IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Madeleine Tourelle>
<12/04/2024>

Outline

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

Executive Summary

- Summary of methodologies

Data Collection: The notebooks use multiple methods to gather data, including web scraping, API collection, and SQL queries, as well as historical data on SpaceX launches.

Data Wrangling: The focus is on cleaning and transforming the raw data into a usable format for modeling, such as converting outcomes into binary labels for prediction and preparing data for machine learning.

Exploratory Data Analysis (EDA): Several notebooks involve in-depth analysis of the data using various visualization techniques, including interactive maps with Folium and other visualization tools like matplotlib and seaborn.

Model Building: Machine learning models (e.g., classification models for predicting landing success) are built using the prepared data. The models are evaluated based on performance metrics like accuracy, precision, and recall.

- Summary of all results

Prediction Accuracy: The notebooks aim to predict Falcon 9 landing success using classification models, with performance metrics evaluated using methods like cross-validation and hyperparameter tuning.

Visual Insights: Interactive visualizations, such as maps and charts, are used to explore the data and derive actionable insights. This helps in understanding patterns in launch site locations, payloads, and other variables affecting landing success.

Data Patterns: The analysis shows correlations between various factors (e.g., launch site, payload, weather) and the success of rocket landings, which can guide future predictions.

Introduction Part 1

- Project background and context

The project revolves around predicting the **success of Falcon 9 rocket landings**, which is a crucial aspect of SpaceX's reusability model. SpaceX's Falcon 9 rocket launches have garnered significant attention because they can reuse the first stage of the rocket, a feature that drastically reduces the cost of space missions. The typical cost for a Falcon 9 launch is about **$62 million**, while other providers charge upwards of **$165 million** for similar missions. This significant cost reduction is due to SpaceX's ability to land and reuse the first stage, an innovation that distinguishes them from traditional space launch providers.

In this project, the aim is to predict the outcome of the first-stage landing based on historical launch data. Successful landing predictions allow SpaceX to optimize its resources and improve its operational costs. Additionally, being able to predict the success of the landing provides valuable insights for competitors bidding against SpaceX for launch services.

The context of this project involves the use of **machine learning** and **predictive analytics** to build a model that can predict the success or failure of the Falcon 9 first stage landing, using various historical data points. These data points include:

- Launch conditions (e.g., weather, payload mass, etc.)
- Type of landing site (e.g., drone ship, ground pad)
- Previous landing outcomes (successful or failed)

The project involves working with the **SpaceX dataset**, which contains records for each launch, including features such as mission type, launch site, success rate, and landing type.

Introduction: Part 2-

- Problems that need answers:

1. **Can we predict if the Falcon 9 first stage will land successfully?**

   The primary goal is to build a predictive model that can take various factors into account (e.g., launch site, payload, weather) and predict whether the first stage of the Falcon 9 rocket will land successfully, which is critical for SpaceX's cost-saving strategy.

2. **What factors contribute to the success or failure of the Falcon 9 landing?**

   By performing Exploratory Data Analysis (EDA), the project investigates which features (such as launch site, payload mass, or weather conditions) are the most influential in determining the landing outcome. This could involve analyzing whether certain landing sites (e.g., drone ships, ground pads) are more likely to result in successful landings or if certain mission types are associated with higher success rates.

3. **How can we optimize landing predictions to improve cost estimates for SpaceX?**

   By developing an accurate model, SpaceX can improve cost forecasting by better predicting whether a landing will be successful, allowing for better planning and resource allocation. This will have a direct impact on the overall cost per launch and will aid in competitive bidding for future space missions.

4. **Can we improve the landing prediction accuracy through better feature engineering and model tuning?**

   The project investigates whether additional features, data transformations, or model tuning can increase the accuracy of the prediction model.This includes exploring various classification algorithms (e.g., logistic regression, decision trees, support vector machines) and using techniques such as hyperparameter optimization and cross-validation.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:

  The notebooks use multiple methods to gather data, including web scraping, API collection, and SQL queries, as well as historical data on SpaceX launches.

- Perform data wrangling

  The focus is on cleaning and transforming the raw data into a usable format for modeling, such as converting outcomes into binary labels for prediction and preparing data for machine learning.

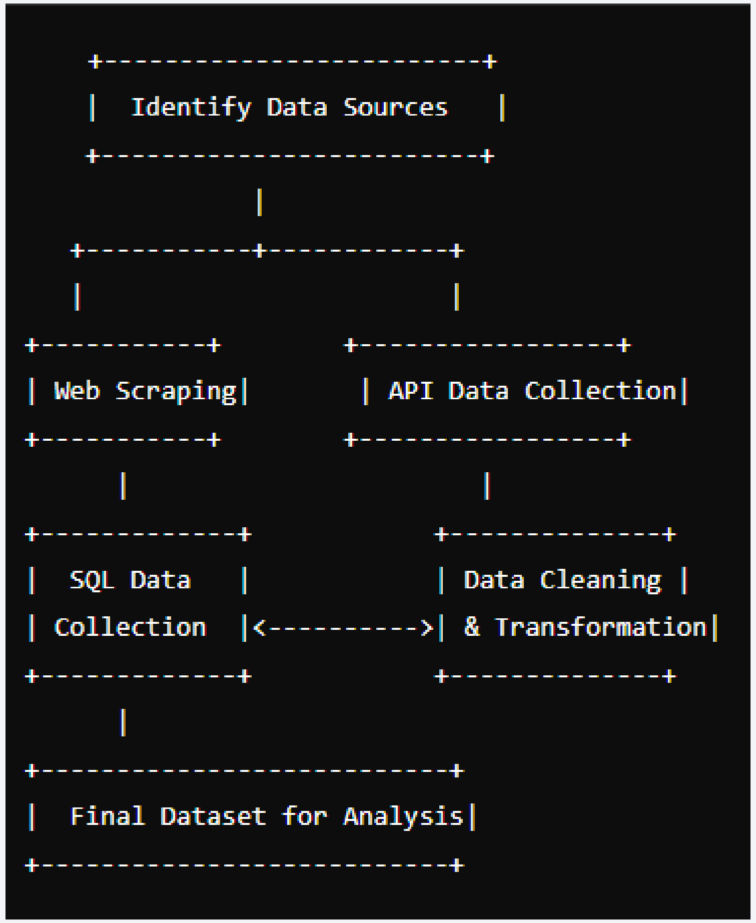
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

  Machine learning models (e.g., classification models for predicting landing success) are built using the prepared data. The models are evaluated based on performance metrics like accuracy, precision, and recall.

Data Collection

- Describe how data sets were collected.

  The data collection process for predicting the success of the Falcon 9 first stage landing involves gathering multiple sources of information related to the historical launches. The datasets are collected from various sources, including **APIs, web scraping, and SQL databases**, and then cleaned and transformed for use in predictive models.



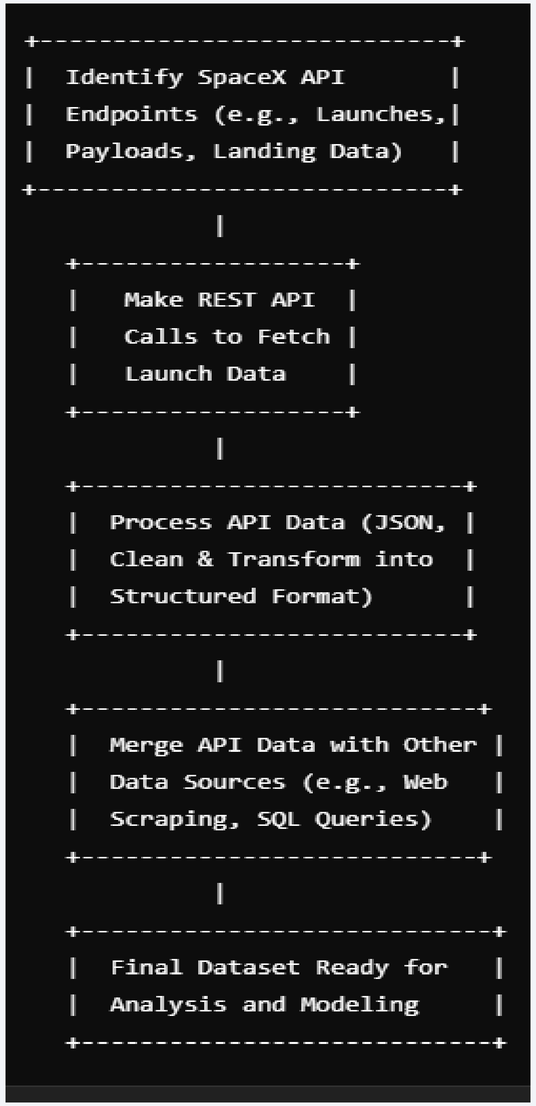
```
    +-------------------------+
    |  Identify Data Sources   |
    +-------------------------+
               |
   +-----------+------------+
   |                        |
+-----------+       +----------------+
| Web Scraping|      | API Data Collection|
+-----------+       +----------------+
      |                       |
+-------------+          +-------------+
|  SQL Data   |          | Data Cleaning |
| Collection  |<--------->| & Transformation|
+-------------+          +-------------+
      |
+---------------------------+
|  Final Dataset for Analysis|
+---------------------------+
```

Data Collection – SpaceX API

- The data collection process through SpaceX REST API calls focuses on gathering structured data directly from the SpaceX API for Falcon 9 rocket launches. This API provides detailed information such as mission outcomes, payload details, landing information, and launch configurations, which are critical for building a predictive model for landing success.

  1. API Endpoints Identification: Identify the relevant endpoints from SpaceX's API to gather historical launch data, including launch details, payloads, and landing outcomes.
  2. API Requests: Make REST API calls using Python libraries like requests to fetch data in JSON format.
  3. Data Processing: Process and clean the API data, transforming it into a structured format for use in machine learning models.
  4. Data Integration: Combine the API data with additional datasets (e.g., web scraping data or SQL data) to create a comprehensive dataset for analysis.

