IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mesut K.
April 2023

Outline

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

Executive Summary

- Summary of methodologies
  - API-based Data Collection
  - Web Scraping for Data Collection
  - Data Wrangling
  - Exploratory Data Analysis using SQL
  - Exploratory Data Analysis with Data Visualization
  - Folium for Interactive Visual Analytics
  - Machine Learning for Predictive Modeling
- Summary of all results
  - Exploratory Data Analysis Findings
  - Screenshots of Interactive Analytics
  - Predictive Analytics Results

Introduction

- Project background and context

  The objective of this project is to develop a machine learning pipeline that can predict whether the first stage of a Falcon 9 rocket will land successfully. The ability to accurately predict the outcome of a rocket launch is crucial for determining the cost of the launch. Space X is able to offer lower costs for their launches compared to other providers because they can reuse the first stage. By predicting the success of the landing, other companies interested in bidding for a rocket launch against Space X can estimate the cost and make an informed decision. Ultimately, this project aims to provide a valuable tool for companies seeking to compete in the space launch market.

- Problems you want to find answers

  Determining all the factors that impact the outcome of landing.

  Examining the correlation between each variable and its effect on the landing result.

  Identifying the optimal conditions necessary to enhance the likelihood of a successful landing.

Section 1

Methodology

Methodology

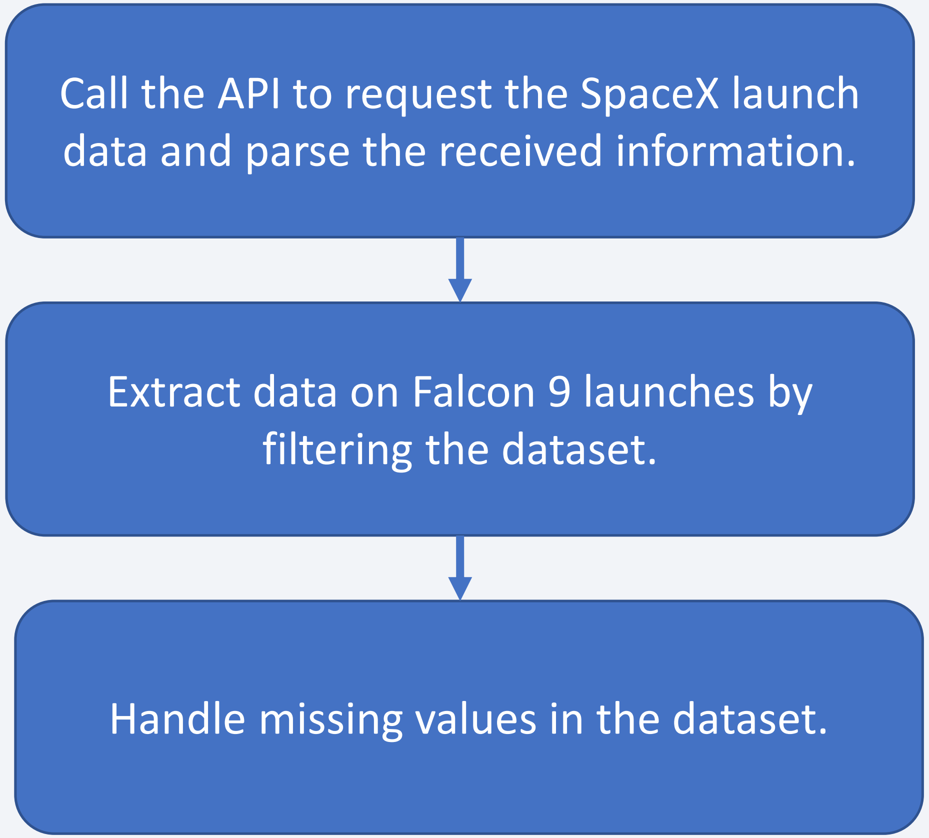
Executive Summary

- Data collection methodology:
  - The data was obtained by utilizing both the SpaceX API and web scraping information from Wikipedia.
- Perform data wrangling
  - Categorical features were encoded using one-hot encoding during the data processing phase.
- 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 collection process involved sending a GET request to the SpaceX API, decoding the response content as JSON using the .json() function, and converting it into a pandas dataframe with the .json_normalize() function. We then cleaned the data, checked for missing values, and filled in any missing values as needed. Additionally, we performed web scraping of Falcon 9 launch records from Wikipedia using BeautifulSoup. Our goal was to extract the launch records as an HTML table, parse the table, and convert it into a pandas dataframe for further analysis.

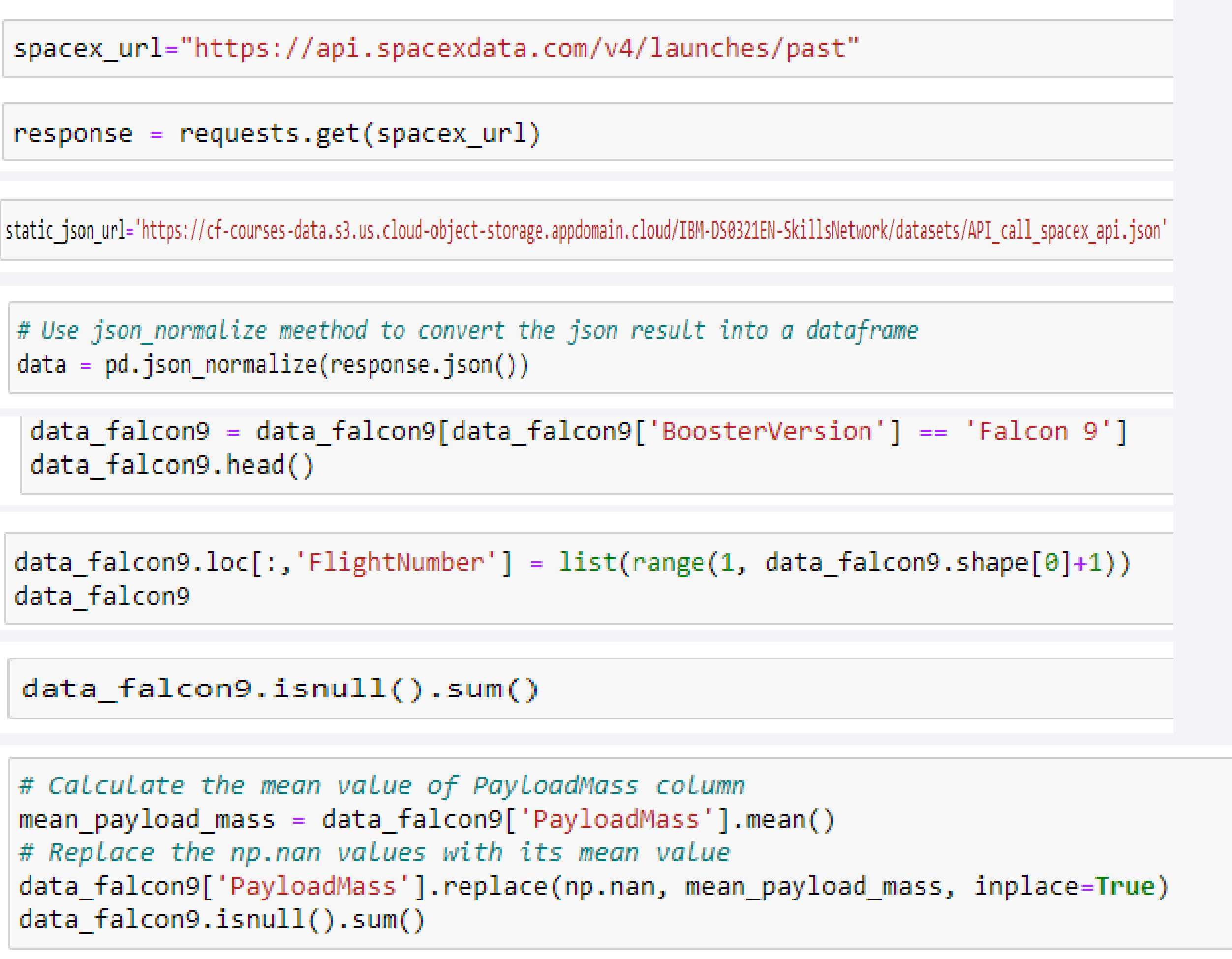
Data Collection – SpaceX API



Call the API to request the SpaceX launch data and parse the received information.

Extract data on Falcon 9 launches by filtering the dataset.

Handle missing values in the dataset.



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

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

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

```
data_falcon9 = data_falcon9[data_falcon9['BoosterVersion'] == 'Falcon 9']
data_falcon9.head()
```

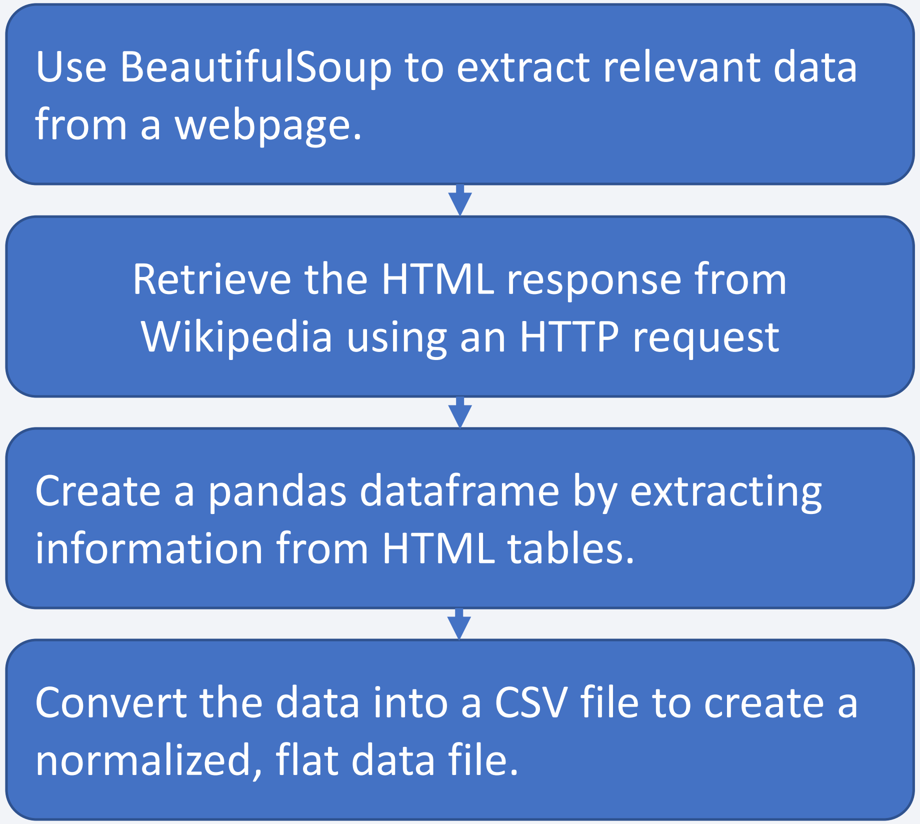
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
data_falcon9.isnull().sum()
```

```
# 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'].replace(np.nan, mean_payload_mass, inplace=True)
data_falcon9.isnull().sum()
```

Github Link

Data Collection - Scraping



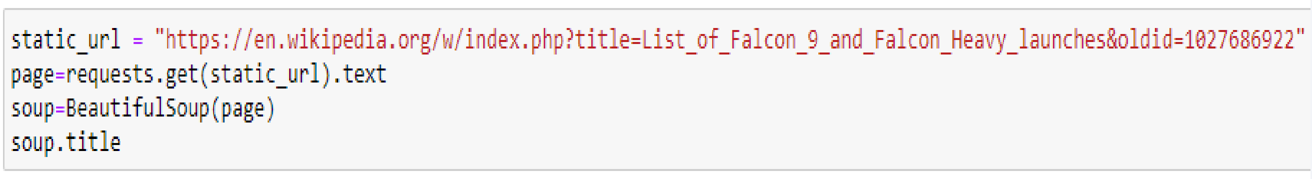
Use BeautifulSoup to extract relevant data from a webpage.

Retrieve the HTML response from Wikipedia using an HTTP request

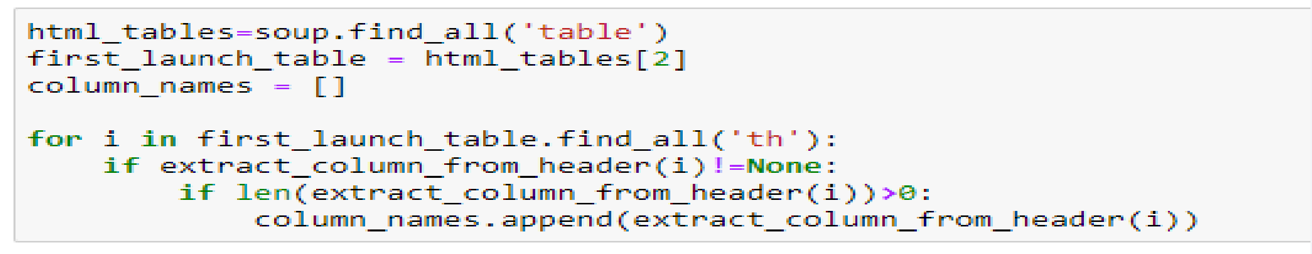
Create a pandas dataframe by extracting information from HTML tables.

Convert the data into a CSV file to create a normalized, flat data file.

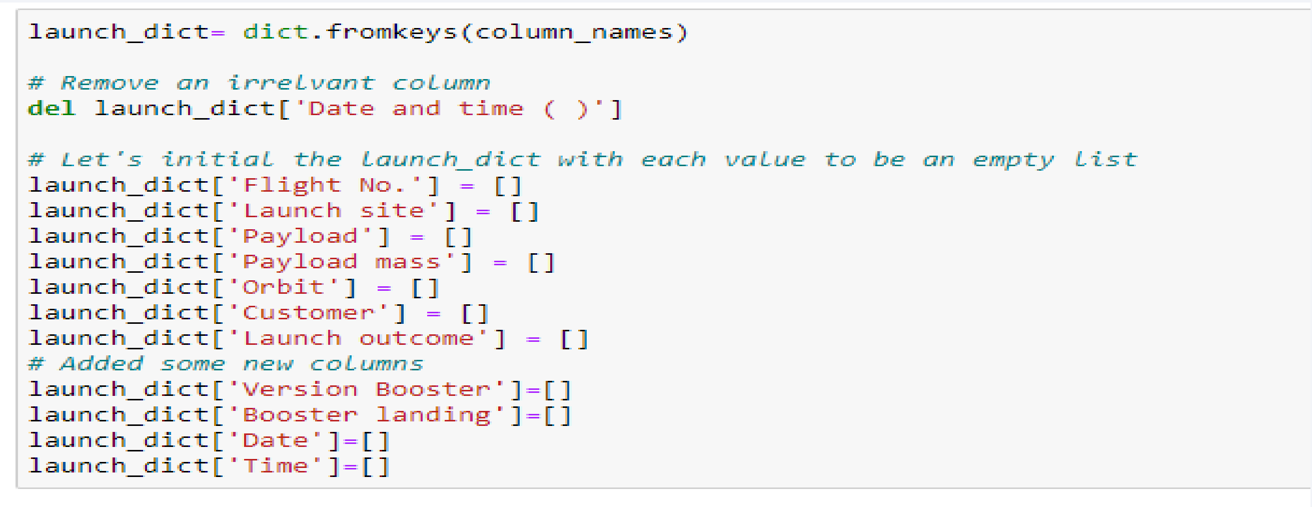
Github Link



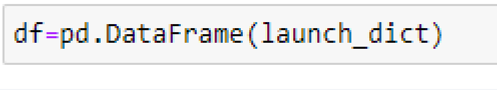
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
page=requests.get(static_url).text
soup=BeautifulSoup(page)
soup.title
```



```
html_tables=soup.find_all('table')
first_launch_table = html_tables[2]
column_names = []

for i in first_launch_table.find_all('th'):
    if extract_column_from_header(i)!=None:
        if len(extract_column_from_header(i))>0:
           column_names.append(extract_column_from_header(i))
```



