IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

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

Executive Summary

- Summary of methodologies
  - Collection of Data through API
  - Collection of Data with Web Scraping
  - Data Wrangling
  - Data Analysis with SQL
  - Data Analysis with Data Visulization
  - Interactive Visual Analytics with Folium
  - Machine Learning Predictions
- Summary of all results
  - Data Analysis Results

Introduction

Project Summary and Context

The project analyzes SpaceX's rocket launches to identify variables affecting landing success, crucial for optimizing reusable rocket technology pioneered by Elon Musk. By studying factors like weather, trajectory, and design, the aim is to enhance landing procedures and increase success rates, advancing cost-effective and reliable space exploration.

Problems

- Finding the variables that influence landing outcome
- Best factors needed to increase successful landing probability

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collected using SpaceX REST API and web scrapping from SpaceX Wikipedia Page
- Perform data wrangling
  - Data was processed using encoding for 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

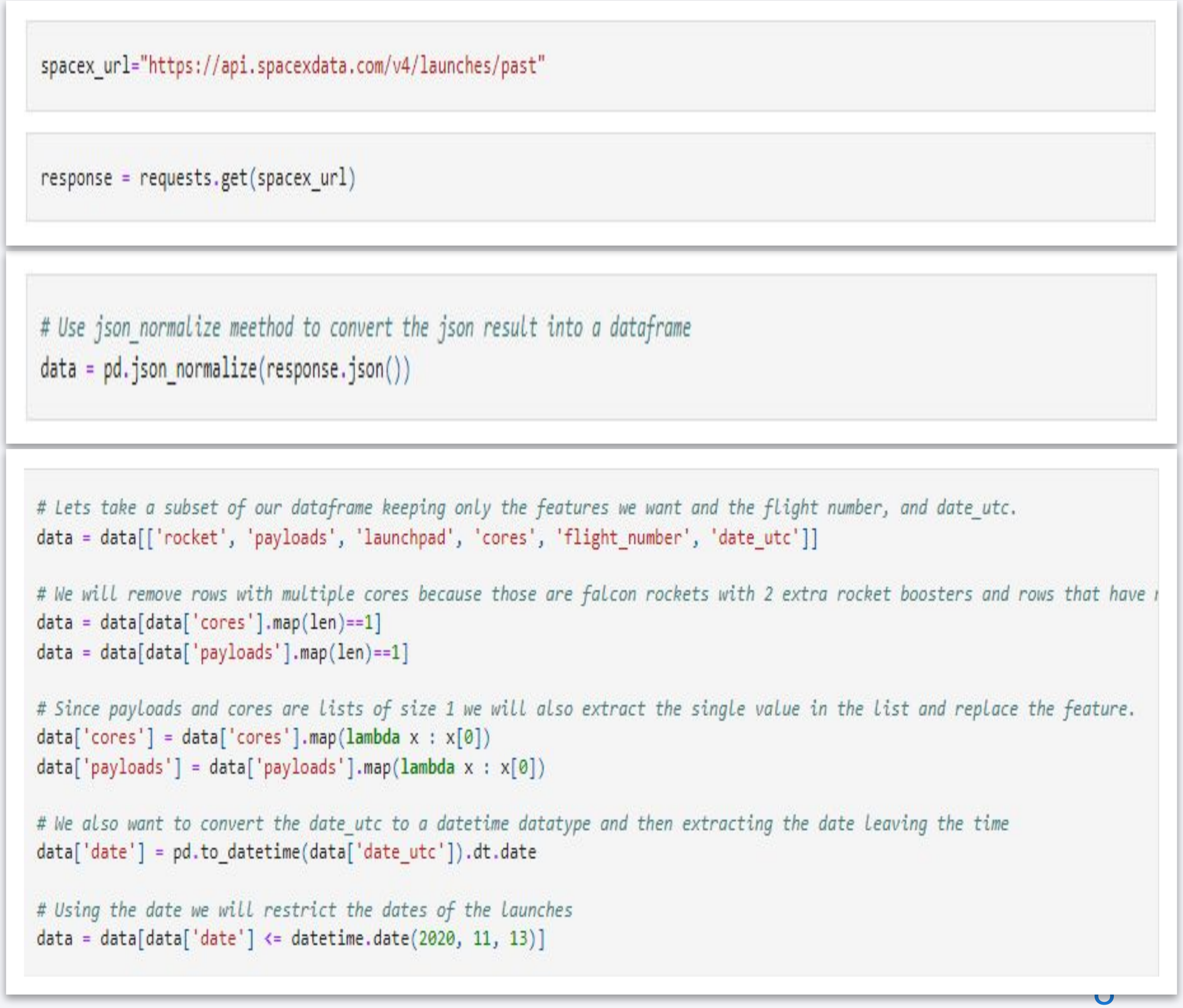
Data Collection

- Data collection involves gathering and measuring information on specific variables within a system to answer questions and evaluate outcomes. In this project, the dataset was obtained through REST API and web scraping from Wikipedia. You need to present your data collection process use key phrases and flowcharts
- Initially, the REST API was utilized by initiating a GET request. The response content was decoded as JSON and transformed into a pandas data frame using json_normalize(). Subsequently, data cleaning procedures were implemented, including the identification and handling of missing values.
- Additionally, for web scraping, Beautiful Soup was employed to extract launch records from HTML tables. The extracted data was parsed and converted into a pandas data frame to facilitate further analysis.

Data Collection – SpaceX API

- Get Request for rocket launch data using REST API
- Using Json_normalize method to convert json result to dataframe
- Data cleaning

https://github.com/A1ty/applied-data-science-capstone/blob/main/notebook_Data_Collection.ipynb



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

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

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

```
# 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
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)]
```

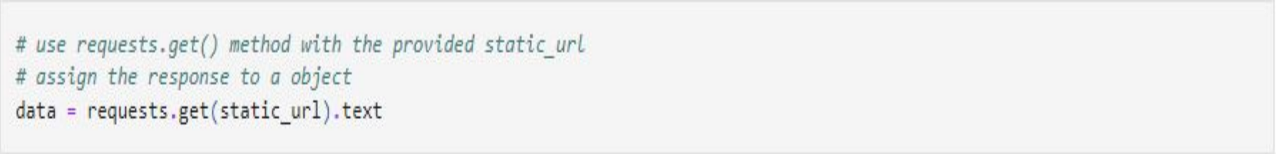
Data Collection - Scraping

- Request the Falcon9 Launch Wiki page from url

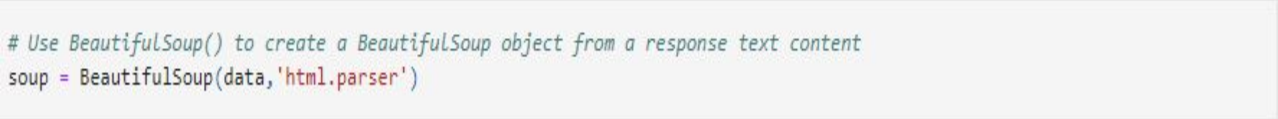
- Create a BeautifulSoup from the HTML response

- Extract all column/variable names from the HTML header

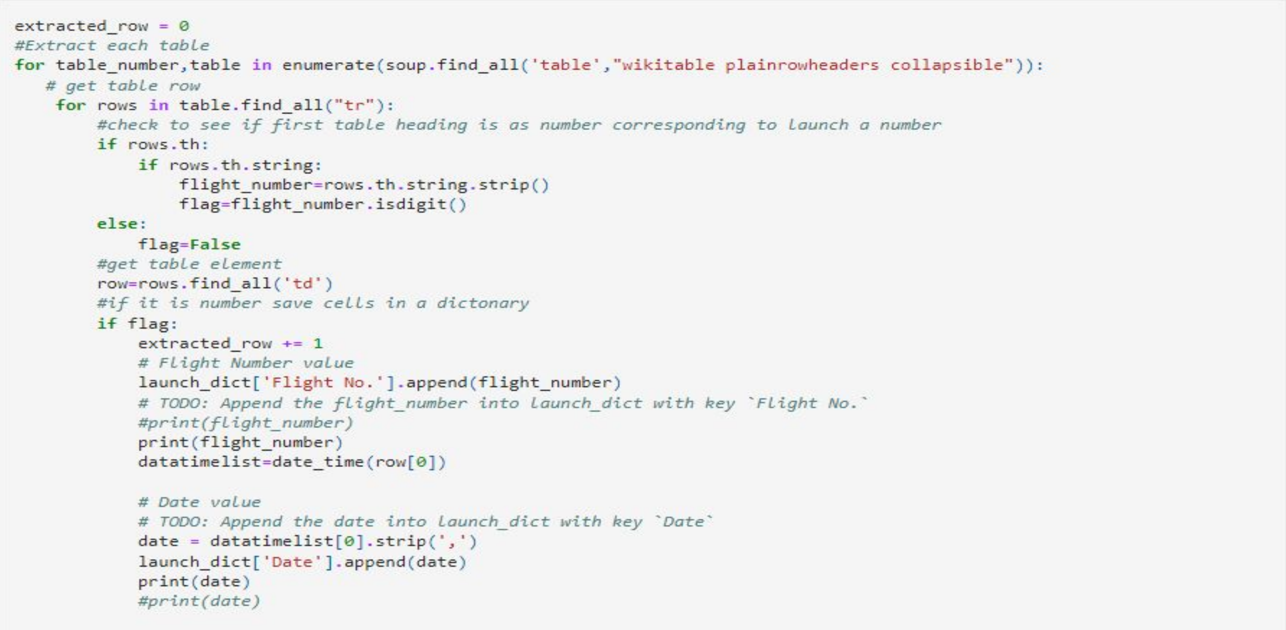
- https://github.com/A1ty/applied-data-science-capstone/blob/main/notebook_Data_Collection_with_Web_Scraping.ipynb



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



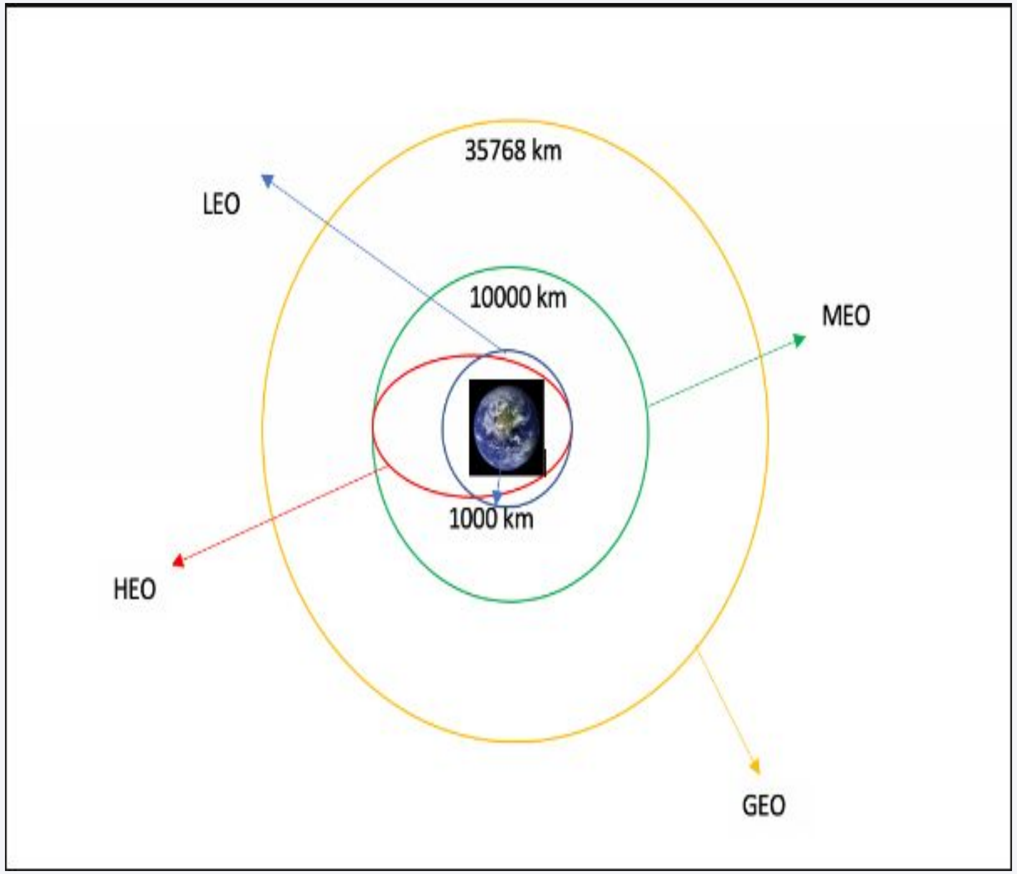
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data,'html.parser')
```



