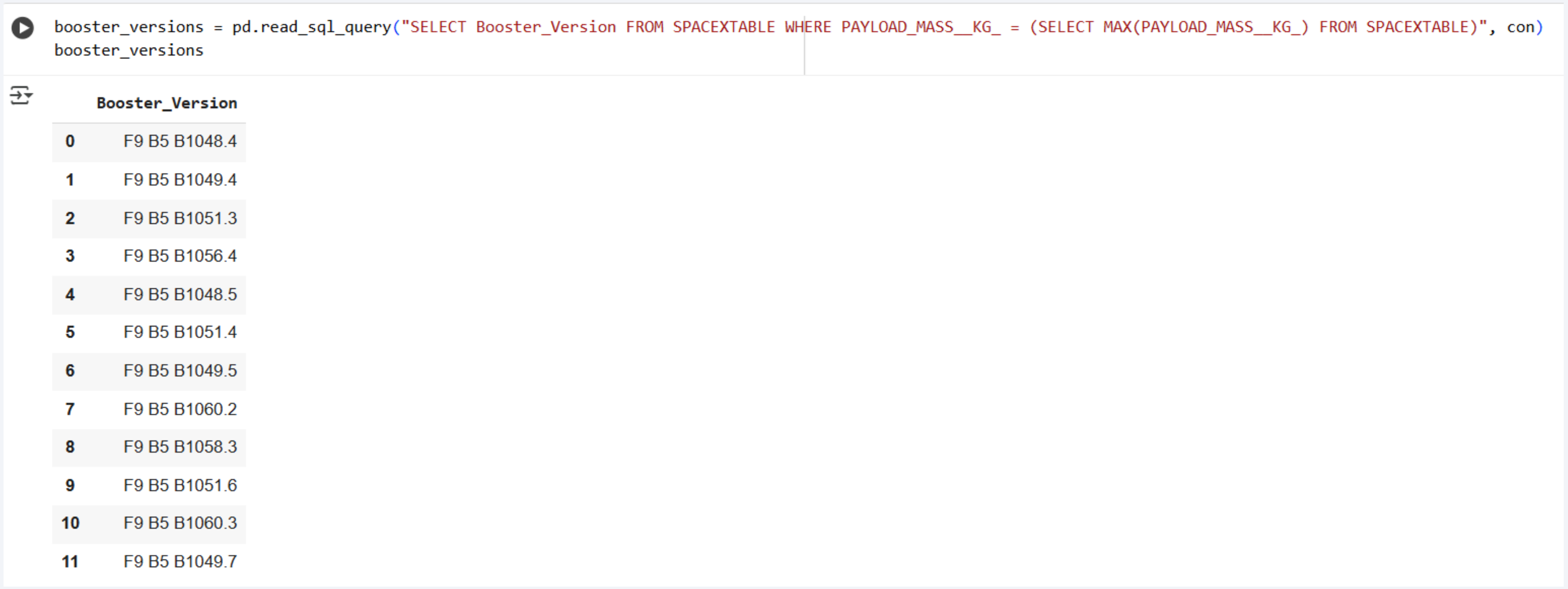


```
mission_outcomes = pd.read_sql_query("SELECT Mission_Outcome, COUNT(*) FROM SPACEXTABLE GROUP BY Mission_Outcome", con)
mission_outcomes
```

```
                    Mission_Outcome  COUNT(*)
0              Failure (in flight)         1
1                          Success        98
2                          Success         1
3  Success (payload status unclear)        1
```

# Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.

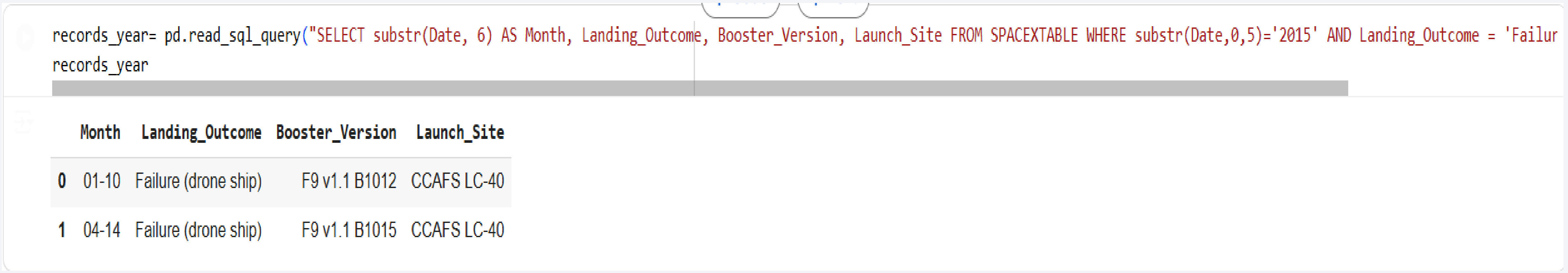


```
booster_versions = pd.read_sql_query("SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)", con)
booster_versions
```

| | Booster_Version |
|---|---|
| 0 | F9 B5 B1048.4 |
| 1 | F9 B5 B1049.4 |
| 2 | F9 B5 B1051.3 |
| 3 | F9 B5 B1056.4 |
| 4 | F9 B5 B1048.5 |
| 5 | F9 B5 B1051.4 |
| 6 | F9 B5 B1049.5 |
| 7 | F9 B5 B1060.2 |
| 8 | F9 B5 B1058.3 |
| 9 | F9 B5 B1051.6 |
| 10 | F9 B5 B1060.3 |
| 11 | F9 B5 B1049.7 |