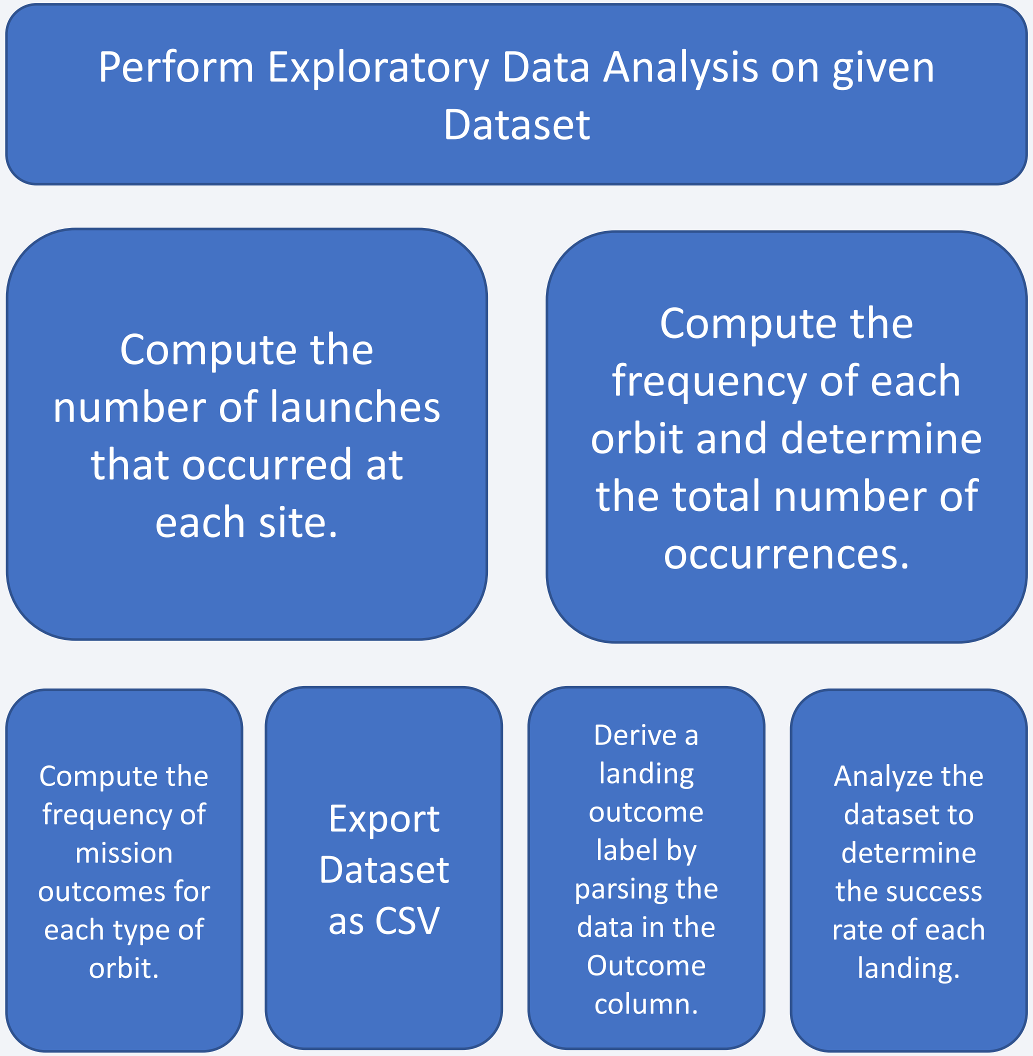
```
launch_dict= dict.fromkeys(column_names)

# 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']=[]
```



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

Data Wrangling

Github Link



Perform Exploratory Data Analysis on given Dataset

Compute the number of launches that occurred at each site.

Compute the frequency of each orbit and determine the total number of occurrences.

Compute the frequency of mission outcomes for each type of orbit.

Export Dataset as CSV

Derive a landing outcome label by parsing the data in the Outcome column.

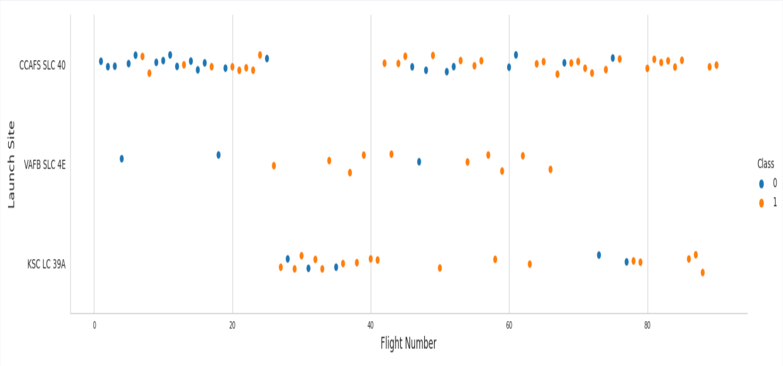
Analyze the dataset to determine the success rate of each landing.

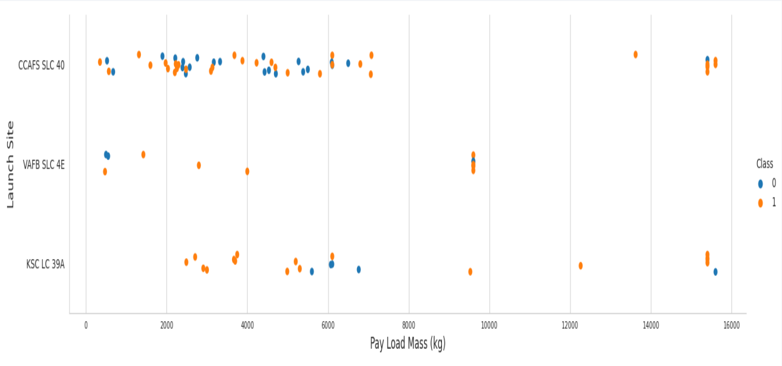
- After conducting exploratory data analysis, we identified the training labels and calculated various statistics.
- We analyzed the data and determined the training labels, as well as the number of launches at each site and the frequency of each orbit.
- Our exploratory data analysis involved calculating launch site counts and orbit occurrences, and determining the training labels based on these findings.
- Using exploratory data analysis, we determined the training labels and created a landing outcome label from the outcome column, exporting the resulting data to a CSV file.
- By analyzing the data, we identified the training labels and used the number of launches at each site and orbit frequency to create a landing outcome label. We then exported this information to a CSV file.

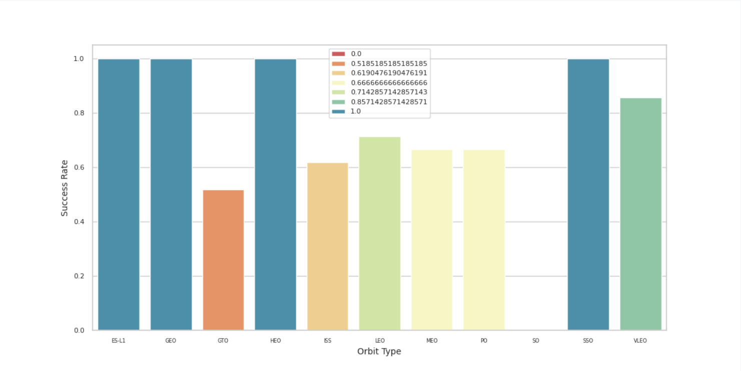
EDA with Data Visualization

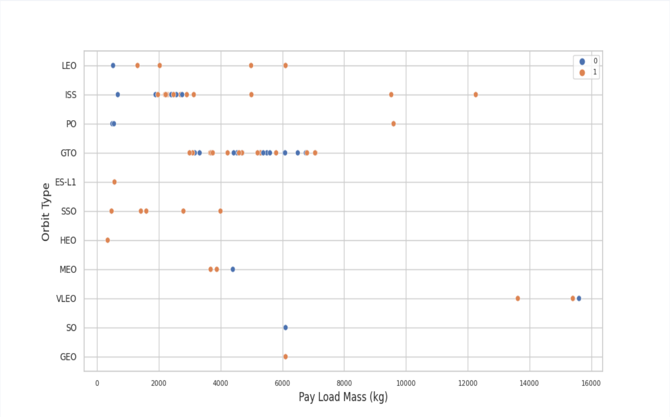
Github Link

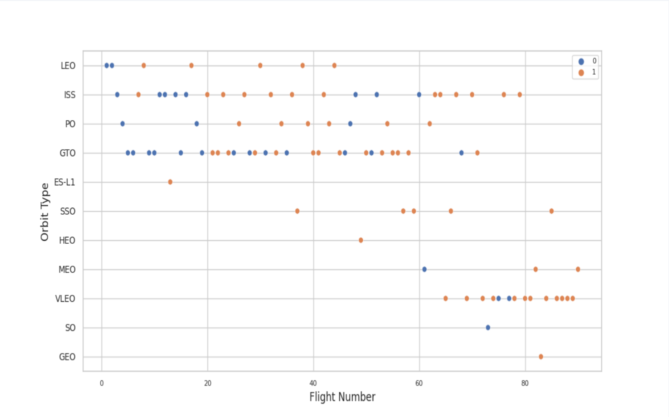
- Our data exploration involved generating visualizations to identify correlations among factors such as flight number and launch site, payload and launch site, success rate for each orbit type, flight number and orbit type, and annual launch success trend.

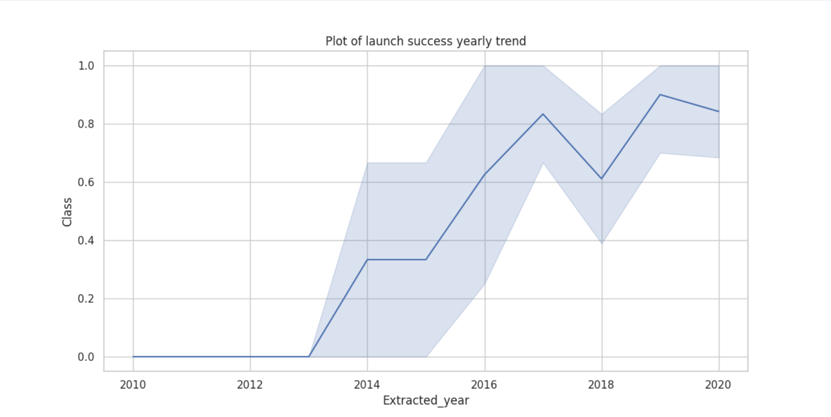












EDA with SQL

Github Link

- We imported the SpaceX dataset into a SQLite database directly from Jupyter notebook.
- We utilized SQL for exploratory data analysis to extract insights from the data. We formulated queries to retrieve various information, such as:
- The unique names of the launch sites involved in the space missions.
- The total payload mass carried by boosters launched under the NASA (CRS) program.
- The average payload mass carried by booster version F9 v1.1.
- The total number of successful and failed mission outcomes.
- The failed landing outcomes on drone ships, along with their corresponding booster versions and launch site names.