```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            launch_dict['Flight No.'].append(flight_number)
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            print(date)
            #print(date)
```

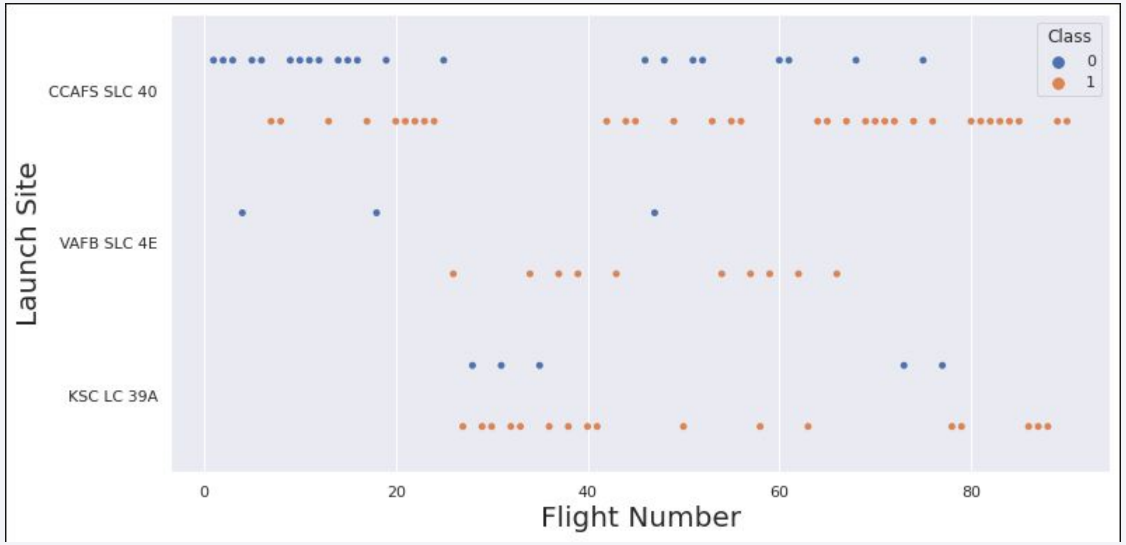
Data Wrangling

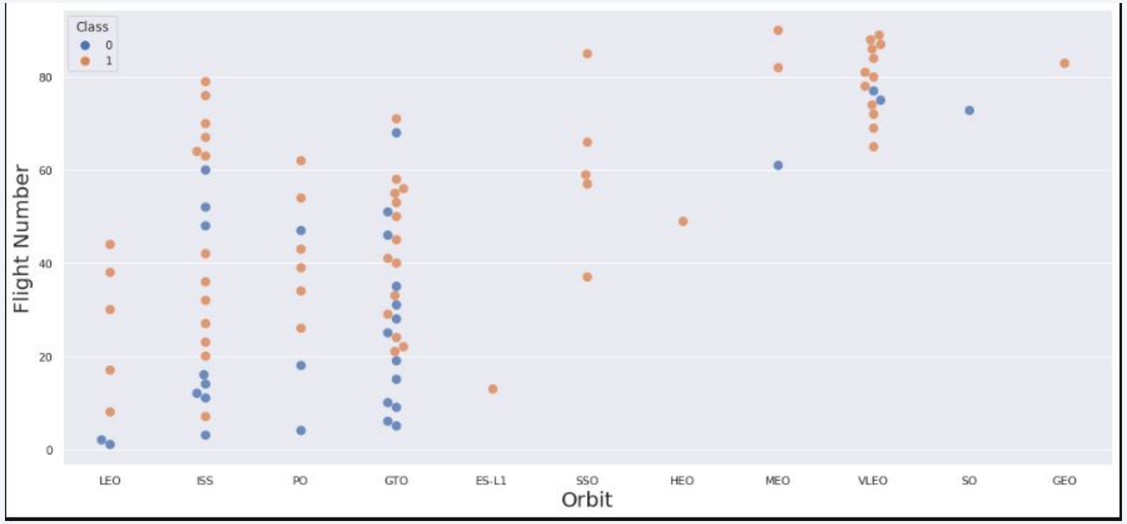


- Data wrangling involves the refinement and consolidation of disorderly and intricate datasets, aiming to streamline accessibility and facilitate Exploratory Data Analysis (EDA).
- Let's start by tallying the number of launches on each site, followed by determining the quantity and frequency of mission outcomes for each orbit type.
- Next, we generate a landing outcome label based on the data in the outcome column. This simplifies subsequent analysis, visualization, and machine learning tasks. Finally, we export the outcome to a CSV file.

- https://github.com/A1ty/applied-data-science-capstone/blob/main/notebook_Data_Wrangling.ipynb

EDA with Data Visualization





- Initially, we utilized scatter plots to examine the correlations between various attributes, including:
    - Payload and Flight Number
    - Flight Number and Launch Site
    - Payload and Launch Site
    - Flight Number and Orbit Type
    - Payload and Orbit Type
- Scatter plots illustrate the interdependence of attributes. When patterns emerge from these graphs, it becomes straightforward to identify the primary factors influencing the success of landing outcomes.

EDA with SQL

- Display the names of the launch sites.
- Show 5 records where launch sites begin with the string 'CCA'.
- Present the total payload mass carried by boosters launched by NASA (CRS).
- Provide the average payload mass carried by booster version F9 v1.1.
- List the date of the first successful landing outcome on a ground pad.
- List the names of boosters that successfully landed on a drone ship with a payload mass greater than 4000 but less than 6000.
- Display the total number of successful and failed mission outcomes.
- List the names of booster versions that carried the maximum payload mass.
- List the failed landing outcomes on drone ships, along with their booster versions and launch site names for the year 2015.
- Rank the count of landing outcomes or successes between the dates June 4, 2010, and March 20, 2017, in descending order.

- https://github.com/A1ty/applied-data-science-capstone/blob/main/notebook_Exploratory_Data_Analysis_with_SQL.ipynb

Build an Interactive Map with Folium

- To visually represent launch data, we plotted an interactive map. Each launch site was marked with a circle marker indicating its latitude and longitude coordinates, accompanied by the launch site name.
- We categorized the launch outcomes (failure and success) into classes 0 and 1, represented respectively by red and green markers on the map within a MarkerCluster.
- Using Haversine's formula, we computed the distances between launch sites and various landmarks to address questions such as:
  a. Proximity of launch sites to railways, highways, and coastlines.
  b. Distance between launch sites and nearby cities.

- https://github.com/A1ty/applied-data-science-capstone/blob/main/notebook_Interactive_Visual_Analytics_with_Folium.ipynb

Build a Dashboard with Plotly Dash

We developed an interactive dashboard using Plotly Dash, enabling users to explore the data flexibly.

- Pie charts were generated to illustrate the total launches from specific sites.
- Scatter plots were created to visualize the relationship between Outcome and Payload Mass (Kg) across various booster versions.