## 2015 Launch Records

- We used a combinations of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

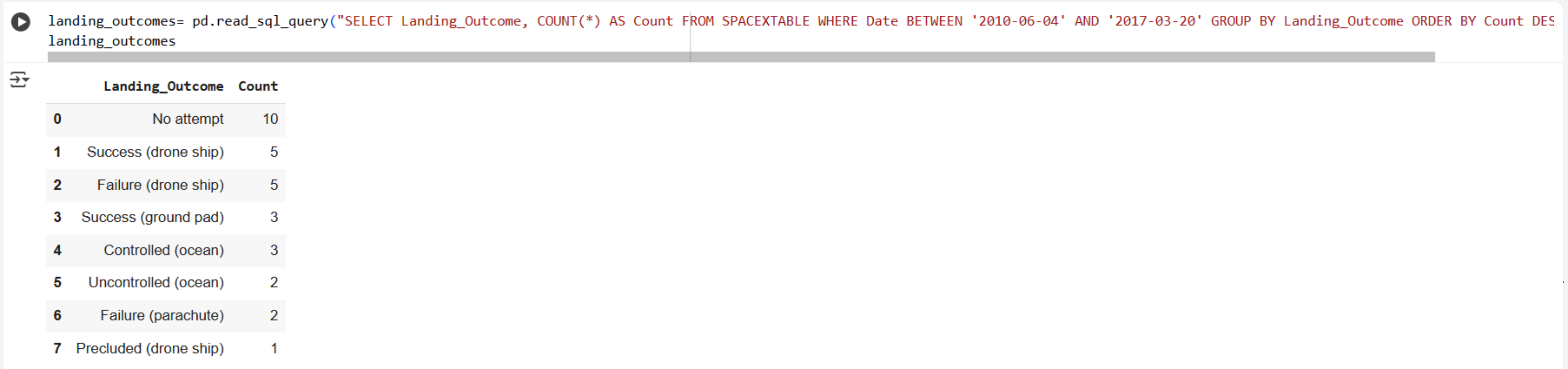


```
records_year= pd.read_sql_query("SELECT substr(Date, 6) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr(Date,0,5)='2015' AND Landing_Outcome = 'Failur
records_year
```

|   | Month | Landing_Outcome | Booster_Version | Launch_Site |
|---|---|---|---|---|
| 0 | 01-10 | Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 |
| 1 | 04-14 | Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 |

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

- We selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.
- We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.



```
landing_outcomes= pd.read_sql_query("SELECT Landing_Outcome, COUNT(*) AS Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY Count DES
landing_outcomes
```

