IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Okoye Perpetua
25th October, 2023

Outline

Executive Summary

SpaceX has a reputation for its cost-efficient and reusable Falcon 9 rockets, setting itself apart from competitors. To understand the intricacies of SpaceX's operations, an exhaustive analysis was done.

The methodology encompassed data collection from SpaceX through APIs and web scraping, followed by data wrangling. Extensive EDA was conducted using SQL and visualization tools. Interactive visual analytic tools like Folium and Plotly provided dynamic insights into the data. Predictive analysis was applied using various classification models to forecast landing outcomes.

Notably, the decision tree model outperformed others in predicting successful landings, offering invaluable insights into launch costs and enhancing the competitive bidding process for rocket launches.

Introduction

- SpaceX has several significant achievements, one of which is its ability to return a spacecraft from low-earth orbit.
- It also offers Falcon 9 rocket launches at a cost of 62 million dollars, significantly less than its competitors.
- The objective of analyzing the dataset is to determine if the first stage will land, hence deducing the cost of a launch.
- This data is invaluable for companies looking to compete with SpaceX for rocket launches.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology

SpaceX data encompassed comprehensive details for each payload, launch site, and other mission-specific variables from their outer space ventures. This data was sourced through the SpaceX API, complemented by web scraping information about Falcon 9 and Falcon Heavy launches from Wikipedia

- Data wrangling

The data was first processed to address gaps and anomalies in the dataset, uncover patterns, and establish labels requisite for the training of supervised models.

- Exploratory data analysis (EDA) using visualization and SQL

A robust EDA was done using an array of visualization tools to enhance and modify data features. SQL played a pivotal role in exploring and reconfiguring the data structure, proving invaluable in problem-solving.

Methodology

Executive Summary

- Interactive visual analytics using Folium and Plotly Dash

Folium were instrumental in examining the geographical coordinates of various launch sites. Plotly Dash was integrated to facilitate dynamic, real-time data interaction with stakeholders, offering a more immersive experience.

- Predictive analysis using classification models

Our analytical approach hinged on a myriad of classification models including SVM, Classification Trees, Logistic Regression, and KNN. The preliminary steps involved preparing the target variables, standardizing data, and splitting the data into training and test segments. Model performance was subsequently gauged on the test dataset using the z-score to assess accuracy.

Data Collection

The data collection process below was done using two tools:

API

1. Libraries & Configurations: Initialized Python libraries like requests, pandas, numpy, and date time. Configured pandas for optimal data display.
2. Data Retrieval: Made an HTTP request to the SpaceX API to fetch rocket launch data. Ensured a successful request by verifying a 200 status code.
3. Data Parsing: Decoded the received data from JSON format and transformed it into a Pandas data frame.
4. Detail Enrichment: Using identification numbers in columns like rocket, payloads, Launchpad, and cores, additional API calls were made to fetch more detailed information which was then integrated into the data frame.

Data Collection

Web scrapping

1. Library Setup: Installed and initialized Python libraries such as beautifulsoup4, requests, and pandas.
2. Web Scraping Initialization: Defined helper functions for efficient data extraction and specified the URL for the "List of Falcon 9 and Falcon Heavy launches" Wikipedia page. An HTTP GET request was made to retrieve the page's content.
3. Data Parsing & Extraction: Using Beautiful Soup, the HTML response was parsed, and column names were extracted from the table headers. These names were stored in a column_names list.
4. Data Structuring: Created an empty dictionary, launch_dict, with column names as keys, then adjusted it by adding or removing relevant columns.

Data Collection – SpaceX API

Flowchart - API

Initialize necessary Python libraries → Configure pandas display settings → Access SpaceX API and make an HTTP request to get rocket launch data → Check if request was successful → Decode received data as JSON → Convert JSON to Pandas dataframe → For columns with IDs (e.g., rocket, payloads, launchpad, cores) (Make additional API calls using IDs ; Append detailed information to the dataframe)

https://github.com/Perpetua4/Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

Flowchart – Webscrapping

Install & Initialize Libraries: beautifulsoup4, requests, pandas → Web Scraping Initialization [Define helper functions for data extraction → Specify URL for Falcon 9 launches Wikipedia page snapshot → Perform HTTP GET request for the specified URL] → Data Parsing & Extraction [Parse the HTML page using BeautifulSoup→ Extract column headers from the HTML table → Append non-empty column names to a list] → Data Structuring [Create an empty dictionary with keys as column names → Initialize dictionary values as empty lists → Adjust dictionary by removing/adding columns as needed]

https://github.com/Perpetua4/Capstone/blob/main/jupyter-labs-webscraping%20(1).ipynb

Data Wrangling

- The goal is to understand patterns in the data and convert mission outcomes into training labels: 1 (successful landing) and 0 (unsuccessful landing).
- Data examination showcases cases where the booster landed or failed to land successfully. Example outcomes:

  "True Ocean" indicates a successful landing in a specific ocean region.

  "False Ocean" indicates an unsuccessful landing in that region.
- Similar outcomes exist for ground pads (RTLS) and drone ships (ASDS).

Data Wrangling

<u>Flowchart</u>

Setup & Initialization →Data Loading →Data Inspection →Data Analysis [Launch Site, Orbit Analysis , Mission Outcome, Landing Outcome]→ Data Export

https://github.com/Perpetua4/Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

Here, patterns within the data were explored. Such visualization charts and the variables they explored are summarized below

- Categorical plot (catplot)
  - Flight Number vs Payload Mass; Flight Number vs Launch Site
- Scatterplot
  - Flight Number vs. Launch Site; Launch Sites vs Payload mass; Success rate of each orbit type; Flight Number vs Orbit type; Payload vs Orbit type

EDA with Data Visualization

- Bar chart
  - Success rate of each orbit; Flight Number vs Orbit type; Launch success yearly trend
- Line plot
  - Year vs average success rate

https://github.com/Perpetua4/Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(10).ipynb

EDA with SQL

SQL queries performed are as follows:

1. %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
2. %sql select * from SPACEXTABLE WHERE Launch_Site like 'CCA%' limit 5;
3. %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer = 'NASA (CRS)';
4. %sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version = 'F9 v1.1';
5. %sql select min(Date) from SPACEXTABLE where Landing_Outcome = 'Success (ground pad)';

EDA with SQL

6. %sql select Booster_Version from SPACEXTABLE where Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ >4000 and PAYLOAD_MASS__K
7. %sql select Mission_Outcome, Count (*) as Total from SPACEXTABLE group by Mission_Outcome having Mission_Outcome like 'success%' or Missi
8. %sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE);
9. %sql select Date,substr(Date,6,2) as month, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE where substr(Date,0,5)='2015

https://github.com/Perpetua4/Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(2).ipynb

Build an Interactive Map with Folium

Map objects created and added to a folium map, include

- Markers - to indicate specific locations of launch sites on the site map
- Circles - to add a highlighted circle area with a text label on a specific coordinate.
- Lines - connect launch sites to specific locations such as railways, coastlines etc. on the site map.

https://github.com/Perpetua4/Capstone/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

Plots/graphs and interactions added to the dashboard include:

Plots/graphs

- Success-pie-chart: This chart displays the total successful launches.
- Success-payload-scatter-chart: This plot showcases the correlation between the Payload Mass and the Launch Success.

Build a Dashboard with Plotly Dash

Interactions

- Site-dropdown - allows the user select a specific Launch Site or "ALL" to view data for all sites. Pie chart and scatter plot will update based on the selection.
- Payload-slider: Allows users to select a range for the Payload Mass (in Kg). The scatter plot updates to only display the data within this range.

https://github.com/Perpetua4/Capstone/blob/main/spacex_dash_app%20(1).py

Predictive Analysis (Classification)

The process used to build, evaluate, improve, and finding the best performing classification model are described as follows

1. Data Preparation:
    - Extraction: Extract the 'Class' column from the data to form a NumPy array, Y.
    - Standardization: Standardize the data X using preprocessing.
    - Splitting the Data: Divide the data into training and test sets.

Predictive Analysis (Classification)

2. Model Development and Evaluation: Hyperparameter tuning was done in models using GridSearchCV; Test accuracy was also computed, and a confusion matrix was plotted.

- Logistic Regression
- Support Vector Machine (SVM)
- Decision Tree
- K-Nearest Neighbors (KNN)

3. Model Comparison: The models' accuracies were compared.

Predictive Analysis (Classification)

Flowchart

Data Preparation → Data Standardization → Data Splitting (Train/Test) → Model Development (Logistic , SVM , Decision Tree, K-Nearest, Regression Neighbors) → Evaluation → Test and Confusion Matrix → Model Comparison

https://github.com/Perpetua4/Capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20(1).ipynb

Results

Exploratory data analysis (EDA) results

EDA - Data Wrangling

- Based on the outcome column, a new classification variable 'Class' was created.
- The data frame now contain a 'Class' column to denote if the landing was successful (1) or not (0).
- The success rate can be determined using the mean of the 'Class' column.
- The modified dataset is ready for export as "dataset_part_2.csv“, sets the stage for predictive modeling by ensuring the data is well-understood, cleaned, and appropriately labeled.

Results

EDA – SQL based on the order of SQL queries in the EDA with SQL section

1. The distinct launch sites in the space mission were displayed. The launch sites are: CCAFS LC-40, VAFB SLC-4E, KSC LC-39A, and CCAFS SLC-40.
2. Displayed 5 records where launch sites begin with the string 'CCA'.
3. The total payload mass carried by boosters launched by NASA (CRS) was calculated as 45,596 KG.
4. The average payload mass carried by booster version F9 v1.1 was calculated as 2928.4 KG.
5. The first successful landing outcome in a ground pad was achieved on 2015-12-22.

Results

EDA – SQL based on the order of SQL queries in the EDA with SQL section

6. Booster names with a successful landing outcome on a drone ship and payload mass between 4000 and 6000 KG were listed. They are: F9 FT B1022, F9 FT B1026, F9 FT B1021.2, and F9 FT B1031.2.

7. Successful and failure mission outcomes count was calculated. There was 1 mission with "Failure (in flight)" outcome, 98 missions with a straightforward "Success" outcome, and a few with other types of success.

8. Booster_versions names which have carried the maximum payload mass were listed. They include versions like F9 B5 B1048.4, F9 B5 B1049.4, and so on.

9. Records for the year 2015 were displayed showing month names, failure landing outcomes in drone ship, booster versions, and launch sites.

Results

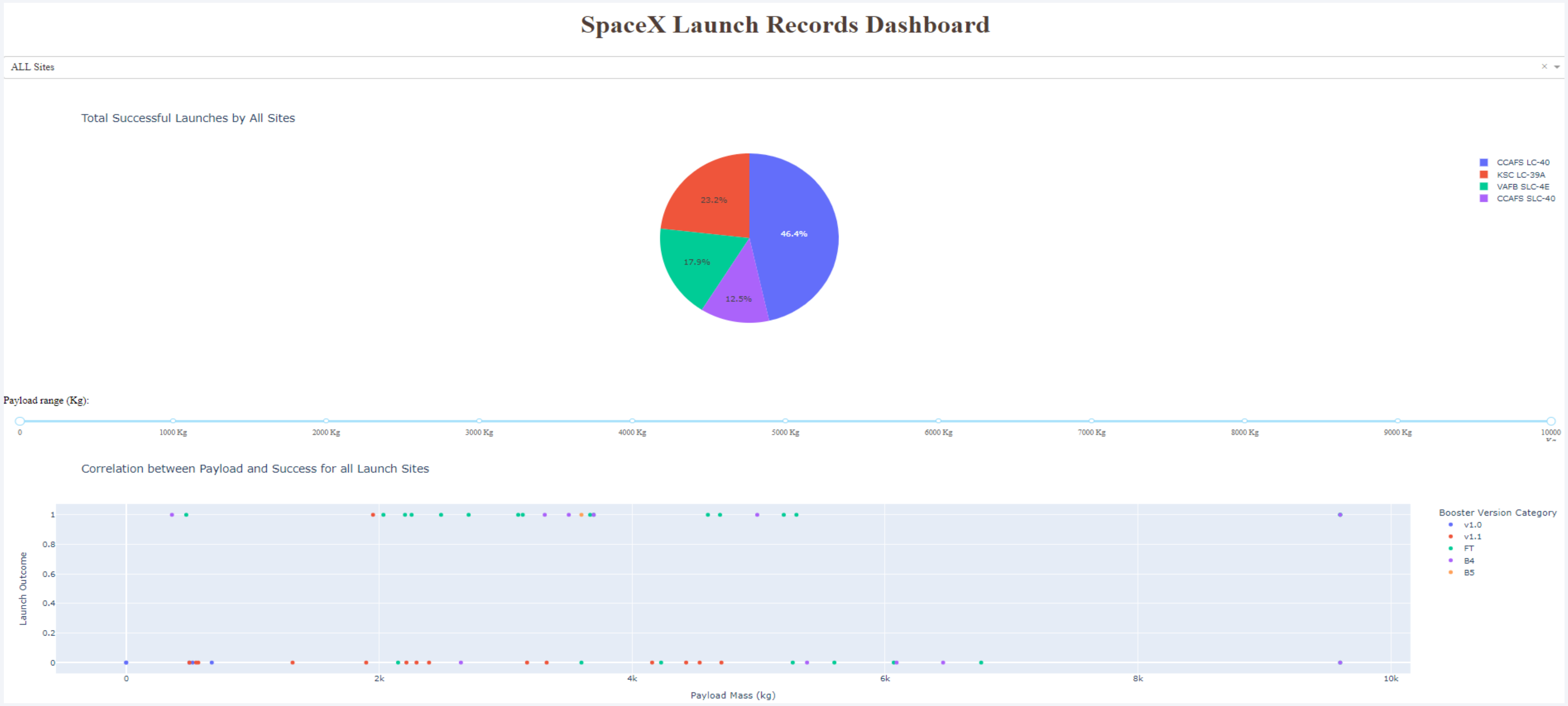
- EDA - Data Visualizations and Observations:
- Flight Number vs. Payload Mass: As flight number increases, there's a higher likelihood of the first stage landing successfully. Heavier payloads appear to decrease the likelihood of a successful landing.
- Launch Site Analysis: Different launch sites have varying success rates. For instance, CCAFS LC-40 has a 60% success rate, while KSC LC-39A and VAFB SLC 4E have a 77% success rate.
- Flight Number vs. Launch Site: The scatter plot visualizations show patterns in launch outcomes based on flight numbers and launch sites.
- Payload vs. Launch Site: For the VAFB-SLC launch site, rockets are not launched for heavy payloads (greater than 10000 kg).

Results

- EDA - Data Visualizations and Observations:
- Success Rate by Orbit Type: A scatter plot and bar chart are used to display the success rates of launches for different orbit types. The goal is to understand if specific orbits have higher success rates.
- Flight Number vs. Orbit Type: In the LEO orbit, success seems to be related to the number of flights, but there's no evident relationship for GTO orbit.
- Payload vs. Orbit Type: With heavier payloads, successful landings are more prevalent for Polar, LEO, and ISS orbits. For GTO, both positive and negative outcomes occur.
- Yearly Trend of Launch Success: A bar plot visualizes the success rate of launches over the years, showcasing a trend in SpaceX's mission successes.

Results

- Interactive analytics demo in screenshots



Results

- Predictive analysis results
- Logistic Regression:
  - Best hyper parameters found were: C: 0.01, penalty: l2, solver: lbfgs.
  - Validation accuracy: 84.64%.

- Support Vector Machine (SVM):
  - Best hyperparameters found were: C: 1.0, gamma: 0.0316, kernel: sigmoid.
  - Validation accuracy: 84.82%.

Results

- Decision Tree:
  - Best hyperparameters were derived from a set of parameters like criterion, splitter, max_depth, etc.
  - Validation accuracy: 88.89%.

- K-Nearest Neighbors (KNN):
  - Best hyperparameters found were: algorithm: auto, n_neighbors: 10, p: 1.
  - Validation accuracy: 84.82%.