- **CAPSTONE PROJECT URL:** https://github.com/mst141umiami/CAPSTONE.git
- **SpaceX API calls notebook file name:** jupyter-labs-spacex-data-collection-api.ipynb



```
+----------------------------+
|  Identify SpaceX API       |
|  Endpoints (e.g., Launches,|
|  Payloads, Landing Data)   |
+----------------------------+
              |
   +------------------+
   |   Make REST API  |
   |   Calls to Fetch |
   |   Launch Data    |
   +------------------+
              |
   +--------------------------+
   |  Process API Data (JSON, |
   |  Clean & Transform into  |
   |  Structured Format)      |
   +--------------------------+
              |
   +----------------------------+
   |  Merge API Data with Other |
   |  Data Sources (e.g., Web   |
   |  Scraping, SQL Queries)    |
   +----------------------------+
              |
   +----------------------------+
   |  Final Dataset Ready for   |
   |  Analysis and Modeling     |
   +----------------------------+
```

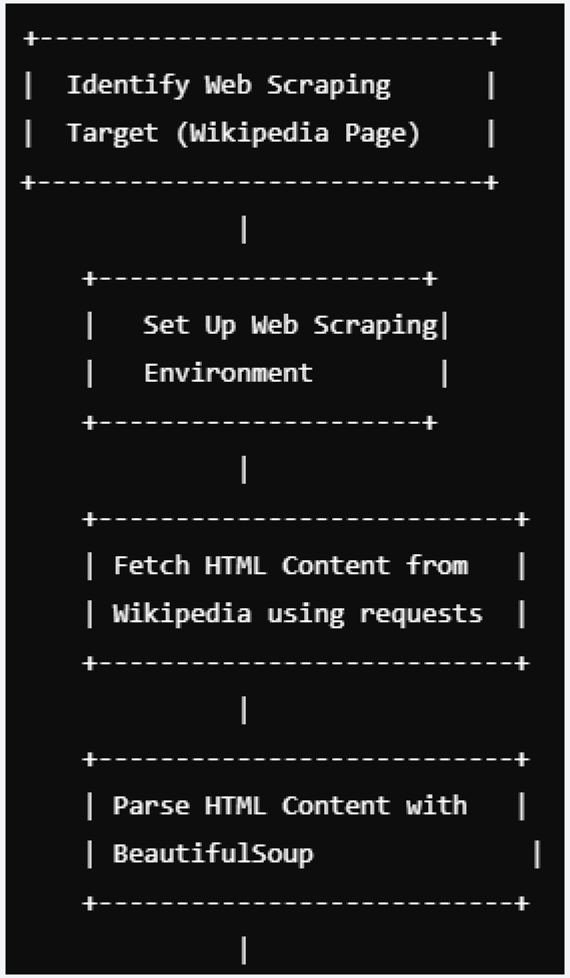
Data Collection - Scraping

- In this project, **web scraping** is employed to collect historical launch records from a Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches".

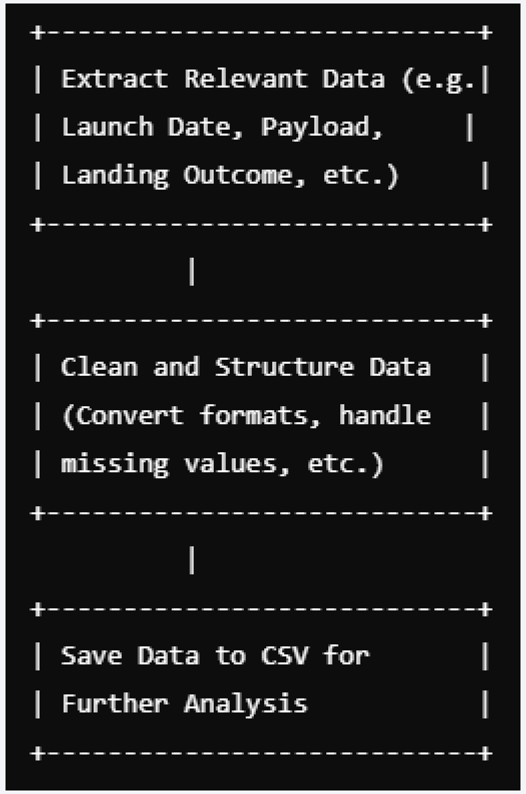
  1. The web scraping process starts by identifying the target webpage (Wikipedia), setting up the required Pythonlibraries (BeautifulSoup, requests), and making GET requests to fetch the HTML content.
  2. The HTML content is parsed using BeautifulSoup, and specific data such as launch date, payload, landing outcome, and launch site is extracted from the HTML tables.
  3. The extracted data is cleaned, structured, and stored in a DataFrame for further analysis and prediction tasks.
  4. Finally, the cleaned data is saved to a CSV file for later use.

- **CAPSTONE PROJECT URL:** https://github.com/mst141umiami/CAPSTONE.git

- **SpaceX Scraping notebook file name:** jupyter-labs-webscraping.ipynb



```
+------------------------------+
|  Identify Web Scraping       |
|  Target (Wikipedia Page)     |
+------------------------------+
              |
    +---------------------+
    |   Set Up Web Scraping|
    |   Environment        |
    +---------------------+
              |
    +---------------------------+
    | Fetch HTML Content from   |
    | Wikipedia using requests  |
    +---------------------------+
              |
    +---------------------------+
    | Parse HTML Content with   |
    | BeautifulSoup              |
    +---------------------------+
              |
```



```
+-----------------------------+
| Extract Relevant Data (e.g.|
| Launch Date, Payload,     |
| Landing Outcome, etc.)      |
+-----------------------------+
          |
+-----------------------------+
| Clean and Structure Data    |
| (Convert formats, handle    |
| missing values, etc.)       |
+-----------------------------+
          |
+-----------------------------+
| Save Data to CSV for        |
| Further Analysis            |
+-----------------------------+
```

Data Wrangling

- **Data wrangling** is the process of cleaning, transforming, and structuring raw data into a usable format for analysis. In the context of predicting the success of Falcon 9 first stage landings, the data wrangling process includes handling missing values, encoding categorical variables, normalizing numerical data, and preparing the dataset for predictive modeling.

  1. The data wrangling process begins by collecting data from various sources, such as web scraping, API calls, and SQL queries. Once the data is collected, it is merged into a single dataset.

  2. Missing values are handled by either imputing or removing them.

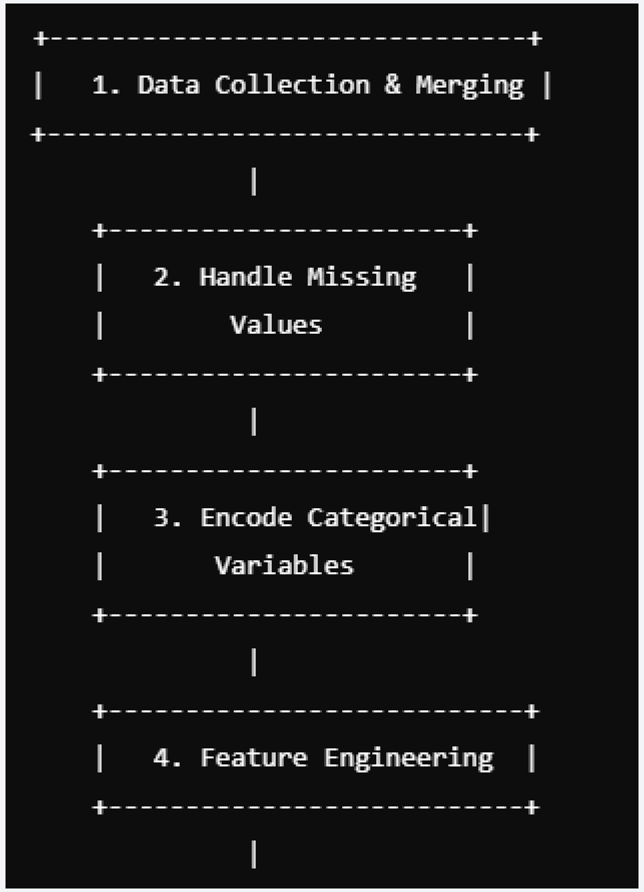
  3. Categorical variables are encoded using methods such as label encoding or one-hot encoding to make them usable by machine learning models.

  4. Feature engineering is done to create new, more informative features, and numerical data is normalized or standardized t o improve model performance.

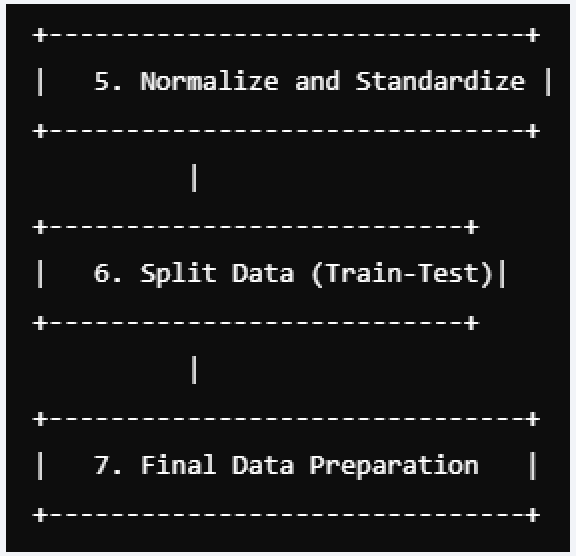
  5. Finally, the data is split into training and testing datasets to evaluate the model's performance.

- **CAPSTONE PROJECT URL:** https://github.com/mst141umiami/CAPSTONE.git

- **SpaceX Wrangling notebook file name:** labs-jupyter-spacex-Data wrangling.ipynb



```
+-------------------------------+
|   1. Data Collection & Merging |
+-------------------------------+
              |
    +----------------------+
    |   2. Handle Missing   |
    |        Values         |
    +----------------------+
              |
    +----------------------+
    |   3. Encode Categorical|
    |       Variables       |
    +----------------------+
              |
    +------------------------+
    |   4. Feature Engineering  |
    +------------------------+
              |
```



```
+-------------------------------+
|   5. Normalize and Standardize |
+-------------------------------+
          |
+--------------------------+
|   6. Split Data (Train-Test)|
+--------------------------+
          |
+-------------------------------+
|   7. Final Data Preparation    |
+-------------------------------+
```

EDA with Data Visualization: Part 1

1. Distribution of Launch Outcomes (Landing Success/Failure)

Chart Type: Bar Chart

Purpose: This chart helps in understanding the distribution of the landing outcome (successful vs. unsuccessful landings).

Why Used: The distribution of the target variable (landing outcome) is important to assess whether the dataset is balanced or if any class imbalances exist.

2. Distribution of Launch Date (Time Series)

Chart Type: Line Plot or Histogram

Purpose: To explore how the launch dates are distributed over time. This allows us to see trends in the frequency of launches and if there's any seasonal variation.

Why Used: Understanding how launch frequency changes over time can highlight trends or patterns and help determine if time-based features (like year, month, or season) influence landing success.

3. Payload Weight vs. Landing Outcome

Chart Type: Box Plot or Violin Plot

Purpose: To examine the relationship between payload weight and landing success/failure.

Why Used: Payload weight is likely a key factor affecting the landing outcome. Visualizing the distribution of payload weights for successful and failed landings helps understand if heavier payloads are correlated with landing failures.

4. Launch Site vs. Landing Outcome

Chart Type: Count Plot or Bar Chart

Purpose: To visualize how different launch sites affect landing success. This chart shows the number of successful and unsuccessful landings at each launch site.

Why Used: Launch site location can influence landing success. For example, some sites might be closer to water or have better weather conditions, which could affect landing success.

EDA with Data Visualization: Part 2

5. Correlation Between Features

Chart Type: Heatmap

Purpose: To explore correlations between numerical features like payload weight, rocket configuration, and other variables.

Why Used: Identifying highly correlated features helps in feature selection and can improve the performance of machine learning models by reducing multicollinearity.

6. Landing Outcome by Rocket Type

Chart Type: Bar Chart or Count Plot

Purpose: To visualize the success rate of landings based on rocket type (e.g., Falcon 9 vs. Falcon Heavy).

Why Used: Different rocket types might have varying landing success rates. This visualization helps identify any trends or differences in the landing outcomes based on rocket type.

7. Weather Conditions and Landing Outcome

Chart Type: Scatter Plot or Box Plot

Purpose: To visualize the relationship between weather conditions (such as wind speed, temperature) and landing success.

Why Used: Weather conditions are likely to impact the success of the rocket landing. This visualization helps identify if extreme weather conditions correlate with landing failures.

8. Landing Outcome by Mission Type

Chart Type: Count Plot

Purpose: To investigate the relationship between mission type (e.g., satellite launch, cargo mission) and landing success.