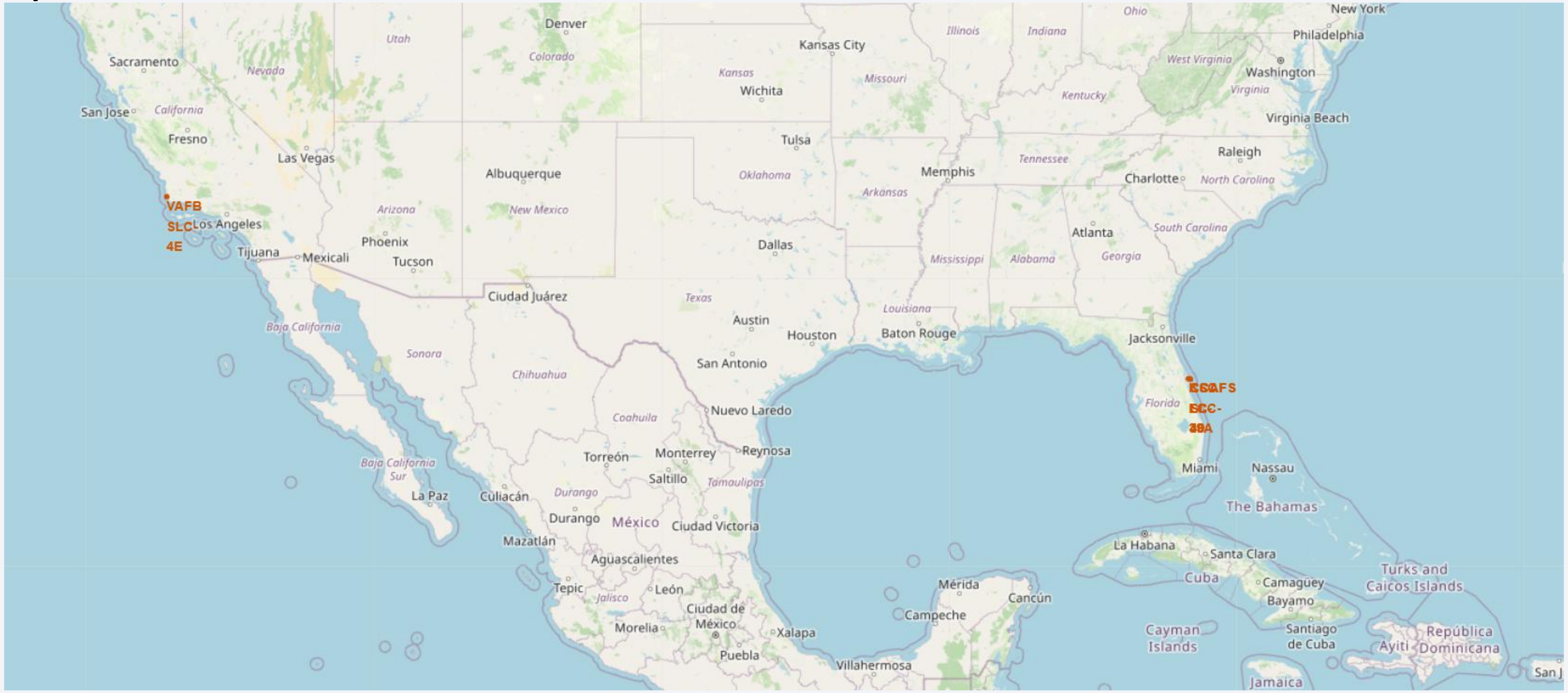
| | Landing_Outcome | Count |
|---|---|---|
| 0 | No attempt | 10 |
| 1 | Success (drone ship) | 5 |
| 2 | Failure (drone ship) | 5 |
| 3 | Success (ground pad) | 3 |
| 4 | Controlled (ocean) | 3 |
| 5 | Uncontrolled (ocean) | 2 |
| 6 | Failure (parachute) | 2 |
| 7 | Precluded (drone ship) | 1 |