Results

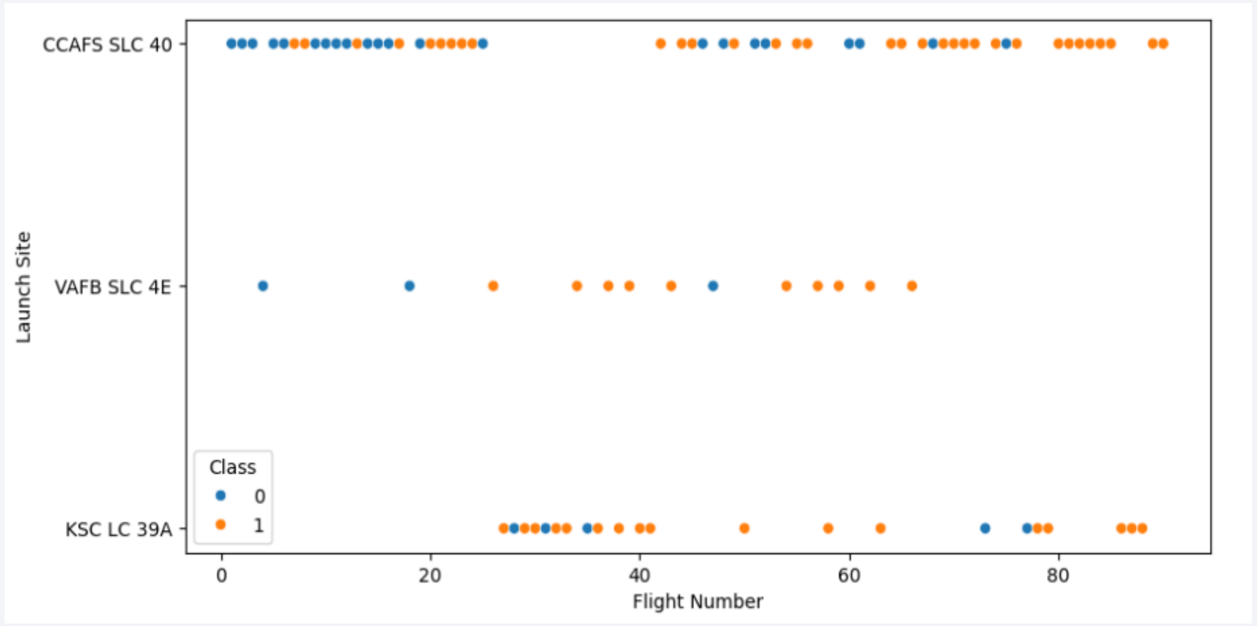
- 3. Comparison and Conclusion:
  - The best performing model was the Decision Tree with a validation accuracy of 88.89%.

Section 2

Insights drawn from EDA

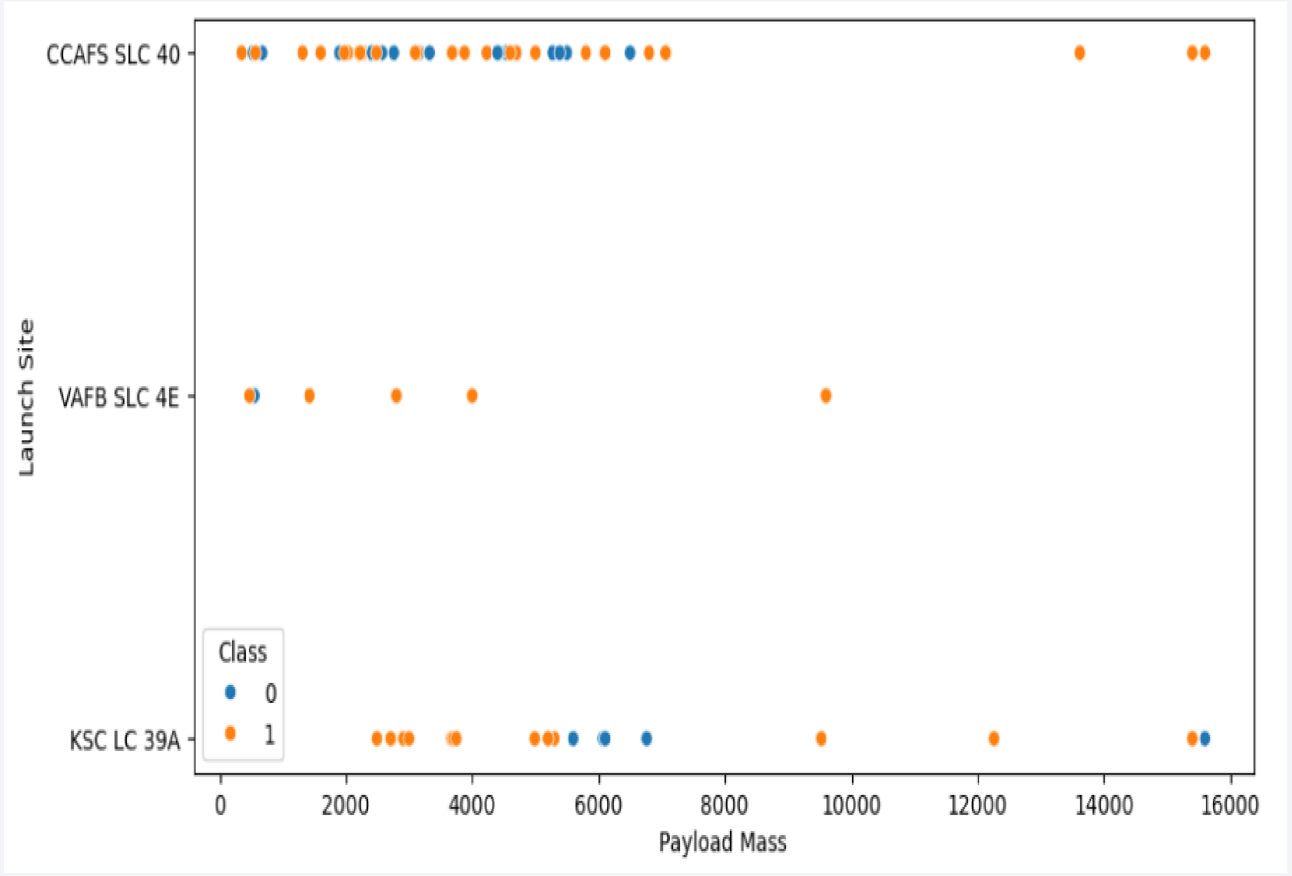
Flight Number vs. Launch Site

- There are two classes of data points, "Class 0" (blue) and "Class 1" (orange).
- The "CCAFS SLC 40" site has the most launches, spread across the entire range of flight numbers.
- Both "Class 0" and "Class 1" launches are spread across all three sites. However, the later flight numbers (60 and above) from the "CCAFS SLC 40" site show a dominance of "Class 1" launches.

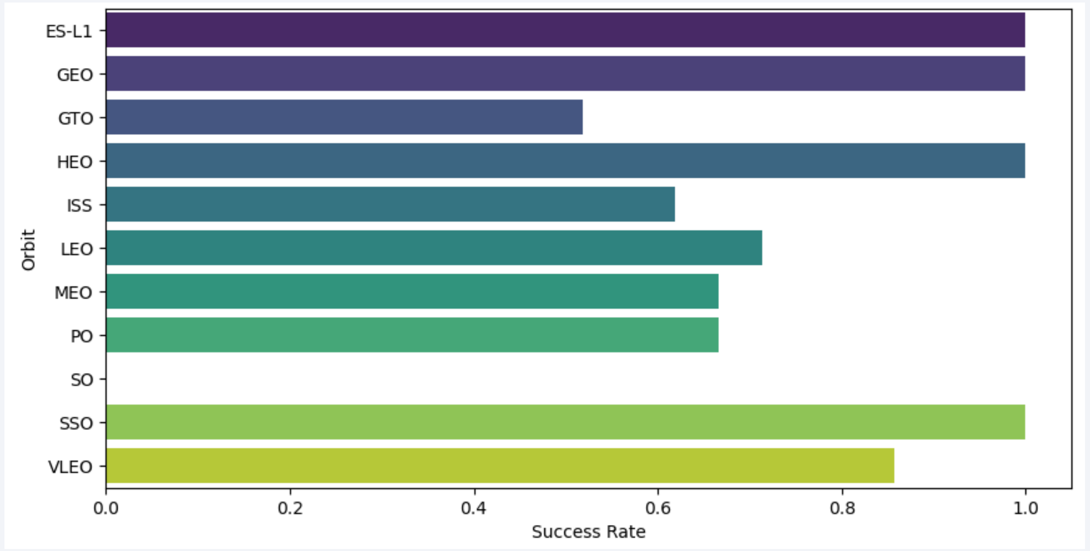


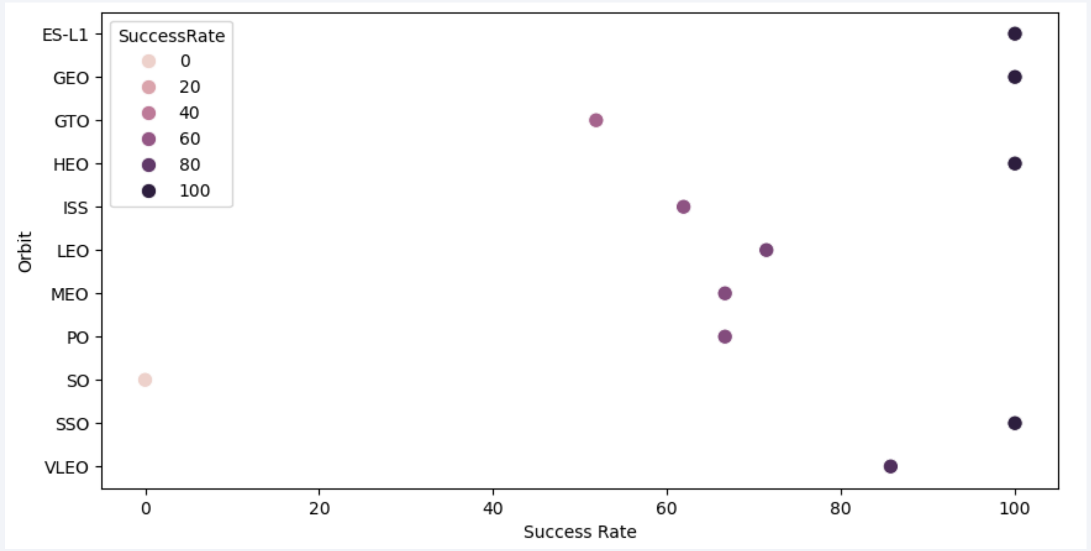
Payload vs. Launch Site

- The "CCAFS SLC 40" site shows a wide range of payload masses. There's a dense concentration of both classes in the lower payload mass range.
- Across all sites, "Class 1" payloads are more common in the very high payload mass category.
- Most of the launches, regardless of the class, are concentrated in the lower payload mass range.



Success Rate vs. Orbit Type

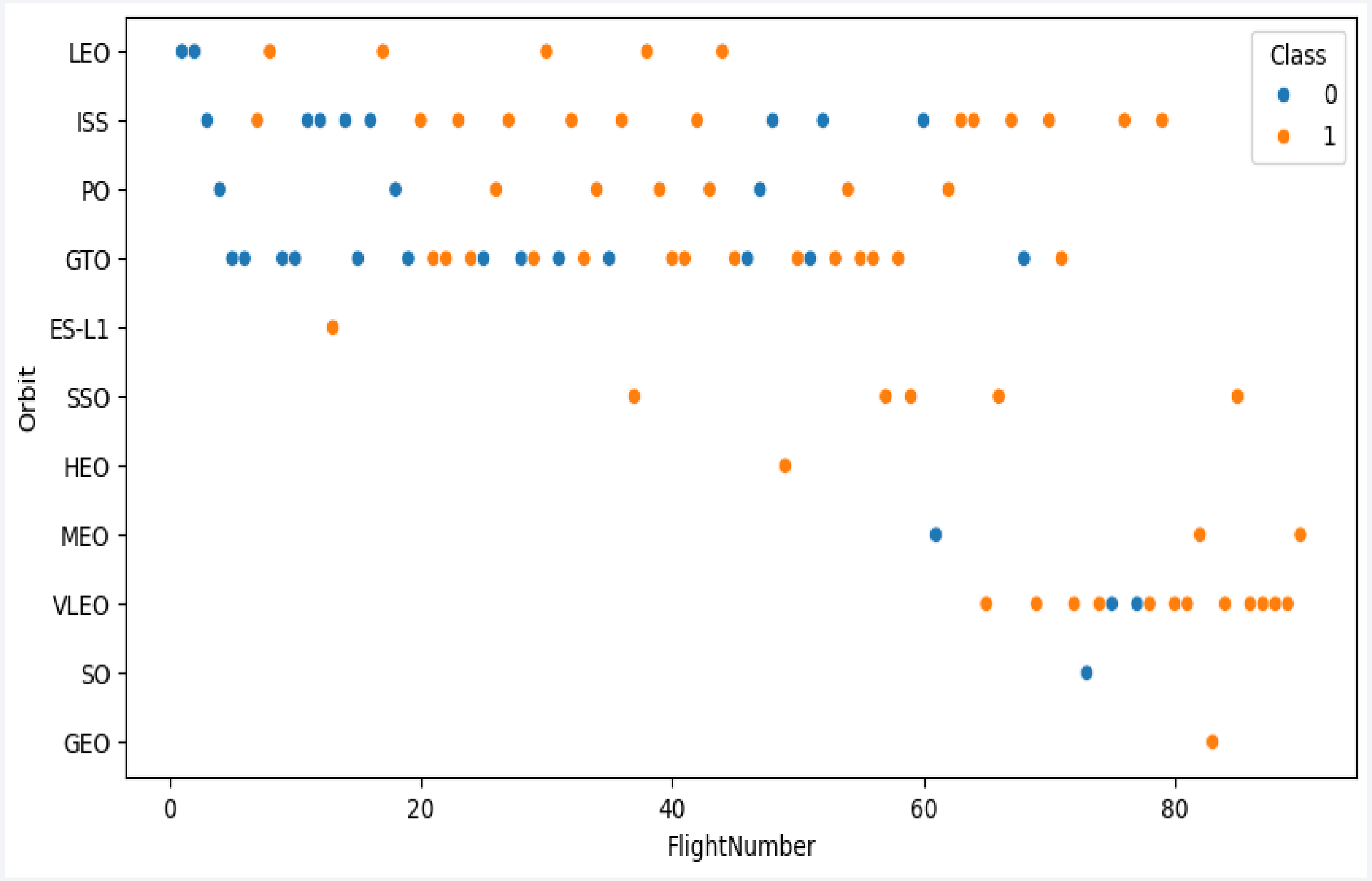




- Several orbits, such as ES-L1, PO, and SO, have success rates below 20% as indicated by the lightest colored dots
- HEO and ISS show the highest success rates, near or at 100%, as denoted by the darkest dots on the extreme right.
- There are no orbits with a success rate specifically at the 20%, 40%, or 60% mark