Build an Interactive Map with Folium

Github Link

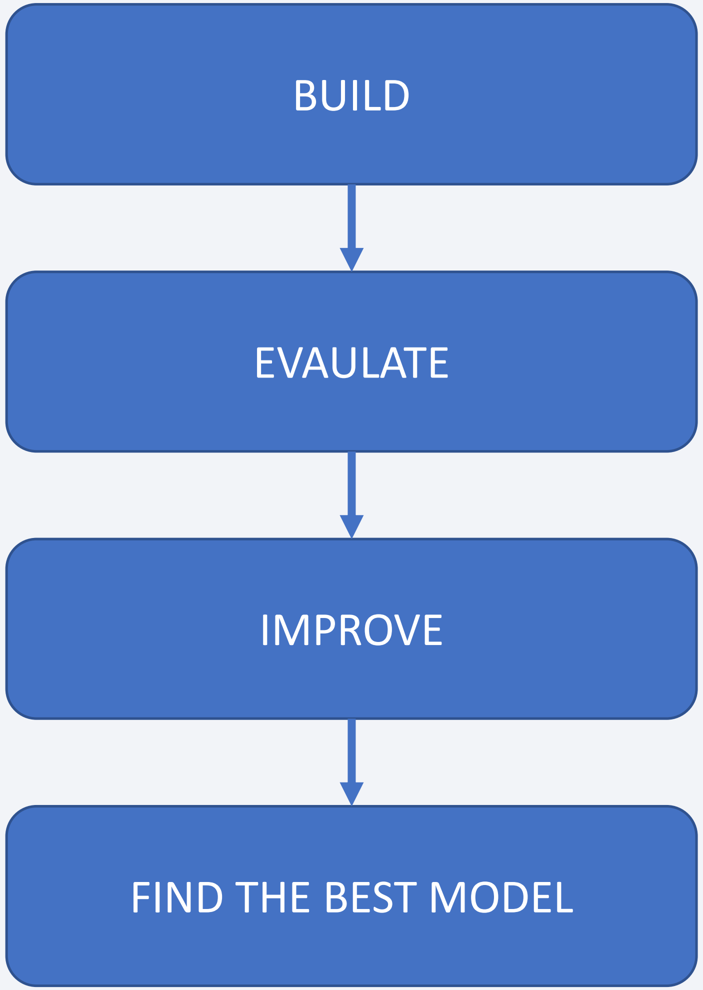
- We utilized Folium to add markers, circles, and lines on the map to indicate the success or failure of launches at each site. All launch sites were marked, and the map objects were added accordingly.
- We created a binary class by assigning launch outcomes (failure or success) to class 0 and 1, respectively. Specifically, class 0 represented failure, while class 1 represented success.
- Using marker clusters labeled with colors, we identified the launch sites that had a relatively high success rate.
- We computed the distances between launch sites and their surroundings to answer questions such as:
  - Are the launch sites located near railways, highways, or coastlines?
  - Is there a certain distance that launch sites need to keep from nearby cities?

Build a Dashboard with Plotly Dash

Github Link

- Using Plotly Dash, we developed a comprehensive and intuitive dashboard that allows users to interact with the data in numerous ways. This dashboard enables users to explore the data from various angles, including filtering, sorting, and selecting variables of interest. The user-friendly interface and interactive features of the dashboard provide a powerful tool for data analysis and exploration.
- In our Plotly Dash dashboard, we included a visually-appealing pie chart that offers a quick and simple way to compare the percentage breakdown of different categories. This chart displays the total number of launches for a specific launch site or all sites, as well as the relative proportions of different data classes. With this pie chart, users can quickly gain insights into the data and identify trends or patterns.
- To demonstrate the correlation between two independent variables, we utilized a scatter plot in our dashboard. Scatter plots are particularly useful for highlighting nonlinear patterns, and our scatter plot showcased the relationship between the result and payload mass (Kg) for different booster versions. This plot allowed users to visualize the data in a clear and meaningful way, making it easier to identify trends and patterns that may not be apparent from raw data.

Predictive Analysis (Classification)


BUILD

EVAULATE

IMPROVE

FIND THE BEST MODEL

- In order to determine the most effective classification method for our dataset, we experimented with four different approaches: logistic regression, support vector machine, decision tree, and k nearest neighbors. Each method has its own strengths and weaknesses, and we carefully evaluated their performance on our data.
- Logistic regression is a statistical method that is commonly used in machine learning to predict binary outcomes. It works by modeling the probability of a certain event occurring given a set of input variables. Support vector machines, on the other hand, are a type of supervised learning algorithm that can be used for both classification and regression tasks. They work by identifying a hyperplane that best separates the different classes in the data.
- Decision trees are a type of supervised learning algorithm that are used for classification and regression tasks. They work by recursively splitting the data into smaller subsets based on the most informative features. Finally, k nearest neighbors is a type of instance-based learning algorithm that is used for classification and regression tasks. It works by finding the k closest training examples in the feature space and using them to predict the class of a new instance.
- By comparing the performance of these four methods on our dataset, we were able to determine which one was the most effective for our specific use case. This information can help us make more informed decisions when using machine learning to analyze similar data in the future.

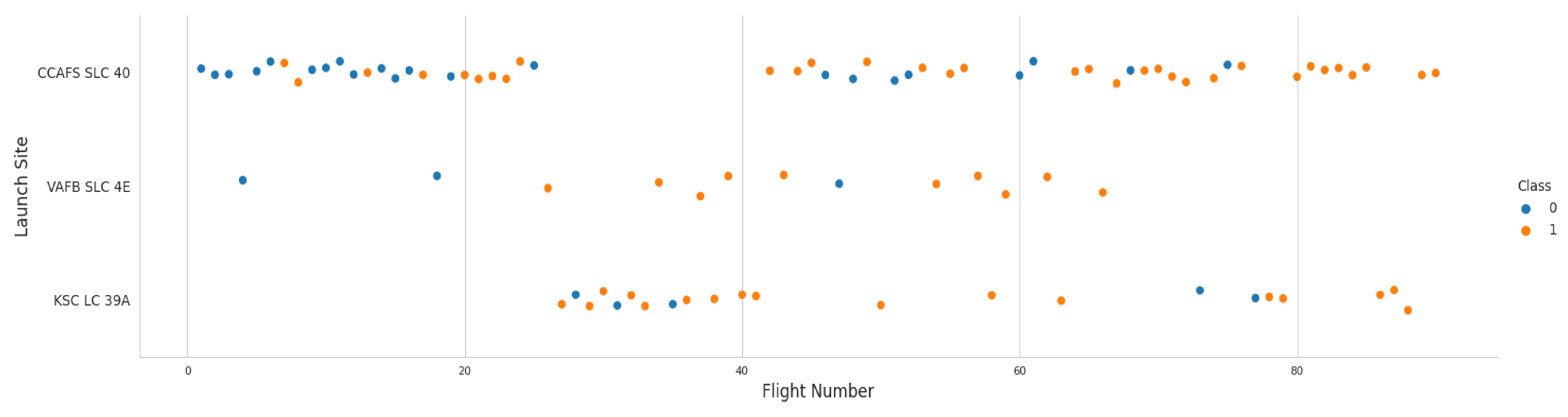
Results

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

Section 2

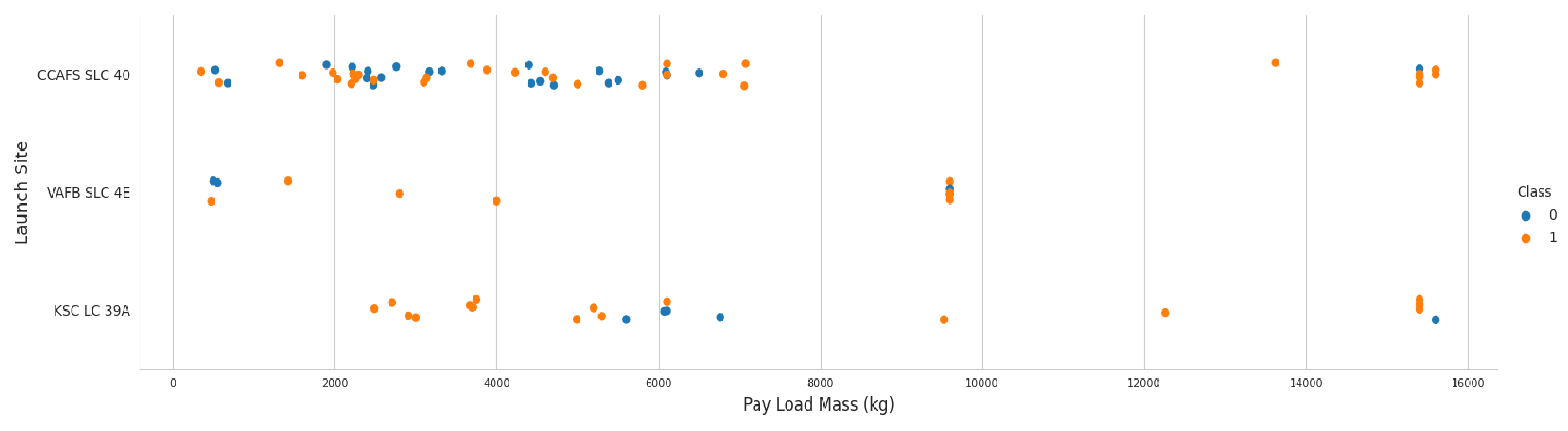
Insights drawn from EDA

Flight Number vs. Launch Site



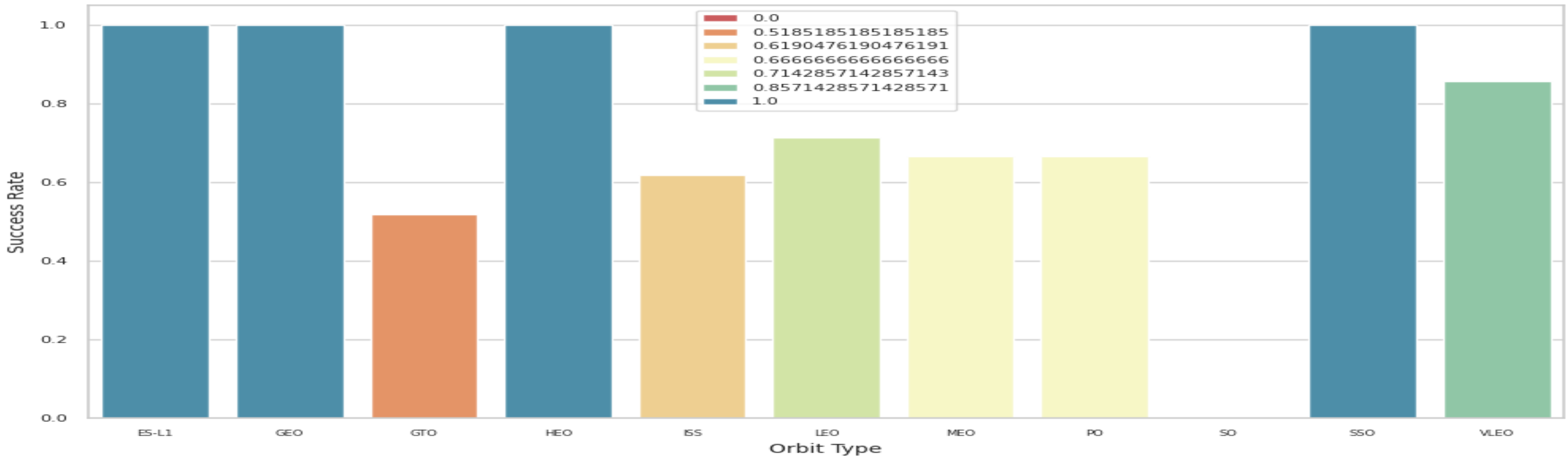
CCAF5 SLC 40 has had the most recent successful launches, making it the best launch location at the moment. VAFB SLC 4E is the second most successful launch site, followed by KSC LC 39A. Additionally, there is a possibility of an increase in success rate over time.

Payload vs. Launch Site



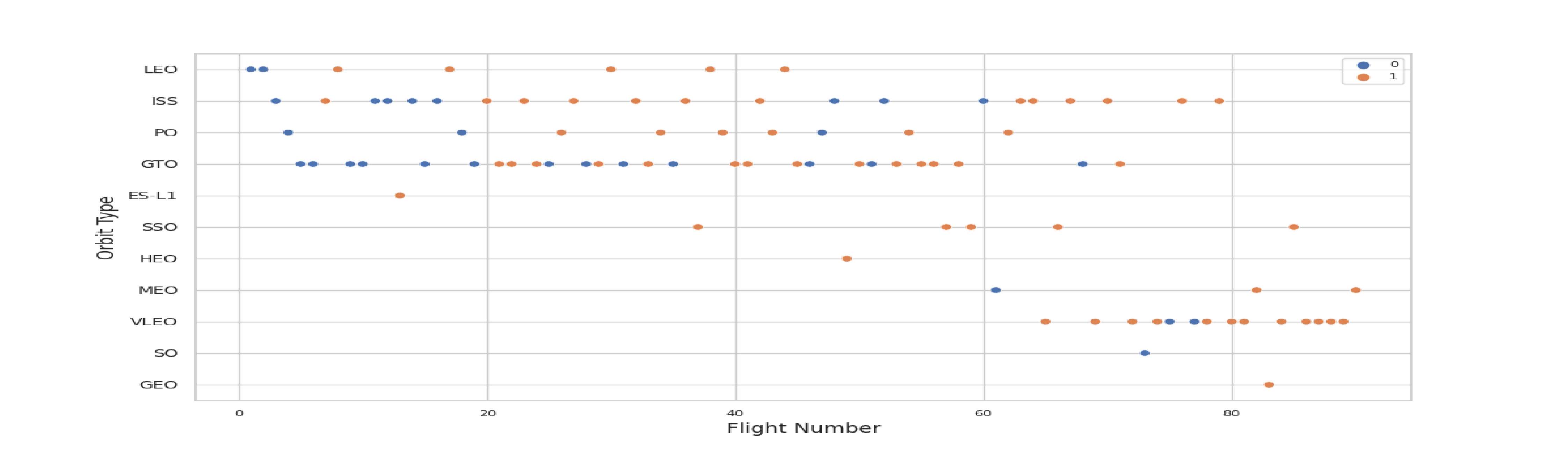
The success rate of heavy payloads exceeding 9,000kg is exceptional, with the possibility of payloads over 12,000kg only at the CCAFS SLC 40 and KSC LC 39A launch sites. As shown in the scatter plot, the probability of success rate significantly increases after the payload mass reaches 7,000 kg. However, there is no clear indication that the success rate at a launch site is directly correlated to the payload's mass.