Section 3

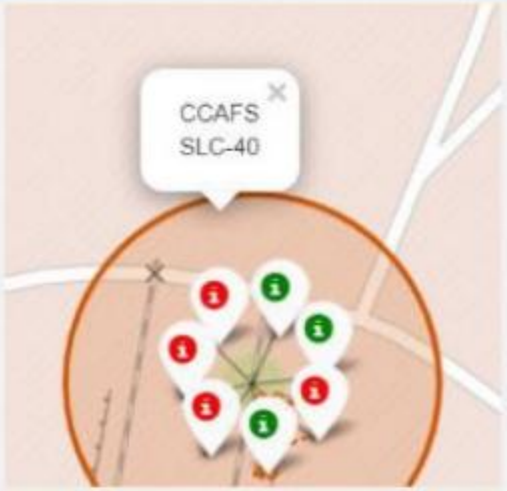
## Launch Sites Proximities Analysis

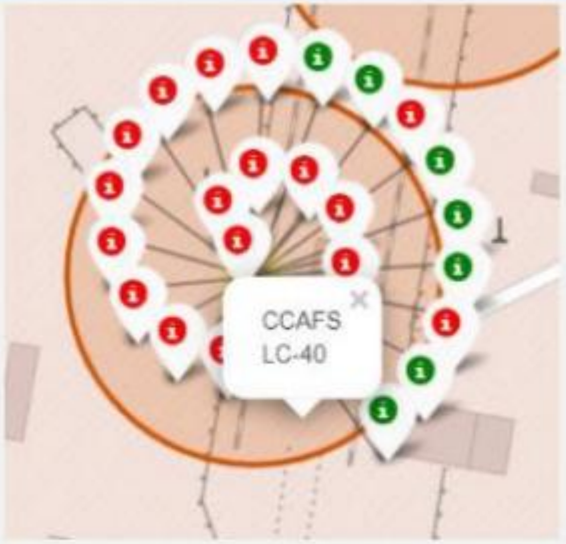
## All launch sites global map markers

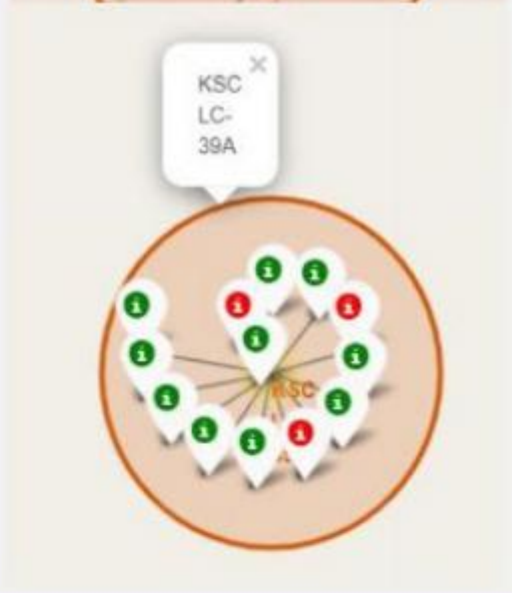
- Space X Launch sites are in USA