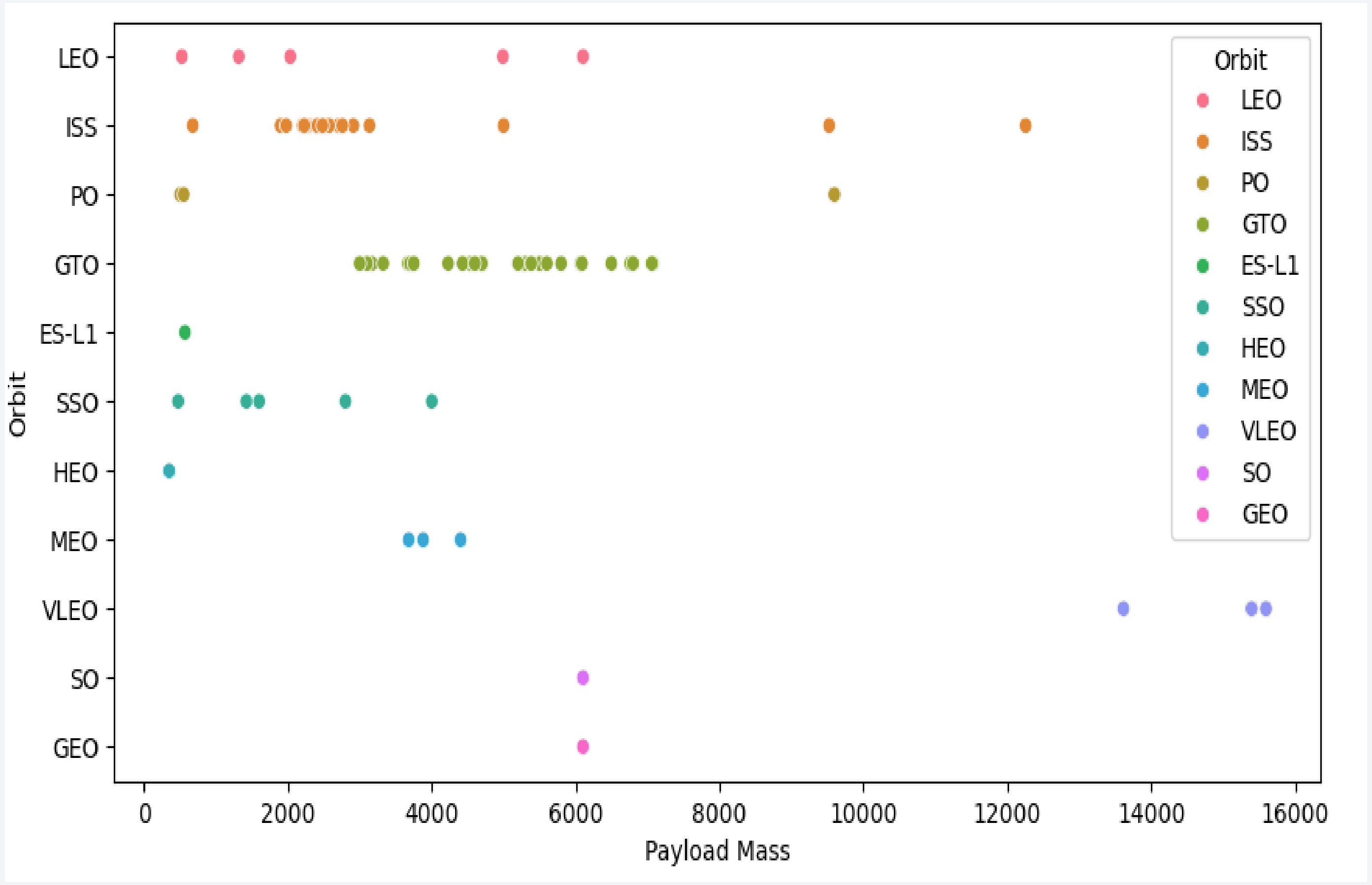
Flight Number vs. Orbit Type

- The orbits LEO, ISS, and GTO show the most launches, with a mix of both Class 0 and Class 1 throughout the range of flight numbers
- As the flight numbers increase (moving right on the X-axis), Class 1 launches (orange dots) appear to be more frequent, especially evident in the LEO, ISS, and GTO orbits.

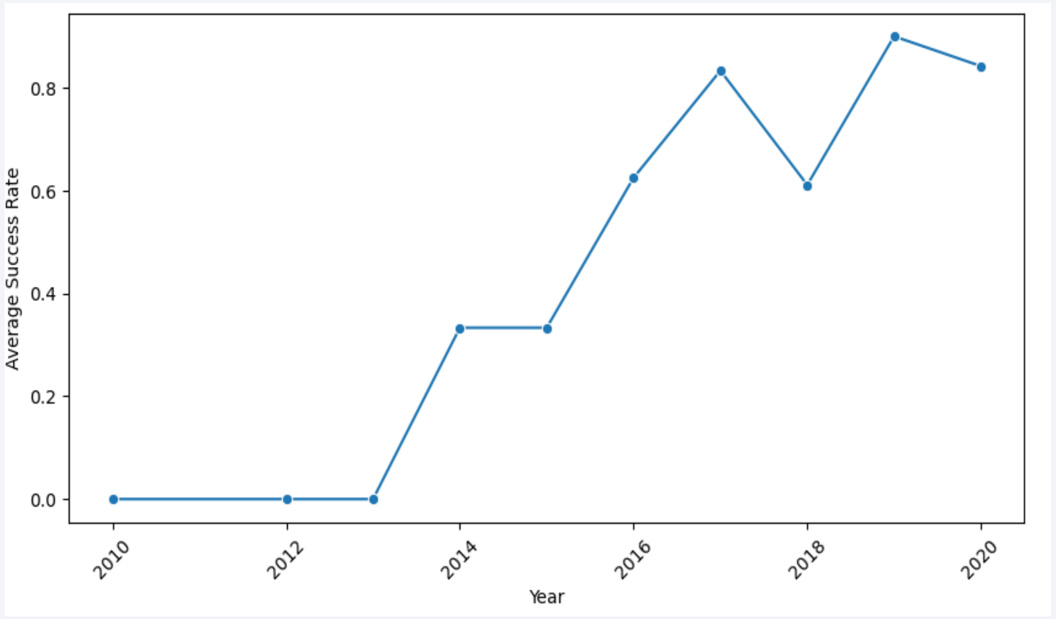


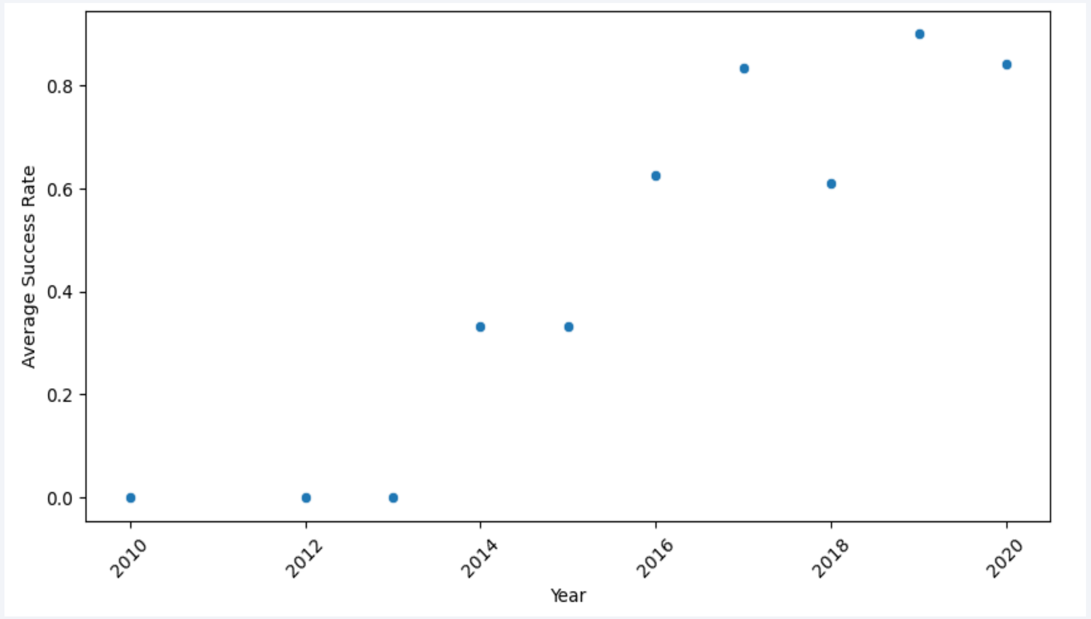
Payload vs. Orbit Type

- The LEO orbit, like other orbits, contains payloads with a wide range of masses, from lightweight to heavier ones, with most payloads under 4000 units.
- The PO orbit seems to have a single data point, indicating a payload of medium mass.



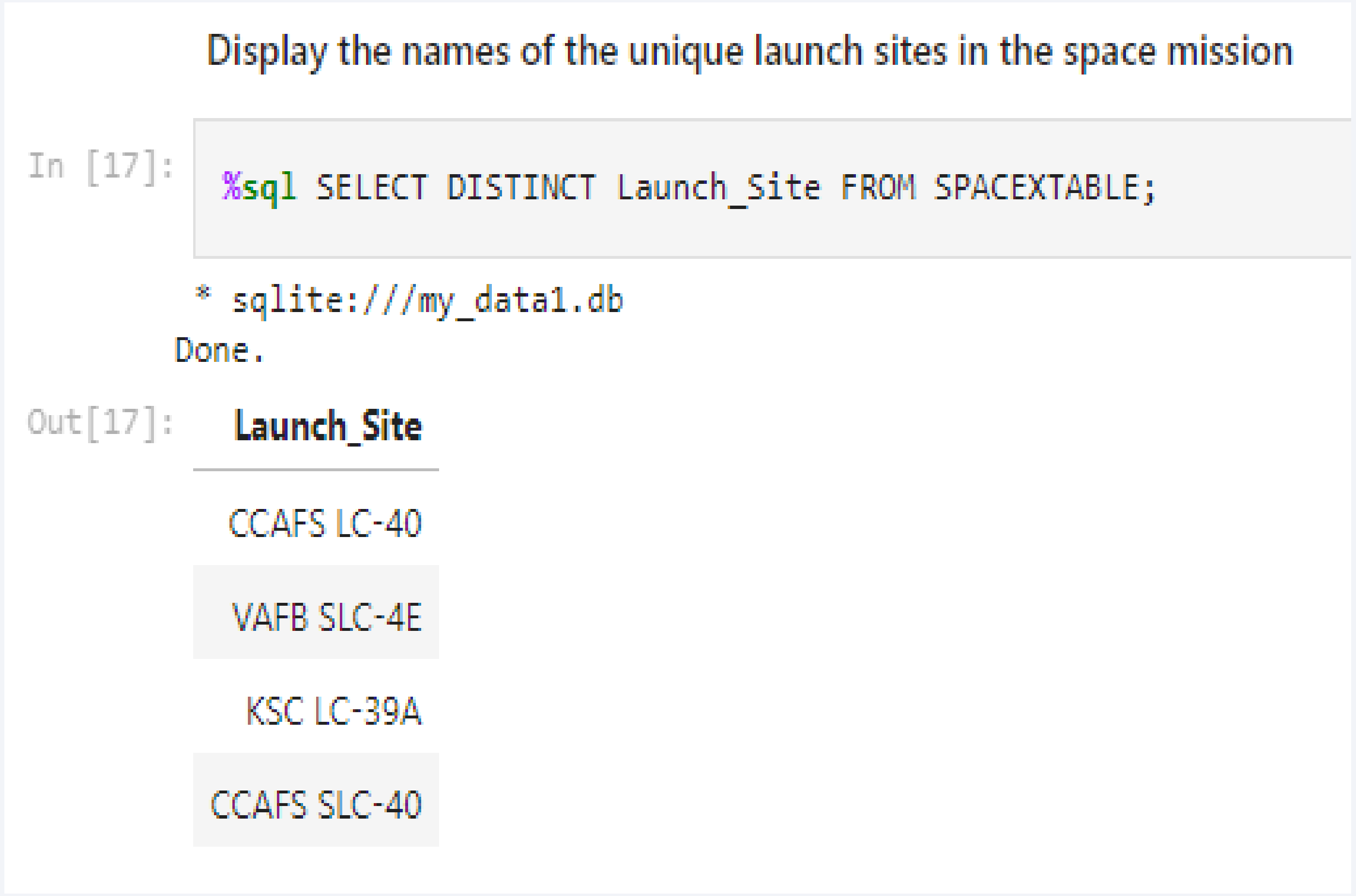
Launch Success Yearly Trend





The plots depict a trend of increasing success in launches over the decade. While the decade began with low success rates, there was a consistent and significant improvement in launch successes, especially in the latter half of the decade.

All Launch Site Names



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

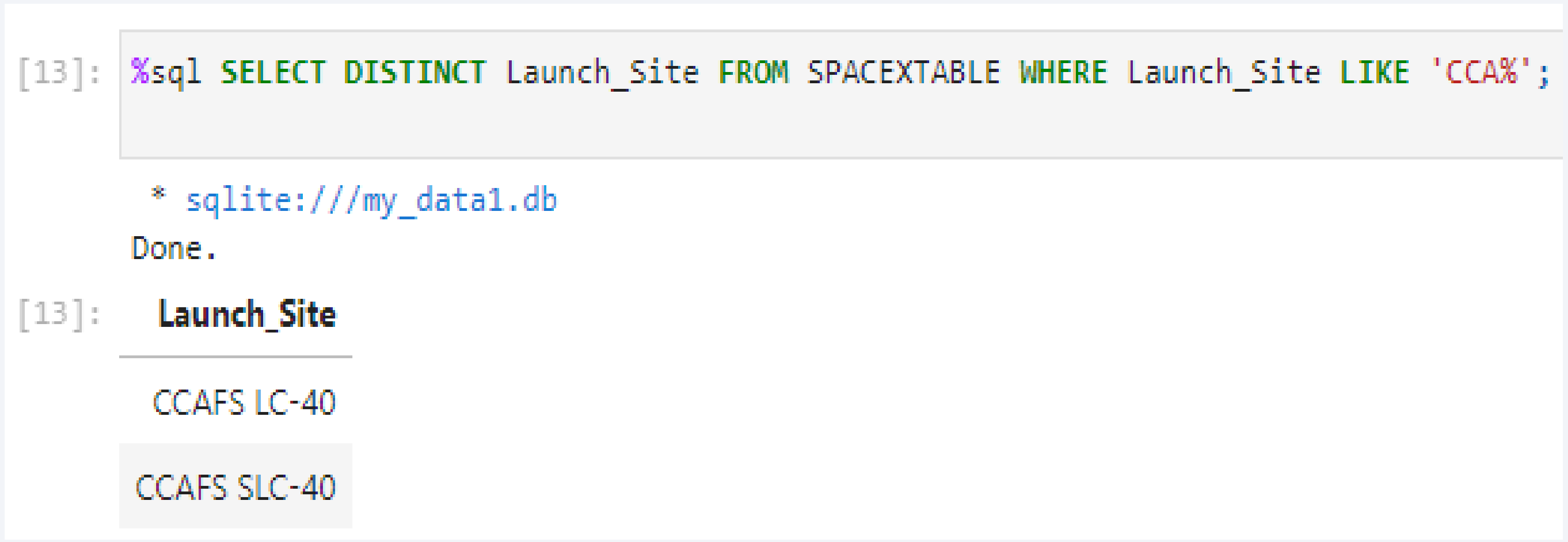
```
In [17]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;

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

Out[17]:

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

The provided SQL query is displaying the unique launch sites used in the space mission by selecting distinct values from the "Launch_Site" column in the "SPACETABLE".

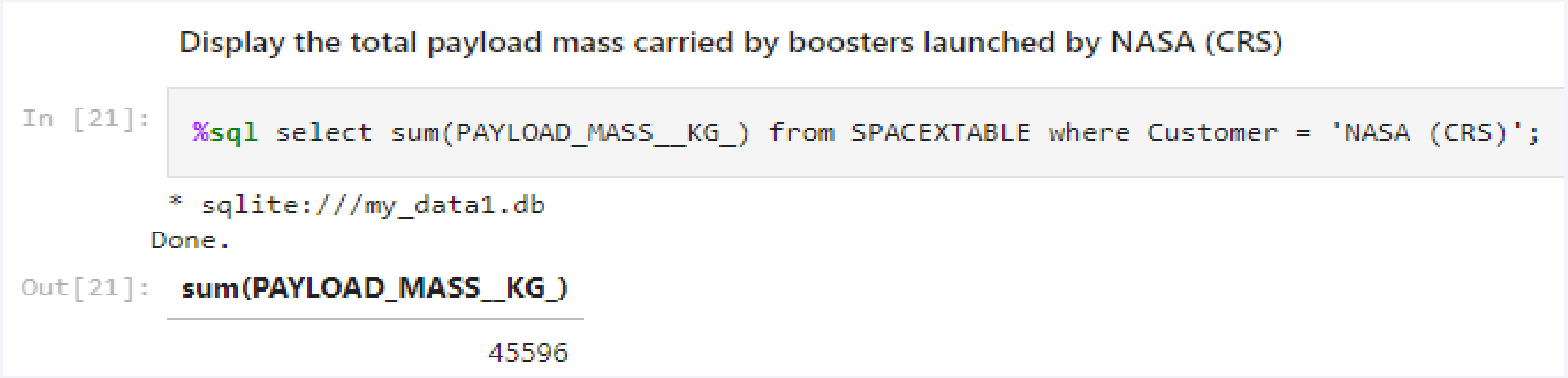
Launch Site Names Begin with 'CCA'



```
[13]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%';

 * sqlite:///my_data1.db
Done.