- https://github.com/A1ty/applied-data-science-capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

Building a Model

- Utilize NumPy and Pandas libraries to load the dataset.
- Preprocess the data, including transformation and splitting into training and test sets.
- Determine the suitable machine learning approach for the problem.
- Define parameters and algorithms for GridSearchCV to optimize.
- Fit the model to the dataset for training using the specified parameters.

Evaluating the Model

- Assess model accuracy.
- Retrieve tuned hyperparameters for each algorithm type.
- Visualize the confusion matrix for performance evaluation.

Improve Model

- Use Feature Engineering and Algorithm Tuning

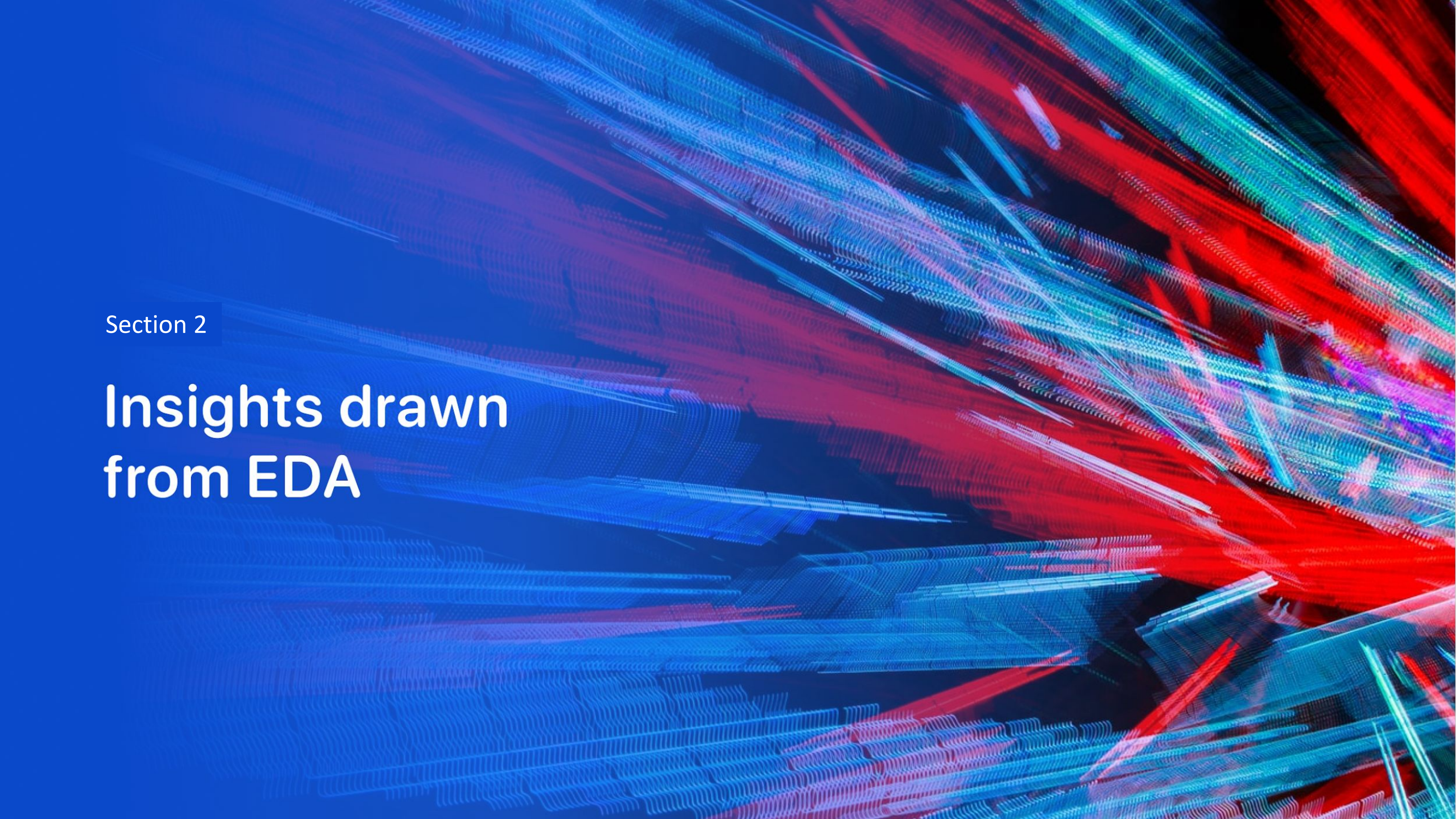
Find the Best Model

•The model with the best accuracy score will be the best performing model.

- https://github.com/A1ty/applied-data-science-capstone/blob/main/spacex_dash_app.py

Results

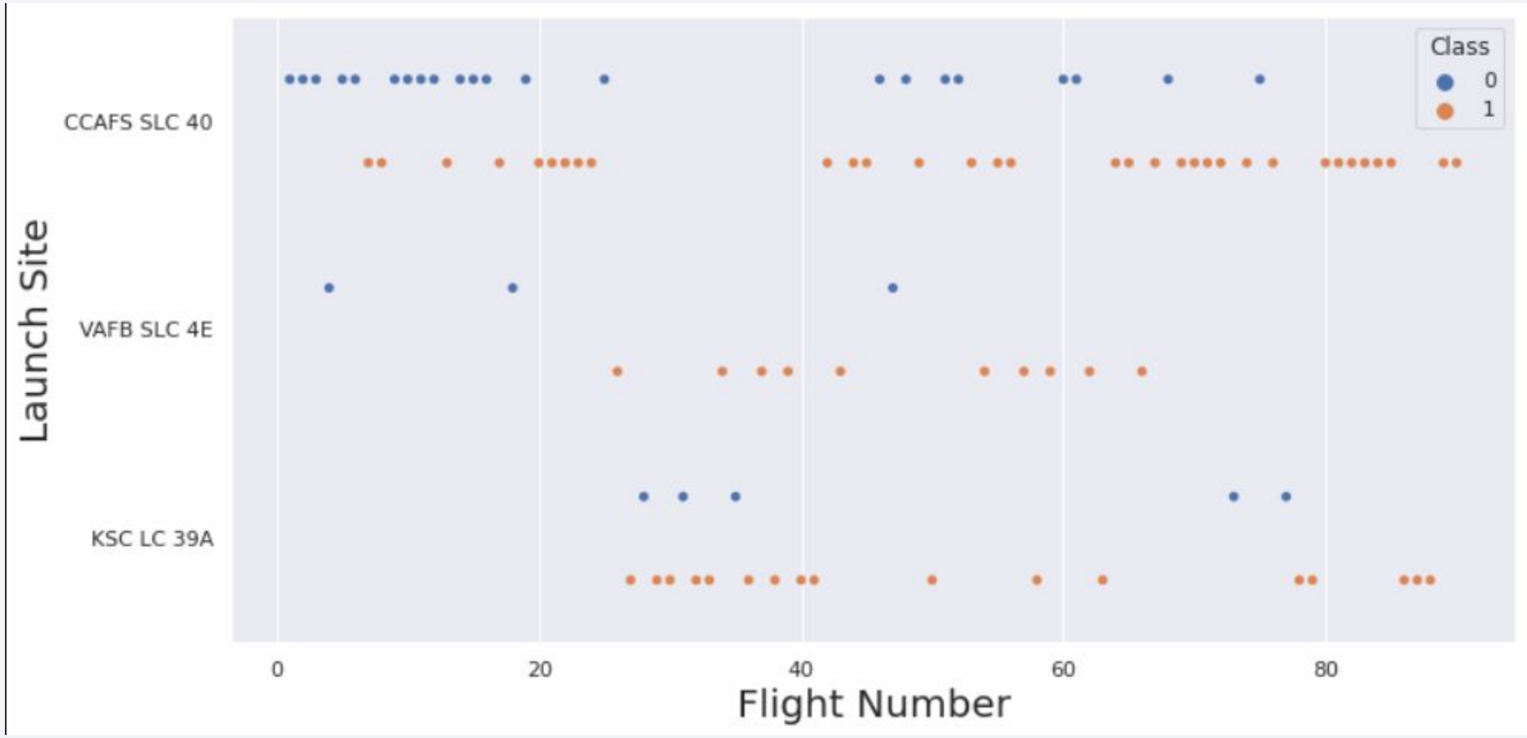
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

Insights drawn from EDA

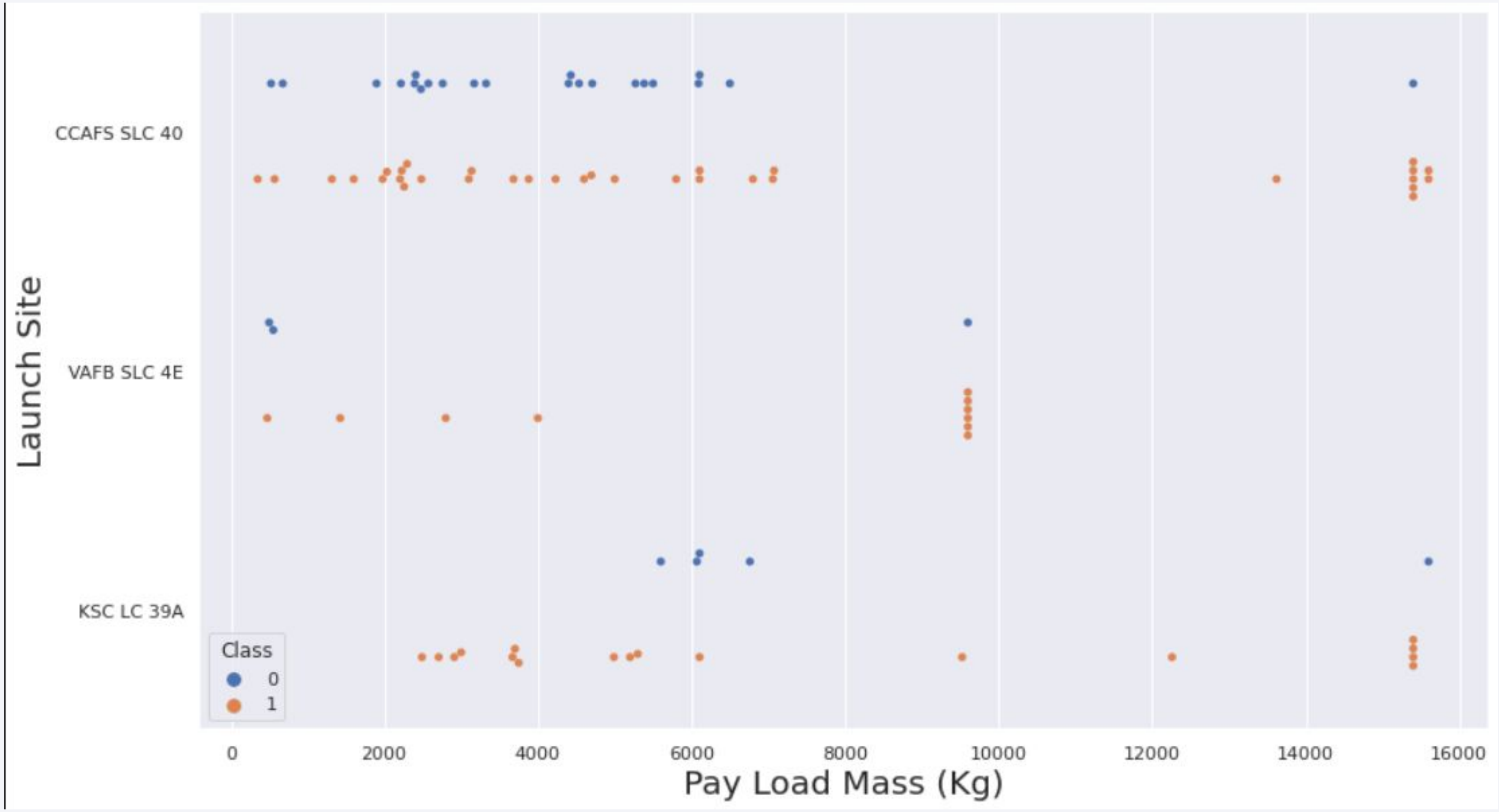
Flight Number vs. Launch Site



- This scatter plot indicates that a higher number of flights from the launch site correlates with a higher success rate.

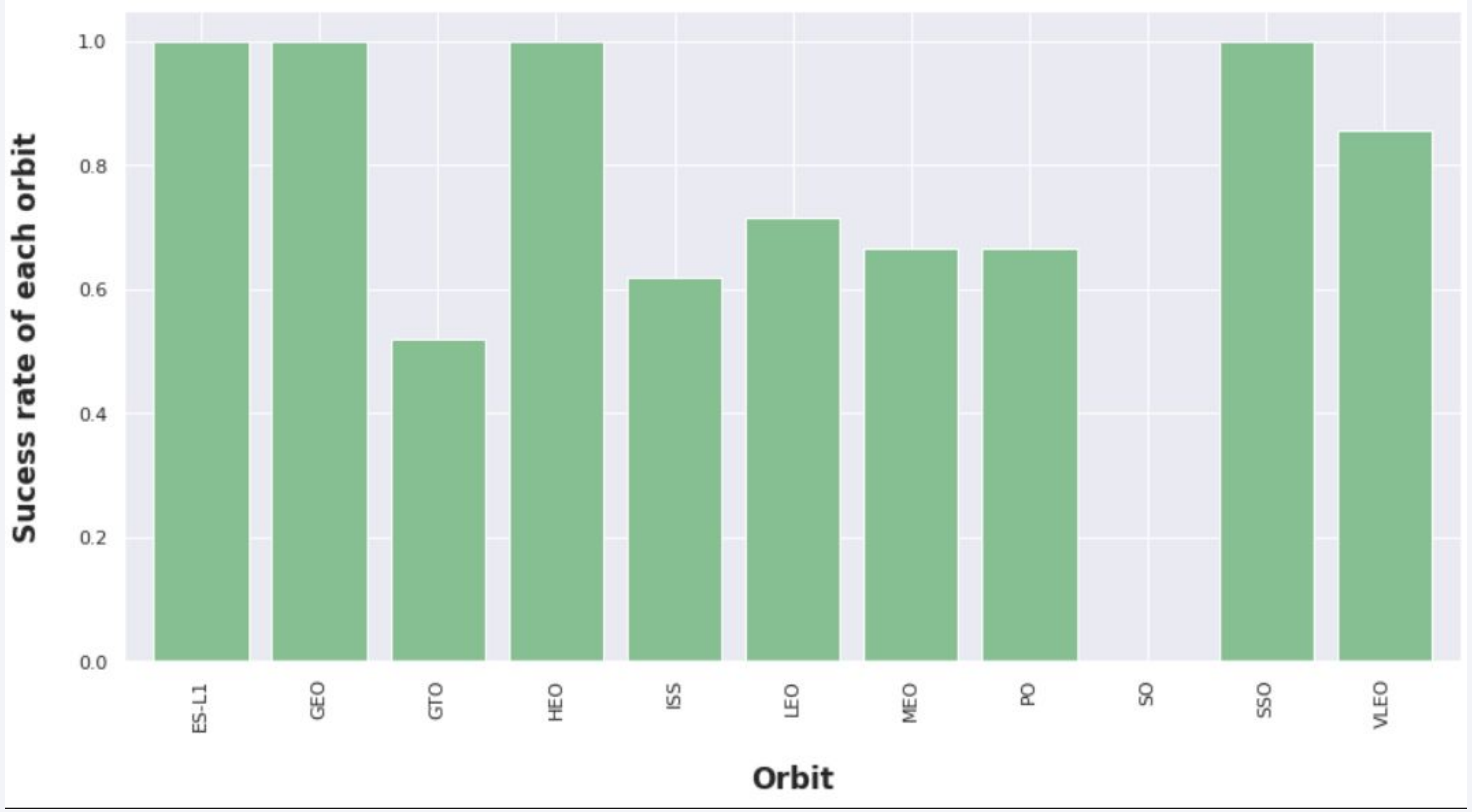
- However, the launch site CCAFS SLC40 displays a weaker pattern in this relationship.

Payload vs. Launch Site



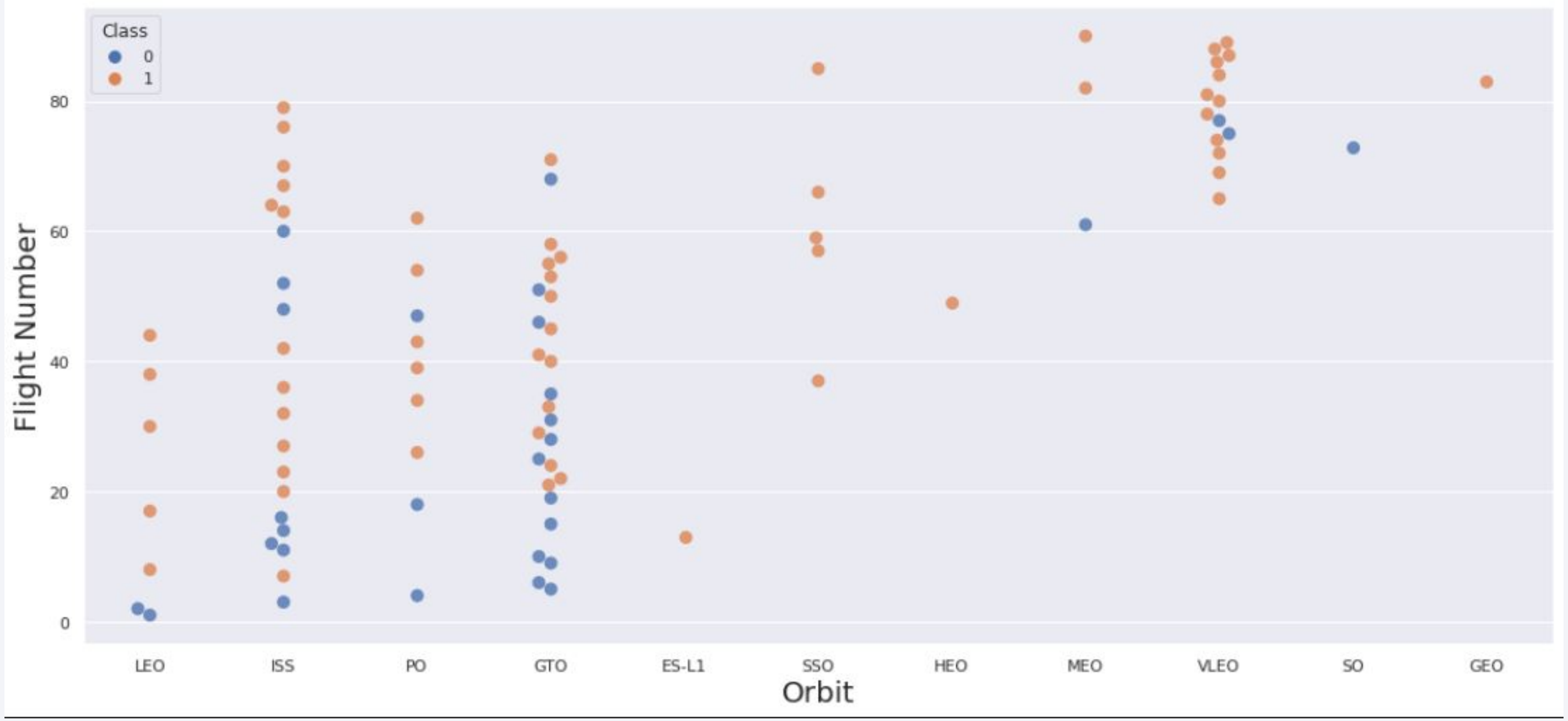
- This scatter plot illustrates that once the payload mass exceeds 7000kg, the probability of success significantly rises.
- However, there's no discernible pattern indicating that the launch site is dependent on the payload mass for the success rate.

Success Rate vs. Orbit Type



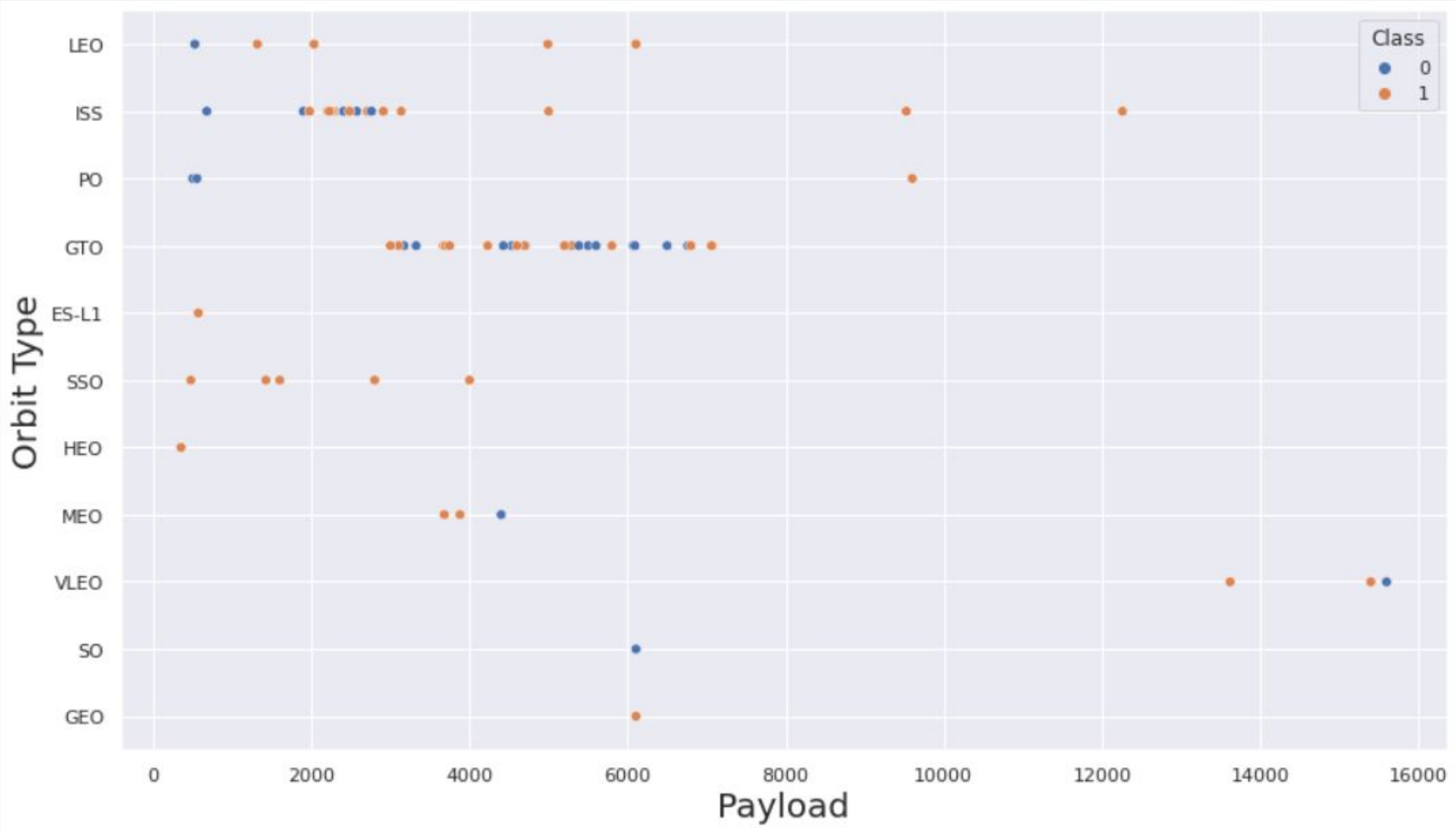
- This figure illustrates how different orbits may influence landing outcomes. Certain orbits, like SSO, HEO, GEO, and ES-L1, exhibit a 100% success rate, while the SO orbit shows a 0% success rate.
- However, upon further analysis, it's evident that some of these orbits have only one occurrence, such as GEO, SO, HEO, and ES-L1. This indicates that more data is needed to discern any patterns or trends before drawing conclusions.

Flight Number vs. Orbit Type



- This scatter plot indicates a general trend: as the flight number increases for each orbit, the success rate tends to be higher, particularly for the LEO orbit.
- However, for the GTO orbit, there is no discernible relationship between the two attributes. Orbits with only one occurrence should be excluded from the above statement, as they require additional data for meaningful analysis.

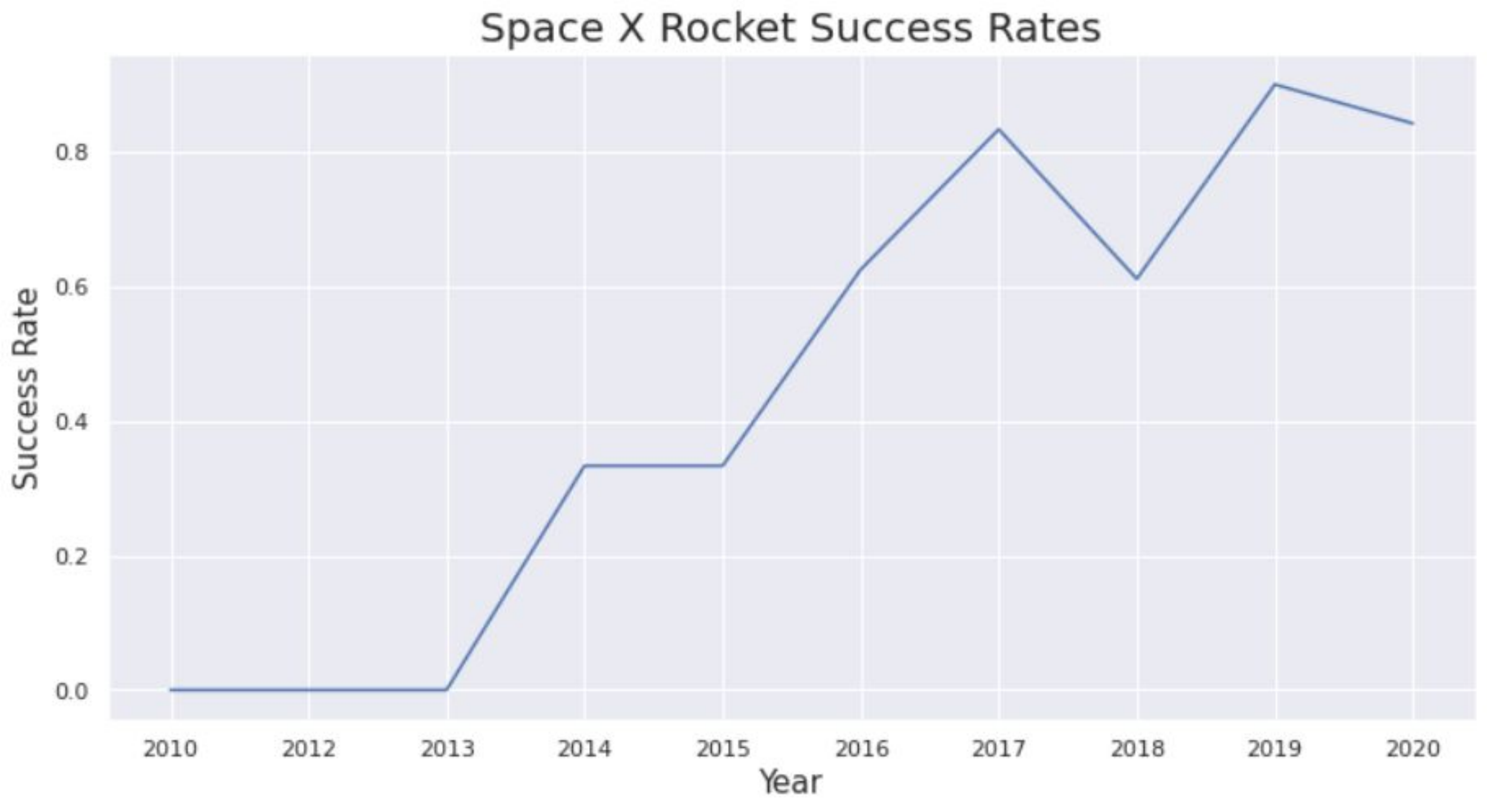
Payload vs. Orbit Type



- Increased payload mass positively affects the success rates of LEO, ISS, and P0 orbits, while it negatively impacts MEO and VLEO orbits.
- However, there seems to be no clear correlation between payload mass and success rate for the GTO orbit.
- Further, SO, GEO, and HEO orbits require additional data to identify any discernible patterns or trends.

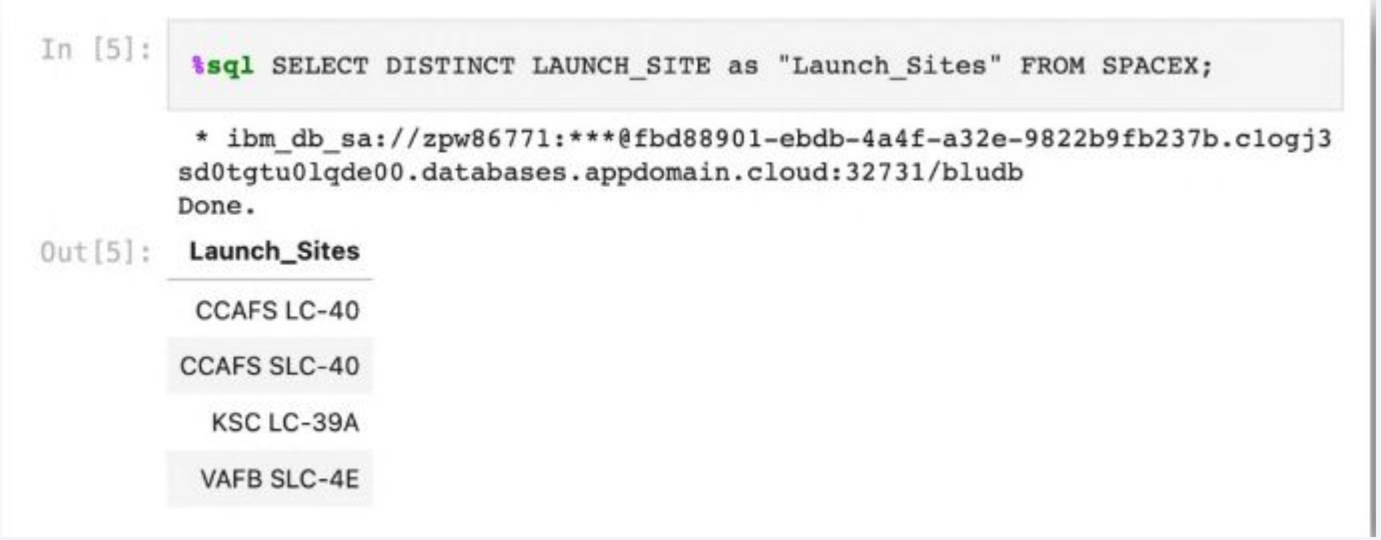
Launch Success Yearly Trend

- These figures clearly illustrate an increasing trend from 2013 until 2020.
- If this trend continues in the following years, the success rate will steadily increase, potentially reaching a 100% success rate.



All Launch Site Names

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



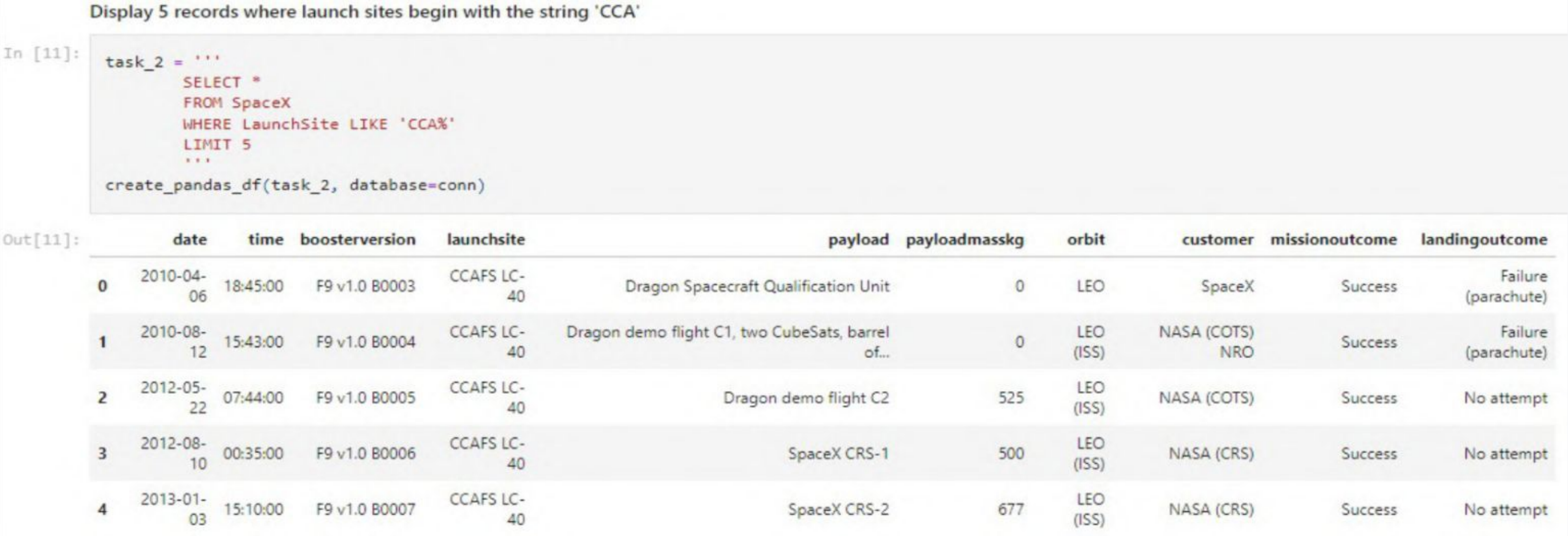
```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEX;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