Success Rate vs. Orbit Type



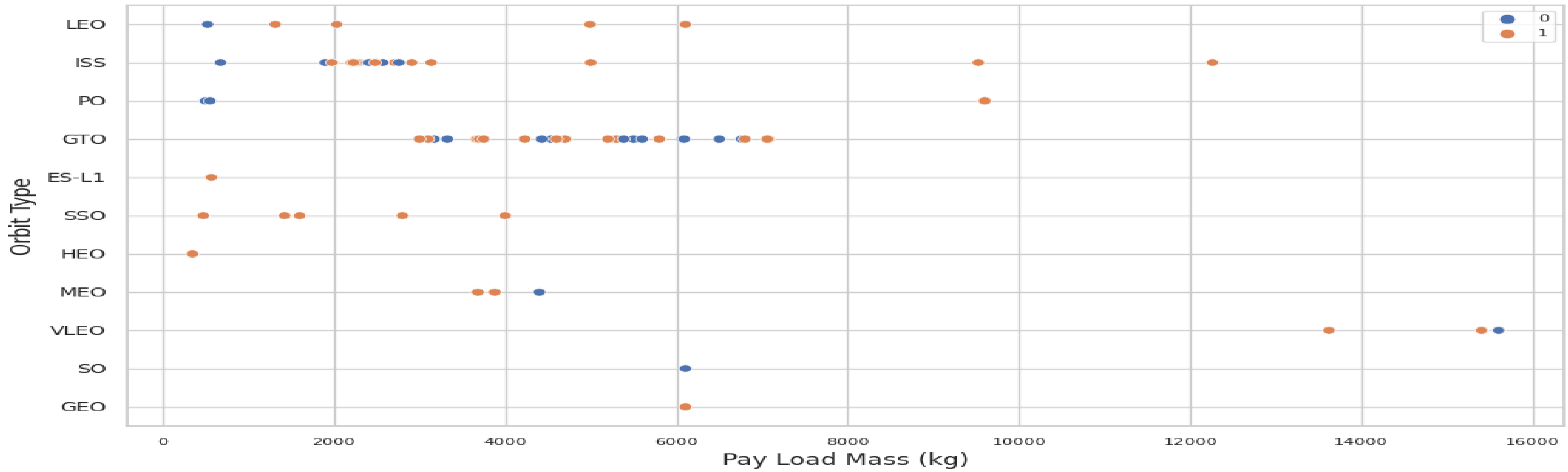
All launches into SSO, HEO, GEO, and ES-L1 orbits were successful, while launches into SO orbit had a 100% failure rate. A bar chart was used to analyze the impact of different orbits on landing outcomes. However, further analysis reveals that certain orbits, including GEO, SO, HEO, and ES-L1, only occurred once, indicating the need for additional data sets to identify patterns or trends before drawing conclusions.

Flight Number vs. Orbit Type



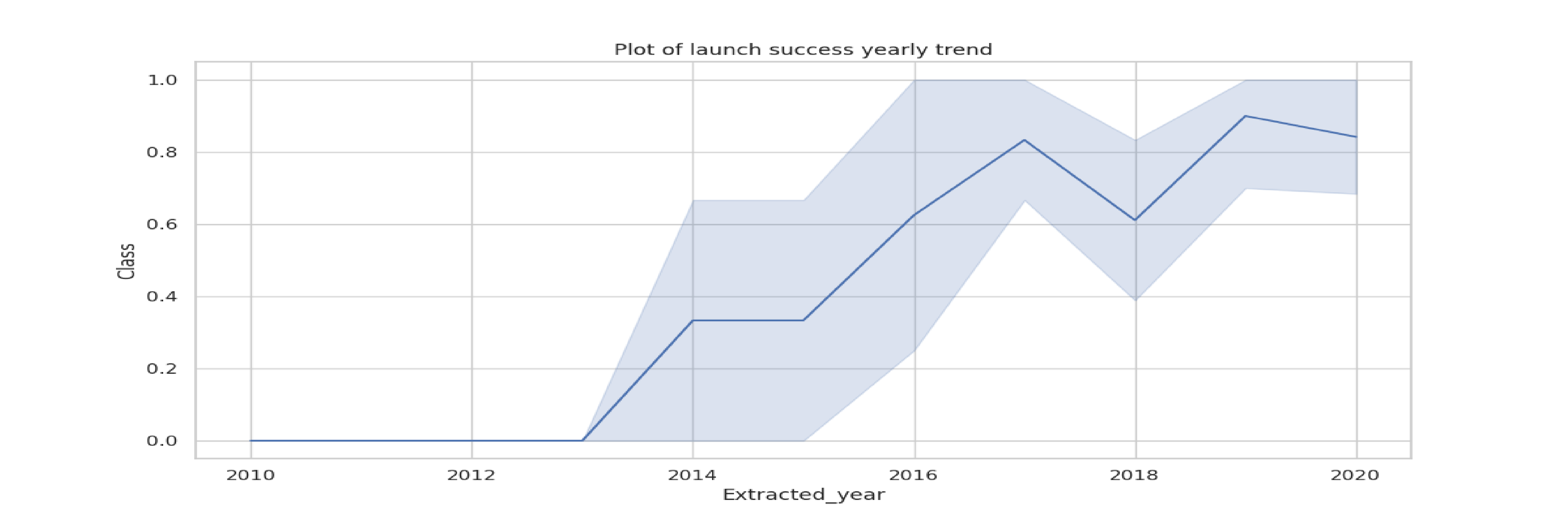
The scatter plot indicates a positive correlation between flight number and success rate for most orbits, except for GTO. Notably, the LEO orbit exhibits a particularly strong correlation. However, caution must be exercised when interpreting this result since orbits with only one occurrence lack sufficient data to draw firm conclusions. It appears that success rates have generally increased over time for all orbits. Additionally, the recent increase in frequency of the VLEO orbit presents a promising economic opportunity.

Payload vs. Orbit Type



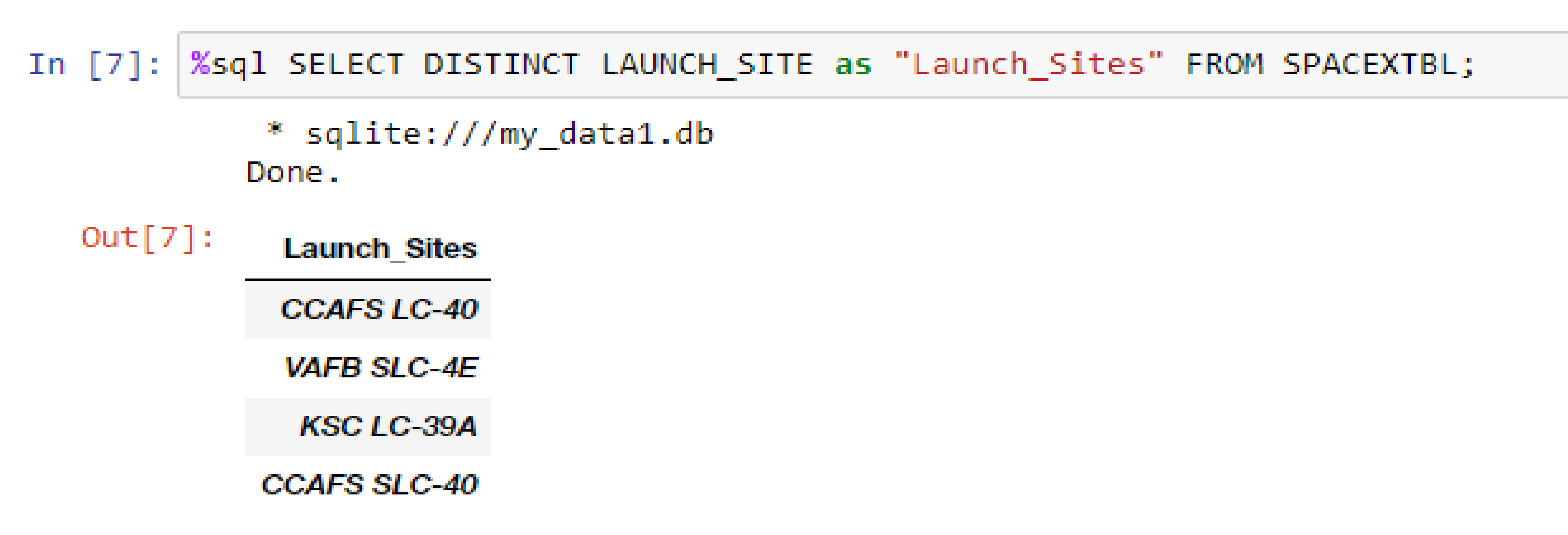
The relationship between payload mass and orbit success is not uniform across all orbits. Specifically, LEO, ISS, and PO orbits show a positive correlation between payload mass and success rate, whereas MEO and VLEO orbits exhibit a negative correlation. The GTO orbit, on the other hand, does not show any clear correlation. However, caution should be exercised when drawing conclusions about SO, GEO, and HEO orbits, as additional data is needed to identify potential trends or patterns. It is worth noting that while heavier payloads may have a negative impact on GTO orbits, they may have a positive impact on LEO and ISS orbits.

Launch Success Yearly Trend



The line chart shows a continuous upward trend in the success rate from 2013 to 2020. If this trend persists in the future, the success rate could potentially reach 100%. The initial three years of the period appear to be a time of technological development and adjustment, as evidenced by the slower increase in success rate during that time.

All Launch Site Names



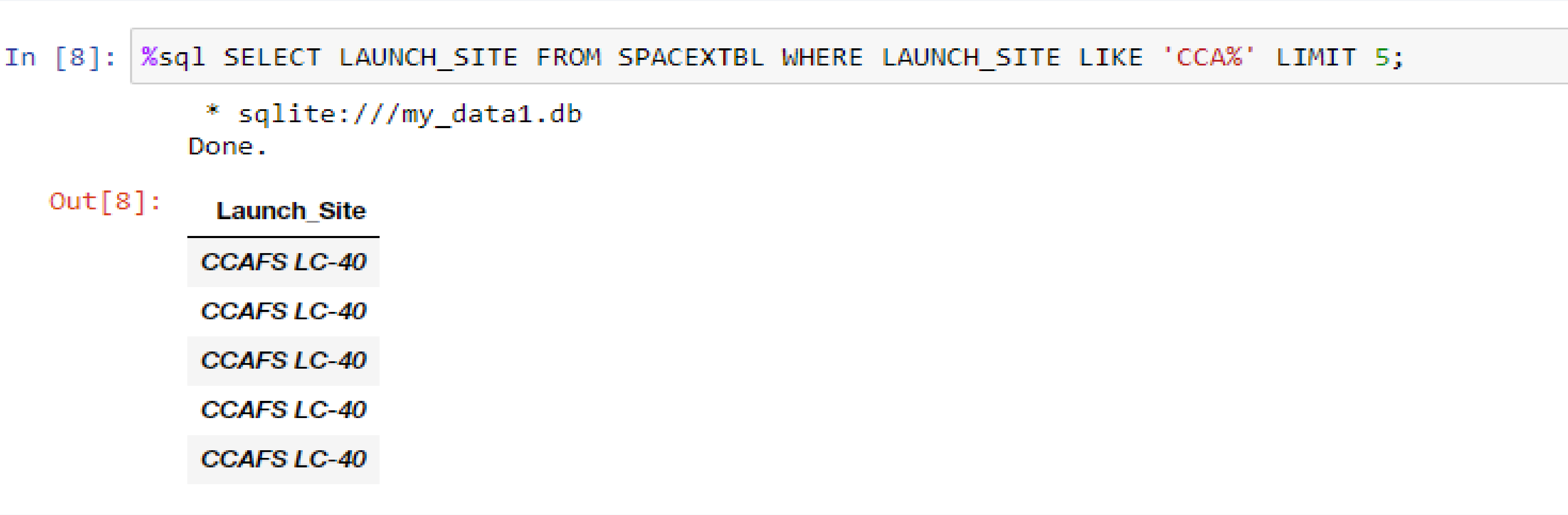
```
In [7]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;

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

Out[7]:

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

We ran a SQL query on the SPACEXTBL dataset with the keyword DISTINCT to retrieve the unique names of the launch sites. This allowed us to obtain a list of all the different launch sites used by SpaceX for their missions. By using the DISTINCT keyword, we eliminated any duplicate values in the "LAUNCH_SITE" column and obtained only the unique names of the launch sites.

Launch Site Names Begin with 'CCA'



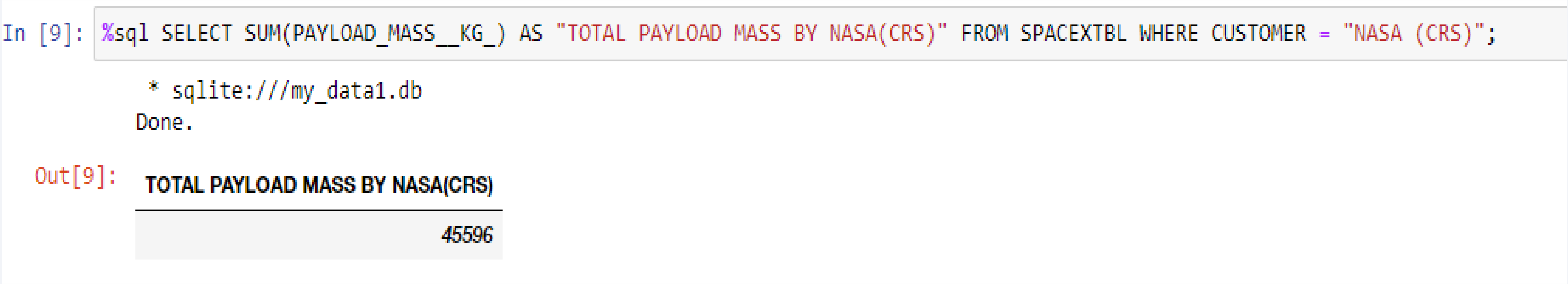
```
In [8]: %sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;

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

Out[8]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

The LIMIT 5 clause specifies that just the first five records from SPACEXTBL will be returned, and the LIKE keyword serves as a wild card by matching any string, so the string "CCA%" implies that the "LAUNCH_SITE" name must begin with CCA. This query allowed us to retrieve the first five unique launch sites from the SPACEXTBL dataset whose names start with "CCA." By using the LIKE keyword, we were able to match any string that started with "CCA," which included launch sites such as "CCAFS SLC 40" and "CCAFSSLC 40 / KRSC LC 39A."