[13]: Launch_Site
      CCAFS LC-40
      CCAFS SLC-40
```

The query retrieves unique names of launch sites that begin with the prefix "CCA".

Total Payload Mass



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

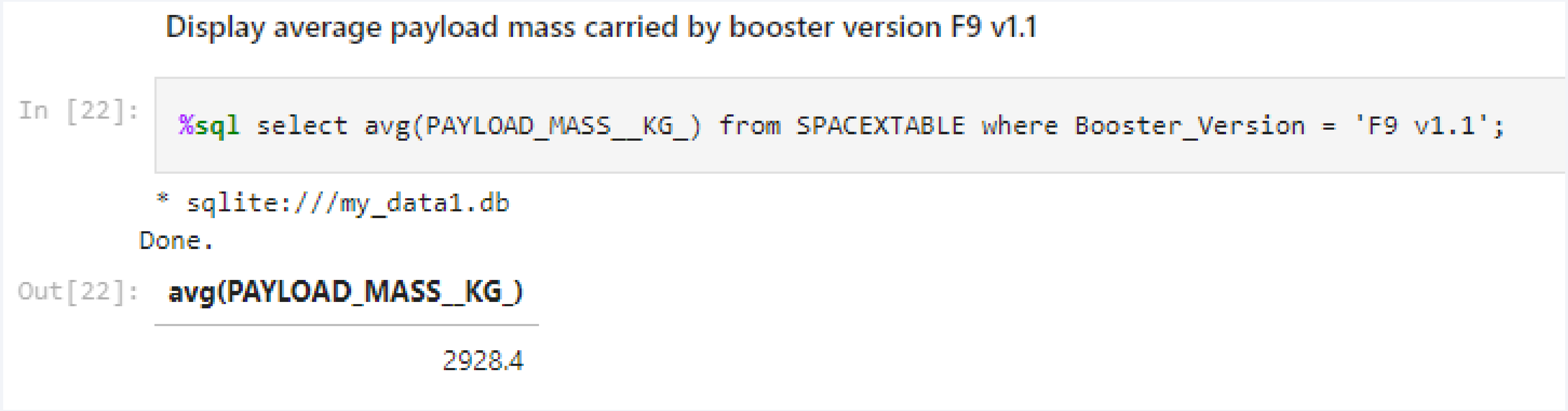
```
In [21]: %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer = 'NASA (CRS)';
```

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

```
Out[21]: sum(PAYLOAD_MASS__KG_)
                          45596
```

The query calculates the total payload mass (using the sum function) for boosters launched by the customer "NASA (CRS)". The total payload mass carried by boosters launched by NASA (CRS) is 45596 kilograms.

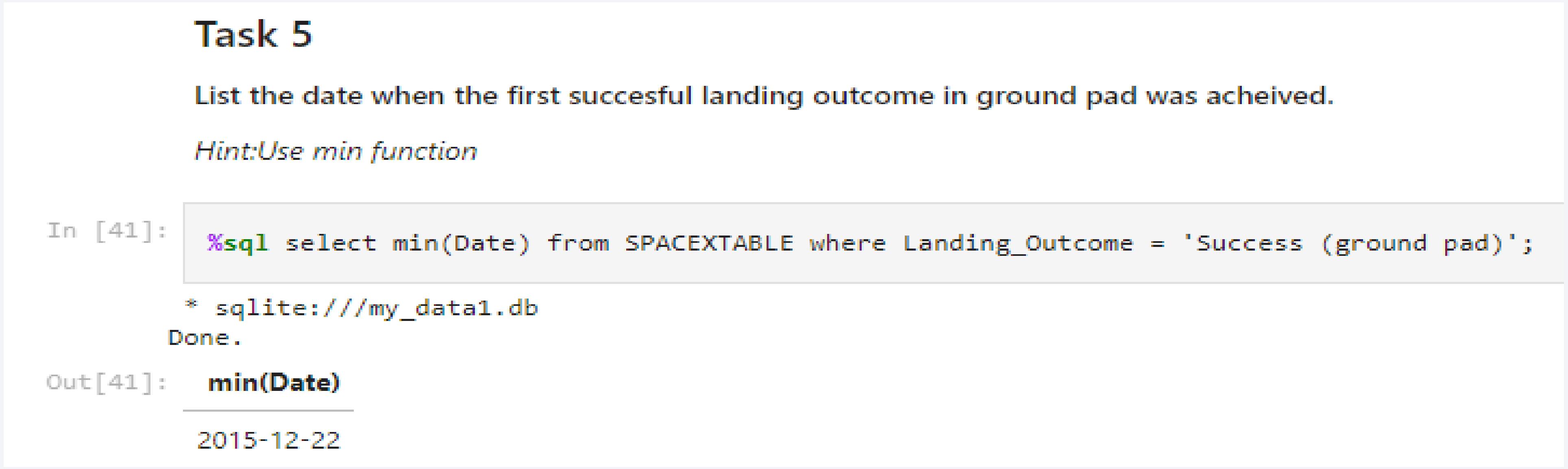
Average Payload Mass by F9 v1.1


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

```
In [22]: %sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version = 'F9 v1.1';

 * sqlite:///my_data1.db
Done.

Out[22]: avg(PAYLOAD_MASS__KG_)
                        2928.4
```

The query calculates the average payload mass (using the avg function) for the booster version "F9 v1.1". Accordingly, the average payload mass carried by booster version F9 v1.1 is approximately 2928.4 kilograms.

First Successful Ground Landing Date



Task 5

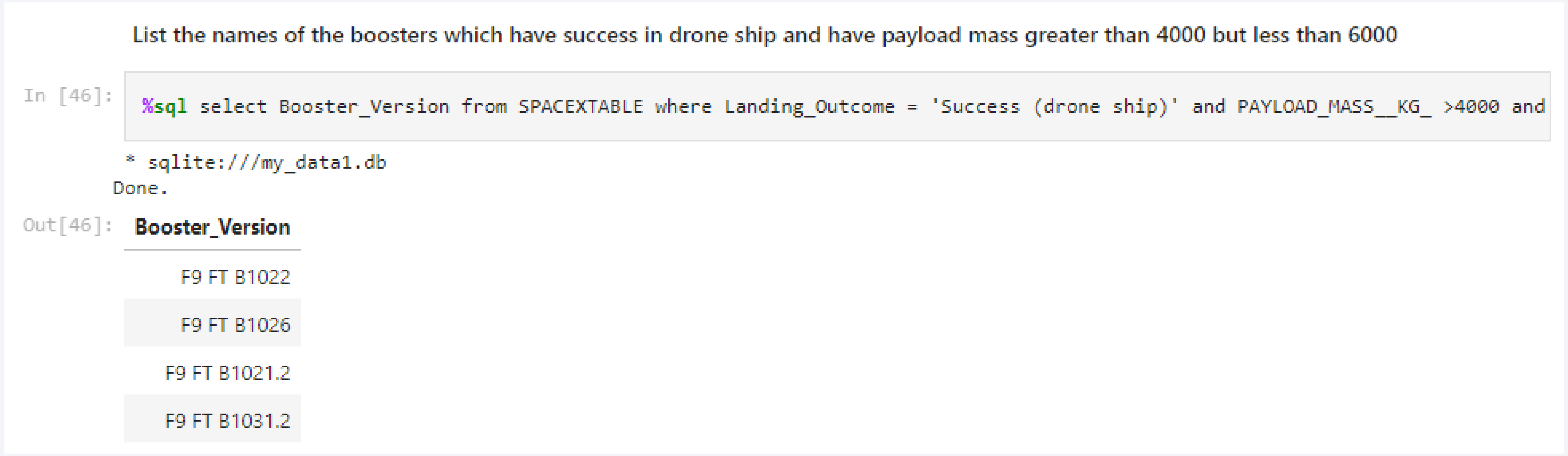
List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
In [41]: %sql select min(Date) from SPACEXTABLE where Landing_Outcome = 'Success (ground pad)';

 * sqlite:///my_data1.db
Done.

Out[41]: min(Date)
         2015-12-22
```

The query identifies the earliest date (using the min function) where a successful landing on a ground pad was achieved. Accordingly, the first successful landing outcome on a ground pad took place on December 22, 2015.

Successful Drone Ship Landing with Payload between 4000 and 6000



List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
In [46]: %sql select Booster_Version from SPACEXTABLE where Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ >4000 and
```

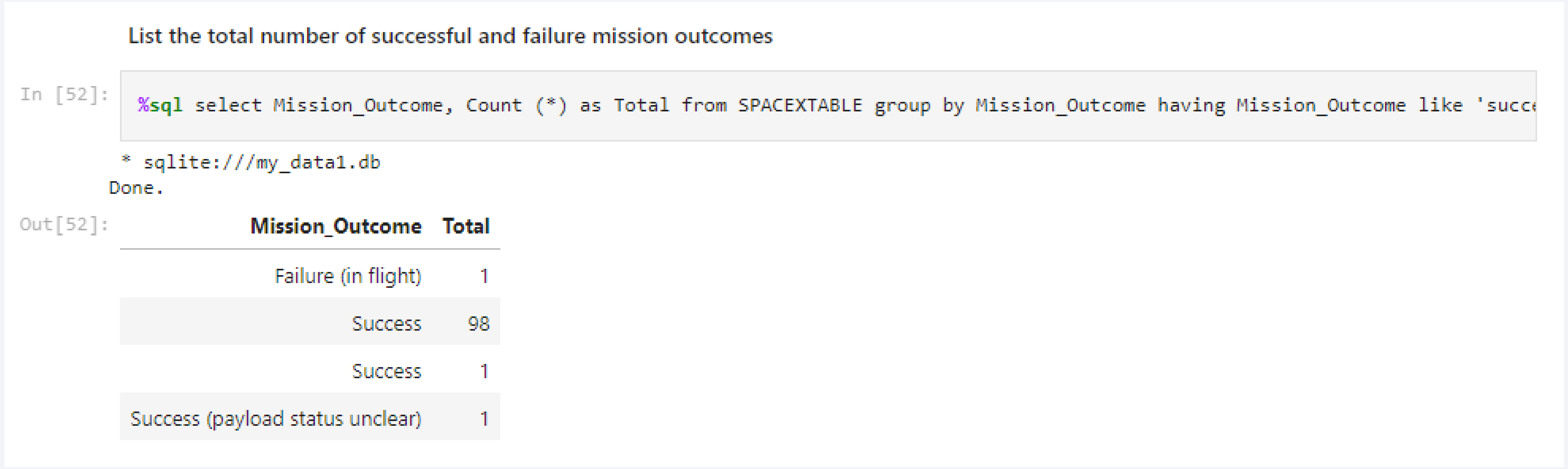
```
 * sqlite:///my_data1.db
Done.
```

Out[46]:

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

The query extracts the versions of boosters. The selected boosters have successfully landed on a drone ship and possess a payload mass ranging between 4000 kg and 6000 kg.

Total Number of Successful and Failure Mission Outcomes


List the total number of successful and failure mission outcomes

```
In [52]: %sql select Mission_Outcome, Count (*) as Total from SPACEXTABLE group by Mission_Outcome having Mission_Outcome like 'succe
```

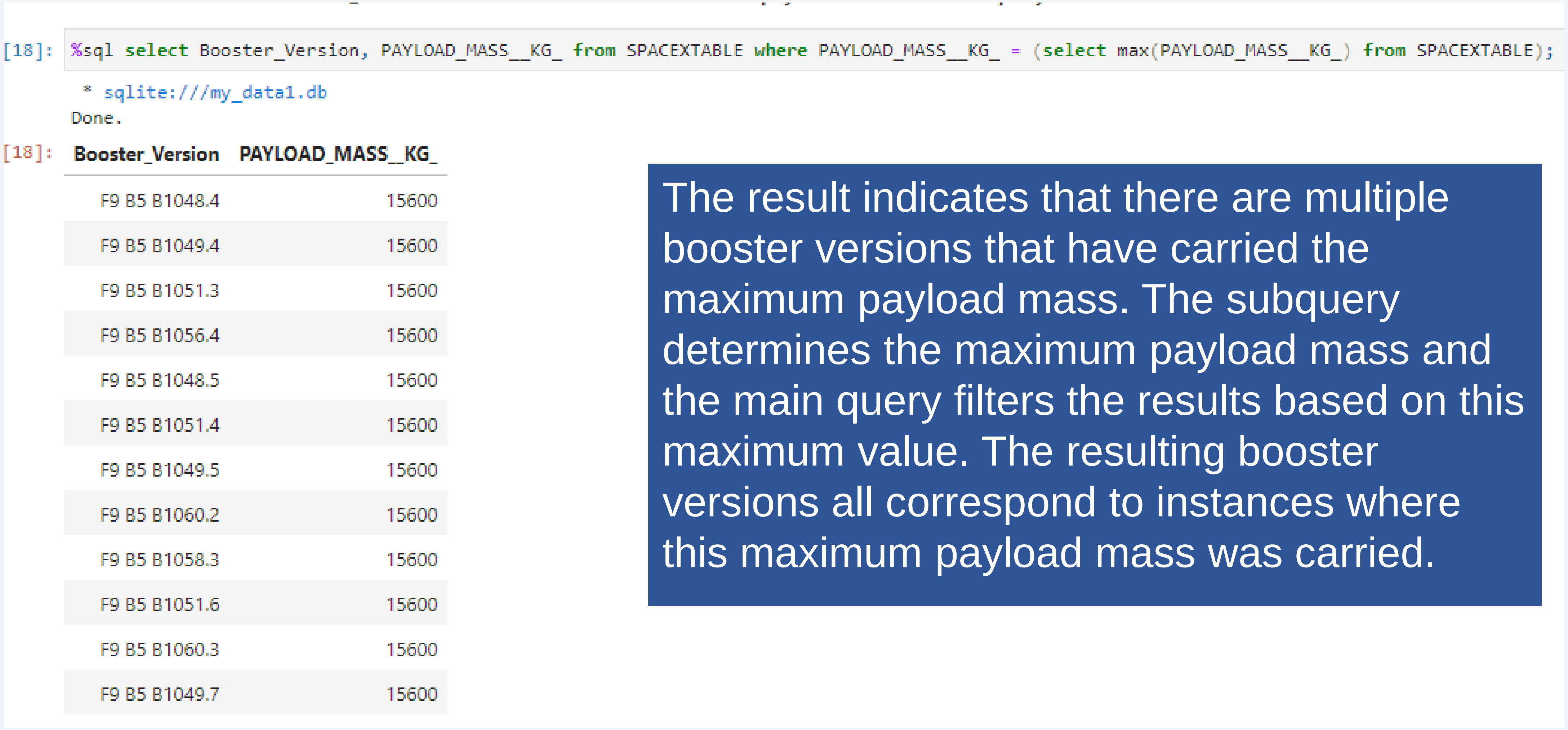
```
 * sqlite:///my_data1.db
Done.
```

Out[52]:

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

There's a minor discrepancy with the data since the "Success" outcome is listed twice. This might be due to some inconsistency in the dataset. The total number of successful missions is 100 (if we sum the two success counts) and failed missions is 1.

Boosters Carried Maximum Payload



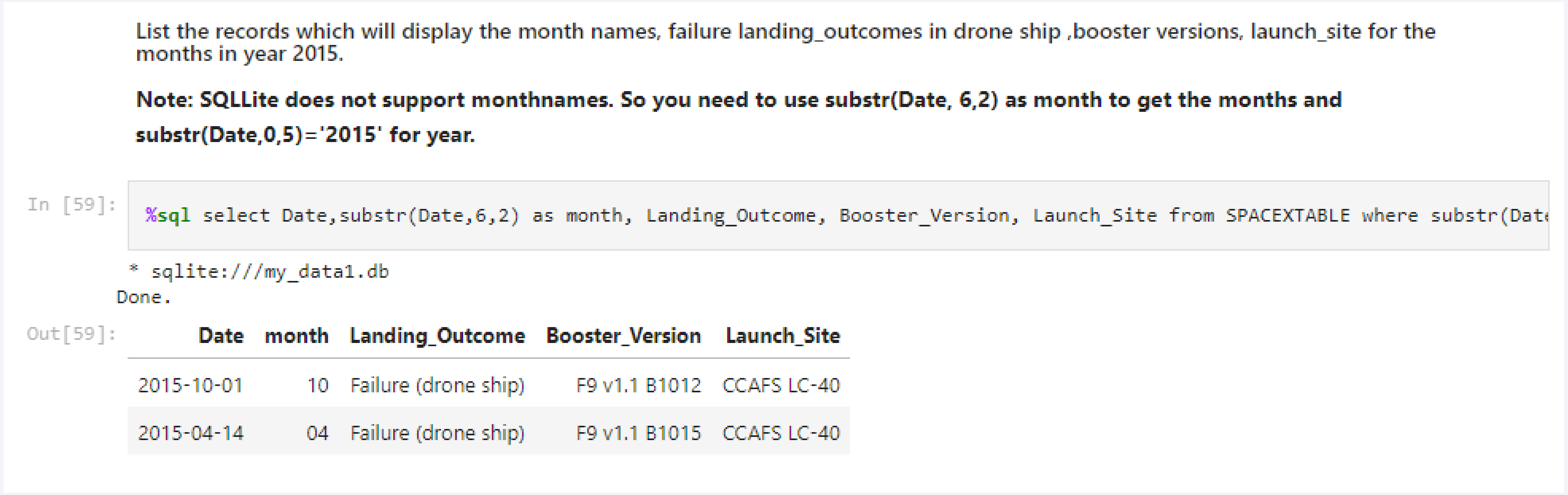
```
[18]: %sql select Booster_Version, PAYLOAD_MASS__KG_ from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE);

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

[18]:

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

The result indicates that there are multiple booster versions that have carried the maximum payload mass. The subquery determines the maximum payload mass and the main query filters the results based on this maximum value. The resulting booster versions all correspond to instances where this maximum payload mass was carried.