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



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

In [11]:

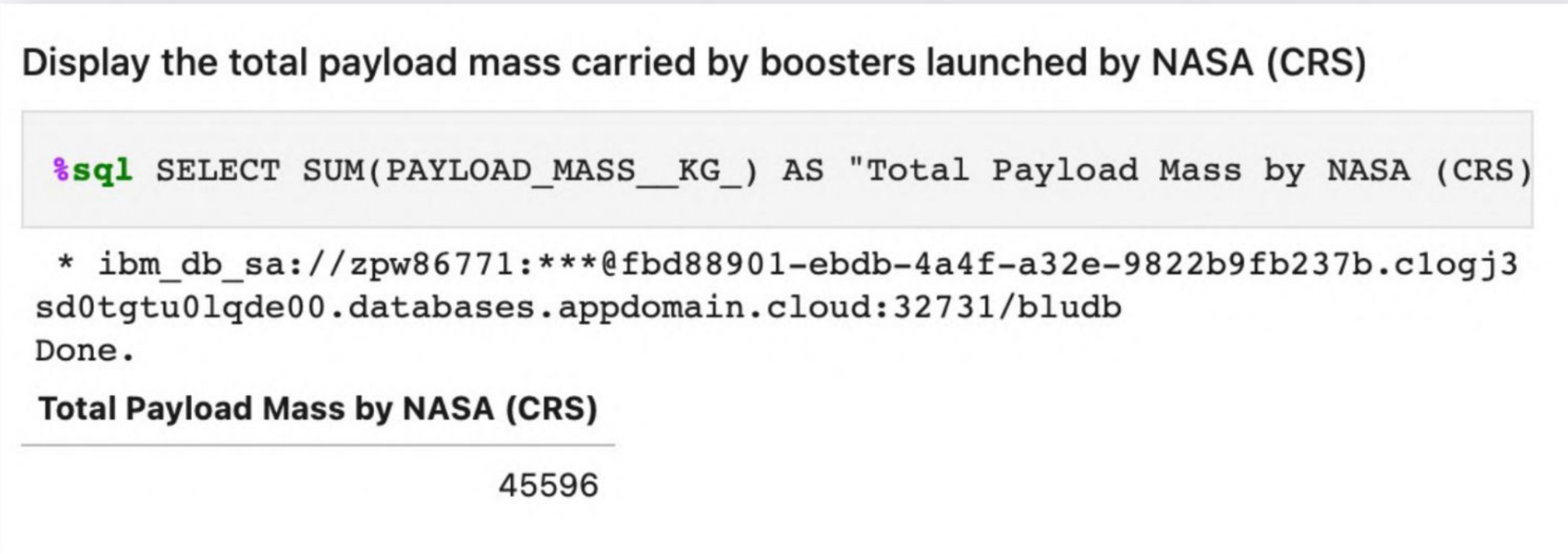
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	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