# Market Launcher with color label

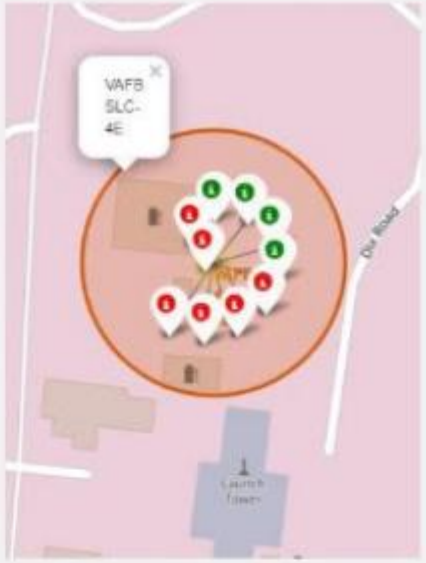






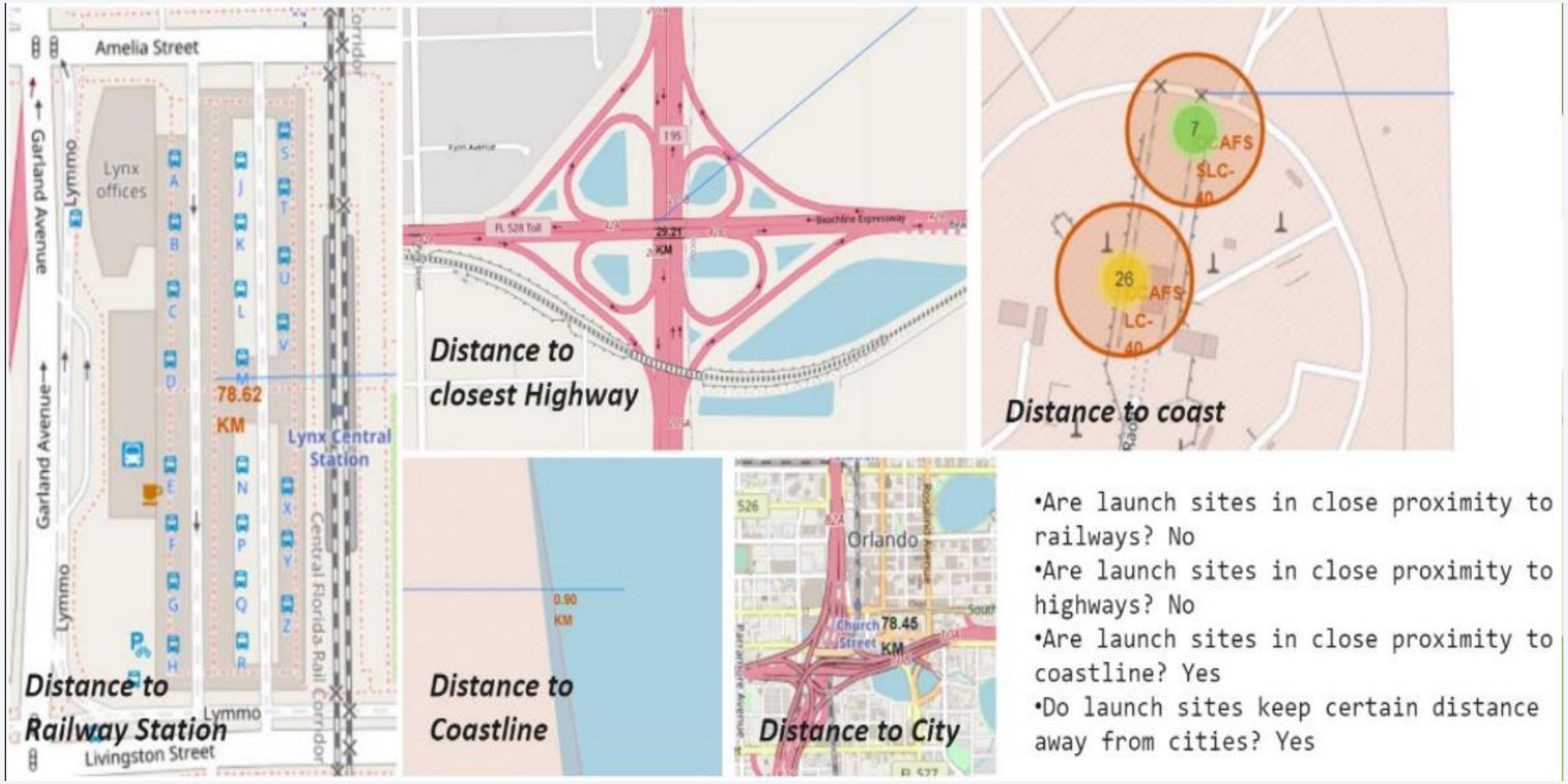
***Florida Launch Sites***

*Green Marker shows successful Launches and Red Marker shows Failures*

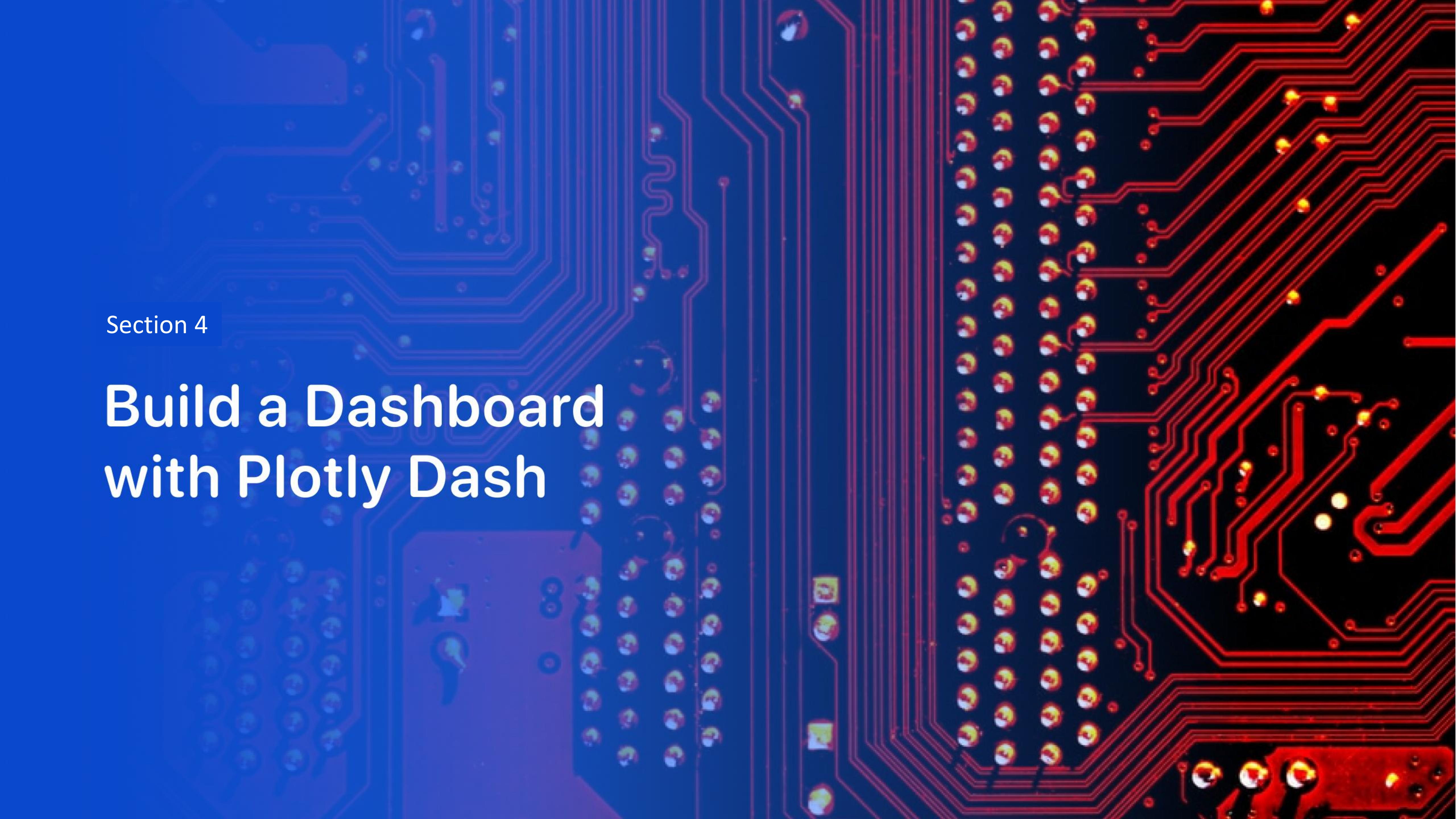


***California Launch Site***

## <Folium Map