2015 Launch Records


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.

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.

```
In [59]: %sql select Date,substr(Date,6,2) as month, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE where substr(Dat
```

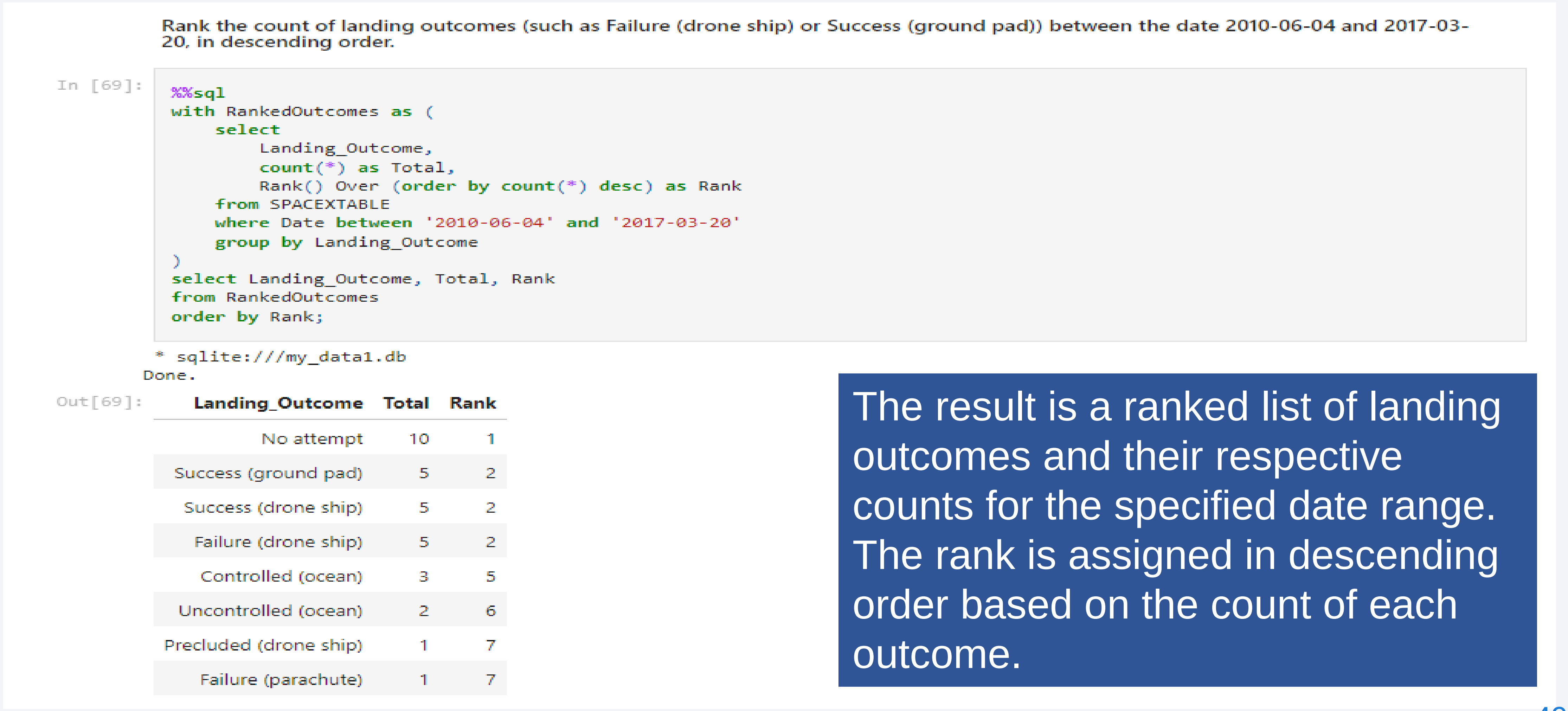
```
 * sqlite:///my_data1.db
Done.
```

Out[59]:

Date	month	Landing_Outcome	Booster_Version	Launch_Site
2015-10-01	10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
2015-04-14	04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

The landing outcomes indicate "Failure (drone ship)" for the displayed entries. The launch site for these records is "CCAFS LC-40". The month is extracted correctly as shown in the results.

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



Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

In [69]:
```
%%sql
with RankedOutcomes as (
    select
        Landing_Outcome,
        count(*) as Total,
        Rank() Over (order by count(*) desc) as Rank
    from SPACEXTABLE
    where Date between '2010-06-04' and '2017-03-20'
    group by Landing_Outcome
)
select Landing_Outcome, Total, Rank
from RankedOutcomes
order by Rank;
```

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

Out[69]:

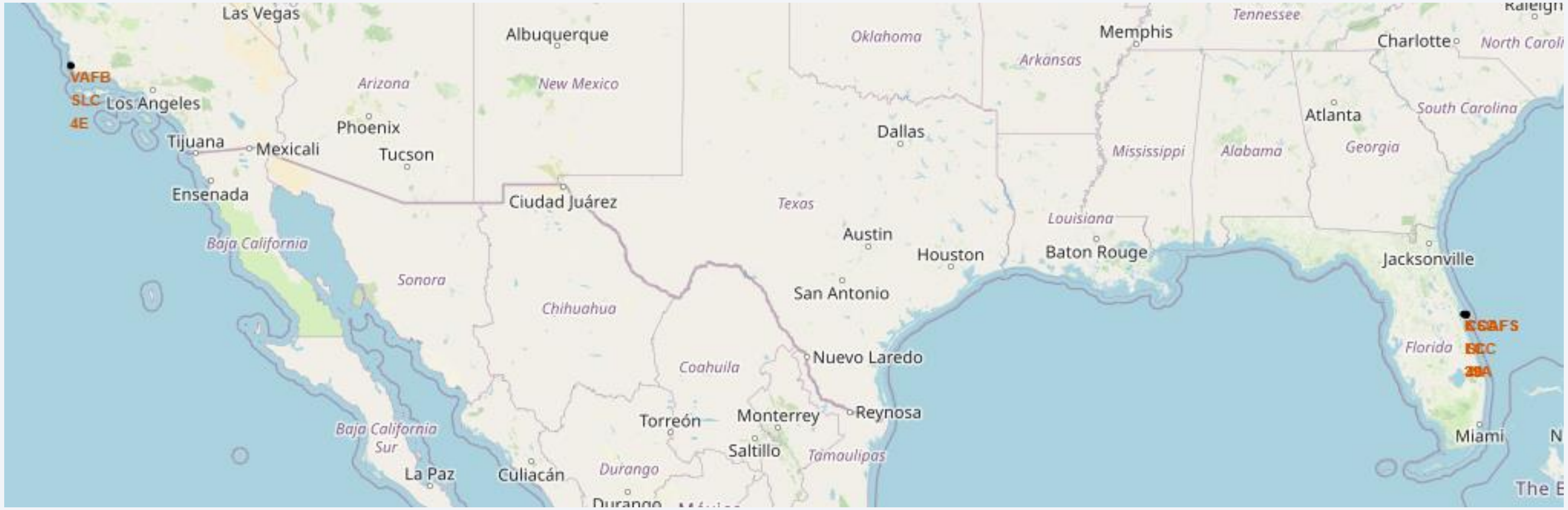
Landing_Outcome	Total	Rank
No attempt	10	1
Success (ground pad)	5	2
Success (drone ship)	5	2
Failure (drone ship)	5	2
Controlled (ocean)	3	5
Uncontrolled (ocean)	2	6
Precluded (drone ship)	1	7
Failure (parachute)	1	7

The result is a ranked list of landing outcomes and their respective counts for the specified date range. The rank is assigned in descending order based on the count of each outcome.

Section 3

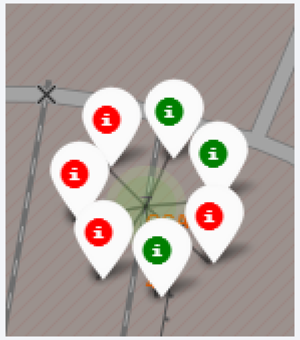
Launch Sites Proximities Analysis

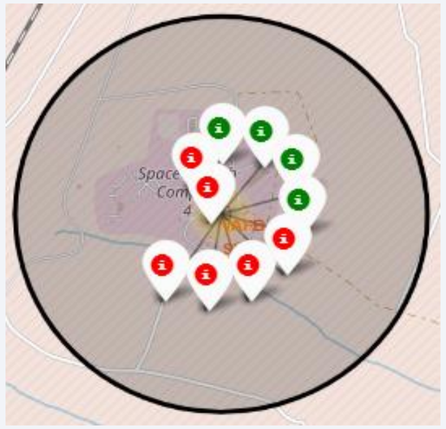
Map with Marked Launch Sites

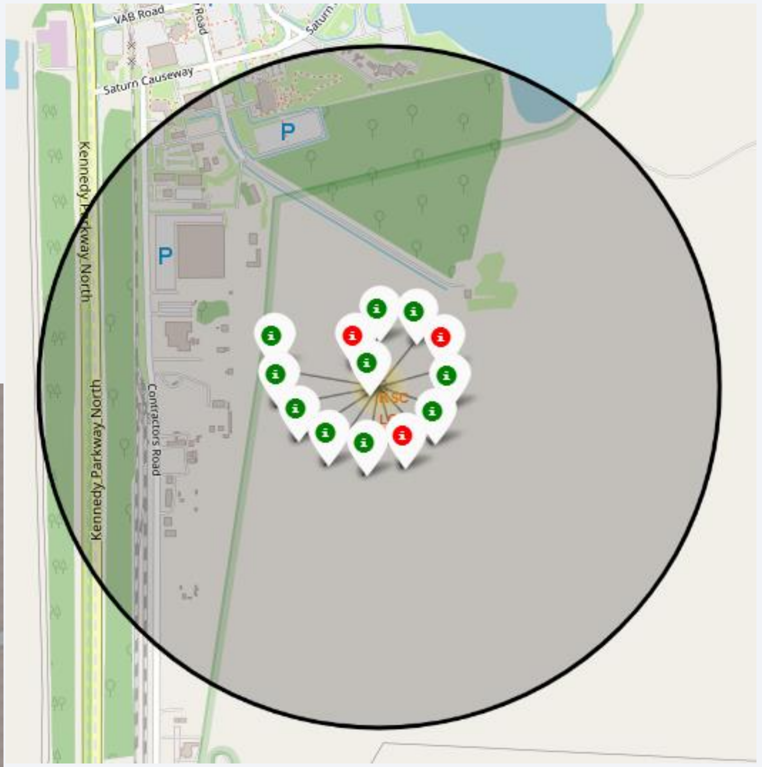


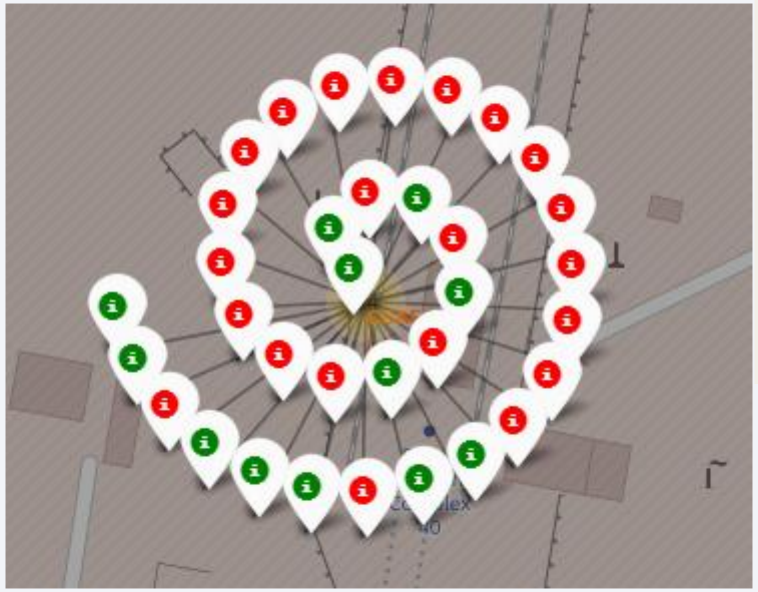
There are markers and annotations on the map representing locations of launch sites. On the western coast, near Los Angeles, we see the marker for "VAFB SLC 4E“ launch site. On the eastern coast, near Florida, markers for other launch sites are visible.