Why Used: Some mission types might be more likely to succeed in landing based on the nature of the payload or the mission's requirements.

- **CAPSTONE PROJECT URL:** https://github.com/mst141umiami/CAPSTONE.git
- **SpaceX EDA notebook file name:**
  edadataviz2.ipynb

EDA with SQL: Part 1

- Loaded Data into SQL Database
  Purpose: Import the SpaceX dataset into a SQL database to enable SQL querying.
  Command: %sql SELECT * FROM spacex_launches;

- Query 1: Retrieving Launch Data:
  Purpose: Get all the launch records, including details about the launch date, payload, and launch site.
  Command: %sql SELECT * FROM spacex_launches;

- Query 2: Filtering Launches by Outcome (Success vs. Failure)
  Purpose: Extract records where the landing was successful or unsuccessful.
  Command: %sql SELECT launch_date, landing_outcome FROM spacex_launches WHERE landing_outcome = 'True';

- Query 3: Count of Launch Successes by Launch Site:
  Purpose: Count the number of successful launches for each launch site.
  Command: %sql SELECT launch_site, COUNT(*) AS success_count FROM spacex_launches WHERE landing_outcome = 'True' GROUP BY launch_site;

- Query 4: Average Payload Weight by Launch Outcome
  Purpose: Get the average payload weight for successful and failed landings.
  Command: %sql SELECT landing_outcome, AVG(payload_weight) AS avg_payload_weight FROM spacex_launches GROUP BY landing_outcome;

- Query 5: Retrieving Launch Dates for Specific Rocket Types
  Purpose: Filter launches by specific rocket types, such as Falcon 9 or Falcon Heavy.
  Command: %sql SELECT launch_date, rocket_type FROM spacex_launches WHERE rocket_type = 'Falcon 9';

EDA with SQL: Part 2

- Query 6: Landing Success Rate by Rocket Type
  Purpose: Calculate the success rate of landings for Falcon 9 versus Falcon Heavy.
  Command: %sql SELECT rocket_type, AVG(CASE WHEN landing_outcome = 'True' THEN 1 ELSE 0 END) AS landing_success_rateFROM spacex_launchesGROUP BY rocket_type;

- Query 7: Landing Success Rate Over Time
  Purpose: Calculate the trend of landing successes over time (e.g., by year).
  Command: %sql SELECT YEAR(launch_date) AS launch_year, AVG(CASE WHEN landing_outcome = 'True' THEN 1 ELSE 0 END) AS landing_success_rateFROM spacex_launchesGROUP BY launch_yearORDER BY launch_year;

- Query 8: Launch Sites with Most Successful Landings
  Purpose: Find the launch sites with the most successful landings.
  Command: %sql SELECT launch_site, COUNT(*) AS successful_landingsFROM spacex_launchesWHERE landing_outcome = 'True'GROUP BY launch_siteORDER BY successful_landings DESC;

- Query 9: Retrieving Launches with Specific Payload Types
  Purpose: Filter the data to get specific payload types (e.g., communication satellites).
  Command: %sql SELECT * FROM spacex_launches WHERE payload_type = 'Communications Satellite';

- **CAPSTONE PROJECT URL:** https://github.com/mst141umiami/CAPSTONE.git

- **EDA SQL notebook file name:**
  jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Markers helped identify key locations (e.g., **launch sites**) on the map.
- Circles visually represented **landing zones**, making it easy to see where landings occurred or could occur.
- Lines were used to represent **rocket flight paths**, helping visualize the trajectories from launch sites to landing zones.
- Popups provided additional detailed **information** when interacting with map objects, offering a better understanding of each data point.
- Heatmaps helped show **regions with high landing success rates**, identifying patterns.
- Custom Icons provided a way to visually **distinguish between different types of launches** or missions.

- **CAPSTONE PROJECT URL:** https://github.com/mst141umiami/CAPSTONE.git
- **EDA SQL notebook file name:** lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- **Dropdown for Launch Site Selection:** A dropdown menu allows the user to filter data based on specific launch sites or view all launch sites at once.
- **Pie Chart:** Displays the total count of successful launches by site. If a specific site is selected from the dropdown, it shows the proportion of successful (class = 1) and failed (class = 0) launches for that site.
- **Range Slider for Payload Mass:** Allows users to filter launches by payload mass (kg), using adjustable minimum and maximum bounds.
- **Scatter Plot:** Shows the correlation between payload mass and launch success (class), with data points color-coded by booster version category. If a specific site is selected, the scatter plot reflects filtered data for that site.

- **CAPSTONE PROJECT URL:** https://github.com/mst141umiami/CAPSTONE.git
- **Plotly Dash file name:**
  mycode3.py

Predictive Analysis (Classification)

- **Prediction Analysis:** The goal was to predict whether the first stage of the Falcon 9 rocket will land successfully based on various features such as payload, launch site, and other mission parameters.

  1. Data Preparation: Preprocessing the data, including handling missing values, encoding categorical features, and splitting the data into training and testing sets.

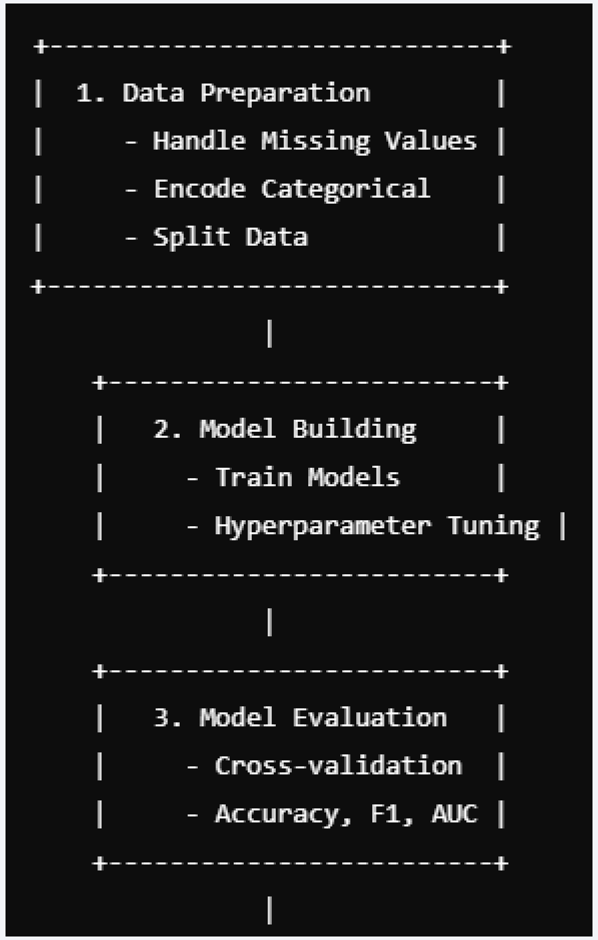
  2. Model Building: Training several classification models (e.g., logistic regression, decision tree, SVM, KNN) and tuning their hyperparameters.

  3. Model Evaluation: Evaluating each model using metrics like accuracy, confusion matrix, and ROC/AUC.

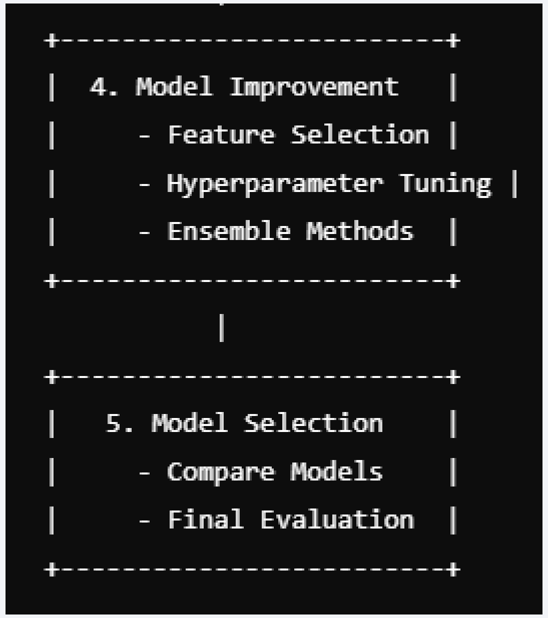
  4. Model Improvement: Enhancing the models through feature selection, hyperparameter optimization, and ensemble methods.

  5. Model Selection: Selecting the best-performing model based on test set accuracy and other evaluation metrics.

- **CAPSTONE PROJECT URL:** https://github.com/mst141umiami/CAPSTONE.git
- **Prediction Analysis notebook file name:** labs-jupyter-spacex-Data wrangling.ipynb



```
+-----------------------------+
|  1. Data Preparation        |
|      - Handle Missing Values |
|      - Encode Categorical   |
|      - Split Data           |
+-----------------------------+
               |
    +-------------------------+
    |   2. Model Building     |
    |      - Train Models     |
    |      - Hyperparameter Tuning |
    +-------------------------+
               |
    +-------------------------+
    |   3. Model Evaluation   |
    |      - Cross-validation |
    |      - Accuracy, F1, AUC |
    +-------------------------+
               |
```



```
+-------------------------+
|  4. Model Improvement   |
|      - Feature Selection |
|      - Hyperparameter Tuning |
|      - Ensemble Methods  |
+-------------------------+
           |
+-------------------------+
|   5. Model Selection    |
|      - Compare Models   |
|      - Final Evaluation |
+-------------------------+
```

Results: EDA

EDA Results: Visualizations and correlations revealed important insights into the relationship between launch site, payload weight, and landing outcomes.

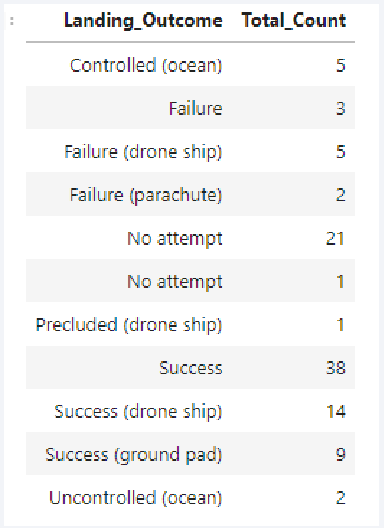
Landing Outcome Distribution: A bar chart was created to show the distribution of successful vs. unsuccessful landings, highlighting the balance or imbalance in the dataset.

Launch Site vs. Landing Outcome: A count plot was used to show the distribution of successful and unsuccessful landings across different launch sites. This helped visualize which sites had higher landing success rates.

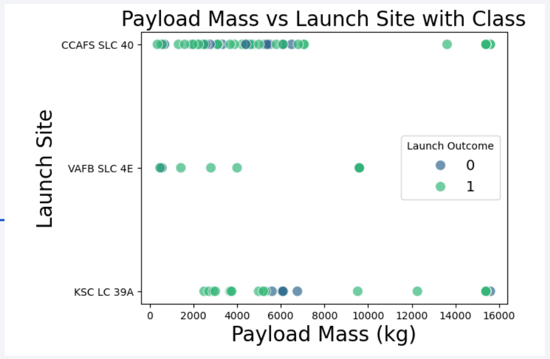
Payload Weight vs. Landing Outcome: A box plot was plotted to show the distribution of payload weights for both successful and unsuccessful landings. This revealed whether payload weight had any influence on the landing success.