Screenshot 3>



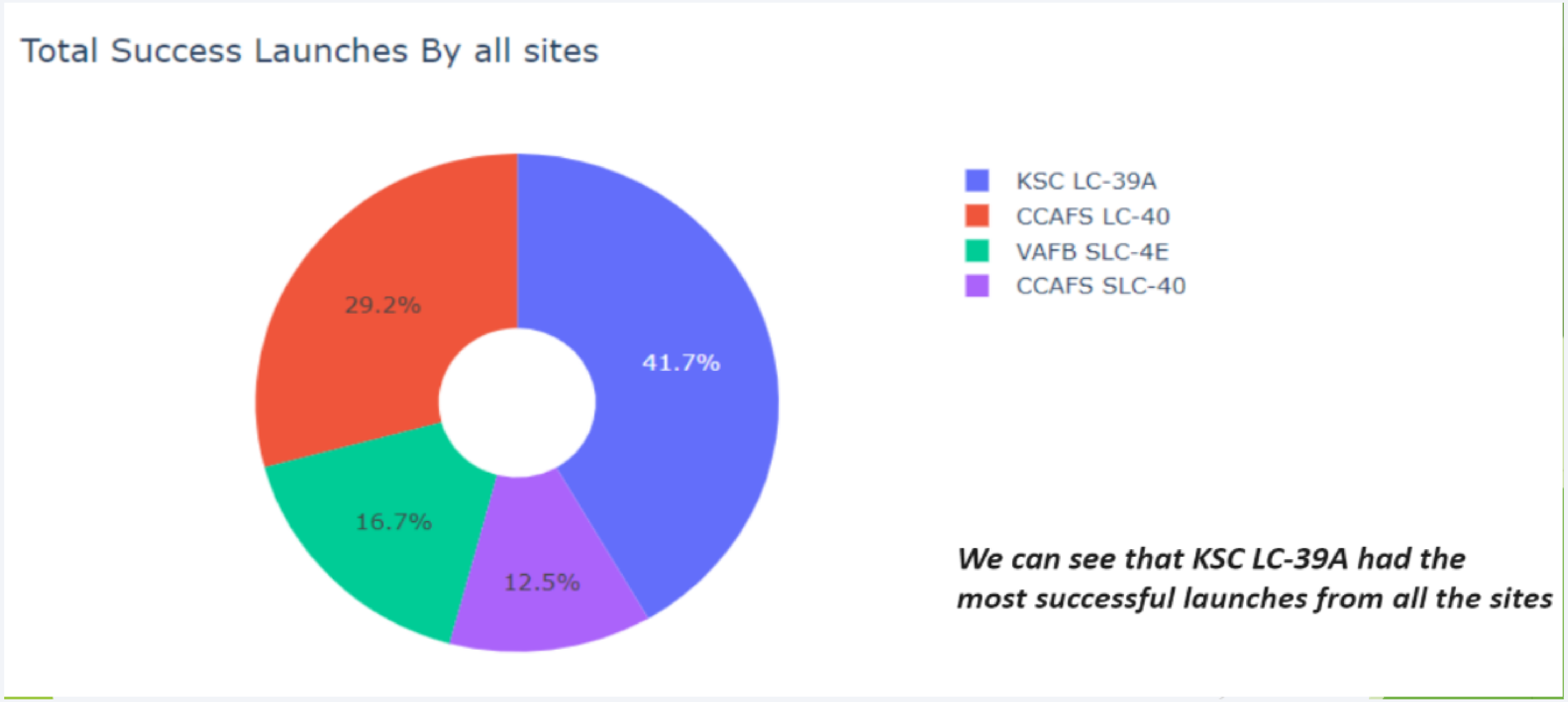
•Are launch sites in close proximity to railways? No
•Are launch sites in close proximity to highways? No
•Are launch sites in close proximity to coastline? Yes
•Do launch sites keep certain distance away from cities? Yes