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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)
```

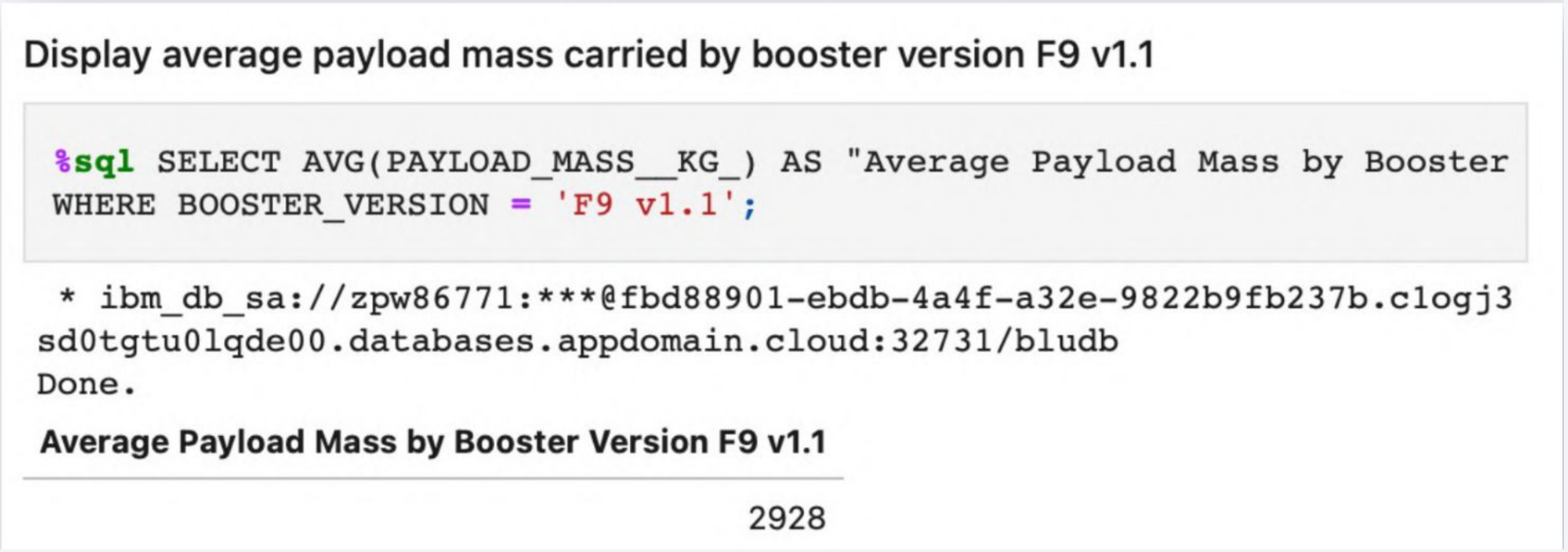
```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Total Payload Mass by NASA (CRS)
45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version

  F9 v1.1 as 2928.4



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

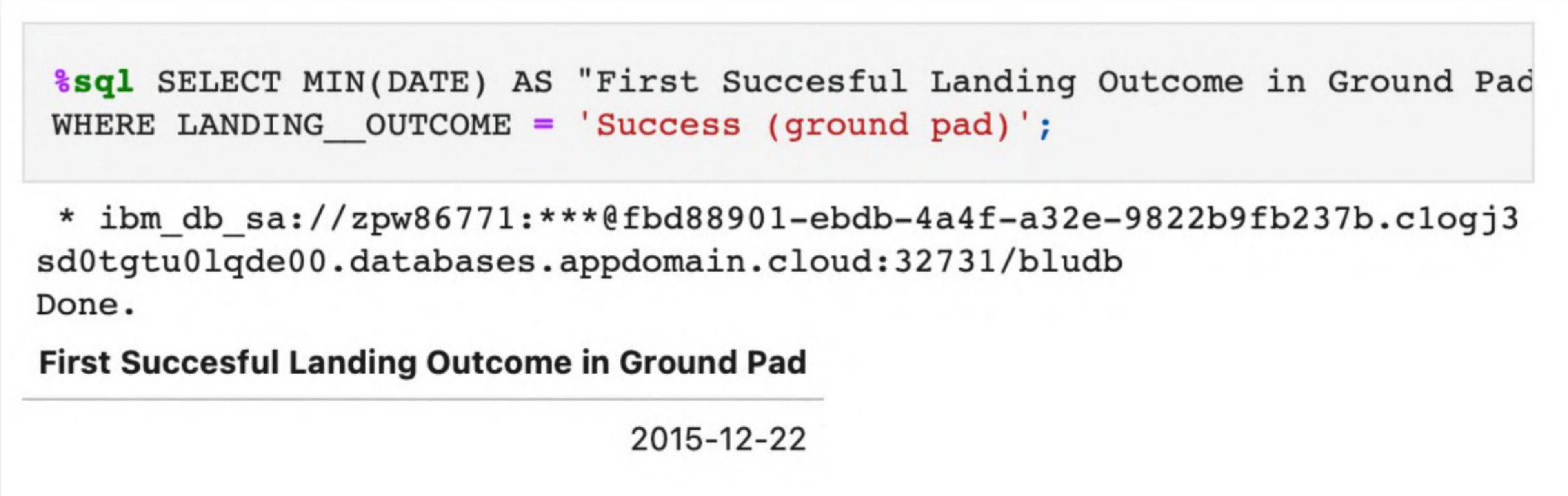
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Average Payload Mass by Booster Version F9 v1.1
2928

First Successful Ground Landing Date

- We use the min() function to find the result
- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015



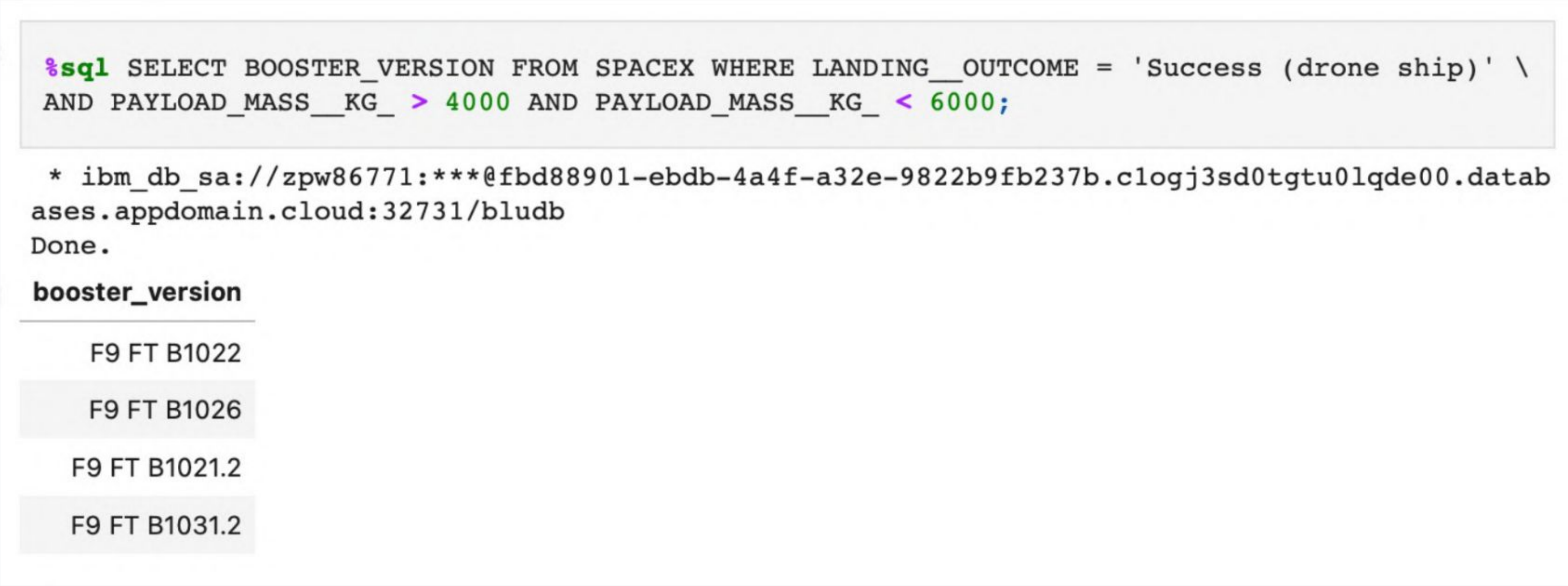
```
%sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

First Succesful Landing Outcome in Ground Pad
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



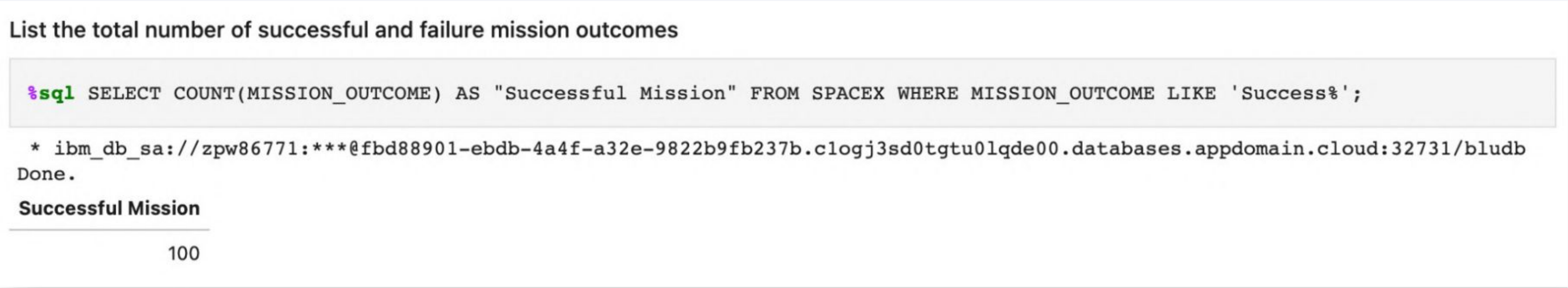
```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.datab
ases.appdomain.cloud:32731/bludb
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

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

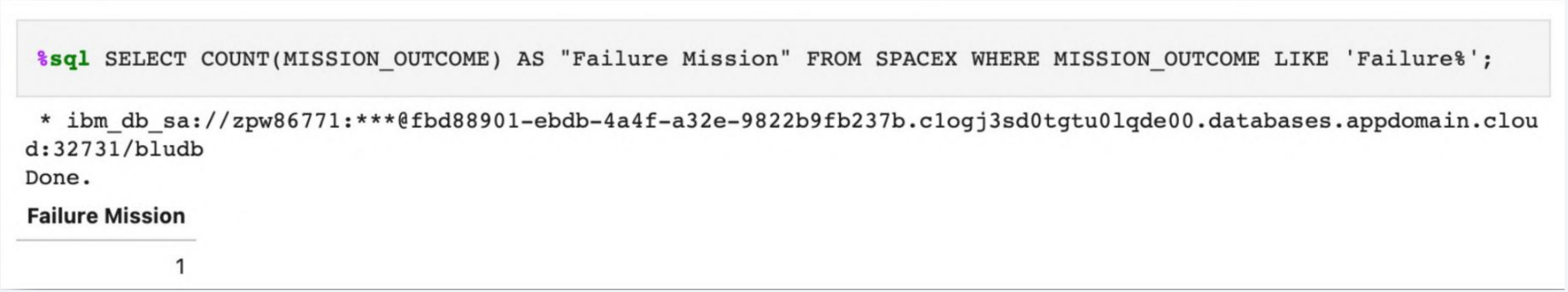


List the total number of successful and failure mission outcomes

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Success%';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Successful Mission
100