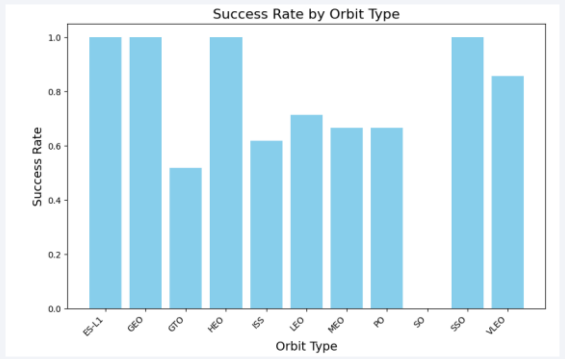
Launch Date Distribution: A histogram was created to show the distribution of launch dates, helping understand the frequency of launches over time.

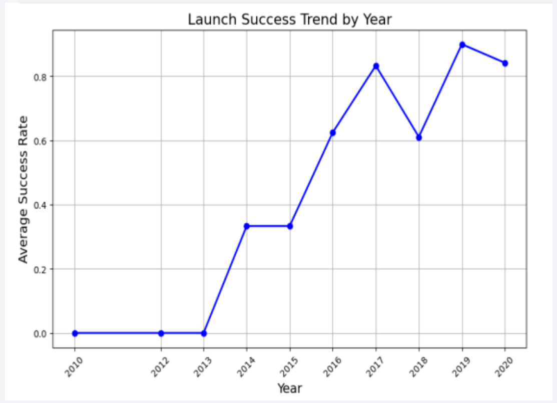
Correlation Heatmap: A heatmap was plotted to reveal the correlation between numerical features such as payload weight, rocket configuration, and landing outcome.

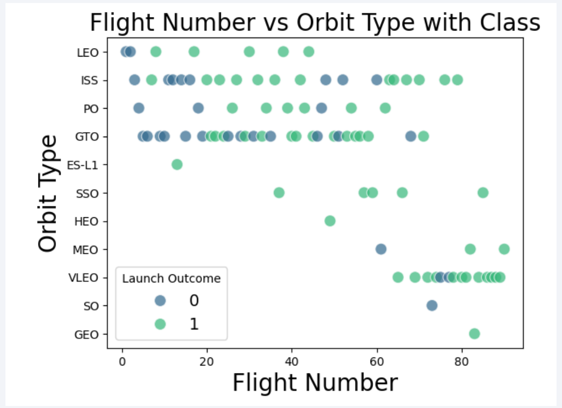


Landing_Outcome	Total_Count
Controlled (ocean)	5
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	21
No attempt	1
Precluded (drone ship)	1
Success	38
Success (drone ship)	14
Success (ground pad)	9
Uncontrolled (ocean)	2

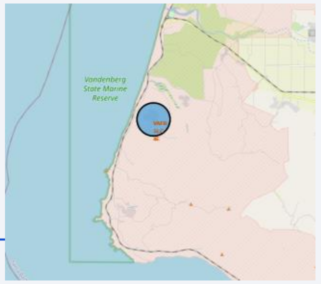




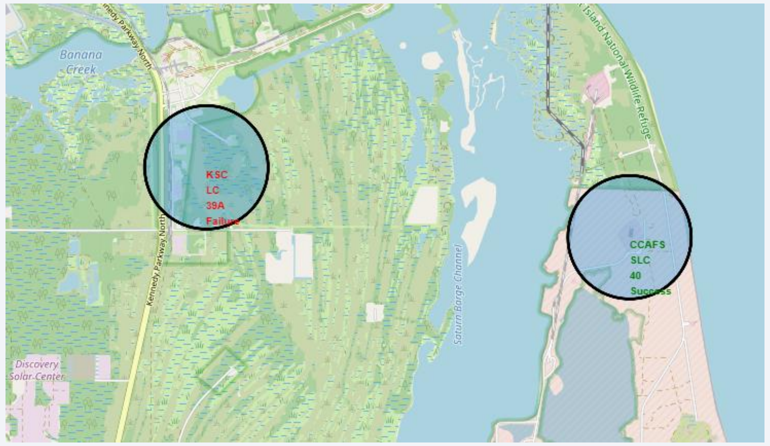




Results: Interactive Analytics



Interactive Analytics: created using Folium to visualize the SpaceX launch sites and landing zones on a map. The interactive map demonstrated the geographic distribution of launch sites and landing zones, offering spatial insights into the success rates of landings.

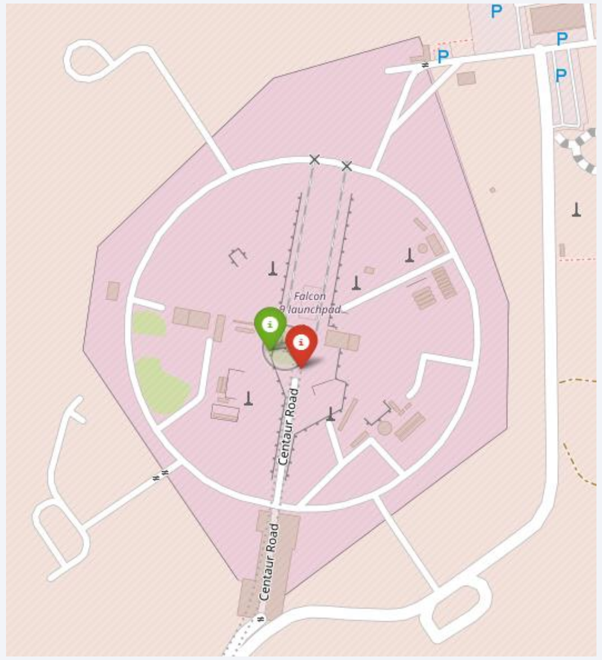


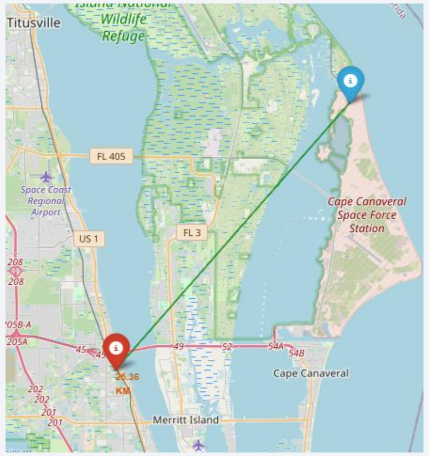
Launch Site Markers: Markers were added to the map to represent the locations of launch sites, such as the Cape Canaveral or Vandenberg launch sites.

Landing Zones: Circles were used to mark the areas where the rocket first stage was supposed to land, including regions over the ocean, drone ships, and ground pads.

Rocket Trajectory Lines: Lines were drawn between launch sites and landing zones to represent the possible rocket flight paths. This helped visualize the distance and trajectory from launch to landing.

Heatmap for Landing Success: A heatmap was added to the map to show areas with high landing success rates, allowing for deeper insights into the spatial distribution of successful landings.





Results: Predictive Analysis

Predictive Analysis: The predictive analysis involved building classification models to predict whether the Falcon 9 rocket's first stage would successfully land. Several models were evaluated, including Logistic Regression, SVM, Decision Trees, and KNN.

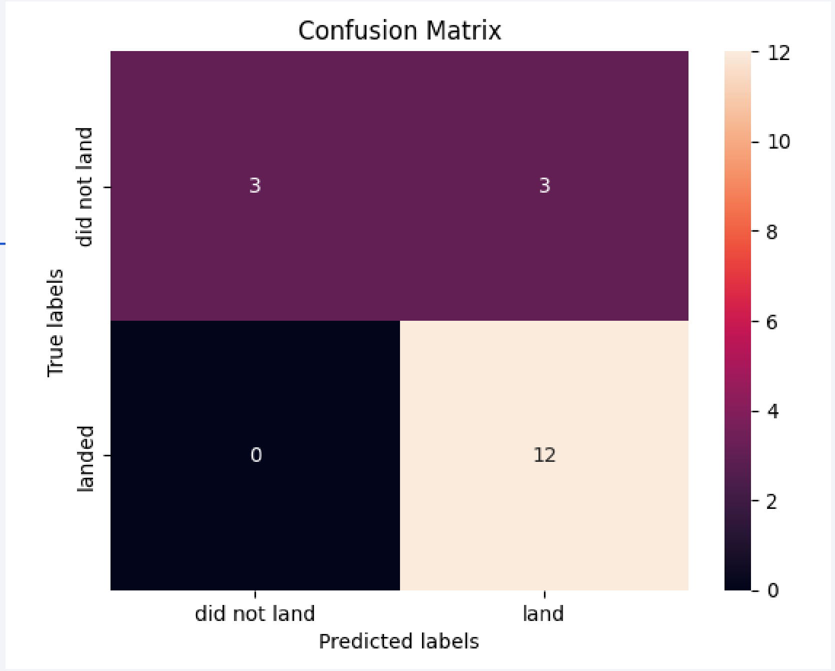
Model Selection: After evaluating multiple classification models, the best-performing model was selected based on accuracy, precision, and recall.

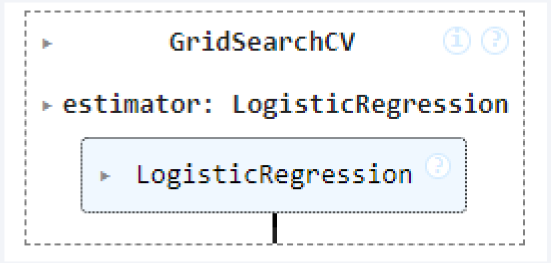
Accuracy Comparison: The Logistic Regression model provided the best accuracy on the test data, achieving an accuracy of around 85% for predicting landing success.

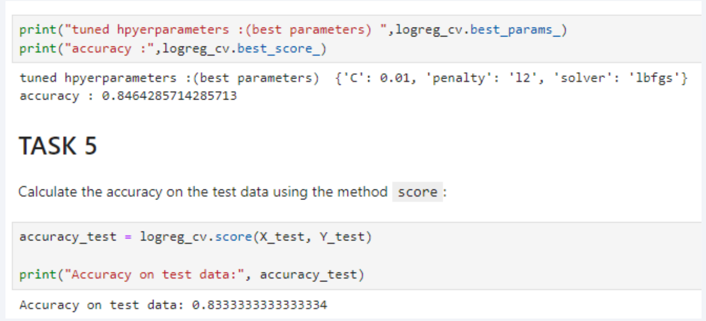
Confusion Matrix: A confusion matrix was used to assess the model's performance, showing how many true positives, false positives, true negatives, and false negatives were predicted.

ROC Curve and AUC: The ROC curve was plotted to evaluate the performance of the classification models, with the AUC (Area Under the Curve) providing a summary of model performance.