Section 4

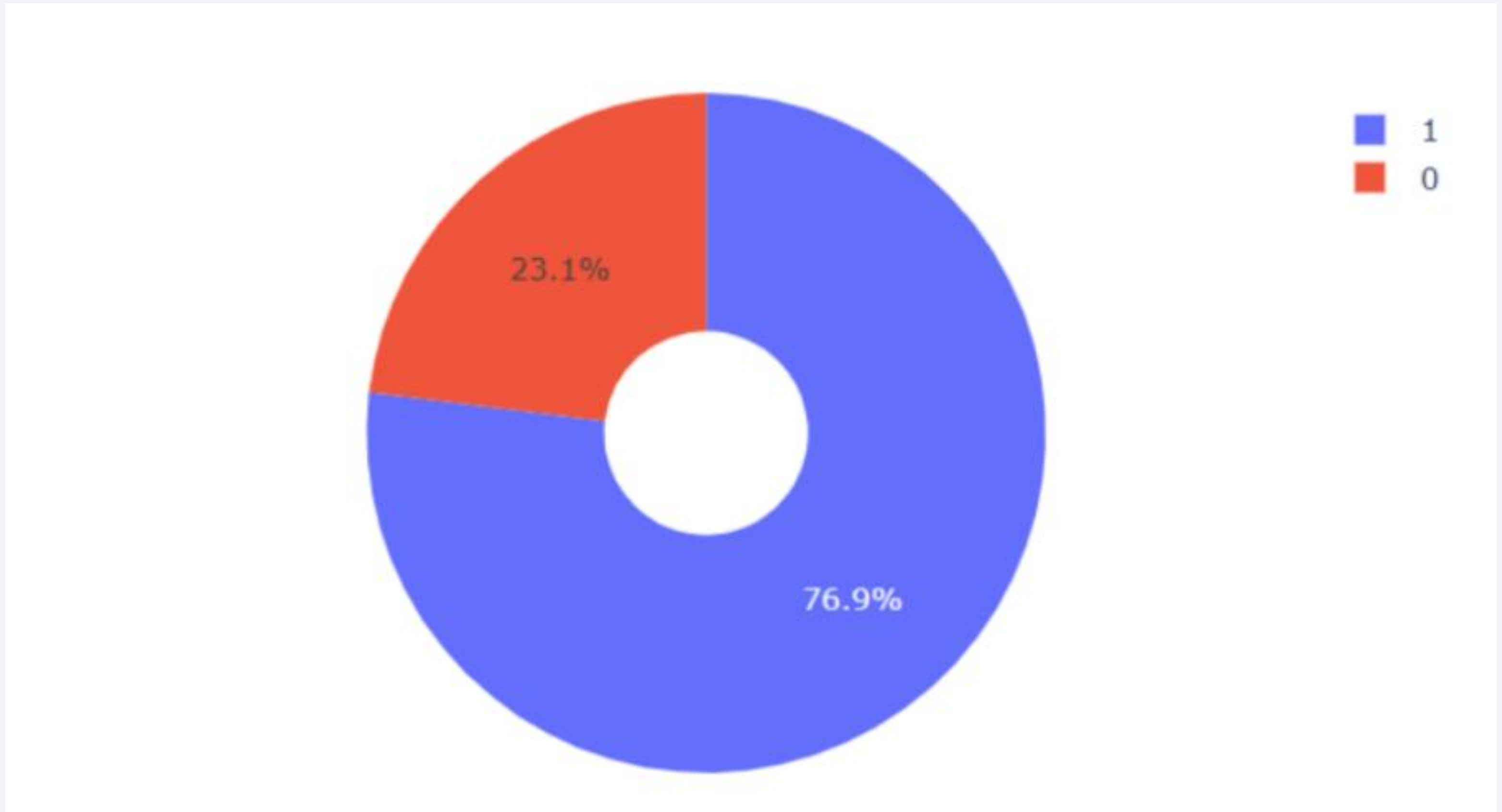
## Build a Dashboard with Plotly Dash

## Pie chart showing the success percentage achieved by each launch site



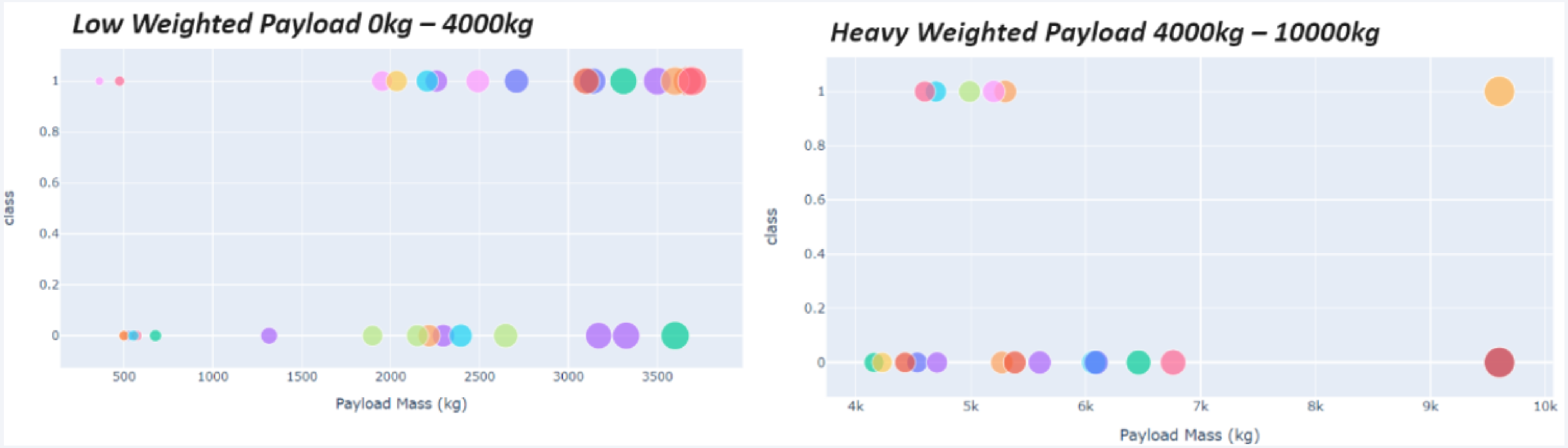
***We can see that KSC LC-39A had the most successful launches from all the sites***