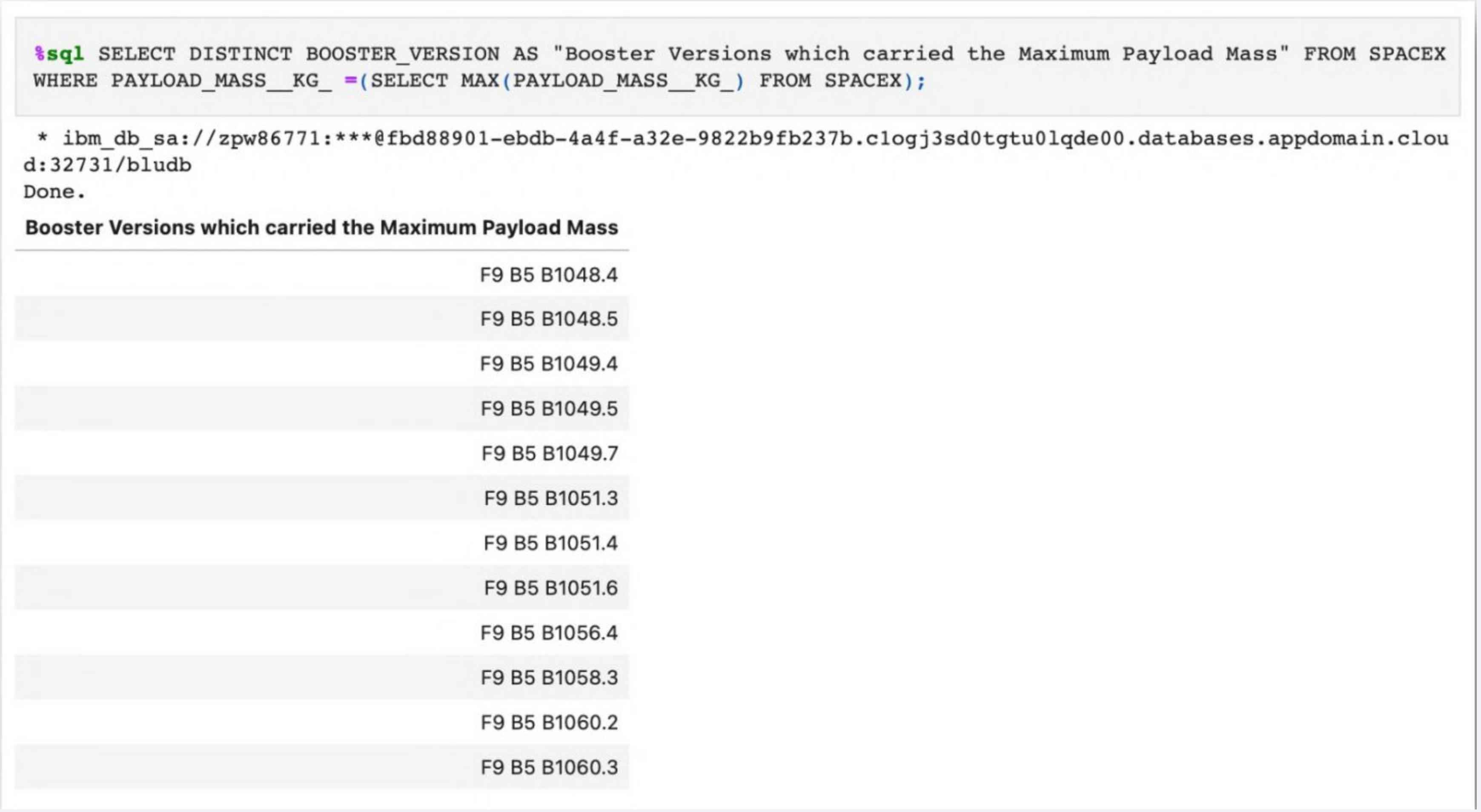


```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Failure Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Failure%';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.clou
d:32731/bludb
Done.
```

Failure Mission
1

Boosters Carried Maximum Payload



```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEX
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX);
```

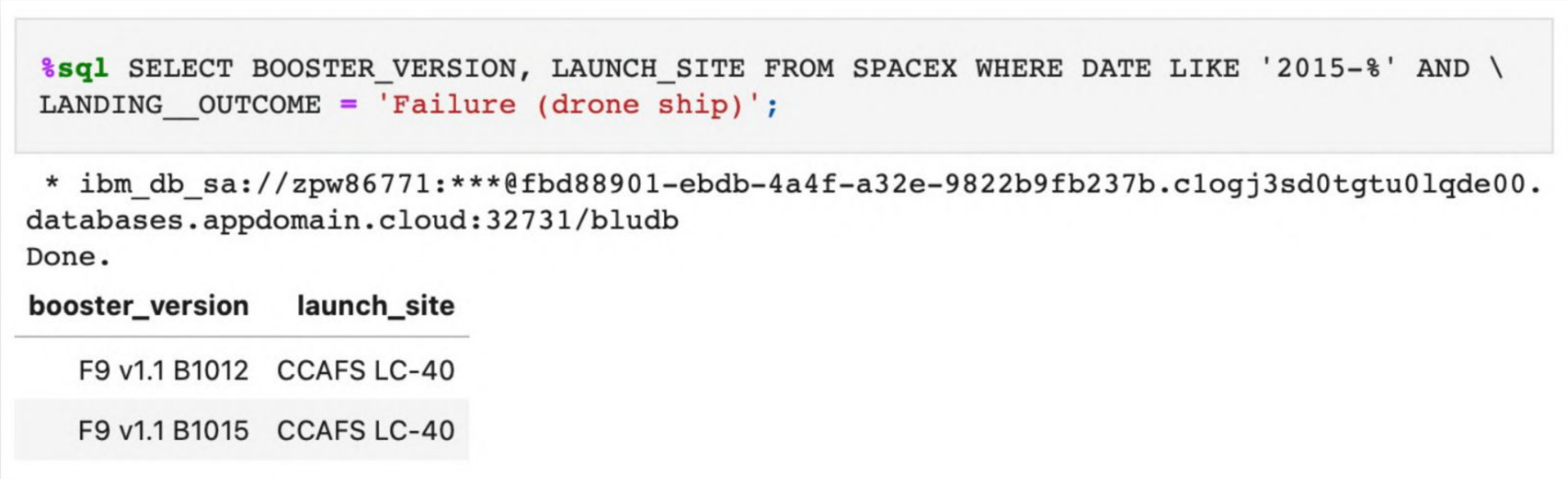
```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lqde00.databases.appdomain.clou
d:32731/bludb
Done.
```

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

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

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



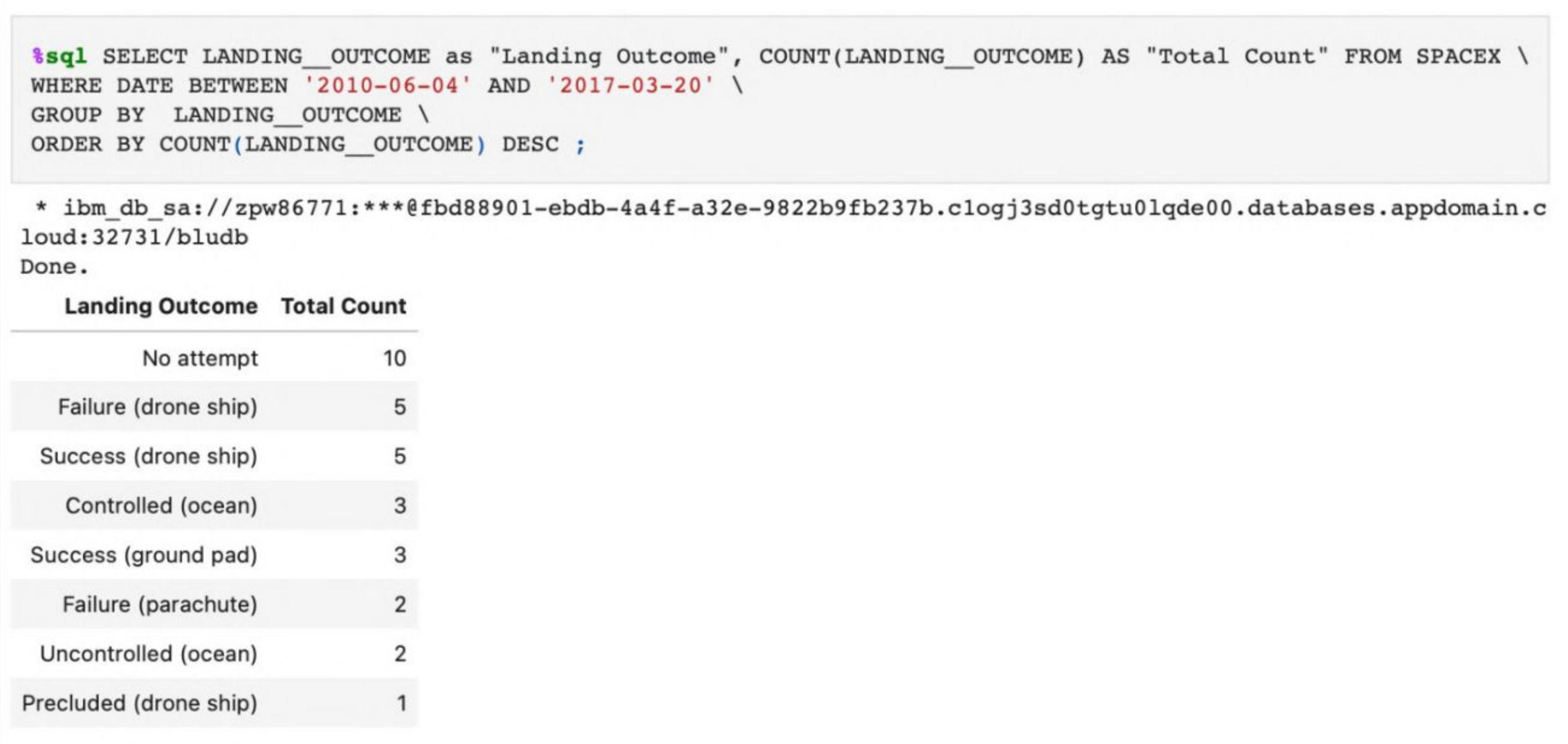
```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE DATE LIKE '2015-%' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lqde00.
databases.appdomain.cloud:32731/bludb
Done.
```

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
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 outcome 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.



```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.c
loud:32731/bludb
Done.
```

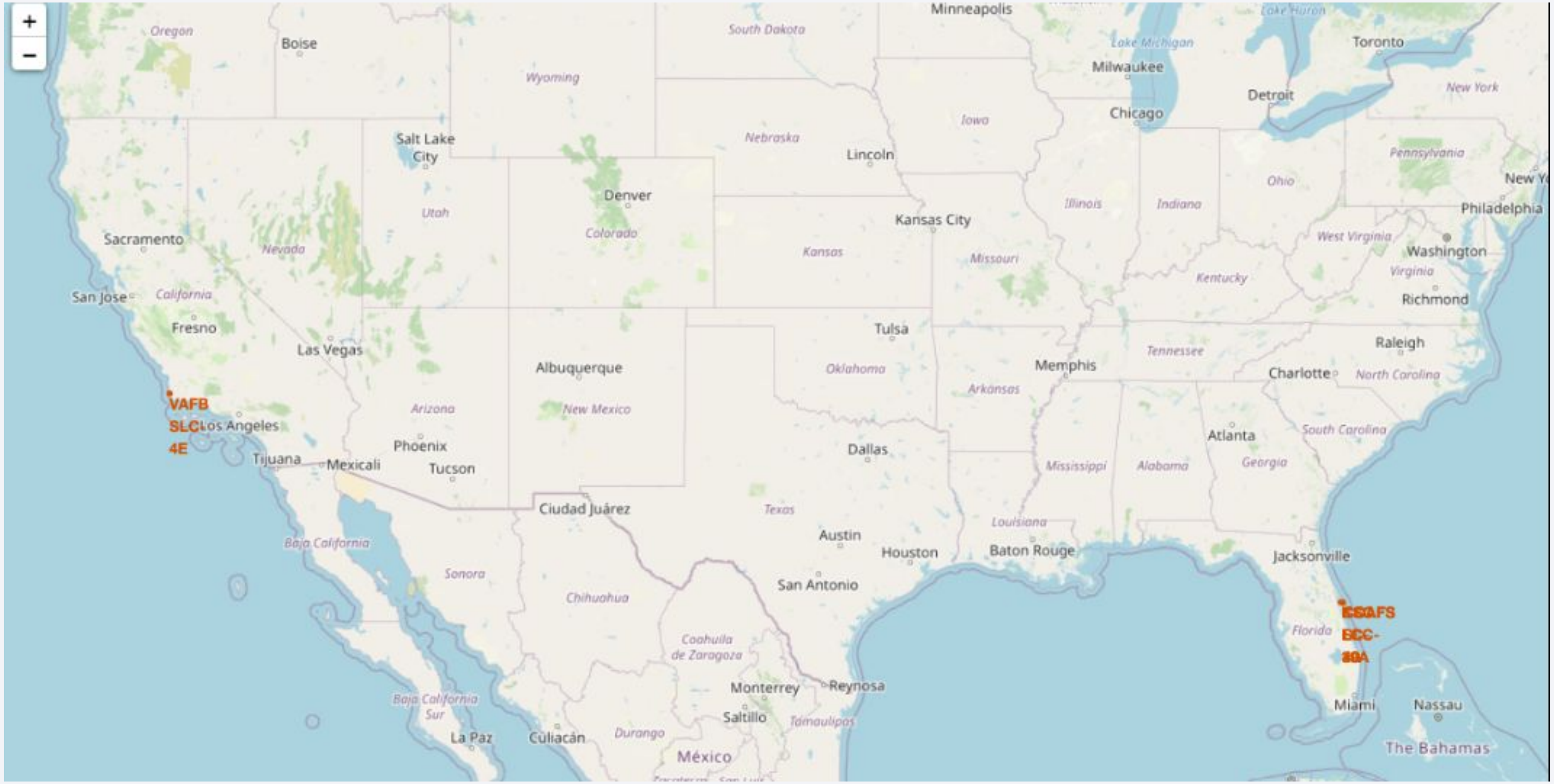
Landing Outcome	Total Count
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Section 3