Final Model: The Logistic Regression model with optimized hyperparameters (found using GridSearchCV) was the final model chosen for deployment due to its high performance. After training and evaluating various models, Logistic Regression was selected as the best model for predicting the success of the Falcon 9 first-stage landing.







```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

TASK 5

Calculate the accuracy on the test data using the method `score` :

```
accuracy_test = logreg_cv.score(X_test, Y_test)

print("Accuracy on test data:", accuracy_test)
```

```
Accuracy on test data: 0.8333333333333334
```

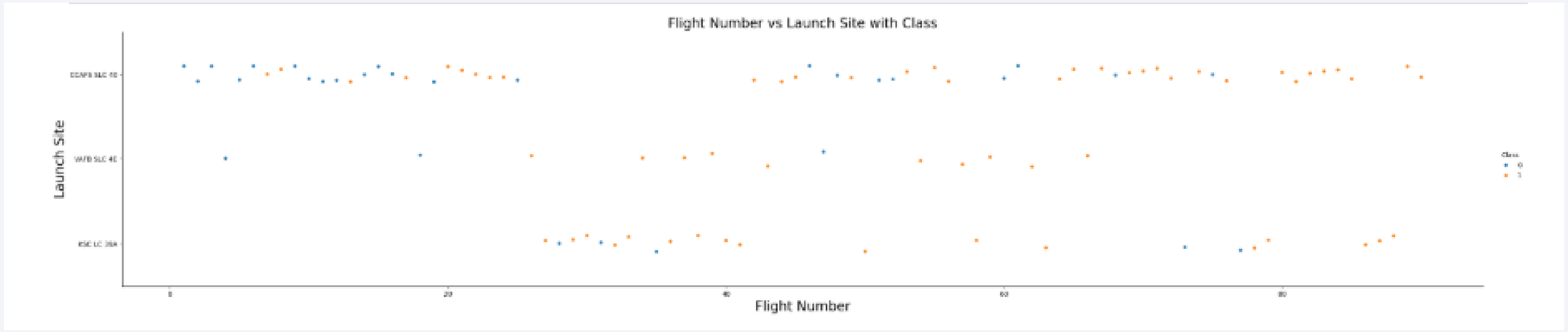
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

Flight Number vs. Launch Site: Flight number does not seem to be strongly correlated with landing outcomes across all sites. **There are no clear trends suggesting that later flights (higher flight numbers) are more likely to succeed or fail**. There seems to be randomness in the success/failure of landings, with points scattered across different flight numbers for each launch site.

Possible Clustered Outcomes: There may be some clustering of successful landings (orange points) within certain flight numbers or launch sites, possibly indicating that **successful landings might happen after a certain number of flights** (indicating experience and optimization of landing strategies). Similarly, there may be clusters of failures (blue points) concentrated in particular flight numbers or launch sites, suggesting that **specific sites or flights may have faced difficulties in landing**.

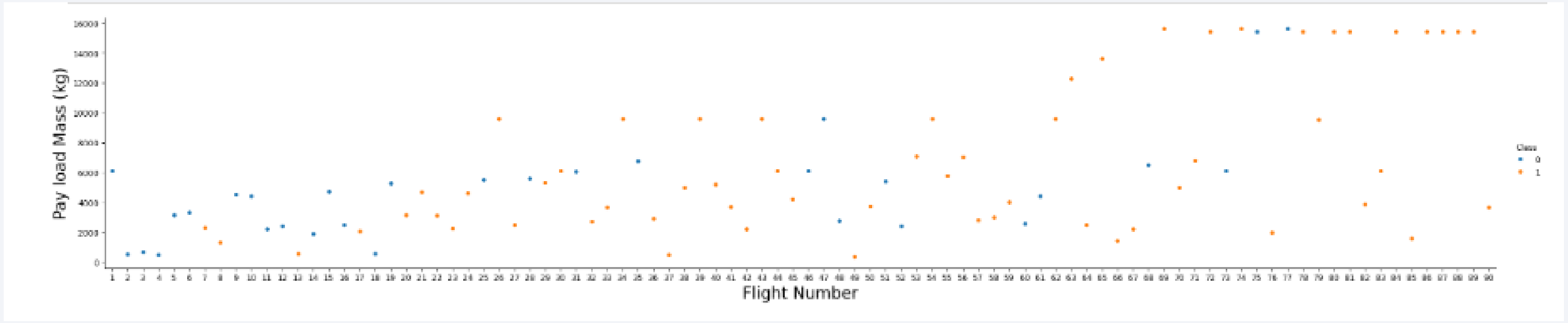


Payload vs. Launch Site

Payload vs. Flight Number: shows that **payload mass increases with flight number**, indicating that SpaceX's missions evolved to carry more complex payloads. Payload mass tends to increase with the flight number. Early flights, particularly those with lower flight numbers, have relatively lower payloads (under 10,000 kg), while later flights have significantly higher payloads, some reaching up to 20,000 kg or more.

Flight Number vs. Landing success: A noticeable trend is that the later flight numbers tend to have more successful landings (orange dots). Early flights (with lower flight numbers) have more blue dots, indicating that landing failures were more common in the early stages of Falcon 9's development.

Payload mass does not show a strong correlation with landing failure, as both successful and unsuccessful landings occur across different payload sizes.

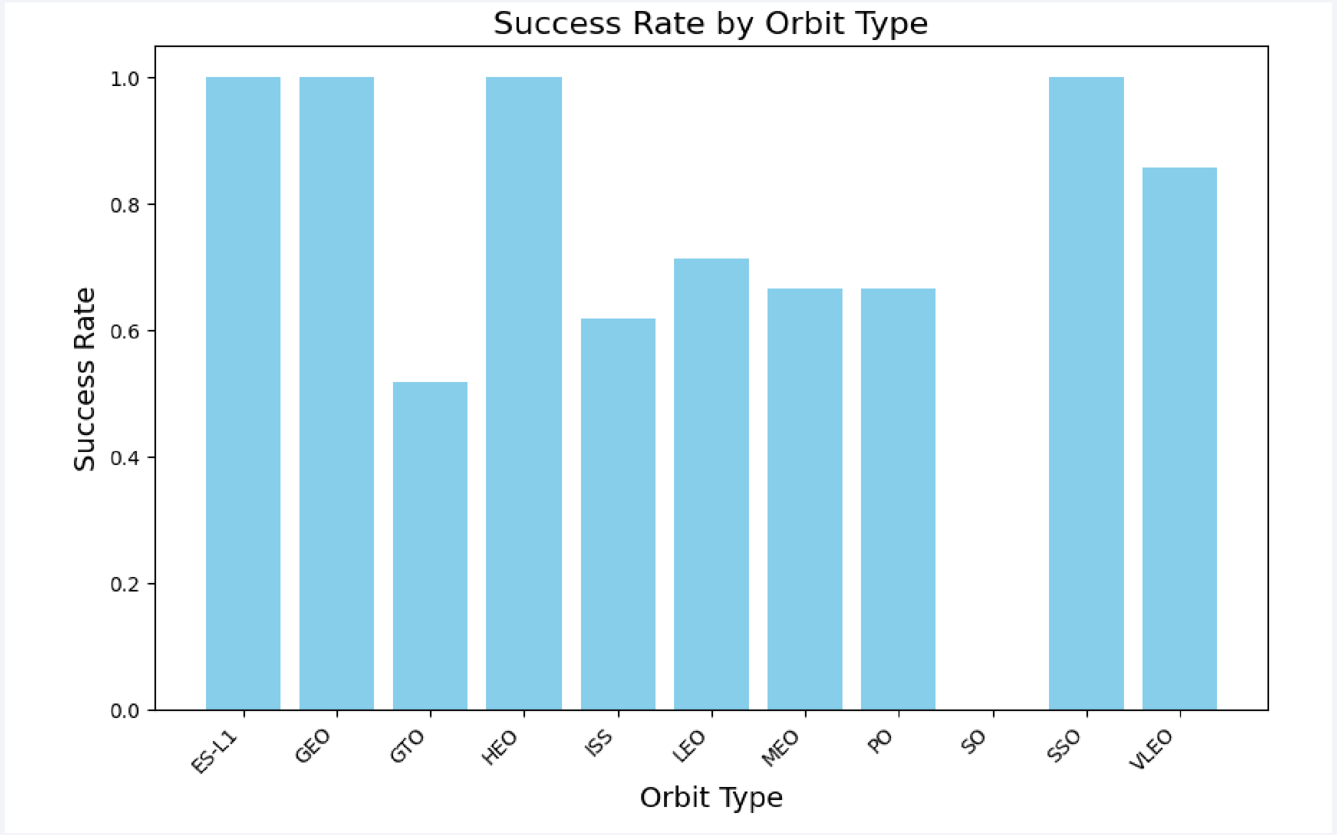


Success Rate vs. Orbit Type

Orbit types like **ES-L1, GEO, SSO, and VLEO** show the **highest success rates**, indicating these missions tend to be more routine or optimized for Falcon 9's capabilities.