Launch outcomes in Sites



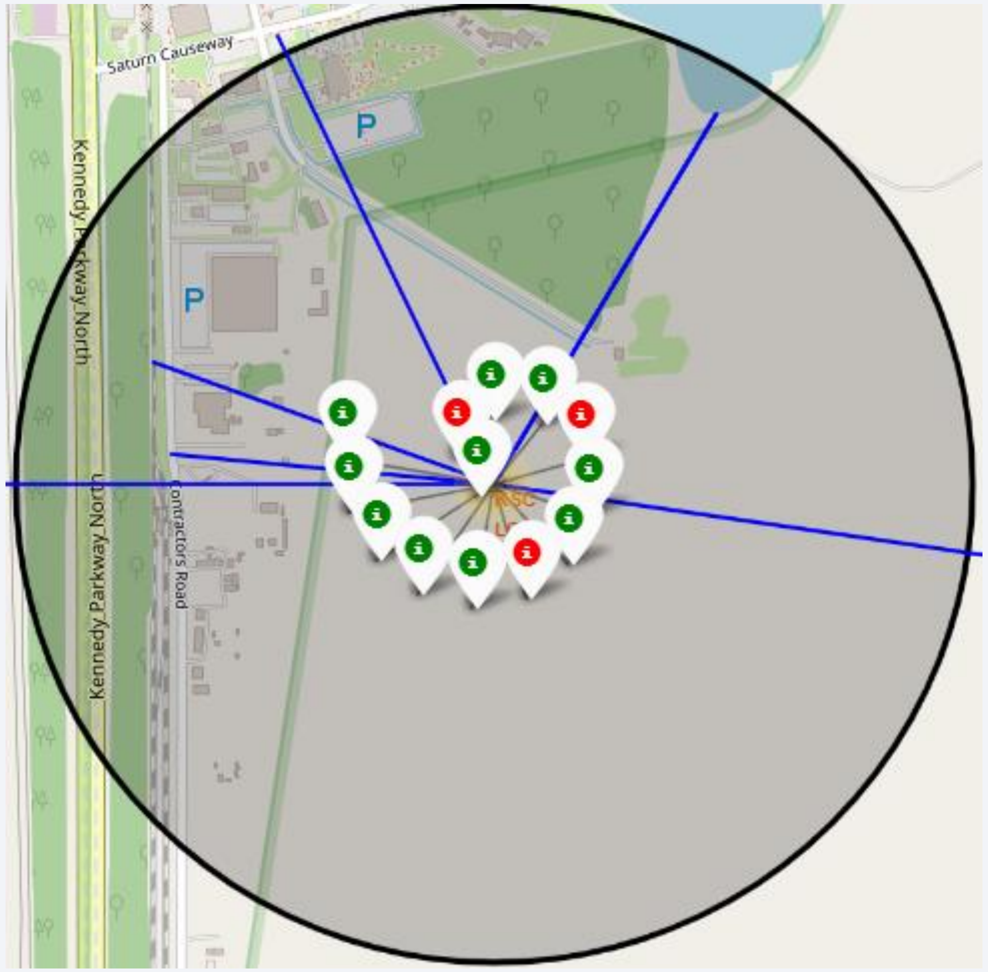




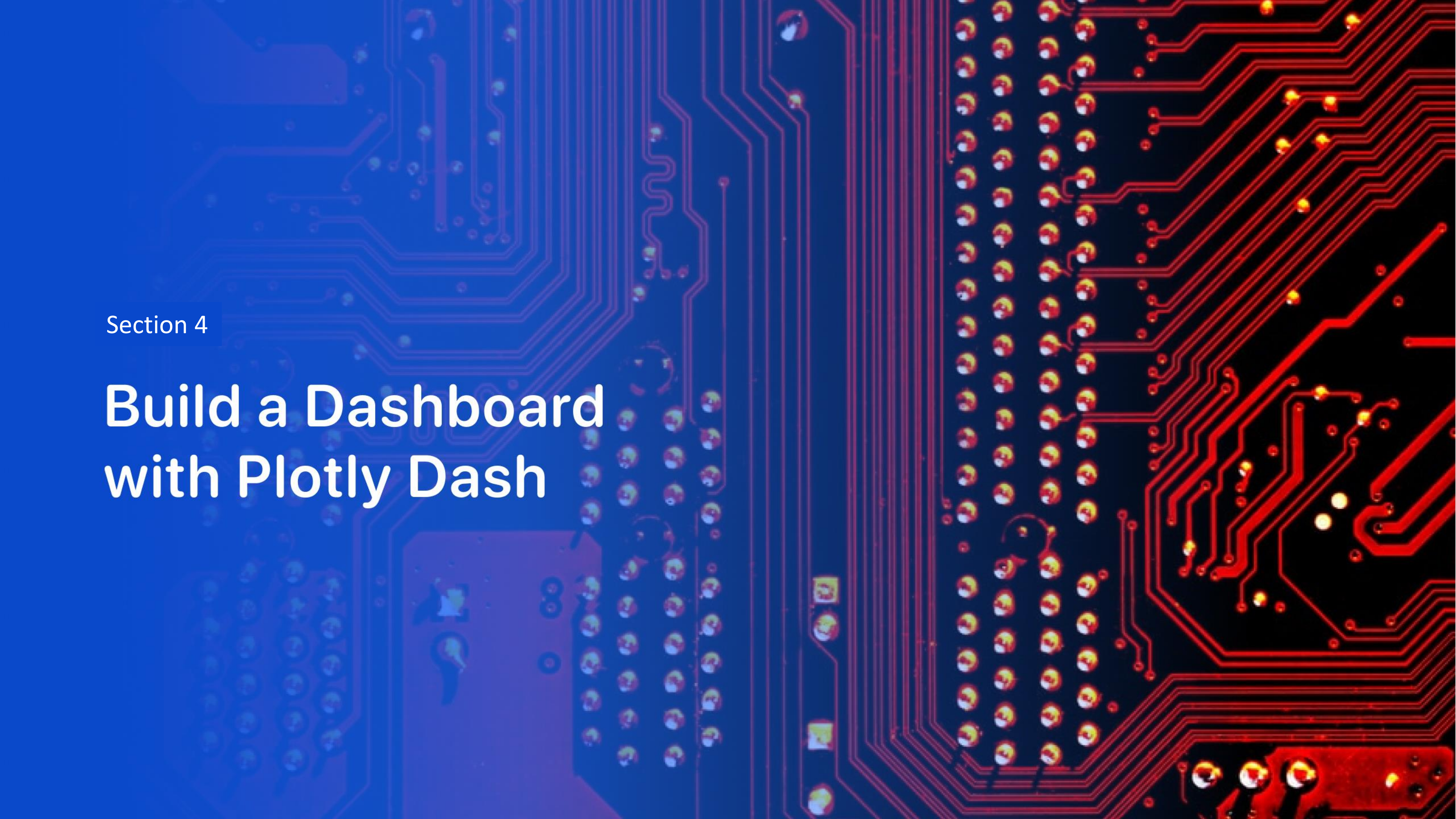


Red Markers indicate failed launch outcomes. Green Markers represent successful launch outcomes. The high number of red markers, especially suggests a significant number of unsuccessful attempts in different launch sites except KSC LC 39A

KSC LC 39A Site Proximity to Locations



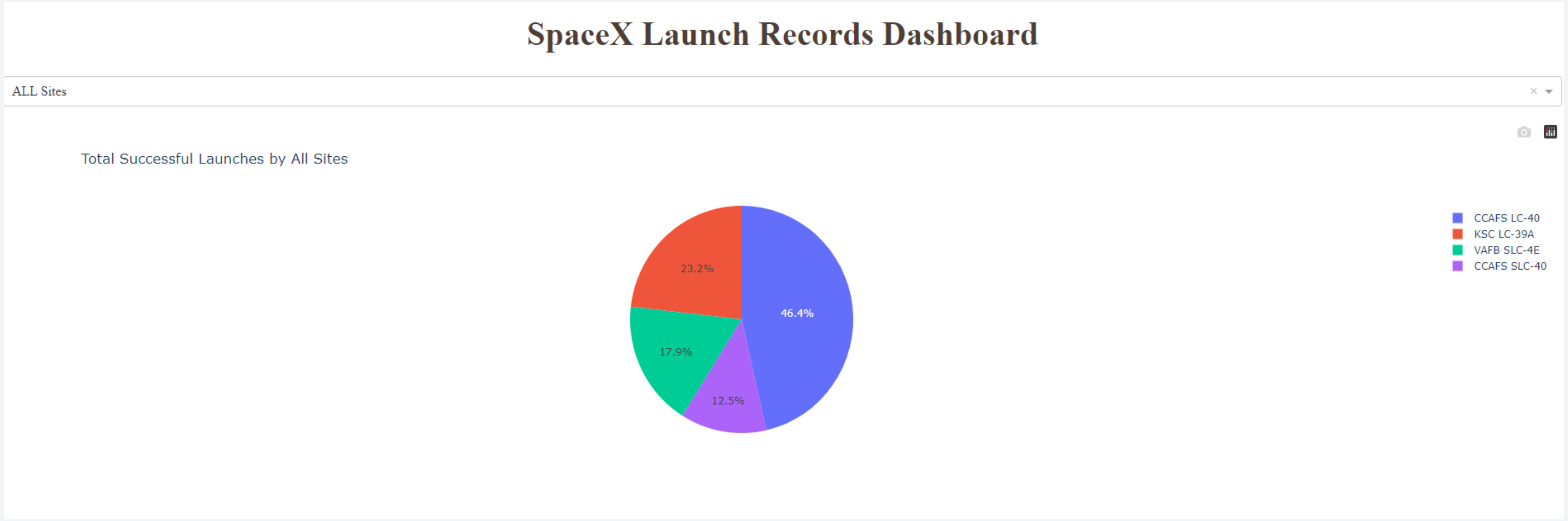
The selected launch site shows its proximities to railway, highway, coastline and city. The lines connecting the launch site to various points represent the distance to those points from the launch site.



Section 4

Build a Dashboard with Plotly Dash

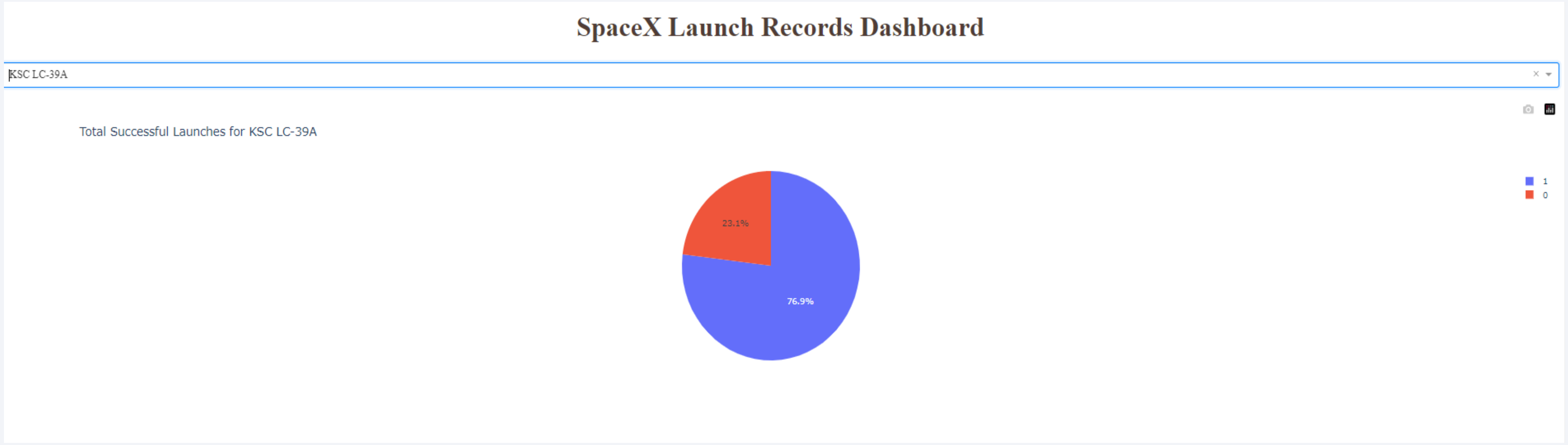
Total Successful Launches by all Sites


SpaceX Launch Records Dashboard

ALL Sites

This dashboard provides insights into the success rate of SpaceX launches across different launch sites, with KSC LC-39A standing out as the most prominent.

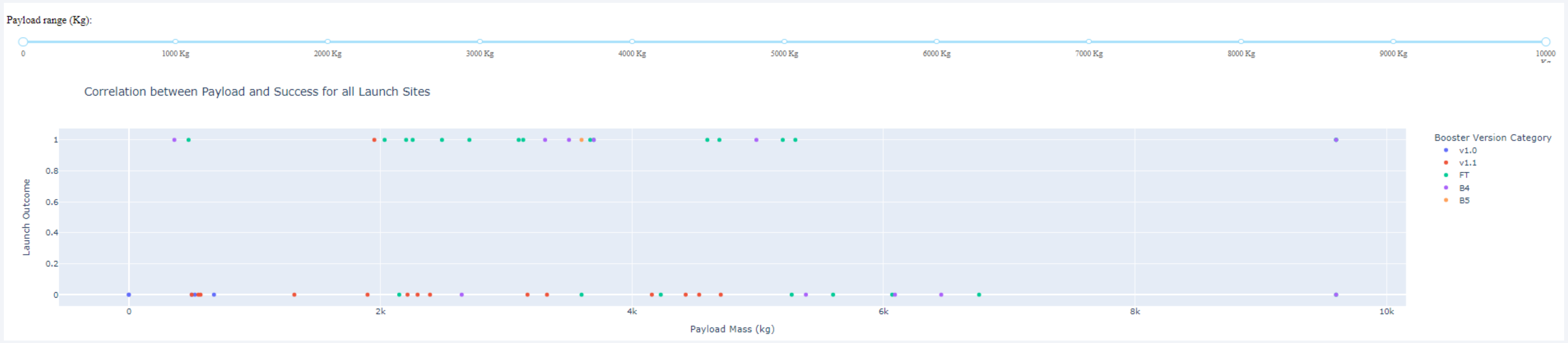
Success Rate of SpaceX Launches at KSC LC-39A Site


SpaceX Launch Records Dashboard

KSC LC-39A

A significant majority (76.9%) of the launches from this site have been successful, while 23.1% were not. The high success rate reflects positively on the operations at the KSC LC-39A launch site.

Payload Mass vs. Launch Success Rate by Booster Version

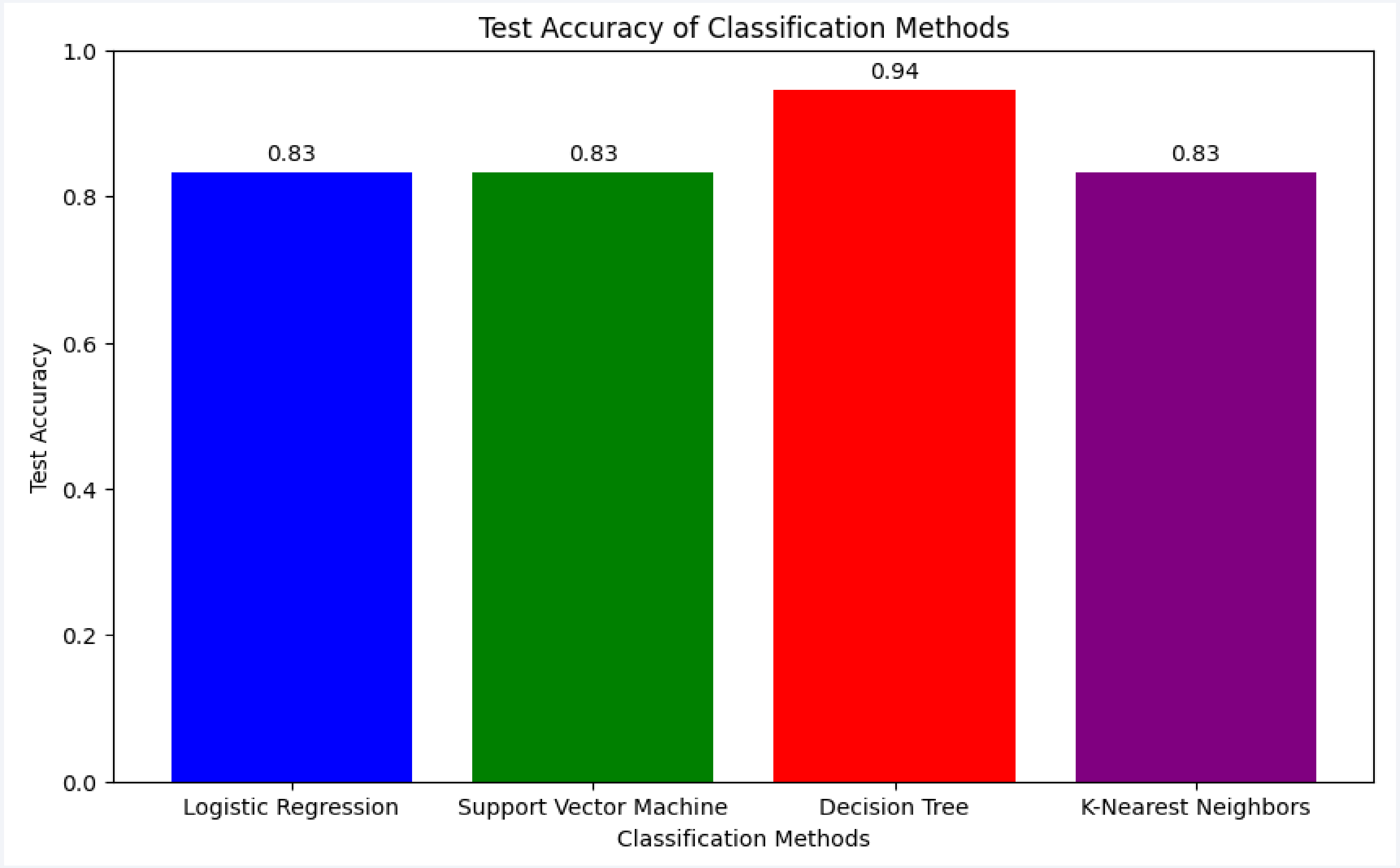


This illustrates a general high success rate across different payload masses and booster versions. While V1.0 seems to be more suited for lighter payloads, B5 boosters demonstrate capabilities to handle both light and very heavy payloads. The B4 booster version appears to be reliable for mid to high payload ranges.