Total Payload Mass



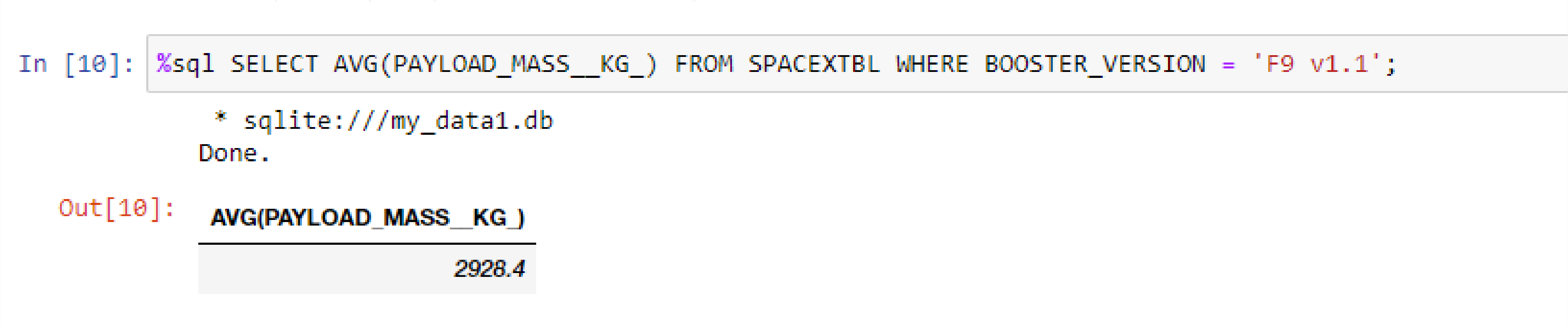
```
In [9]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS "TOTAL PAYLOAD MASS BY NASA(CRS)" FROM SPACEXTBL WHERE CUSTOMER = "NASA (CRS)";

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

Out[9]:

TOTAL PAYLOAD MASS BY NASA(CRS)
45596

The SUM function is used to aggregate the values in the column labeled PAYLOAD MASS KG. With the WHERE clause, we can restrict our data collection and obtain exact results for the Customer column with a NASA (CRS) value. This SQL query allowed us to obtain the sum of all the values in the "PAYLOAD MASS KG" column for only those rows where the "CUSTOMER" column had a value of "NASA (CRS)." By using the WHERE clause, we were able to restrict our data collection to only those rows that met our criteria, allowing us to obtain precise results.

Average Payload Mass by F9 v1.1



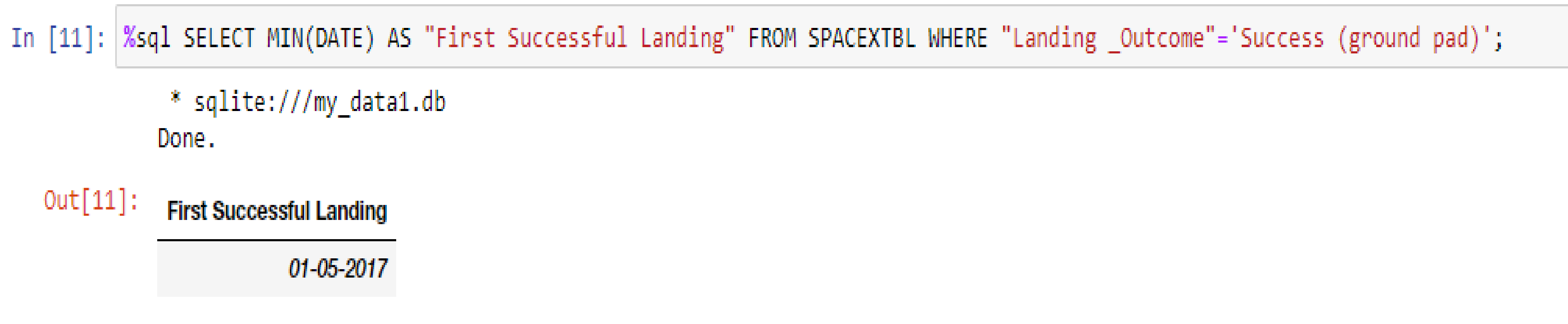
```
In [10]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';

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

Out[10]:

AVG(PAYLOAD_MASS__KG_)
2928.4

The average value in the field PAYLOAD MASS KG is calculated by the function AVG. With the WHERE clause, we can restrict the data set to just include rows about Booster version F9 v1.1 for analysis. This SQL query allowed us to calculate the average value of the "PAYLOAD MASS KG" column for only those rows where the "BOOSTER VERSION" column had a value of "F9 v1.1." By using the AVG function, we were able to calculate the average value of the "PAYLOAD MASS KG" column, and by using the WHERE clause, we were able to restrict the data set to only those rows that met our criteria for analysis.

First Successful Ground Landing Date



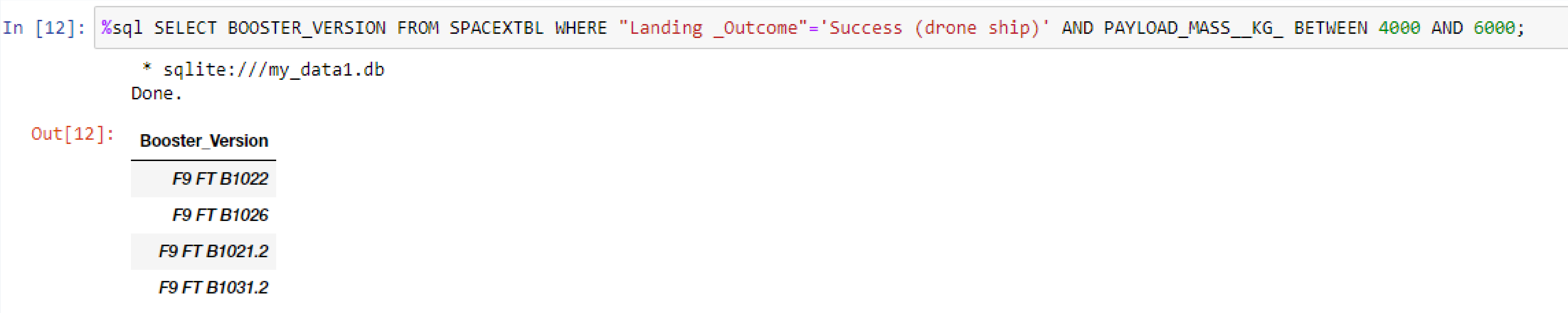
```
In [11]: %sql SELECT MIN(DATE) AS "First Successful Landing" FROM SPACEXTBL WHERE "Landing _Outcome"='Success (ground pad)';

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

Out[11]:

First Successful Landing
01-05-2017

To obtain the earliest date in the Date column, we applied the MIN() function. We then filtered the data set to only include successful landings on ground pads by utilizing the WHERE clause with the value "Success (ground pad)" in the Landing_Outcome column.

Successful Drone Ship Landing with Payload between 4000 and 6000

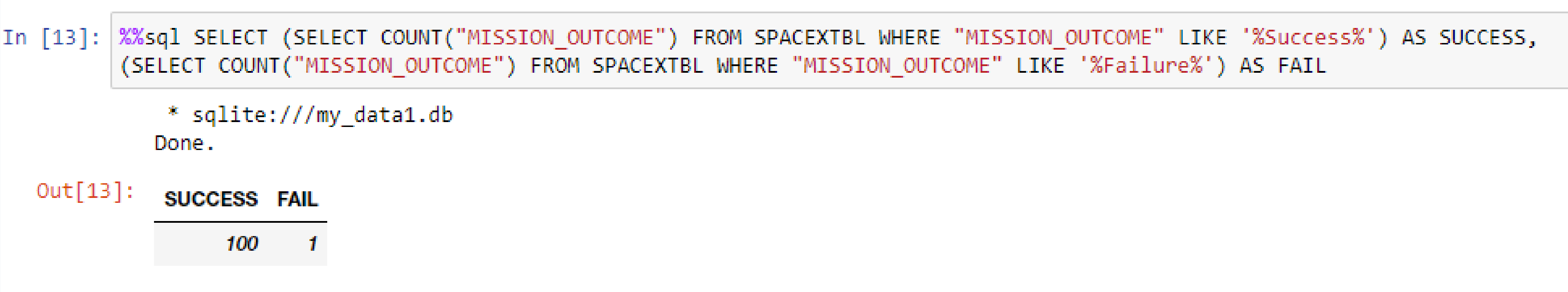


```
In [12]: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE "Landing _Outcome"='Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;

          * sqlite:///my_data1.db
         Done.

Out[12]:
```

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

In order to extract specific data, we used SQL to select the BOOSTER_VERSION column from the table. To narrow down the results, we applied a WHERE clause with conditions for Landing_Outcome to be "Success" (drone ship), and the Payload MASS KG to be either 4000 or 6000.

Total Number of Successful and Failure Mission Outcomes



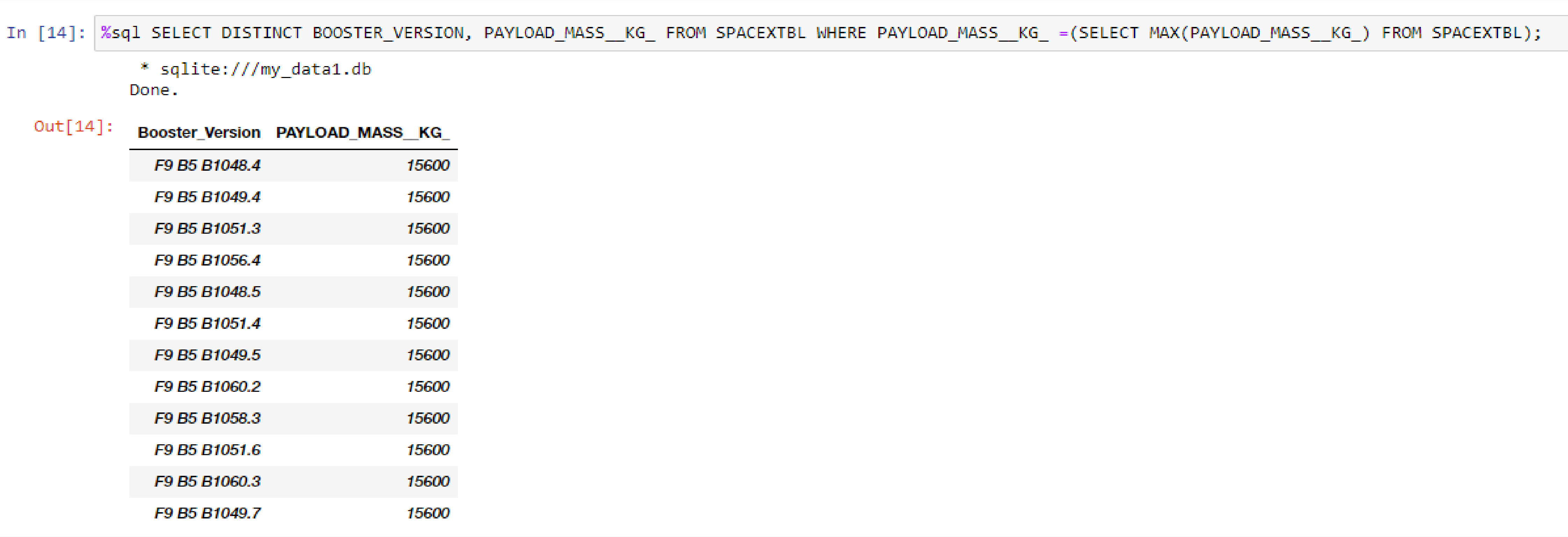
```
In [13]: %%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Success%') AS SUCCESS,
         (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS FAIL

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

Out[13]:

SUCCESS	FAIL
100	1

To narrow the data, we employed nested queries with the WHERE clause and used a LIKE '%' statement to filter "Success" and "Failure" statements in the MISSION OUTCOME column. Additionally, we utilized subqueries to aggregate the results and created a summary by counting records for each subquery.