## 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

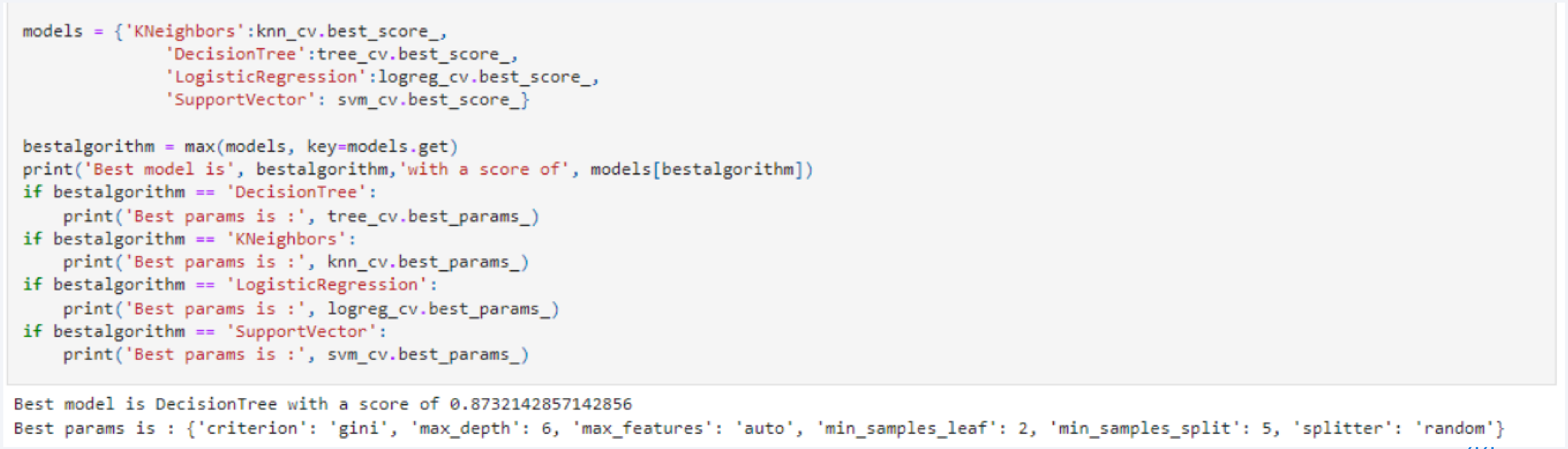


Section 5

## Predictive Analysis (Classification)

## Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

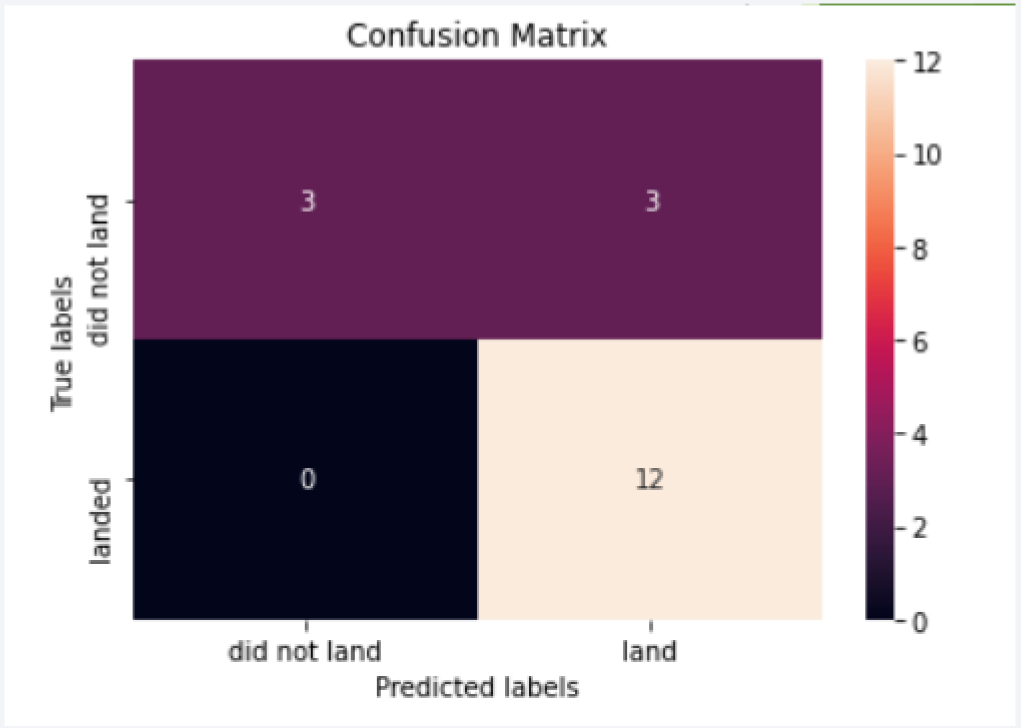


```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

## 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.

## Thank you!