Section 5

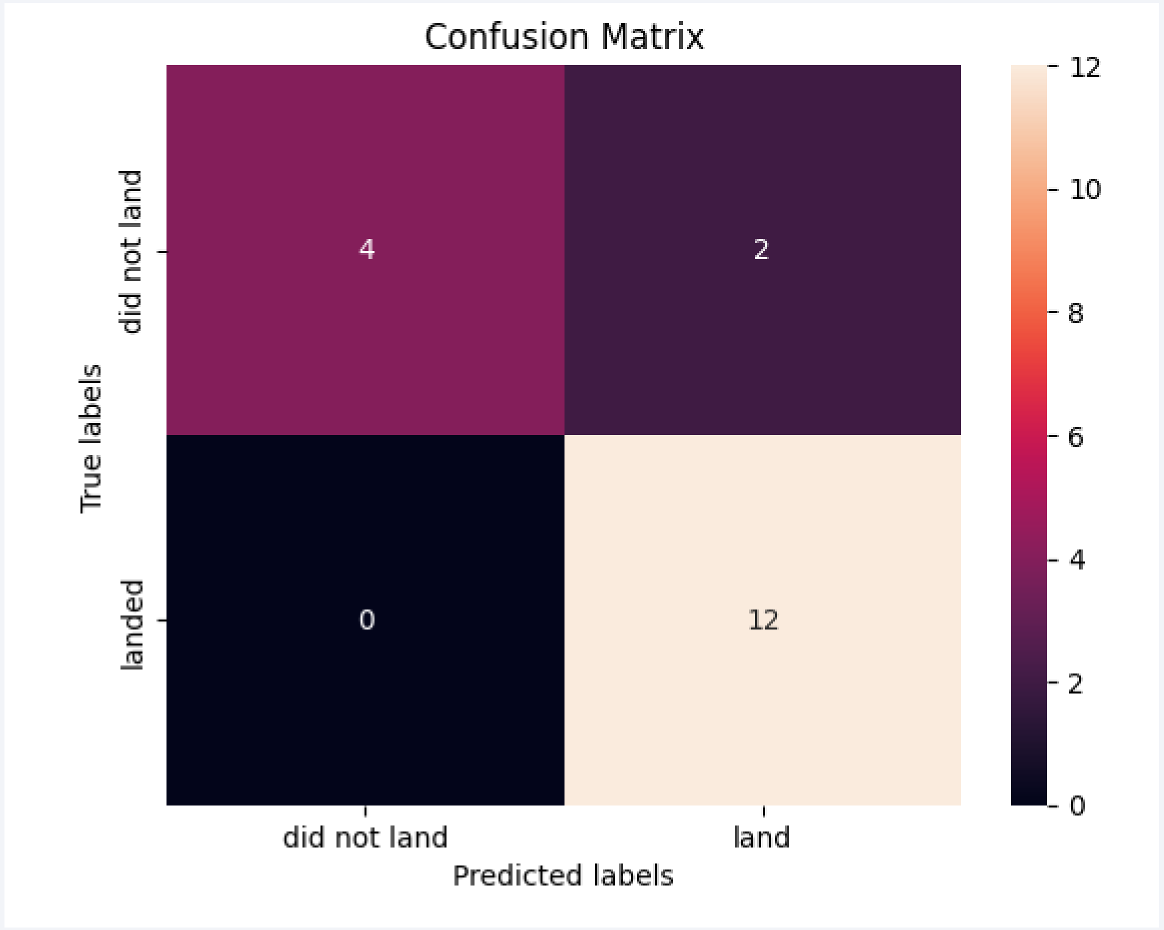
Predictive Analysis (Classification)

Classification Accuracy



The model with the highest classification accuracy is the "Decision Tree" with a test accuracy of 0.94.

Confusion Matrix – Decision Tree



- High True Positives (TP): The model has done a good job predicting the "land" class as there were 12 correct predictions.
- High True Negatives (TN): The model correctly predicted the "did not land" class 4 times.
- Low False Positives (FP): The model made 2 mistakes by predicting "land" when it should have been "did not land".
- No False Negatives (FN): This is ideal since the model never mistakenly predicted "did not land" for instances that actually "landed".
- Overall, this confusion matrix suggests that the classification model performs well, especially for predicting the "landed" class.

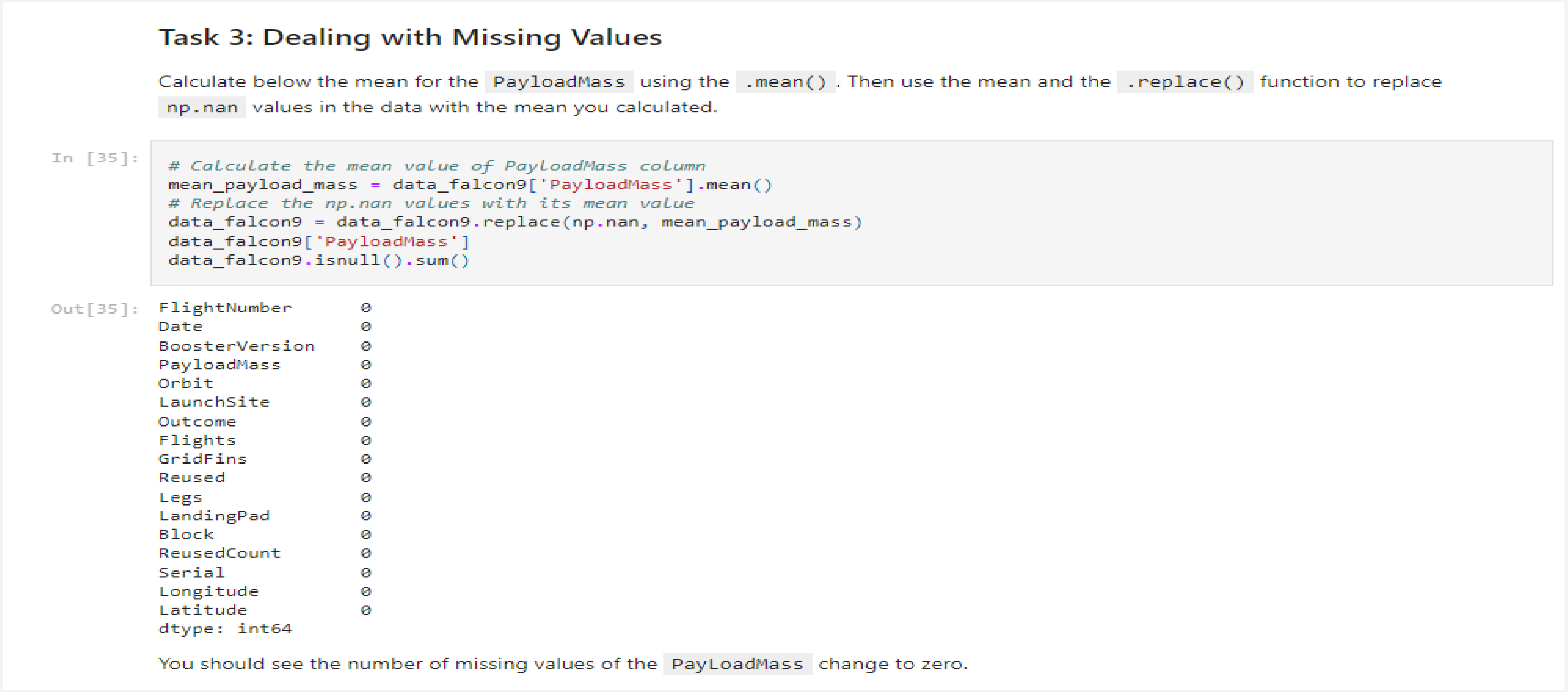
Conclusions

- **EDA - Data Wrangling:**
  - Efficient in cleaning data for further use.
  - Classified the 'class' column into "successful" and "failure" mission launch outcomes.
- **EDA - SQL Queries:**
  - Assessed launch sites.
  - Calculated the count of landing outcomes in the 'class' column.
  - Evaluated various payload masses by flight number and booster version.
- **EDA - Visualizations:**
  - Effectively explored relationships between variables.
  - Analyzed the impact of these variables on the success rate of landing outcomes.

Conclusions

- **Dash Tool:**
  - Offers real-time interactive tools for users.
  - Indicates success rates by all and individual sites.
  - Showcases the correlation with payload mass.
- **Folium Map:**
  - Provides an interactive tool for users.
  - Allows for assessing spatial locations of launch sites and their outcomes.
- **Conclusion:**
  - Among various predictive analyses, the decision tree proved the most effective in predicting successful landing outcomes.
  - It also provided insights into the cost of a launch.

Appendix – Code snippet for dealing with missing values



Task 3: Dealing with Missing Values

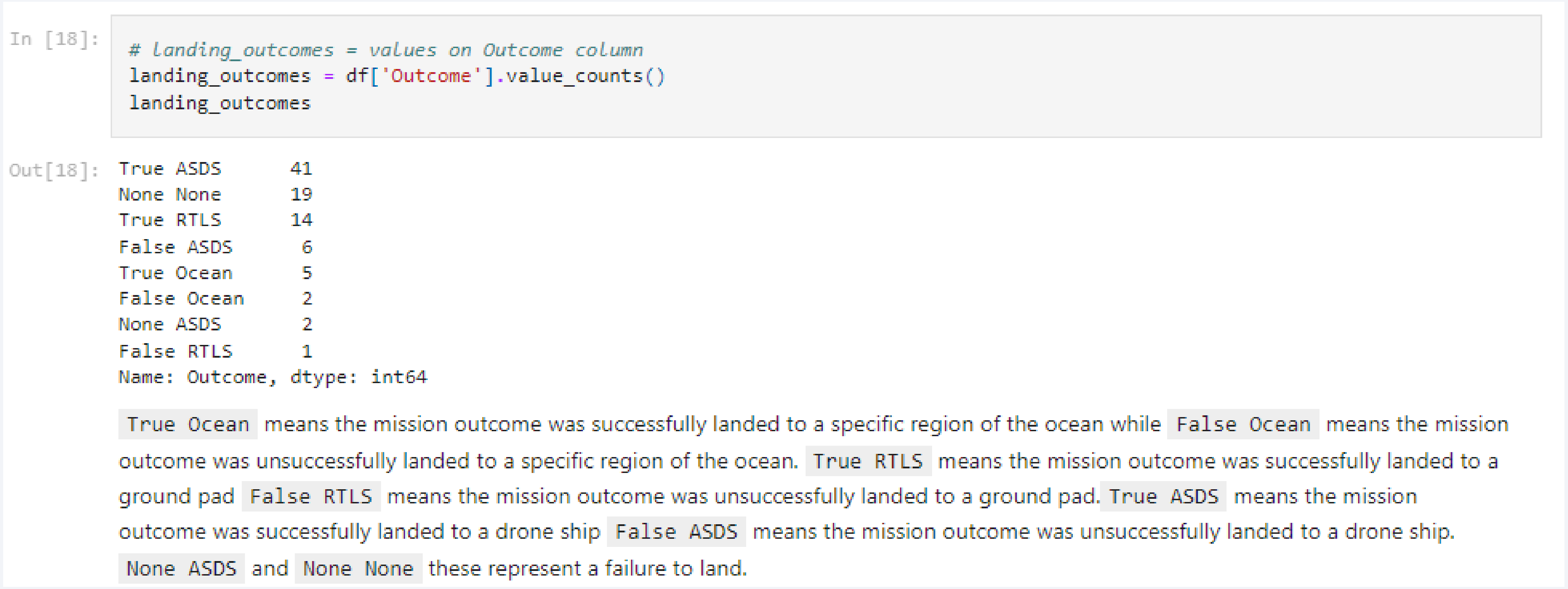
Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [35]:
```
# 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 = data_falcon9.replace(np.nan, mean_payload_mass)
data_falcon9['PayloadMass']
data_falcon9.isnull().sum()
```

Out[35]:
```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad        0
Block             0
ReusedCount       0
Serial            0
Longitude         0
Latitude          0
dtype: int64
```

You should see the number of missing values of the `PayLoadMass` change to zero.

Appendix – Code snippet for counting landing outcomes

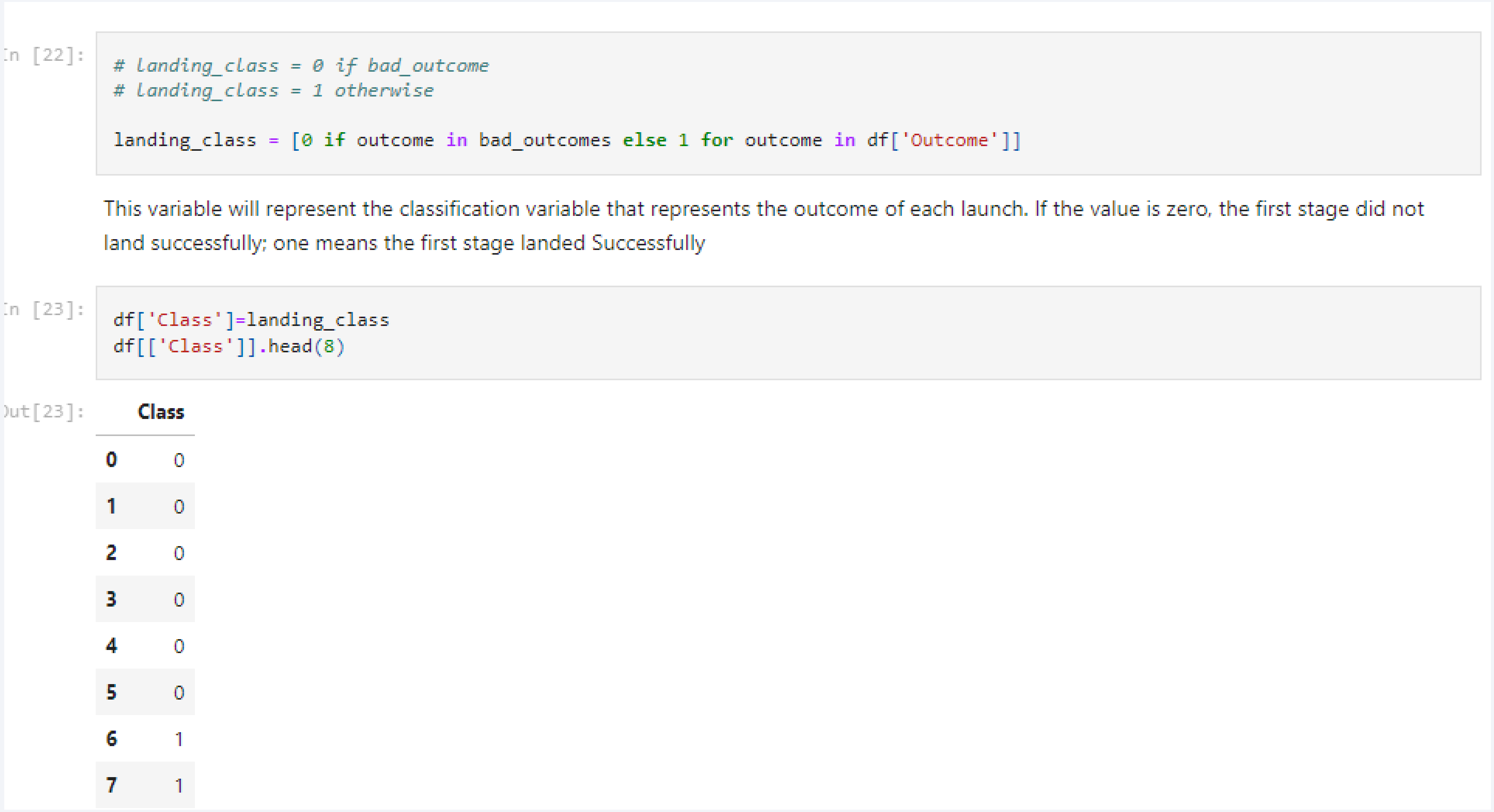


```
In [18]:  # landing_outcomes = values on Outcome column
          landing_outcomes = df['Outcome'].value_counts()
          landing_outcomes
```

```
Out[18]:  True ASDS      41
          None None      19
          True RTLS      14
          False ASDS      6
          True Ocean      5
          False Ocean     2
          None ASDS       2
          False RTLS      1
          Name: Outcome, dtype: int64
```