Boosters Carried Maximum Payload



```
In [14]: %sql SELECT DISTINCT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

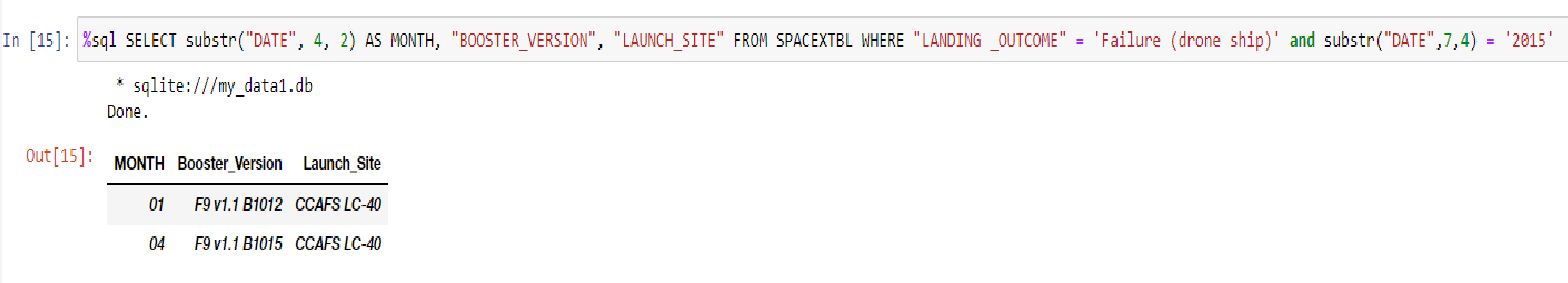
```
 * sqlite:///my_data1.db
Done.
```

Out[14]:

Booster_Version	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

To identify the booster with the highest payload mass, we employed a subquery with a WHERE clause and the MAX() function. We first used the DISTINCT keyword in the main query to retrieve only unique values from the BOOSTER VERSION column. Then, in the subquery, we obtained the maximum payloads and matched them with their respective booster versions using the WHERE clause.

2015 Launch Records



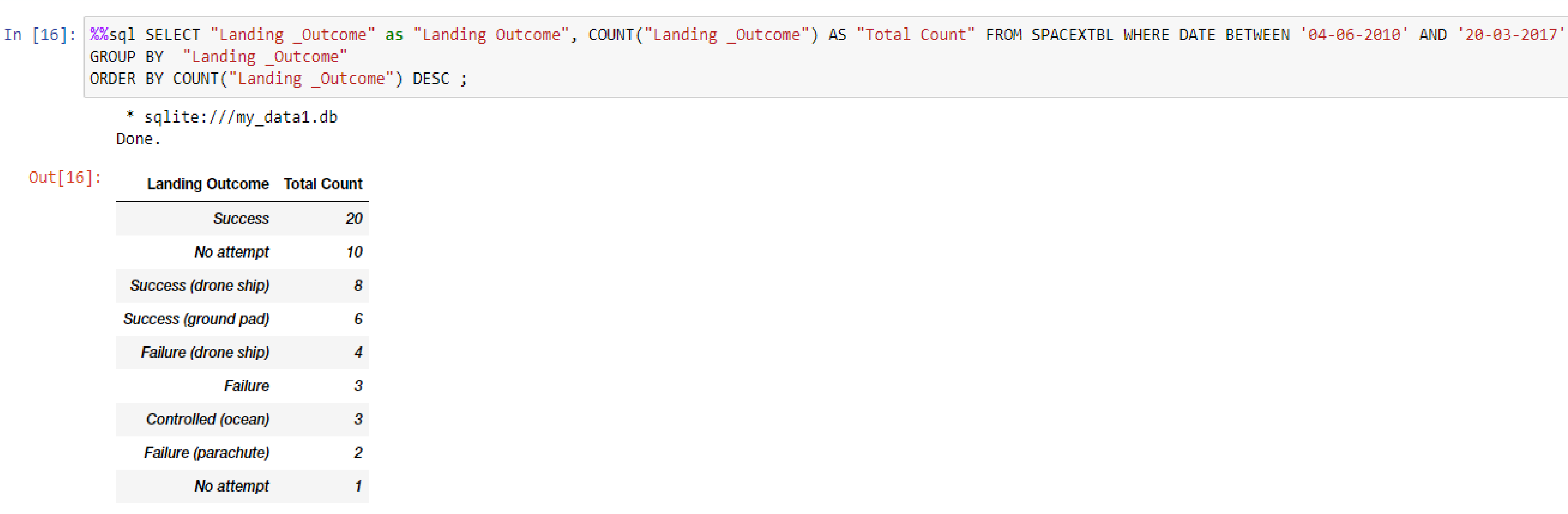
```
In [15]: %sql SELECT substr("DATE", 4, 2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE" FROM SPACEXTBL WHERE "LANDING _OUTCOME" = 'Failure (drone ship)' and substr("DATE",7,4) = '2015'

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

Out[15]:

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

We extracted the BOOSTER_VERSION and LAUNCH_SITE columns and modified the DATE column using substr(Date, 4, 2) to only retrieve months. We then filtered the LANDING OUTCOME column to retrieve only "Failure (drone ship)" values using the WHERE clause. Finally, we used the AND statement in the WHERE clause to select only the year 2015 by adding the condition substr(Date,7,4)='2015'.

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



In [16]:

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

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

Out[16]:

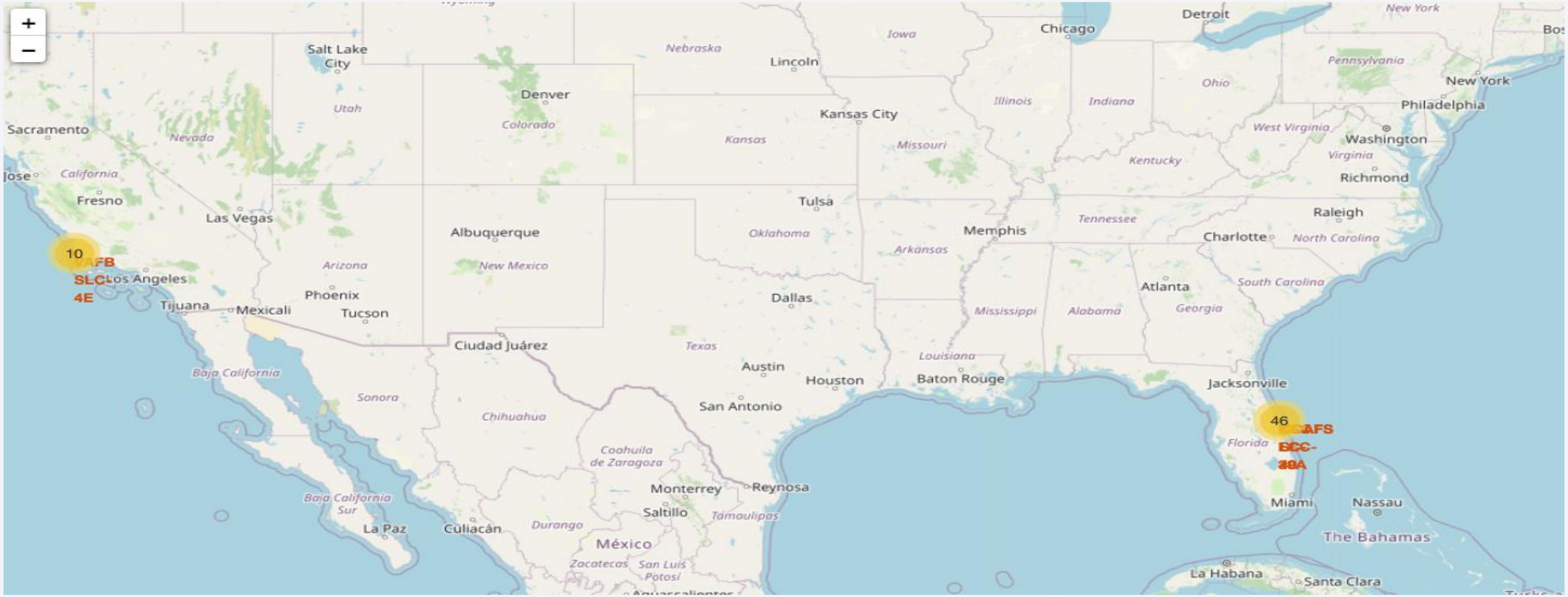
Landing Outcome	Total Count
Success	*20*
No attempt	*10*
Success (drone ship)	*8*
Success (ground pad)	*6*
Failure (drone ship)	*4*
Failure	*3*
Controlled (ocean)	*3*
Failure (parachute)	*2*
No attempt	*1*

We filtered the data by applying the WHERE clause to choose landing outcomes between the dates 2010-06-04 and 2010-03-20. We then selected Landing_Outcomes and counted the number of occurrences using the COUNT function. The resulting data was grouped using the GROUP BY clause and ordered in descending order using the ORDER BY clause.

Section 3

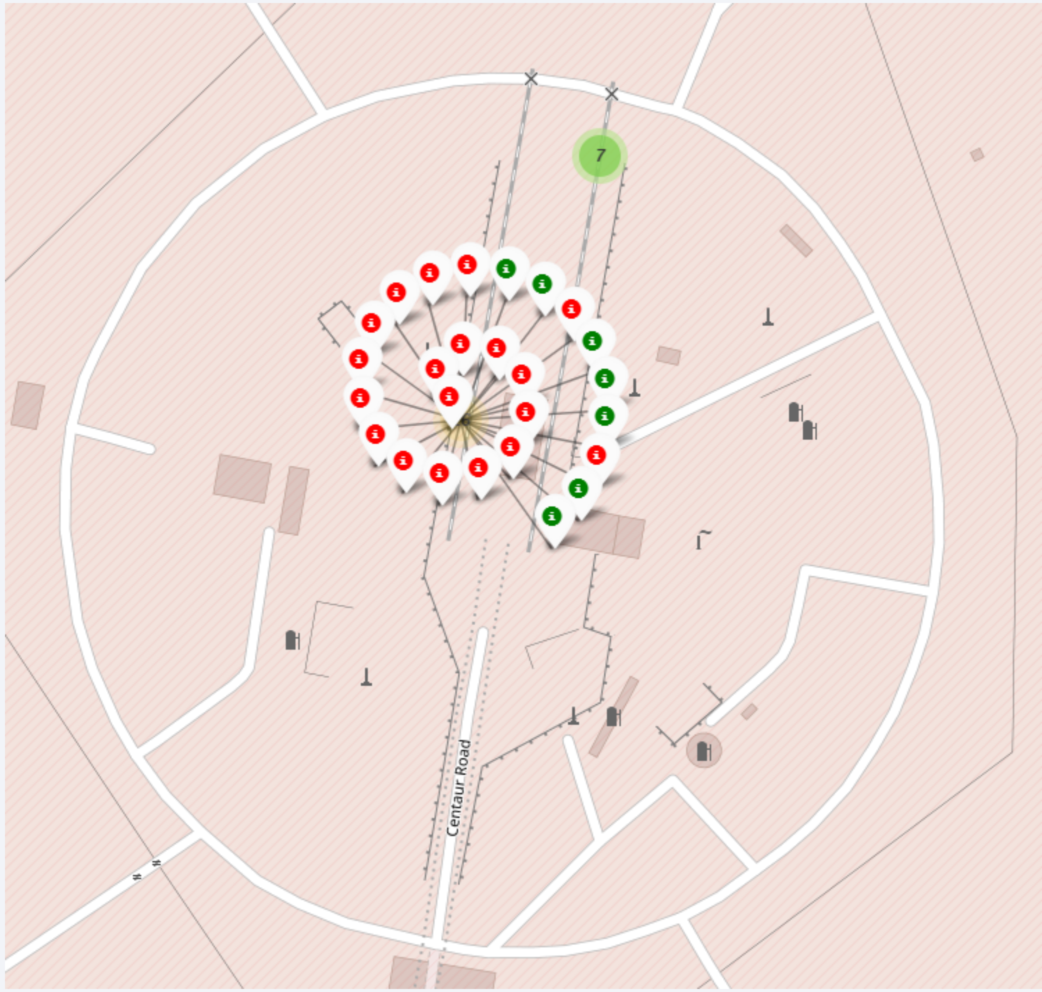
Launch Sites Proximities Analysis

Global map markers for all launch sites.

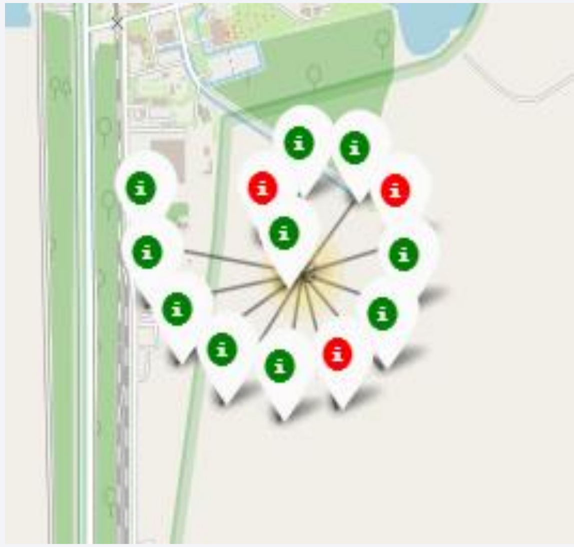


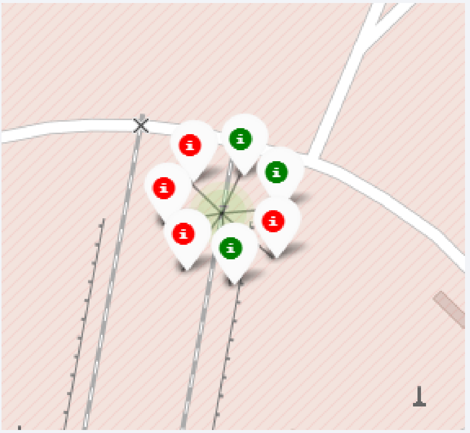
From the map, it can be observed that all the launch sites operated by SpaceX are situated within the boundaries of the United States.

Global map markers for all launch status.