Polar orbits (PO) show a **notably lower success rate**, likely due to more complex mission demands or different constraints during launch and recovery.

The overall trend shows that some orbital missions may be more successful due to either the inherent nature of the mission or the technical ability to handle those types of missions effectively.

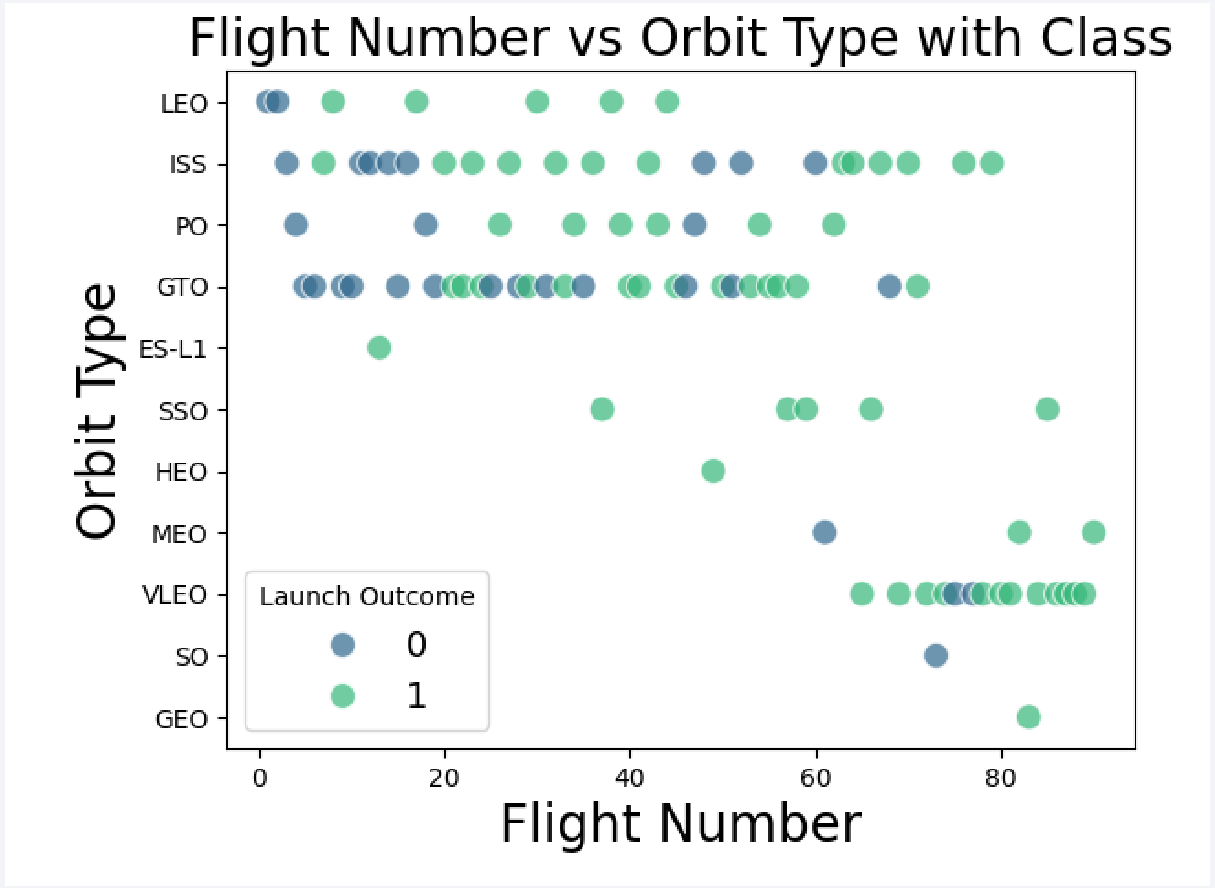


Flight Number vs. Orbit Type

Flight Number: Later flight numbers tend to show **higher success rates across all orbit types.**

Orbit Types: Some orbits like LEO, ISS, and SSO have higher success rates, while GTO and HEO show more variability, indicating challenges in landing these types of missions.

Improvements in Success: The pattern suggests that as SpaceX gained more experience, the success rate improved, particularly for missions to geostationary orbits and other challenging destinations.

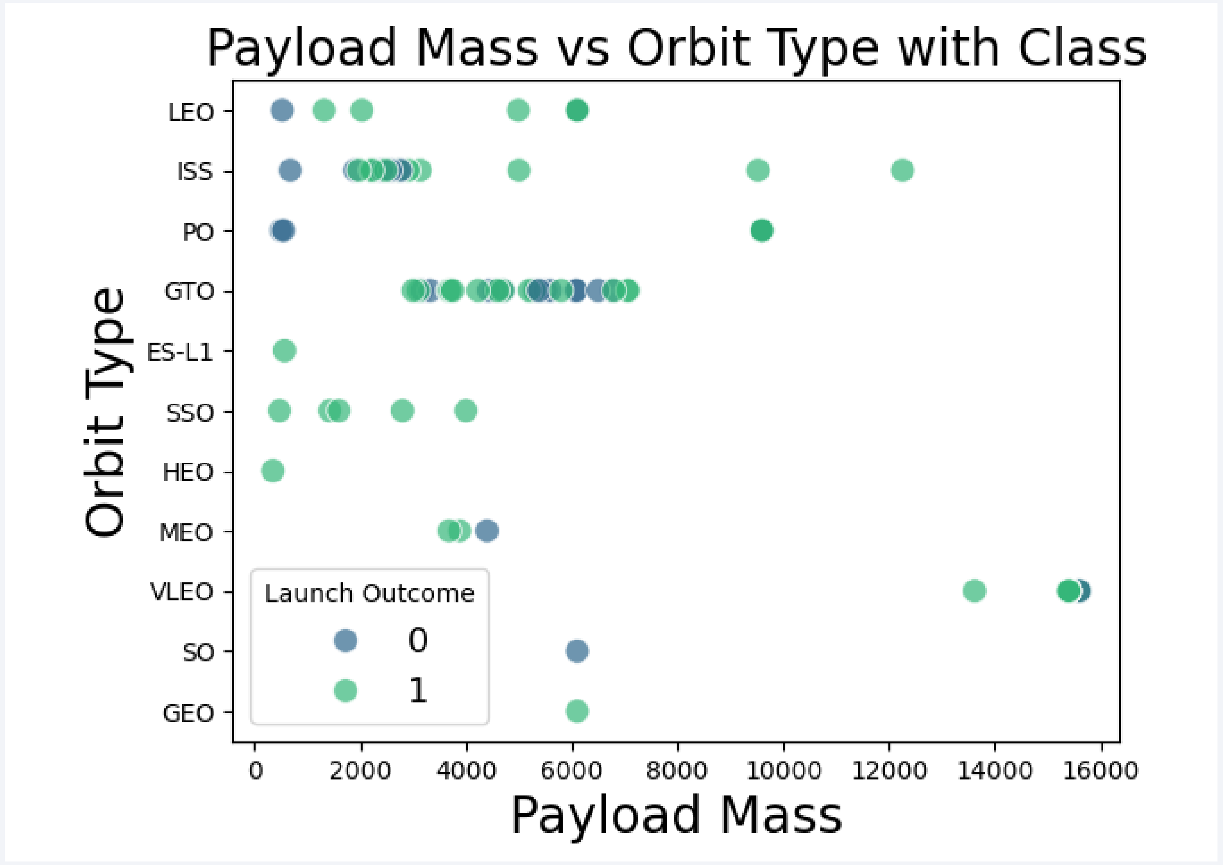


Payload vs. Orbit Type

Mass vs. Orbit Type: Larger payloads are generally associated with missions to GEO, GTO, and SSO orbits, whereas LEO and ISS missions tend to have smaller payloads.

Landing Success and Payload: Missions with larger payloads, particularly those to GEO, GTO, and SSO, **show a higher success rate** (green points). Smaller payloads, particularly for PO, MEO, and SSO orbits, have a more mixed success/failure pattern.

Overall Success: The graph suggests that **as payload mass increases, the success rate improves in some orbit types**, particularly for GEO and GTO orbits.



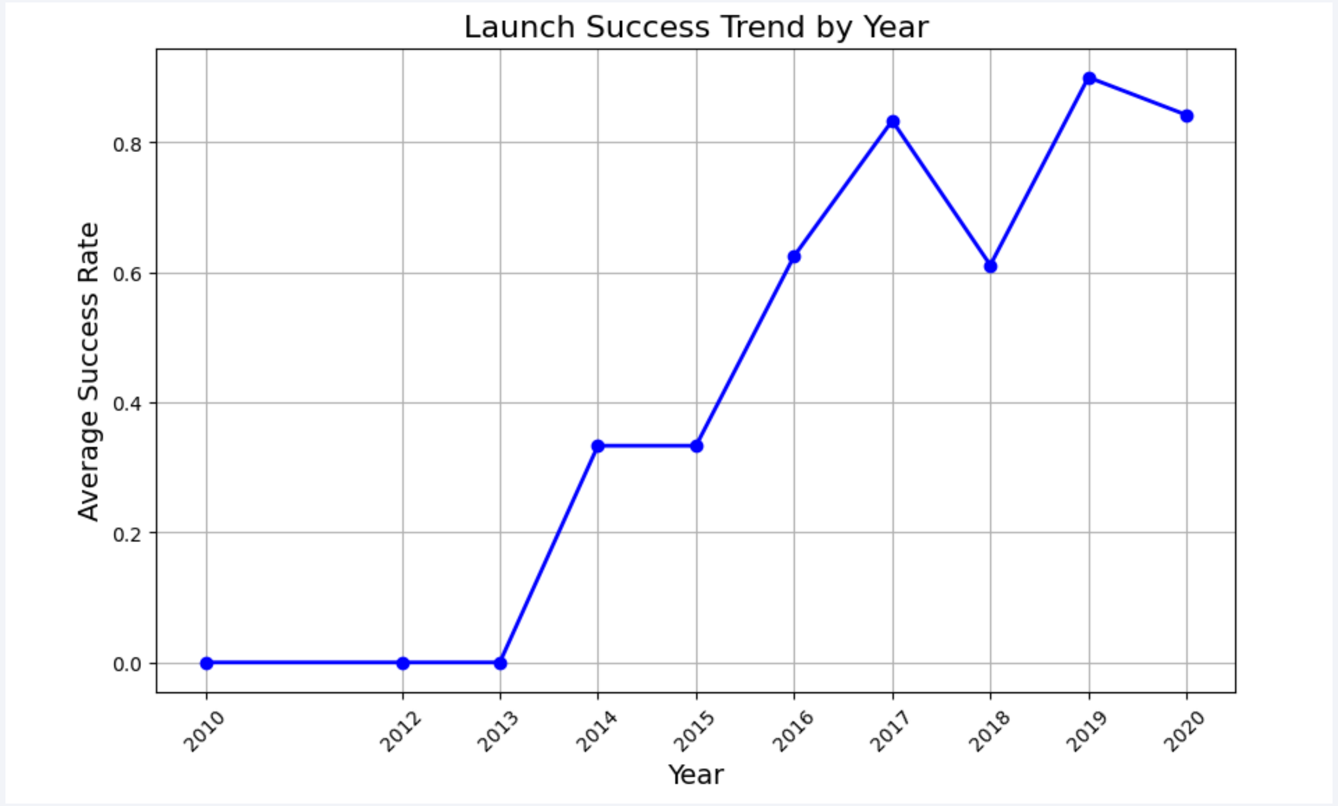
Launch Success Yearly Trend

Early Years (2010-2013): Low success rates, possibly due to early-stage testing and technological challenges.

Improvement (2014-2017): Rapid improvement in launch success rates, indicating growing expertise and refinement of SpaceX's technology.

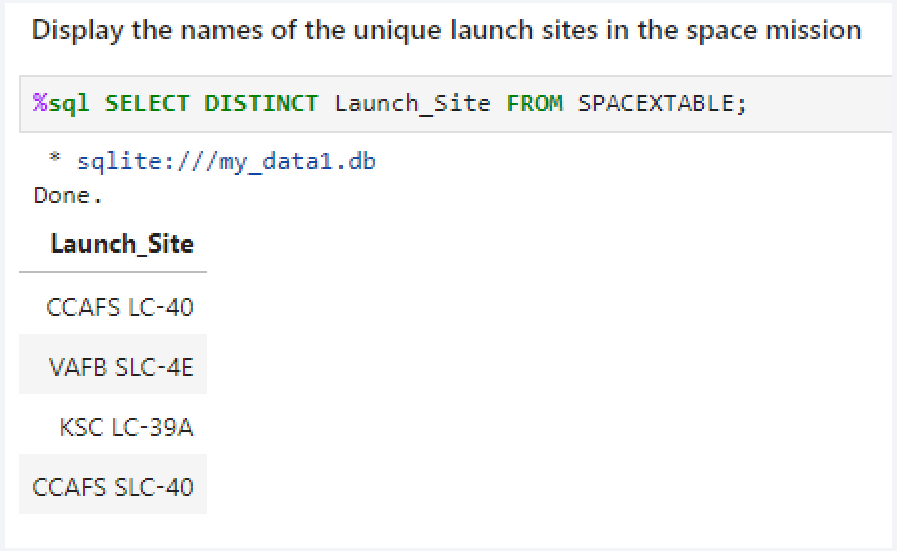
Peak Success (2018): The success rate peaks, reflecting near-perfect mission success.

Slight Decline (2019-2020): A minor dip in success rate but still relatively high, showing SpaceX's continued dominance in the space industry.



All Launch Site Names

The code below retrieves the unique values from the "Launch_Site" column in the dataset. This query retrieves a list of unique launch sites from the SPACEXTABLE. It returns each distinct launch site (without any duplicates) present in the table.



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

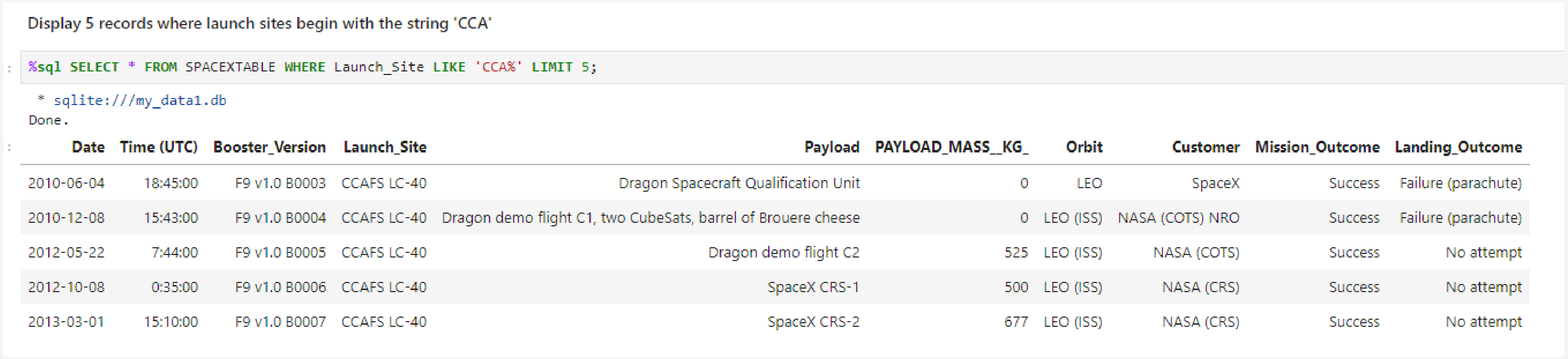
```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

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

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

Launch Site Names Begin with 'CCA'

The code below filters the dataset for rows where the "Launch_Site" column starts with CCA and then display the first 5 records.