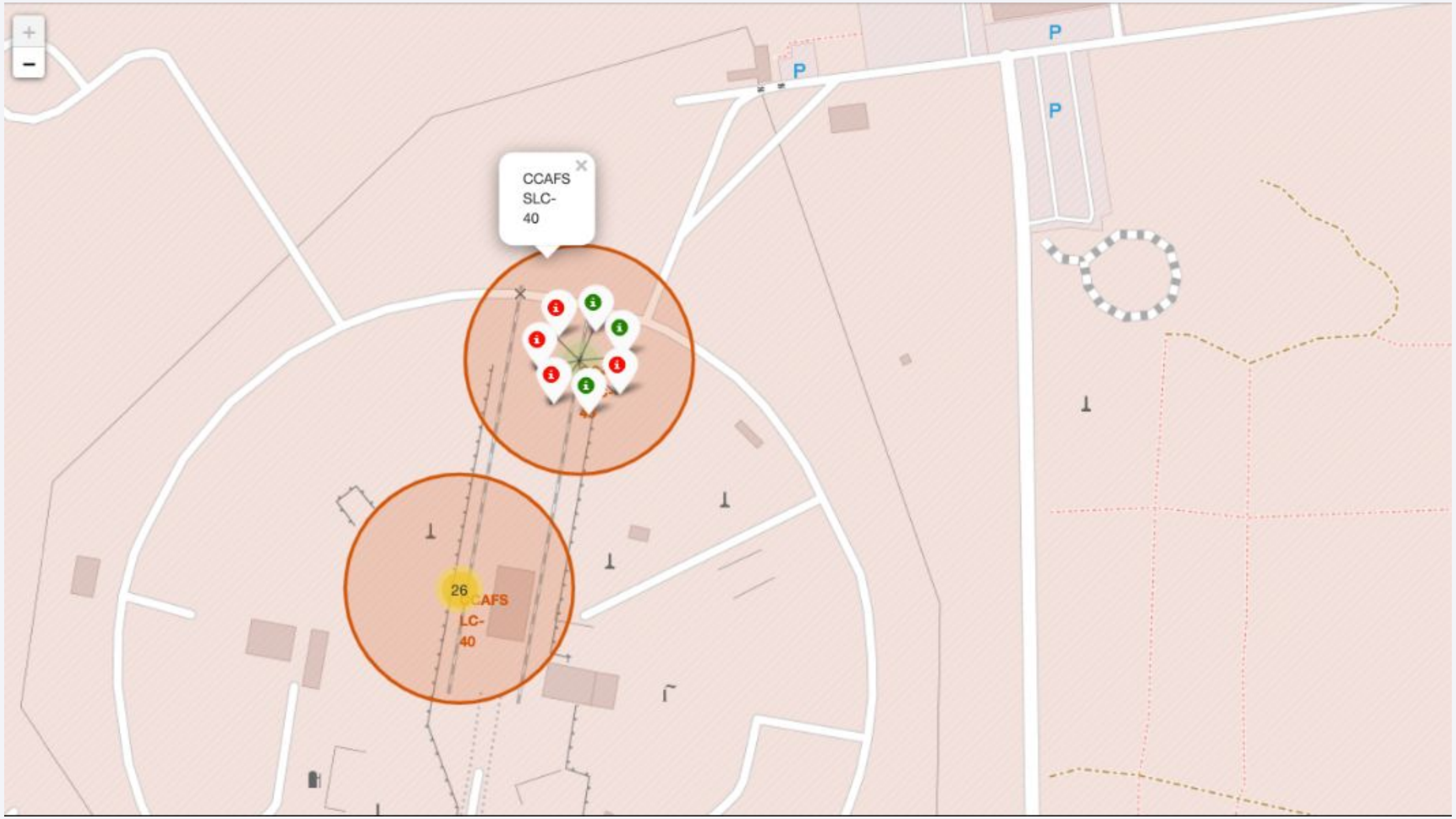
Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

We can see that all the SpaceX launch sites are located inside the United States



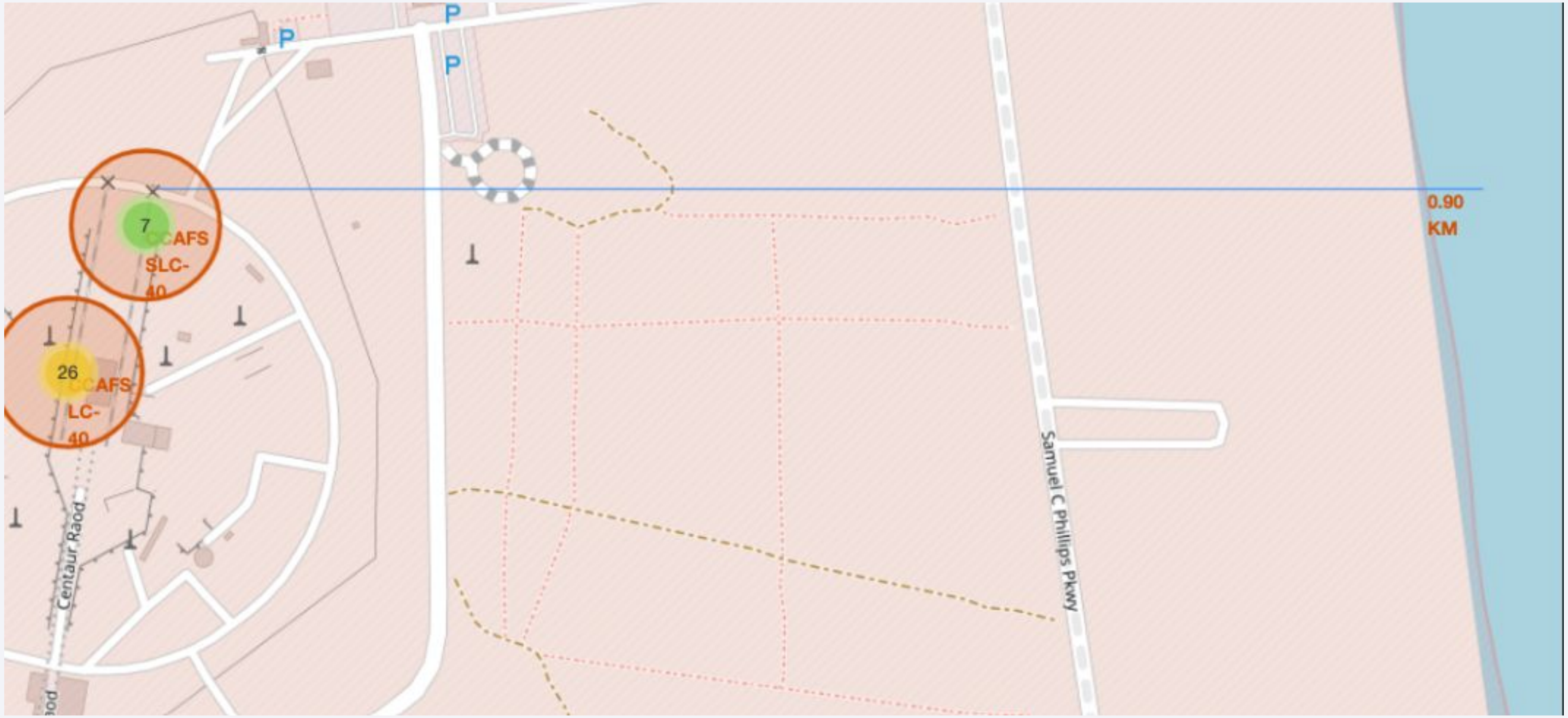
<Folium Map Screenshot 2>



Green Markers = Successful Launches

Red Markers = Failed Launches

<Folium Map Screenshot 3>



Are launches close to railroads? No

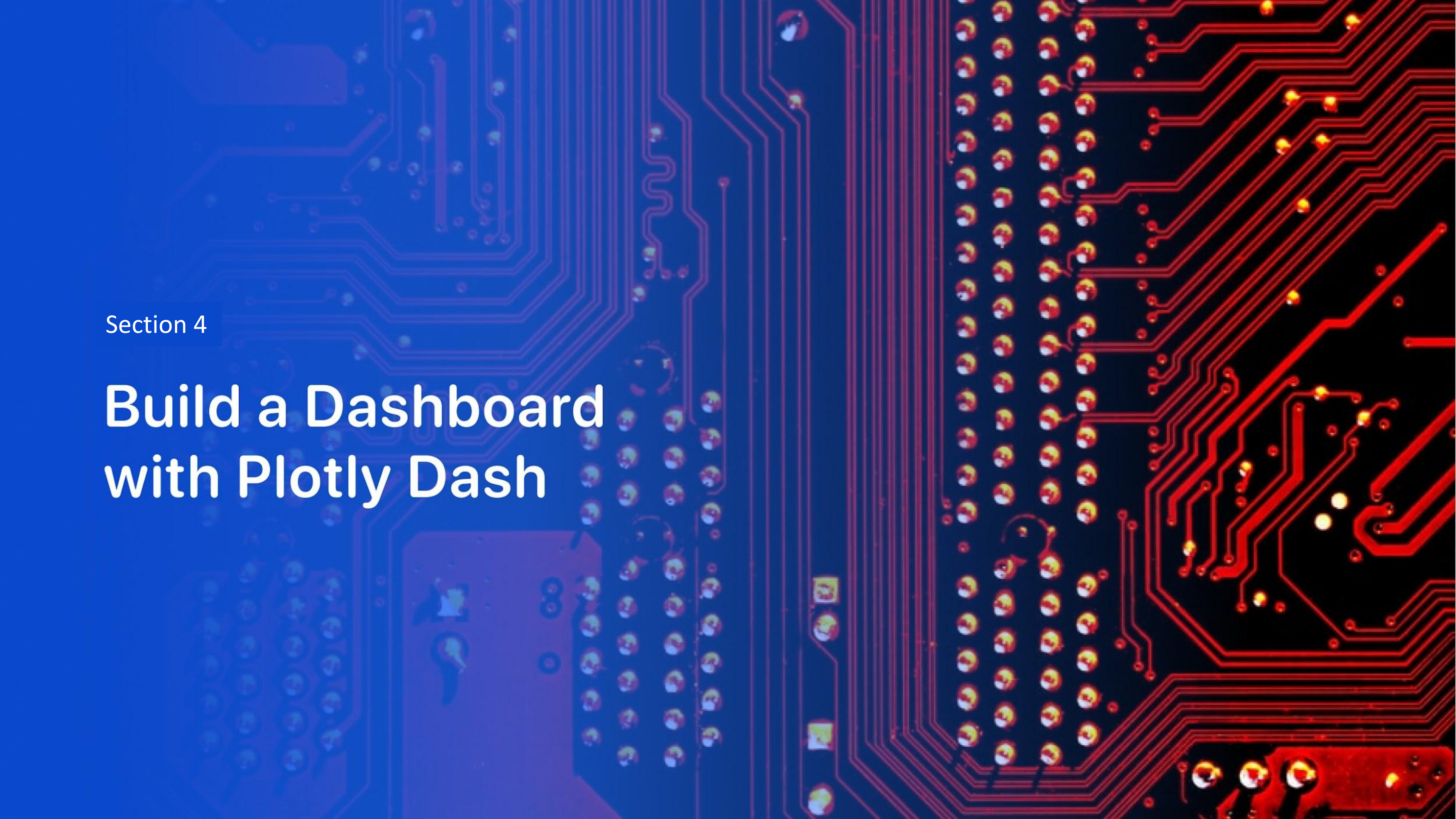
Are launches close to highways? No

Are launches close to coastlines? Yes

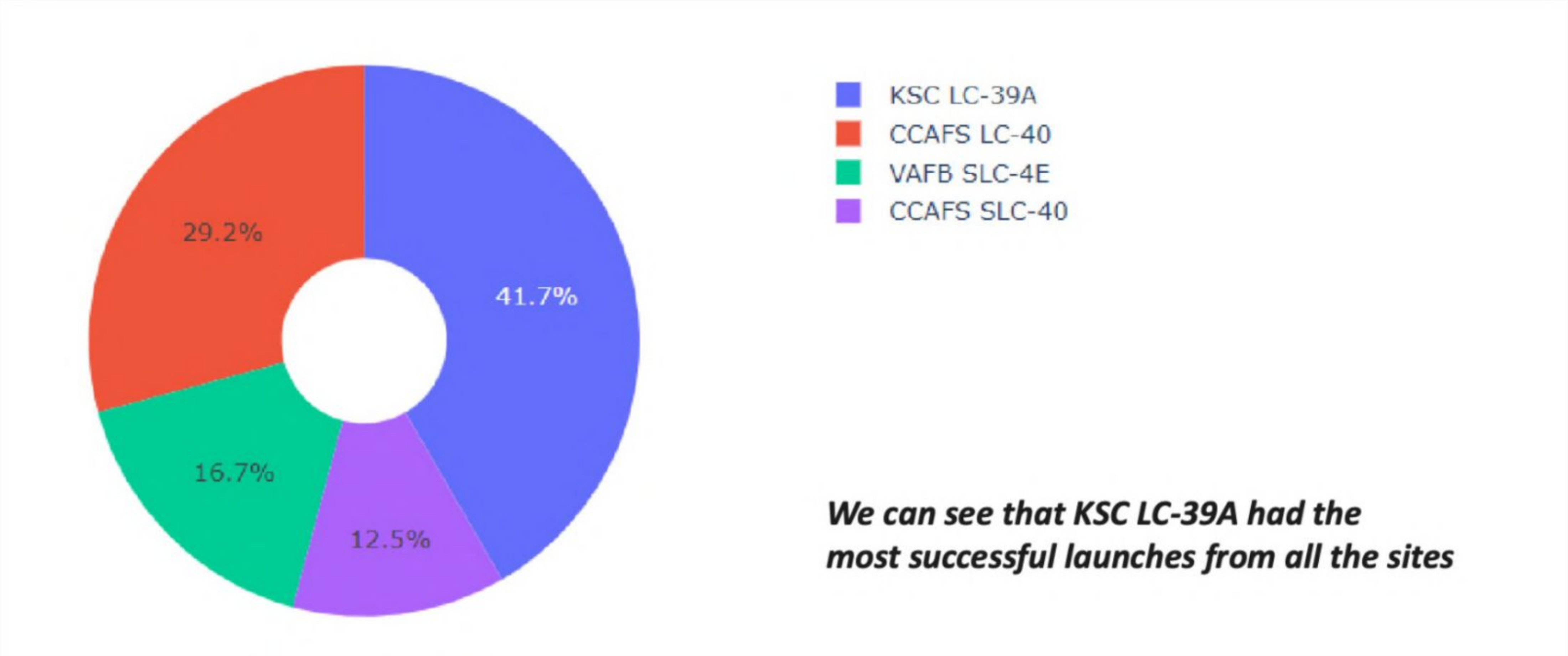
Are launches a certain distance away from cities? Yes