Space Launch Complex 4
Dix Road

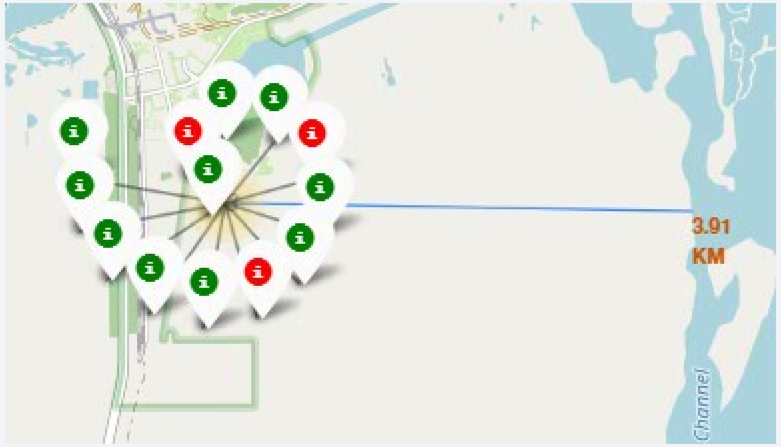




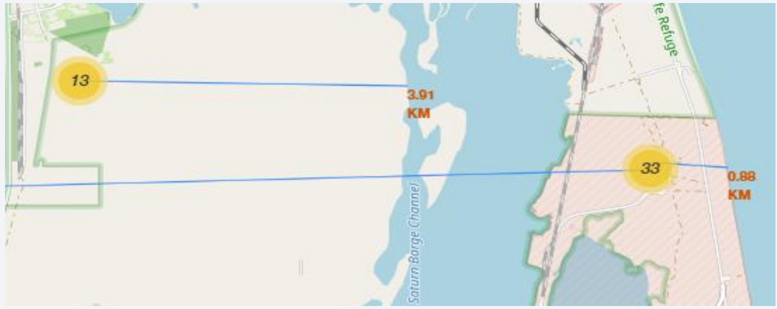
Green Marker
Succesfull

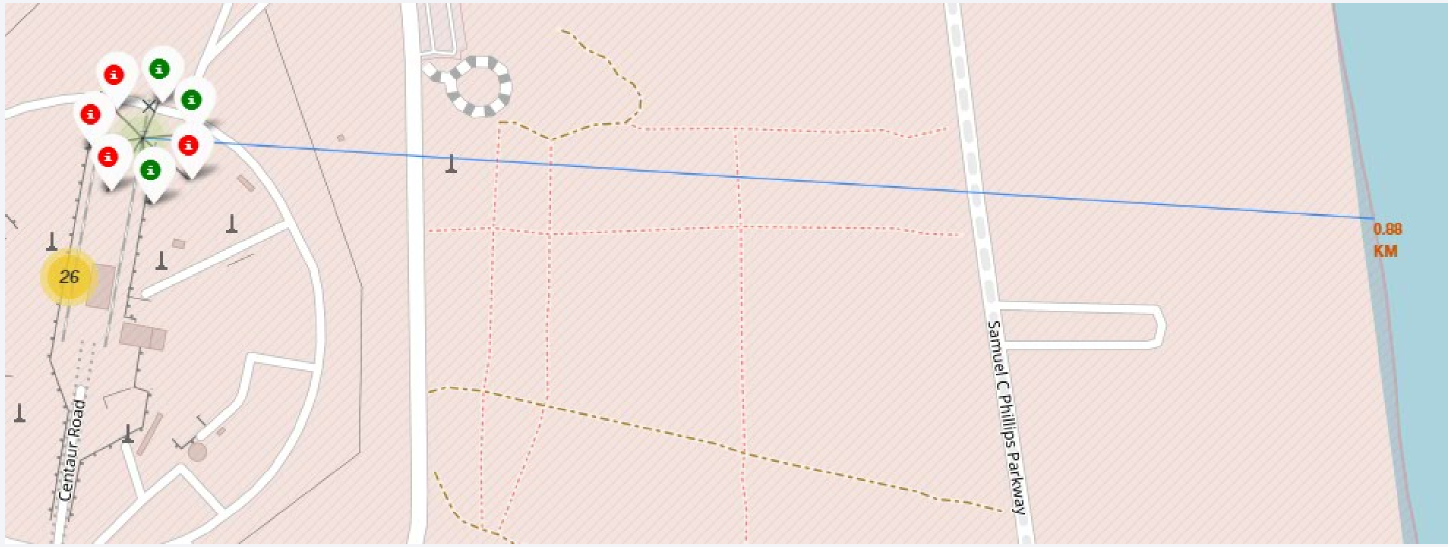
Red Marker
Failure

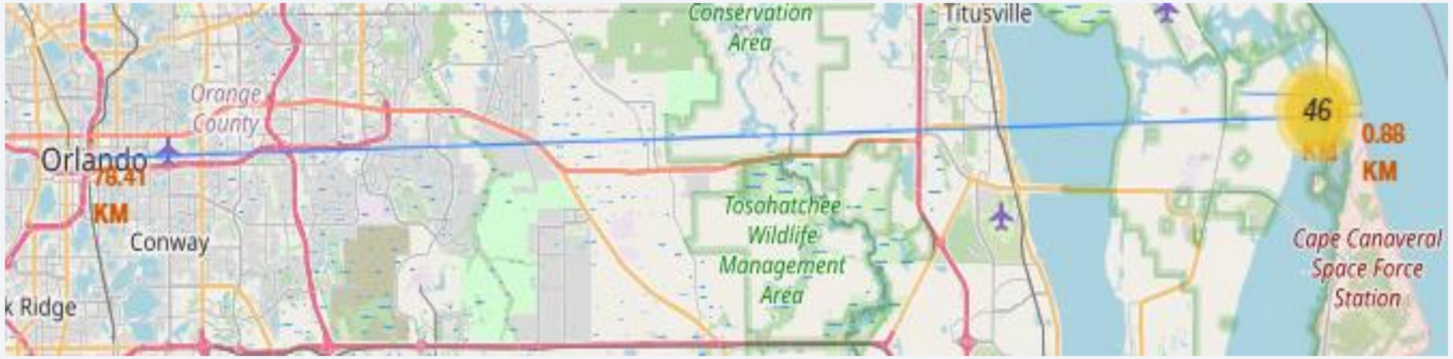
Proximity of launch sites to landmarks



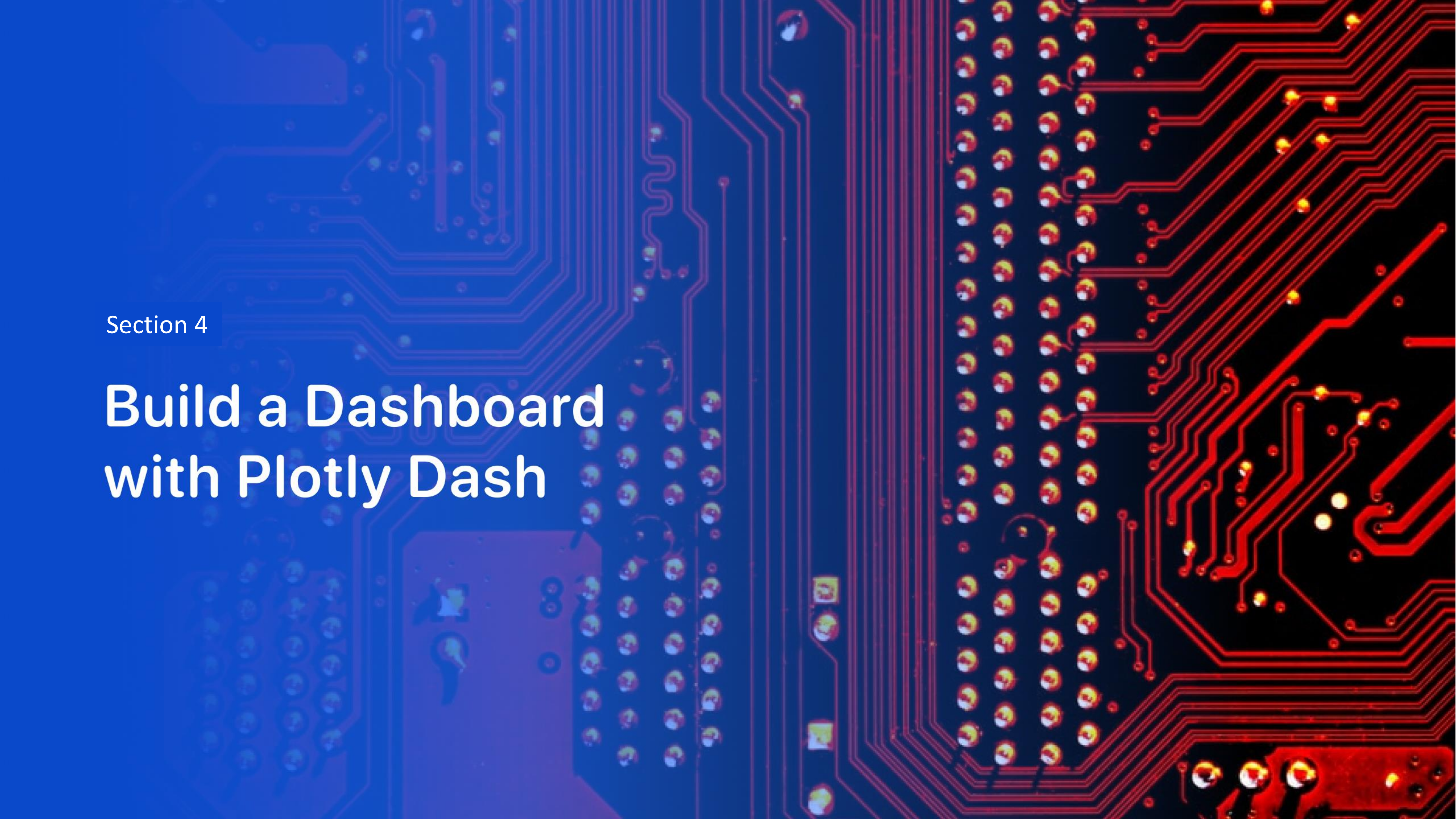
10
Lompoc
Sespe Wilderness
Lancaster
Palmdale
Santa Barbara
Fillmore
Santa Clarita
Ventura
Oxnard
Simi Valley
Thousand Oaks
Glendale
Channel Islands National
Malibu
Santa Monica
Los Angeles







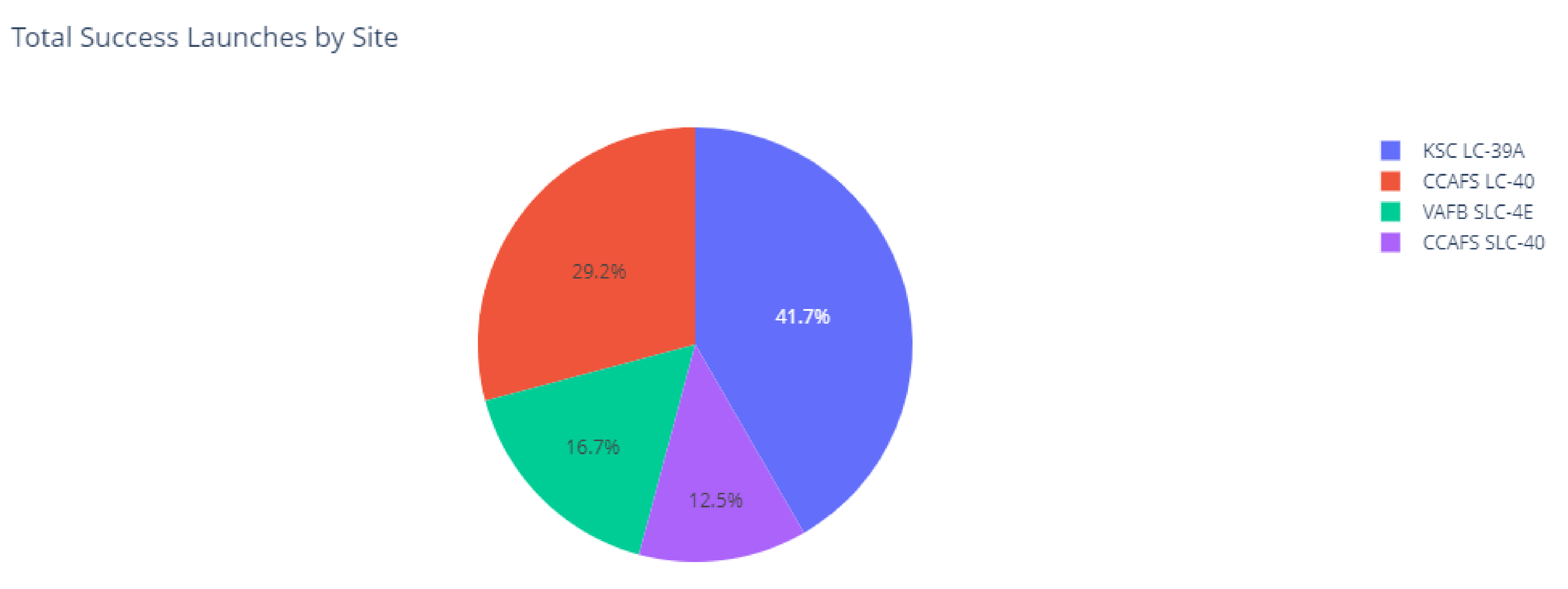
Using the Haversine formula, we calculated the distances between launch sites and specific landmarks to analyze their location-based trends. The distances between CCAFS LC-40 and the city center, as well as KSC LC-39A and the coast, are displayed in the image below.



Section 4

Build a Dashboard with Plotly Dash

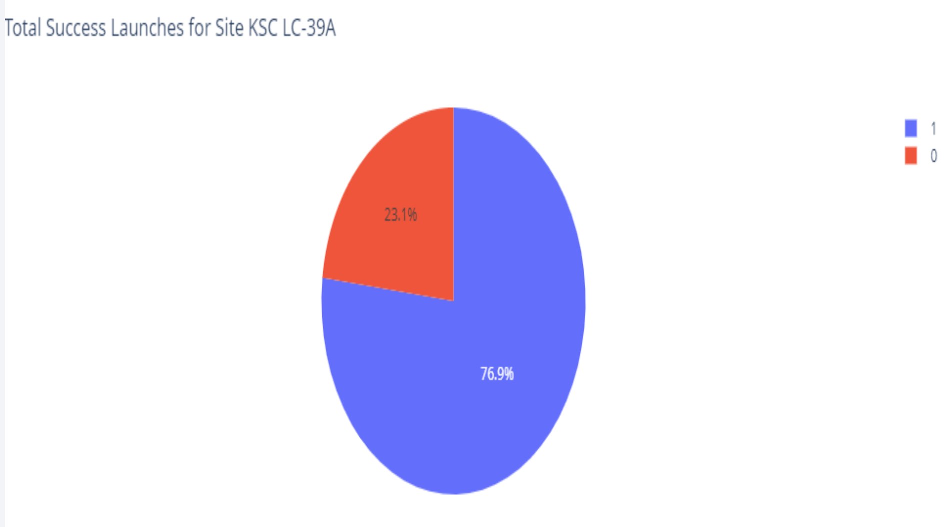
Top performing launch sites ranked by success rate.