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

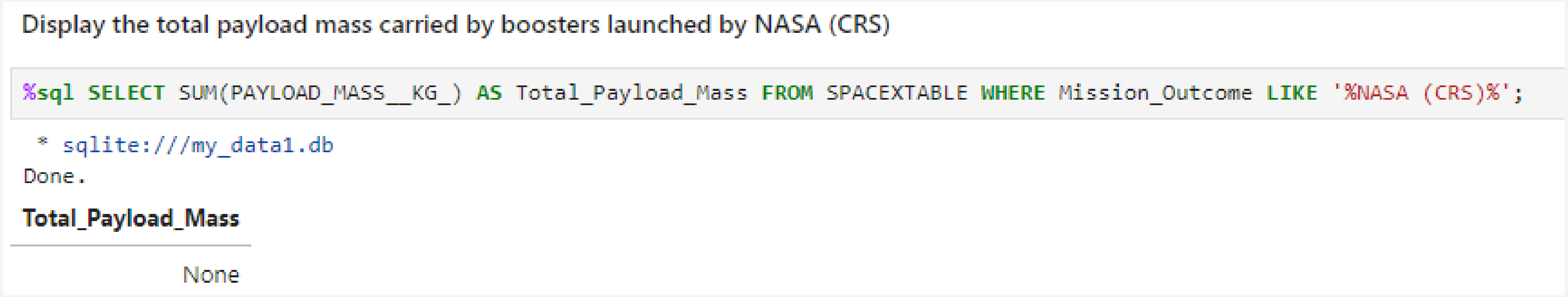
```
%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

The code below sums up the payload mass for all the records in the "PAYLOAD_MASS__KG_" column. The result returns the payload carried by the boosters launched by NASA under the column name Total_Payload_Mass



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

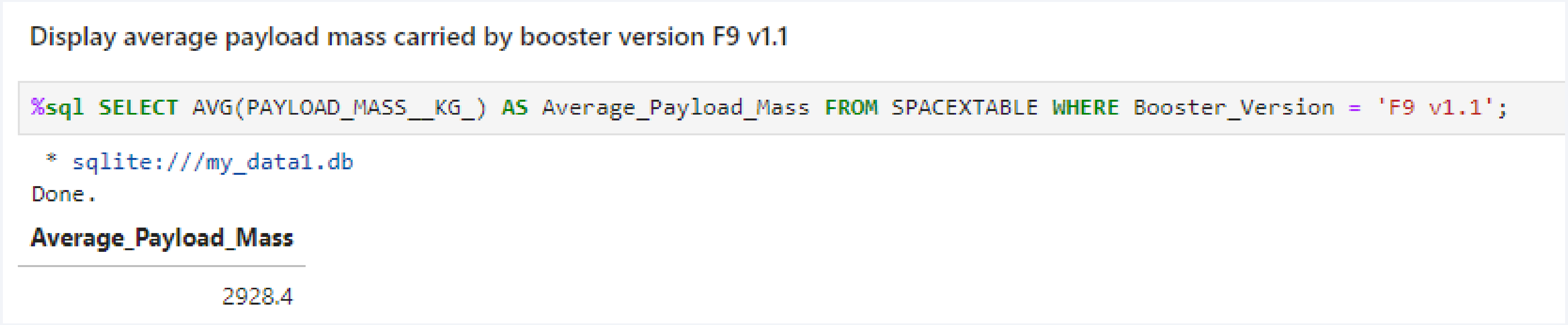
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Payload_Mass FROM SPACEXTABLE WHERE Mission_Outcome LIKE '%NASA (CRS)%';
```

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

Total_Payload_Mass
None

Average Payload Mass by F9 v1.1

The code below calculates the average payload mass (in kilograms) for launches where the booster version is 'F9 v1.1'. It filters the records in the SPACEXTABLE table to only include those launches where the Booster Version is 'F9 v1.1'.The result returns the average payload mass for F9 v1.1 launches under the column name Average_Payload_Mass.


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

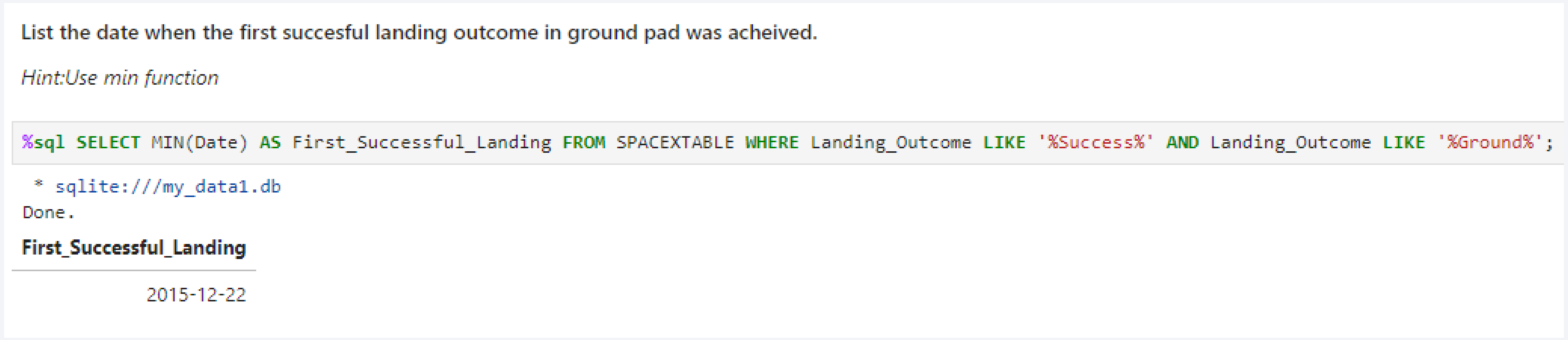
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Payload_Mass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1';
```

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

Average_Payload_Mass
2928.4

First Successful Ground Landing Date

The code below finds the earliest date of a successful landing on the ground. It does so by filtering the SPACEXTABLE for launches where the landing outcome indicates both a successful landing and a ground-based landing, as well as using the MIN(Date) function to ensure that only the earliest date that matches the conditions is returned.



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

Hint:Use min function

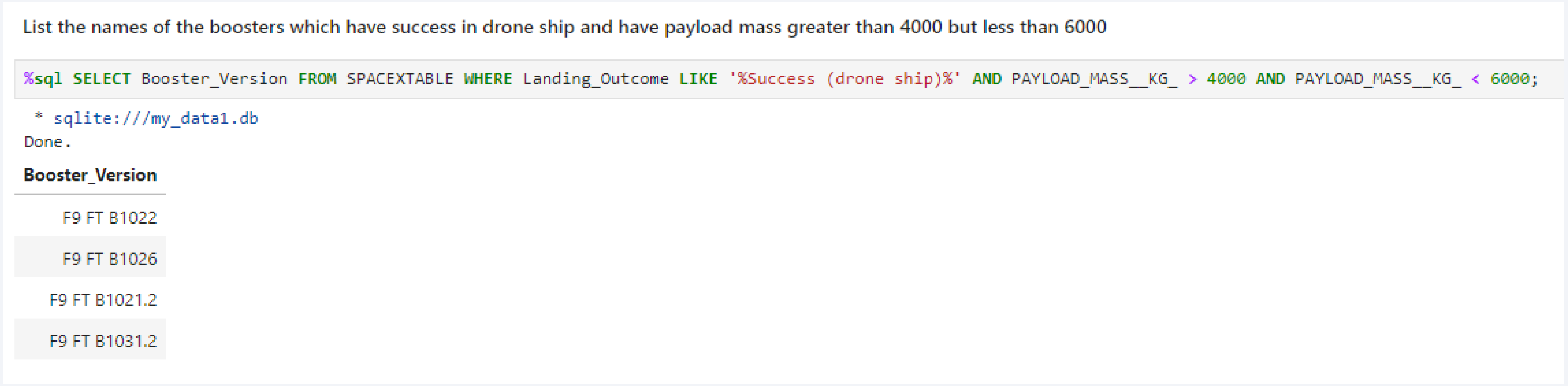
```
%sql SELECT MIN(Date) AS First_Successful_Landing FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%Success%' AND Landing_Outcome LIKE '%Ground%';
```

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

First_Successful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

The code below lists the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000



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

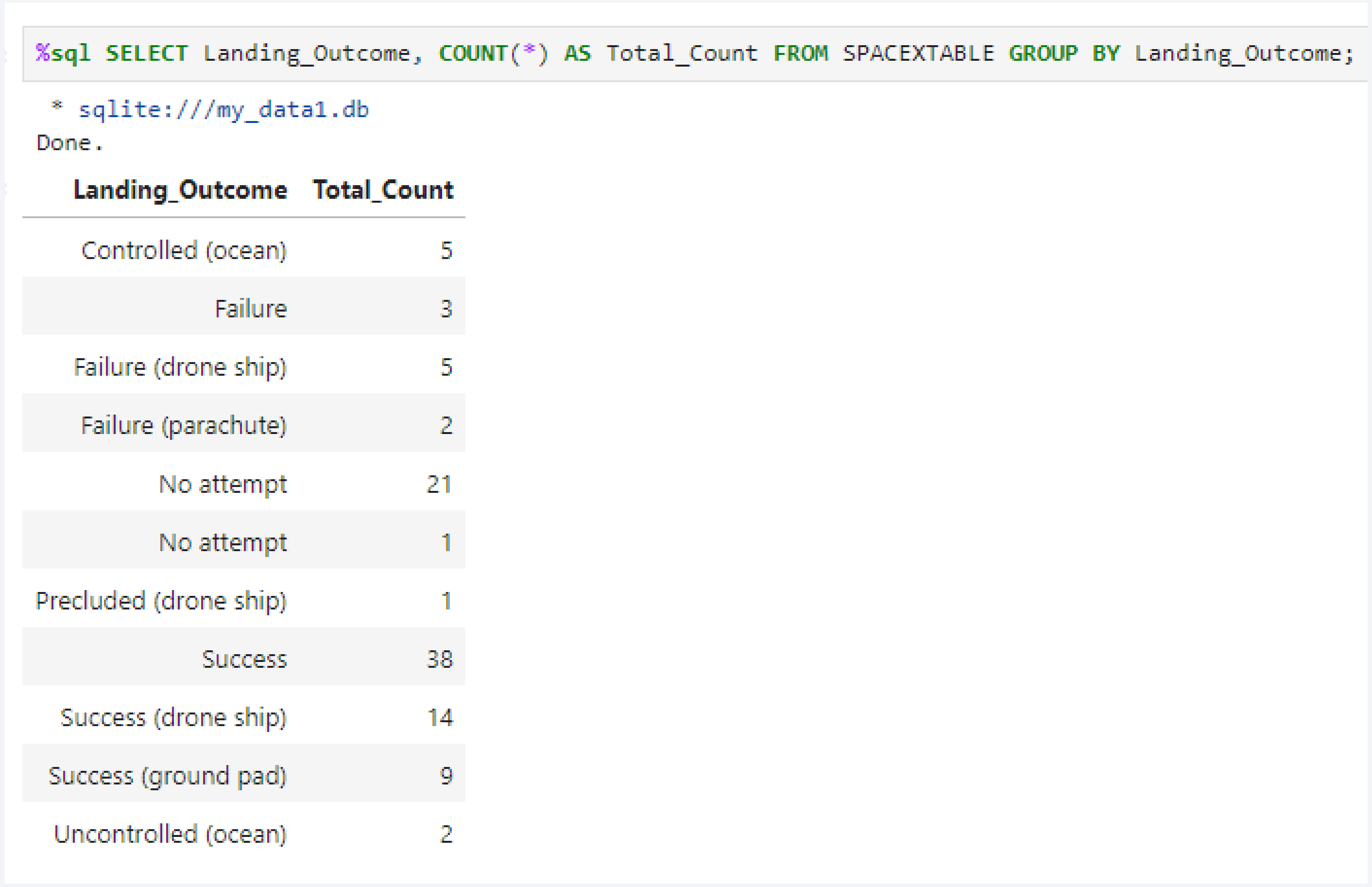
```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%Success (drone ship)%' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