Section 4

Build a Dashboard with Plotly Dash

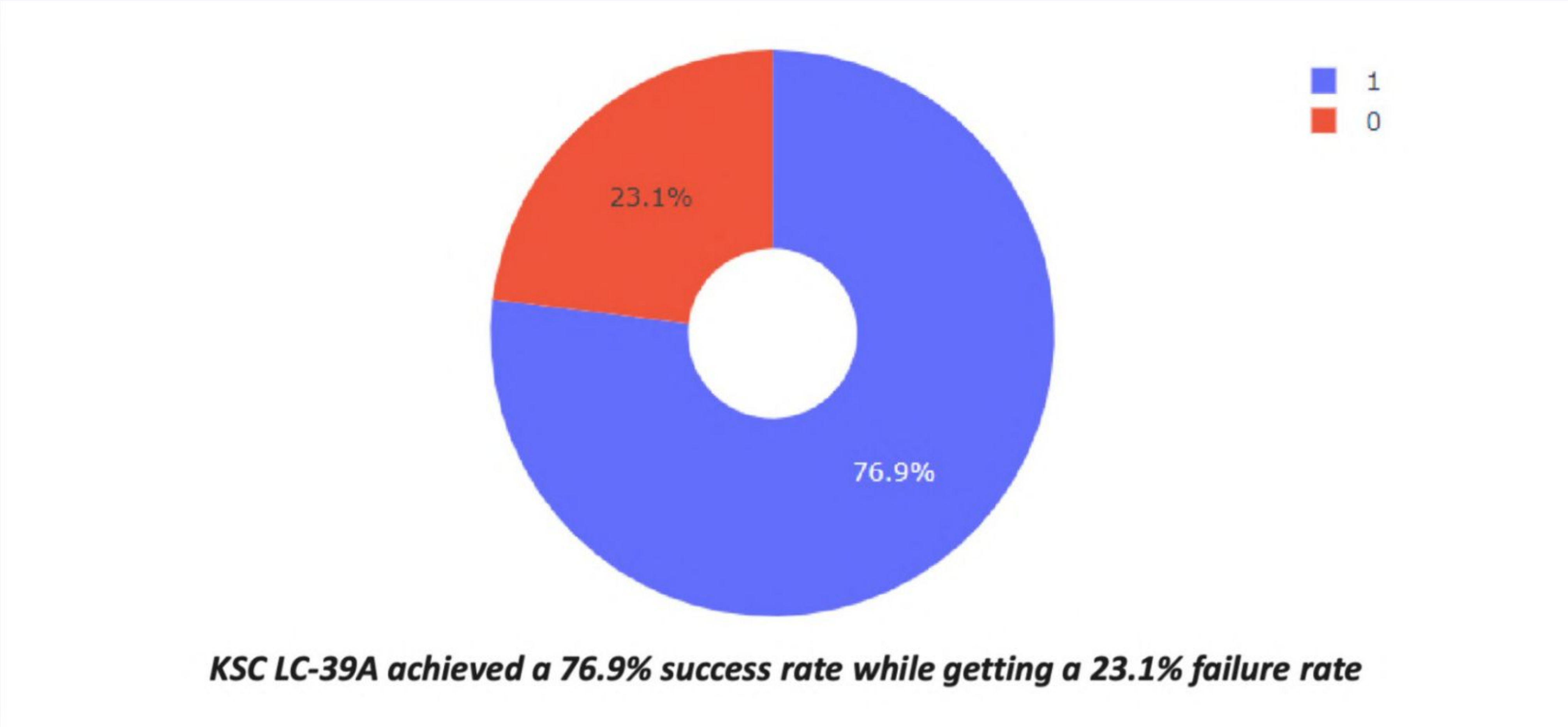


Successful % by each site



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

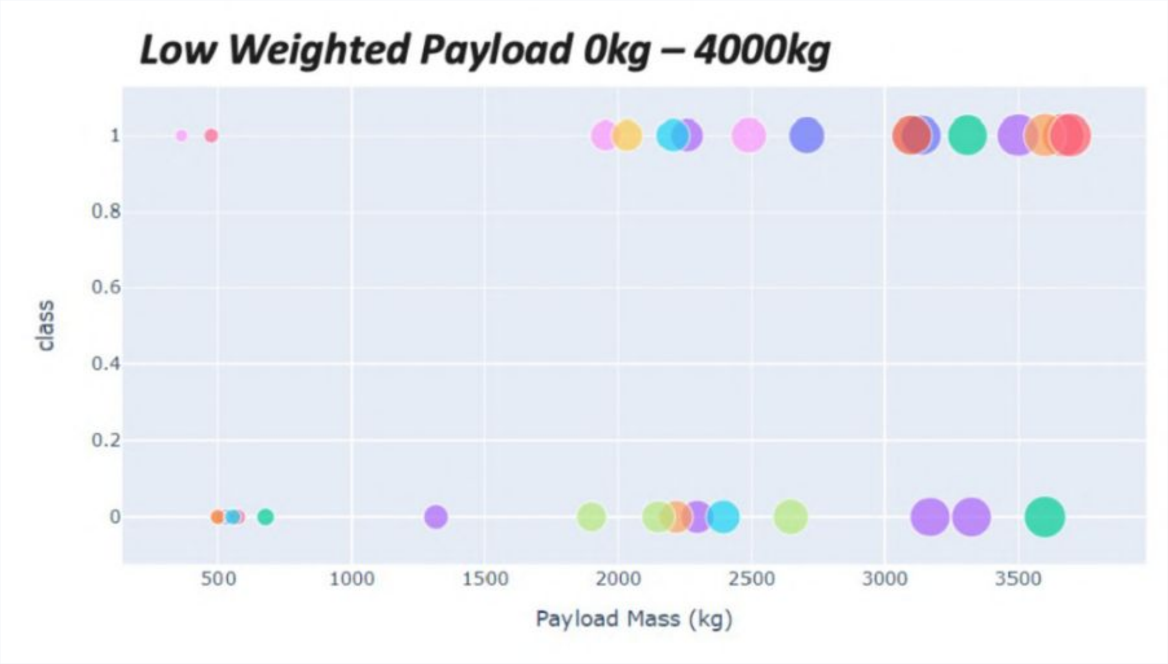
Highest launch-success ratio: KSC LC-39A

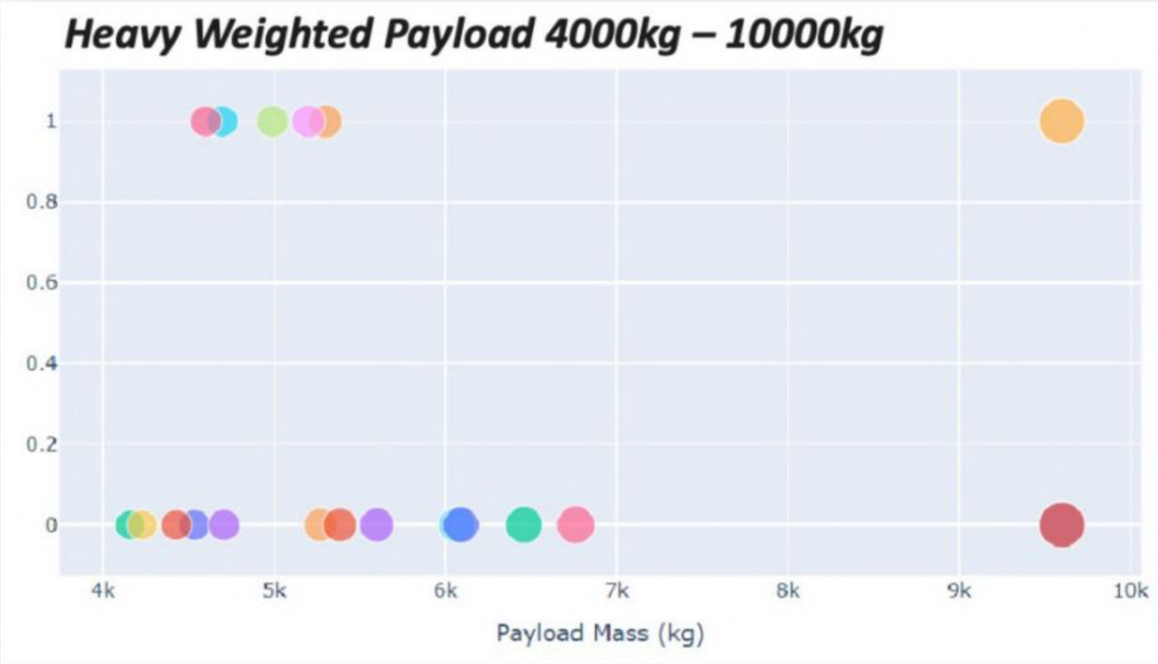


KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Payload vs Launch Outcome Scatter Plot

We can see that all the success rate for low weighted payload is higher than heavy weighted payload



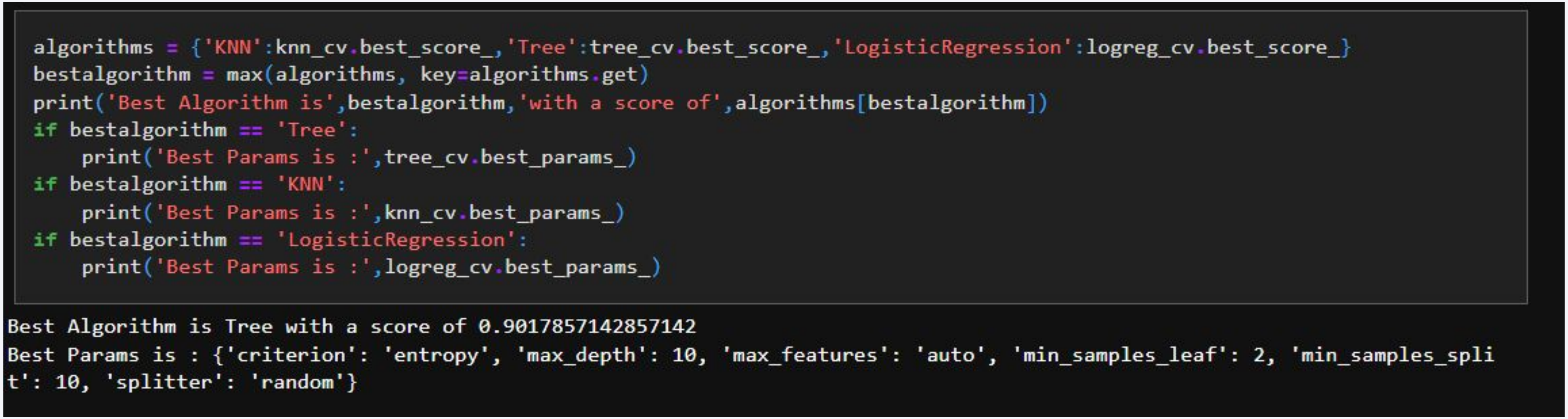


Section 5

Predictive Analysis (Classification)

Classification Accuracy

As we can see, by using the code as below: we could identify that the best algorithm to be the Tree Algorithm which have the highest classification accuracy.

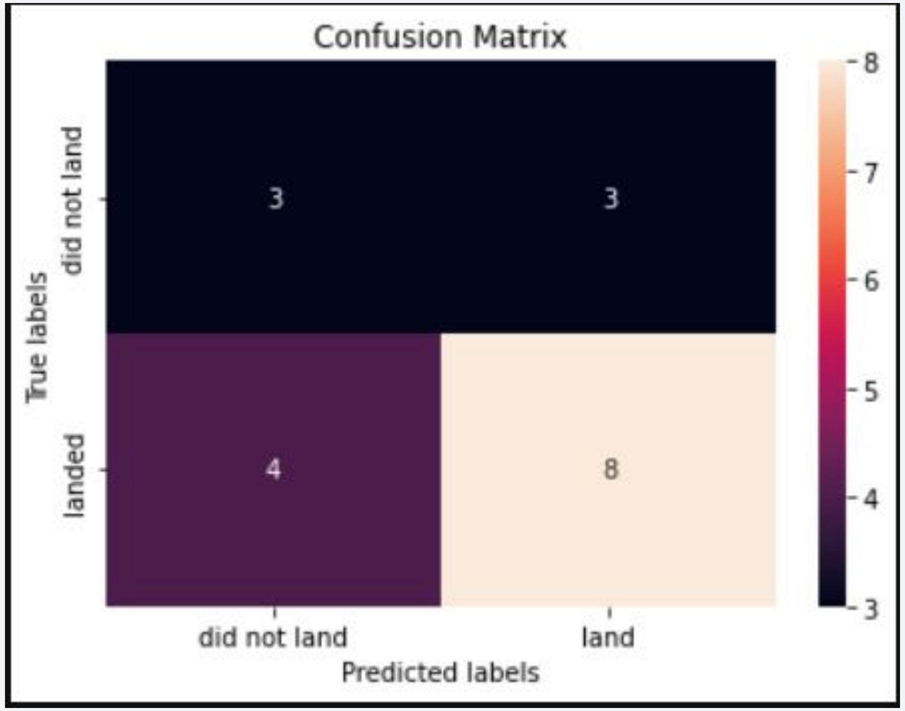


```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.9017857142857142
Best Params is : {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_spli
t': 10, '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

In summary:

- The Tree Classifier Algorithm proves to be the most effective machine learning approach for this dataset.
- Lighter payloads (4000kg and below) demonstrate better performance compared to heavier payloads.
- Since 2013, the success rate of SpaceX launches has steadily increased, indicating a promising trend towards perfecting launches in the future.
- KSC LC-39A stands out with the highest success rate among all launch sites, at 76.9%.
- The SSO orbit boasts the highest success rate of 100%, with more than one occurrence.

Thank you!