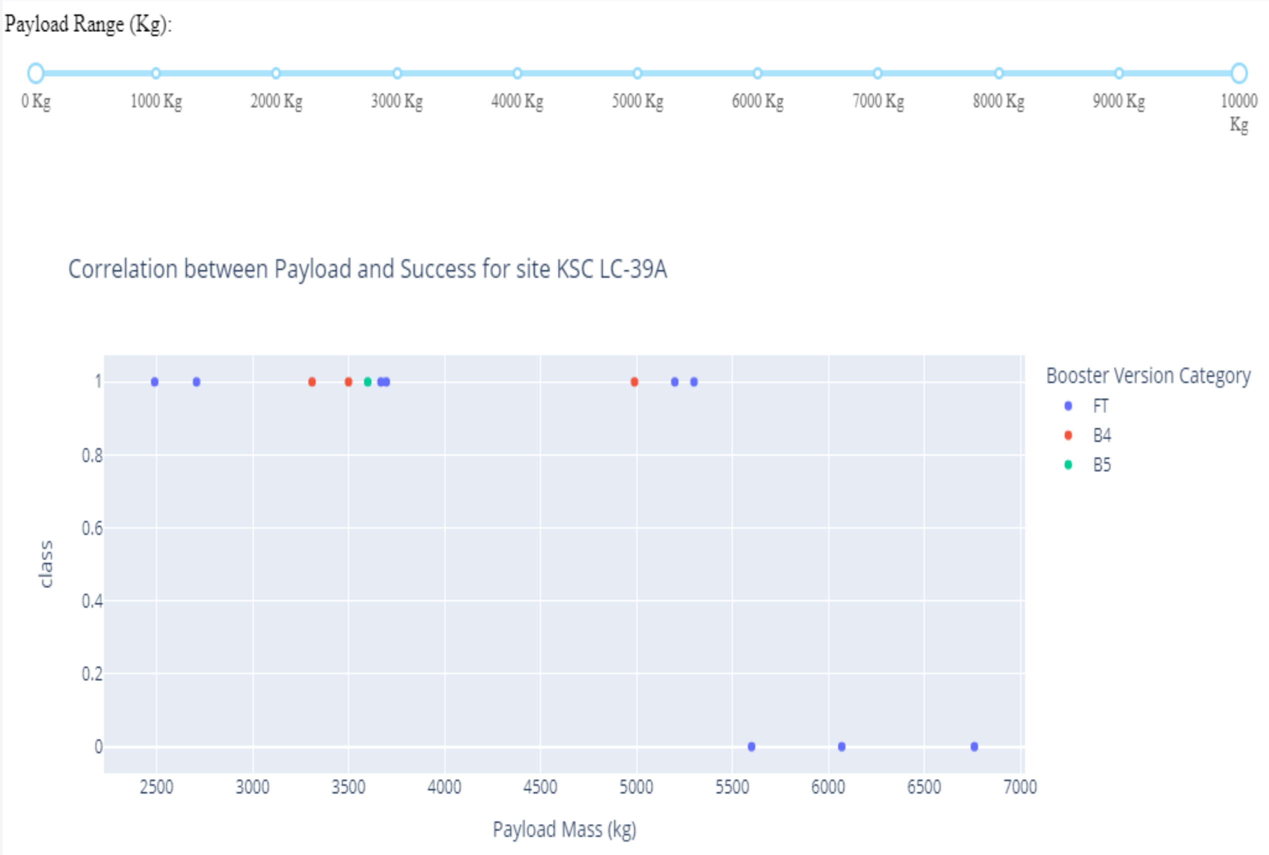


KSC LC-39A was found to have the highest rate of successful launches with 41.7%, followed by CCAFS LC-40 with a rate of 29.2%. The third-highest success rate location was VAFB SLC-4E with 16.7%, and CCAFS SLC-40 was the fourth launch location with a success rate of 12.5%.

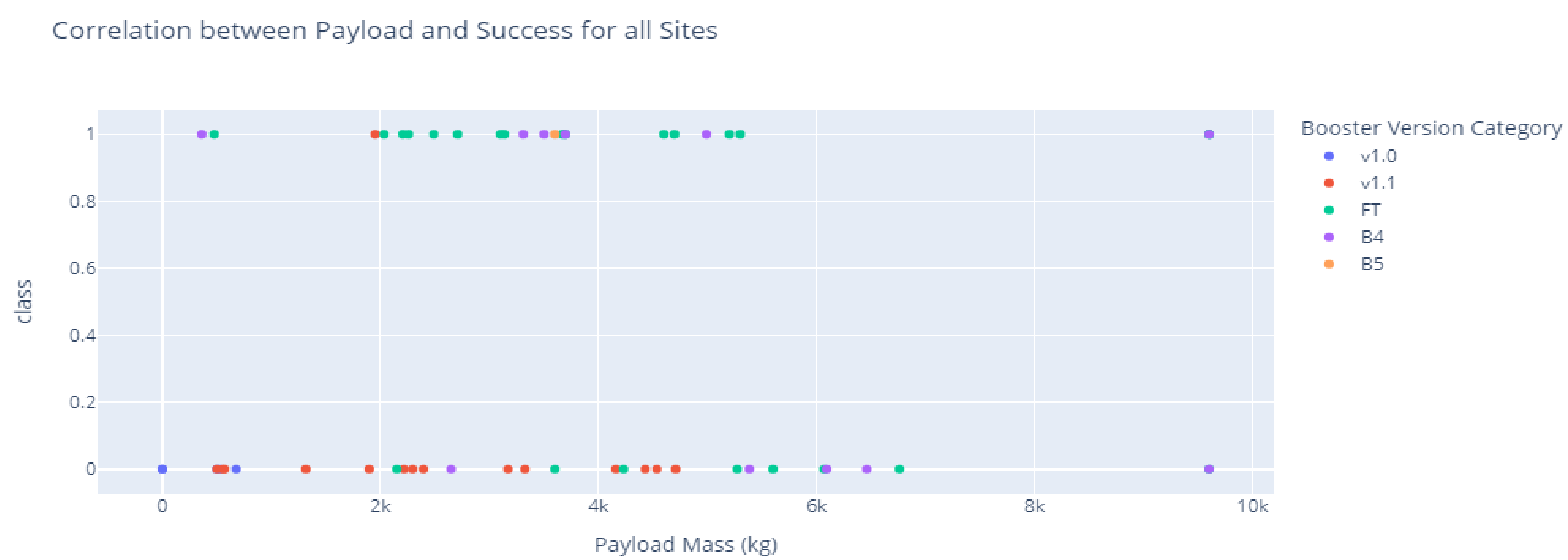
Launch site success ratio pie chart: Highest

The success rate for KSC LC-39A was 76.9% with a failure rate of 23.1%.





Scatter Plot Generated to Visualize the Relationship Between Launch Outcome and Payload

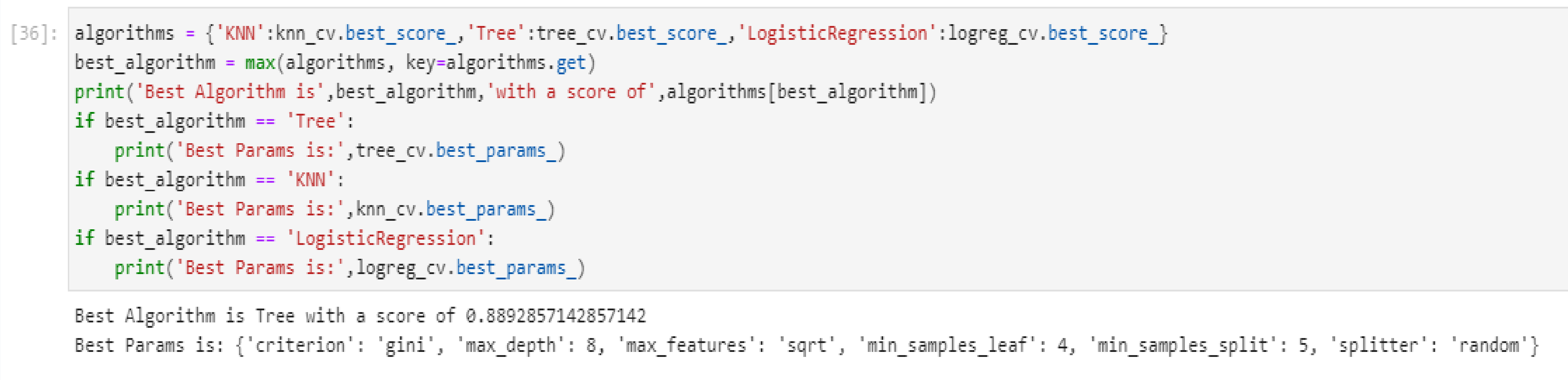


According to the scatter plot, launches with payloads less than 2,000 kg had a higher failure rate compared to successful launches. The success rate increased for payloads between 2,000 and 4,000 kg, and this range had the most successful launches. The failure rate became greater than the success rate for payloads between 4,000 and 6,000 kg. Only one launch with a payload greater than 6,000 kg was successful, while all others failed.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

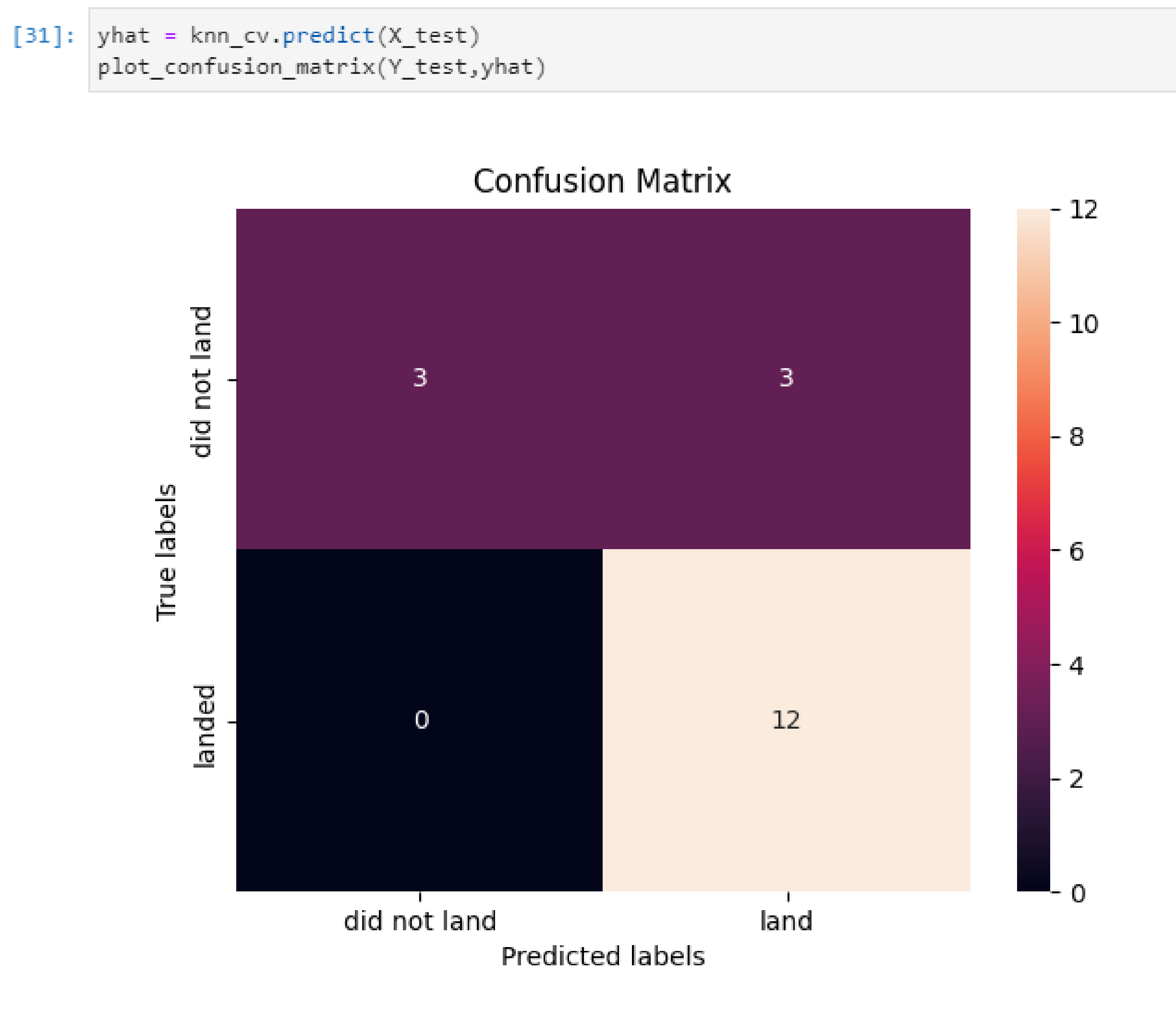


```
[36]: algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
      best_algorithm = max(algorithms, key=algorithms.get)
      print('Best Algorithm is',best_algorithm,'with a score of',algorithms[best_algorithm])
      if best_algorithm == 'Tree':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

```
Best Algorithm is Tree with a score of 0.8892857142857142
Best Params is: {'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 5, 'splitter': 'random'}
```

The classification tree algorithm achieved the highest accuracy among all models for the training dataset, while all classification algorithms performed equally well for the test dataset. The above code confirms the superiority of the classification tree algorithm in terms of accuracy.

Confusion Matrix



```
[31]: yhat = knn_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```

The confusion matrix for the decision tree classifier indicates that the classifier is able to differentiate between the various classes, but there is a significant issue with false positives. This means that the classifier is incorrectly identifying unsuccessful landings as successful landings.

Conclusions

Based on our analysis of the dataset, we can draw the following conclusions:

- The classification tree algorithm is the most effective machine learning technique for predicting launch outcomes.
- There is a positive correlation between the number of launches at a site and the success rate of those launches. As the number of launches increases, so does the success rate.
- Launches with lighter payloads (4,000 kg or less) had a higher success rate than those with heavier payloads.
- Launching payloads over 6,000 kg carries a high risk of failure.
- Starting in 2013, SpaceX's success rate for launches increased each year and is expected to continue improving in the future.
- KSC's LC-39A launch site has the highest success rate (76.9%) among all launch sites.
- SSO orbits have the highest success rate (100%) among all orbital trajectories, followed by GEO, HEO, SSO, and ES L1, which are the most commonly used trajectories.

Thank you!