`True Ocean` means the mission outcome was successfully landed to a specific region of the ocean while `False Ocean` means the mission outcome was unsuccessfully landed to a specific region of the ocean. `True RTLS` means the mission outcome was successfully landed to a ground pad `False RTLS` means the mission outcome was unsuccessfully landed to a ground pad. `True ASDS` means the mission outcome was successfully landed to a drone ship `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

Appendix – Code snippet for ‘class’ column classification



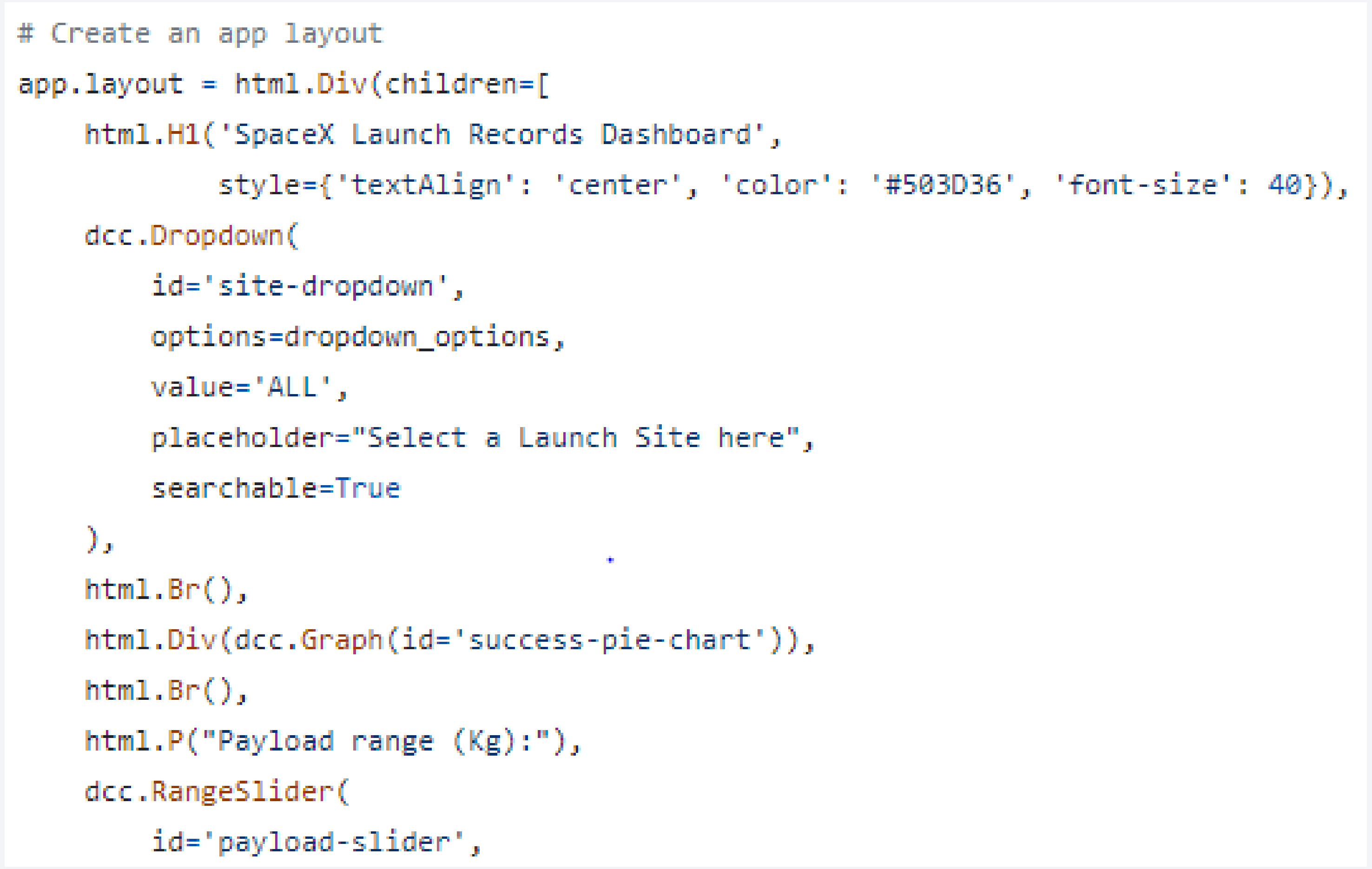
```
In [22]:  # landing_class = 0 if bad_outcome
          # landing_class = 1 otherwise

          landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

```
In [23]:  df['Class']=landing_class
          df[['Class']].head(8)
```

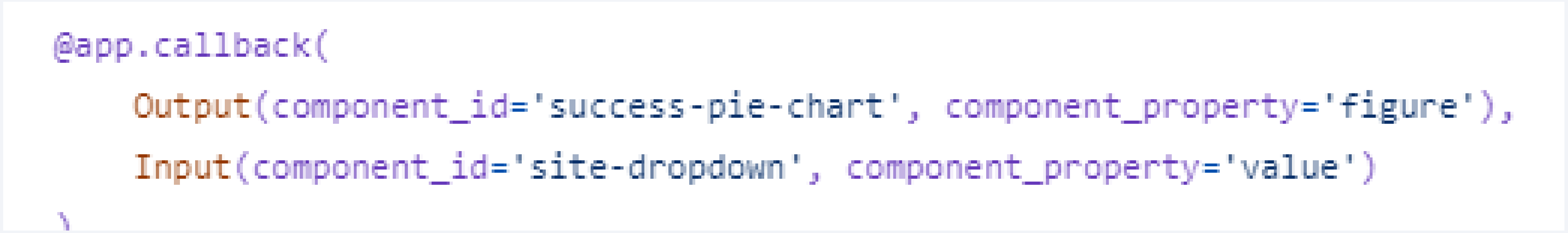
```
Out[23]:
   Class
0      0
1      0
2      0
3      0
4      0
5      0
6      1
7      1
```

Appendix – Code snippet for incorporating markers using Dash

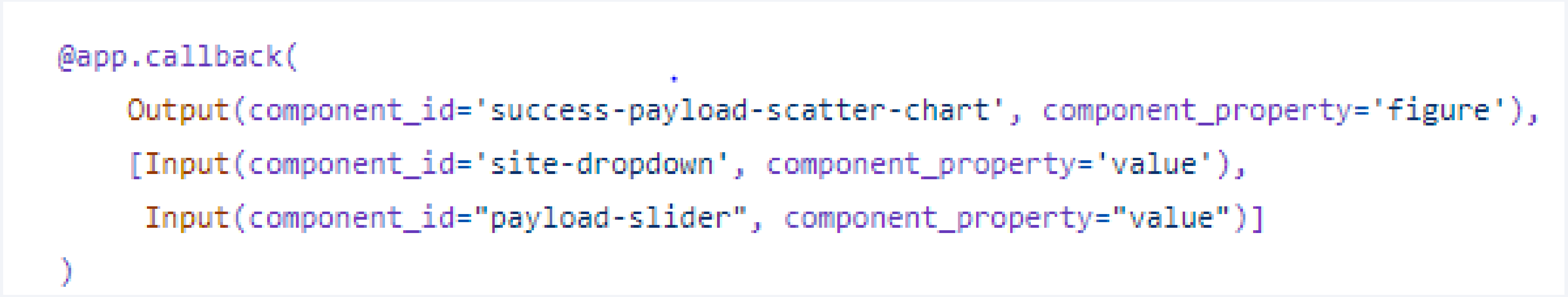


```
# Create an app layout
app.layout = html.Div(children=[
    html.H1('SpaceX Launch Records Dashboard',
            style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),
    dcc.Dropdown(
        id='site-dropdown',
        options=dropdown_options,
        value='ALL',
        placeholder="Select a Launch Site here",
        searchable=True
    ),
    html.Br(),
    html.Div(dcc.Graph(id='success-pie-chart')),
    html.Br(),
    html.P("Payload range (Kg):"),
    dcc.RangeSlider(
        id='payload-slider',
```

Appendix – Code snippet for incorporating charts and plots using Dash

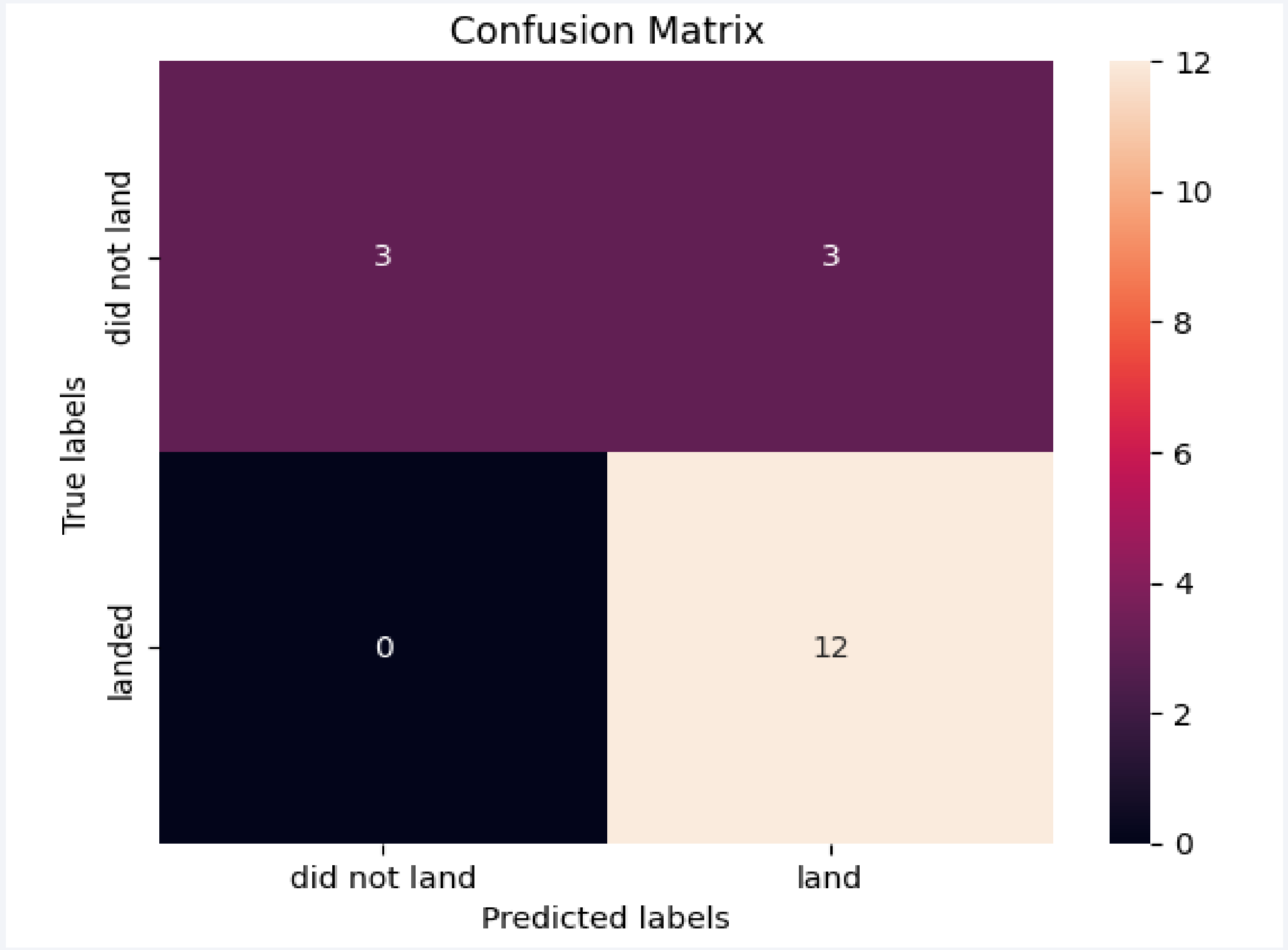


```
@app.callback(
    Output(component_id='success-pie-chart', component_property='figure'),
    Input(component_id='site-dropdown', component_property='value')
)
```

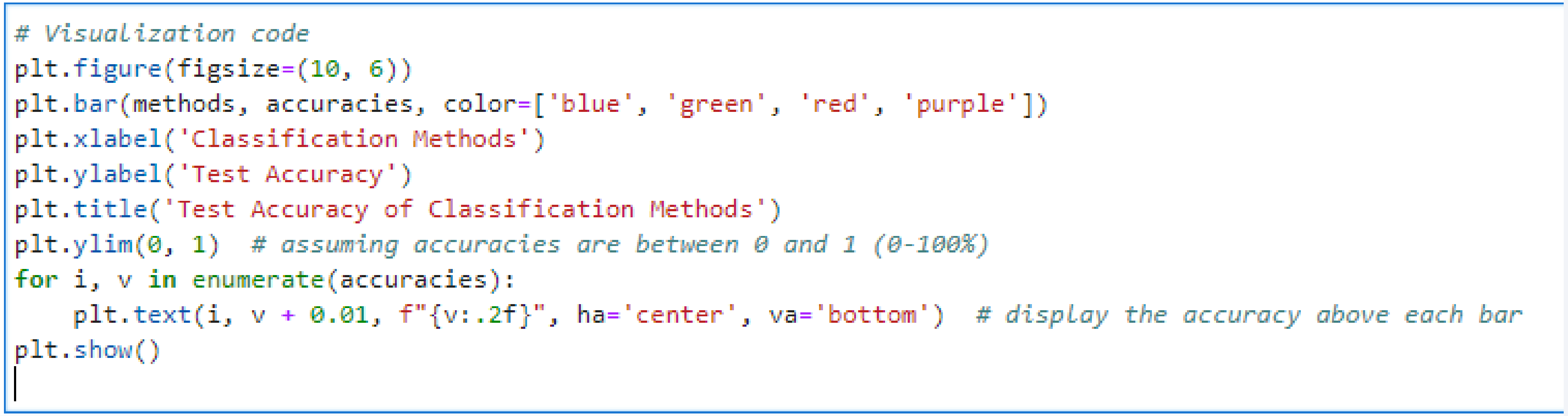


```
@app.callback(
    Output(component_id='success-payload-scatter-chart', component_property='figure'),
    [Input(component_id='site-dropdown', component_property='value'),
     Input(component_id="payload-slider", component_property="value")]
)
```

Appendix – Confusion matrix for logistic regression, SVM and KNN



Appendix – Code snippet for plotting the test accuracy of classification methods



```
# Visualization code
plt.figure(figsize=(10, 6))
plt.bar(methods, accuracies, color=['blue', 'green', 'red', 'purple'])
plt.xlabel('Classification Methods')
plt.ylabel('Test Accuracy')
plt.title('Test Accuracy of Classification Methods')
plt.ylim(0, 1)  # assuming accuracies are between 0 and 1 (0-100%)
for i, v in enumerate(accuracies):
    plt.text(i, v + 0.01, f"{v:.2f}", ha='center', va='bottom')  # display the accuracy above each bar
plt.show()
```

Thank you!