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

This code produces the total number of successful and failure mission outcomes.



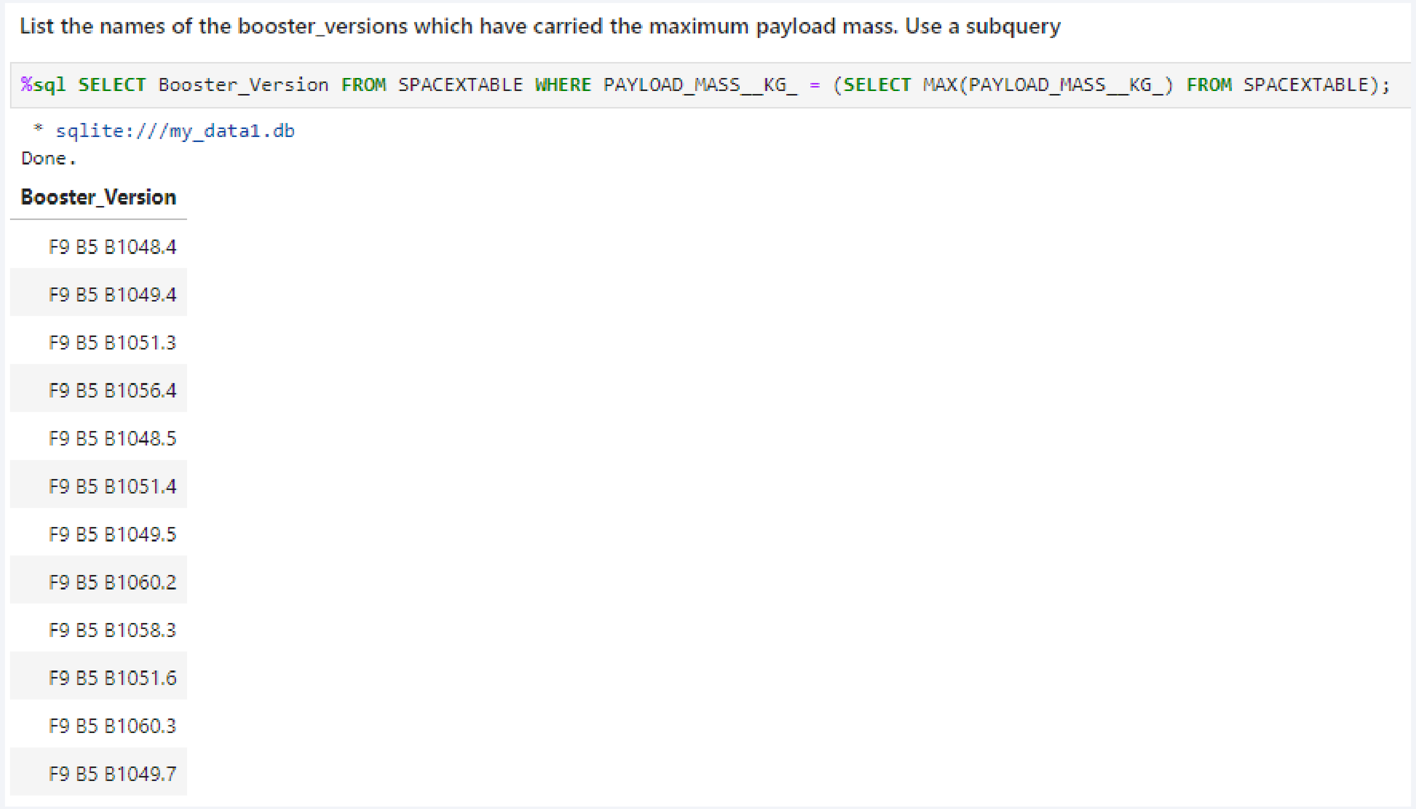
```
%sql SELECT Landing_Outcome, COUNT(*) AS Total_Count FROM SPACEXTABLE GROUP BY Landing_Outcome;
```

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

Landing_Outcome	Total_Count
Controlled (ocean)	5
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	21
No attempt	1
Precluded (drone ship)	1
Success	38
Success (drone ship)	14
Success (ground pad)	9
Uncontrolled (ocean)	2

Boosters Carried Maximum Payload

This code lists the names of the booster which have carried the maximum payload mass.



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

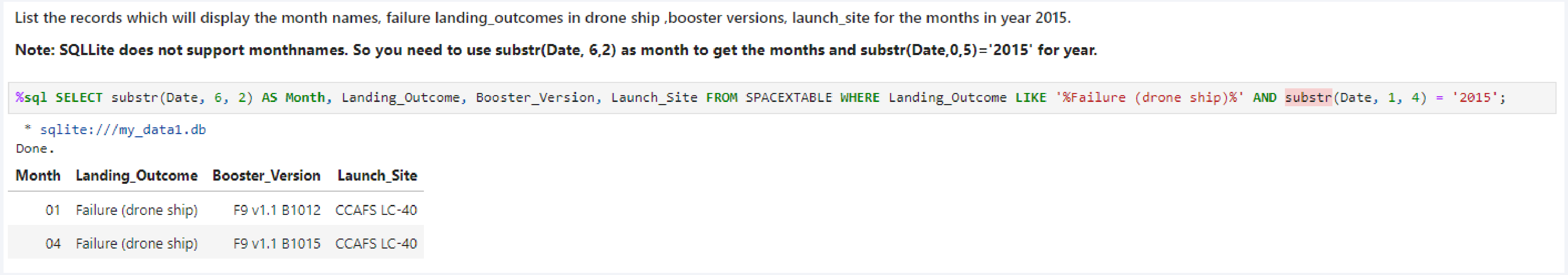
```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
```

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

The coide below lists the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015



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.

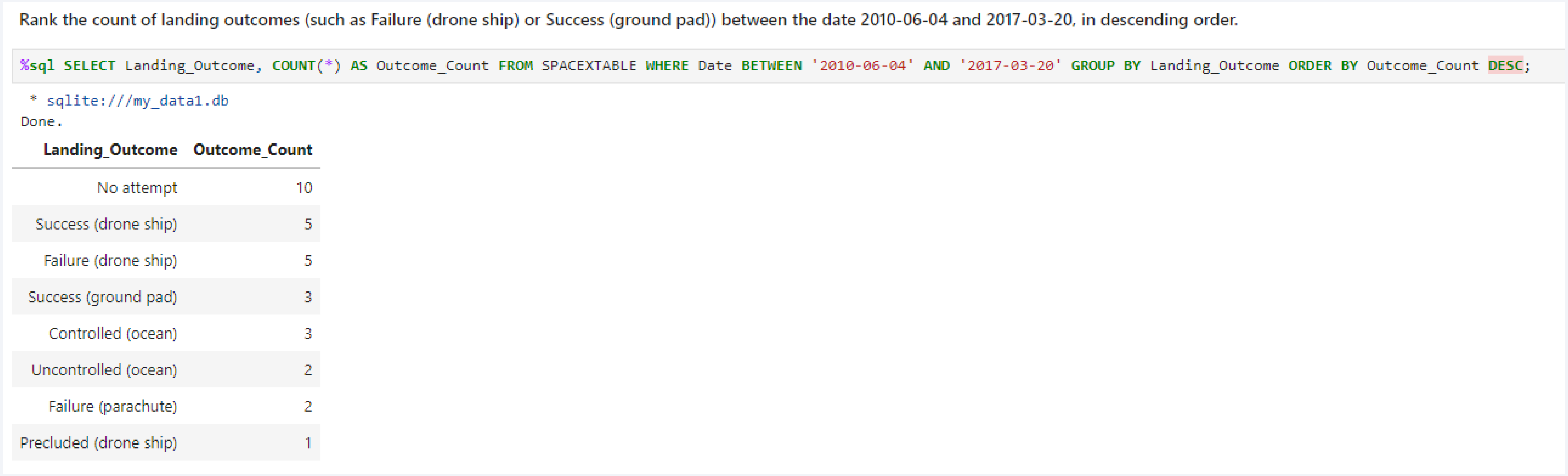
```
%sql SELECT substr(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%Failure (drone ship)%' AND substr(Date, 1, 4) = '2015';
```

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

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

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

The code below ranks 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



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.

```
%sql SELECT Landing_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY Outcome_Count DESC;
```

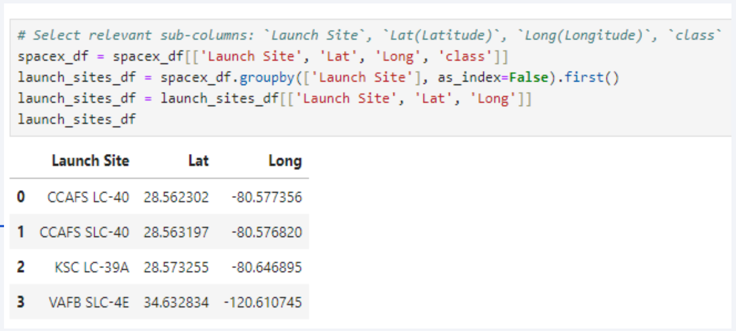
```
 * sqlite:///my_data1.db
Done.
```

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

Section 3

Launch Sites Proximities Analysis

<Launch Site Locations>

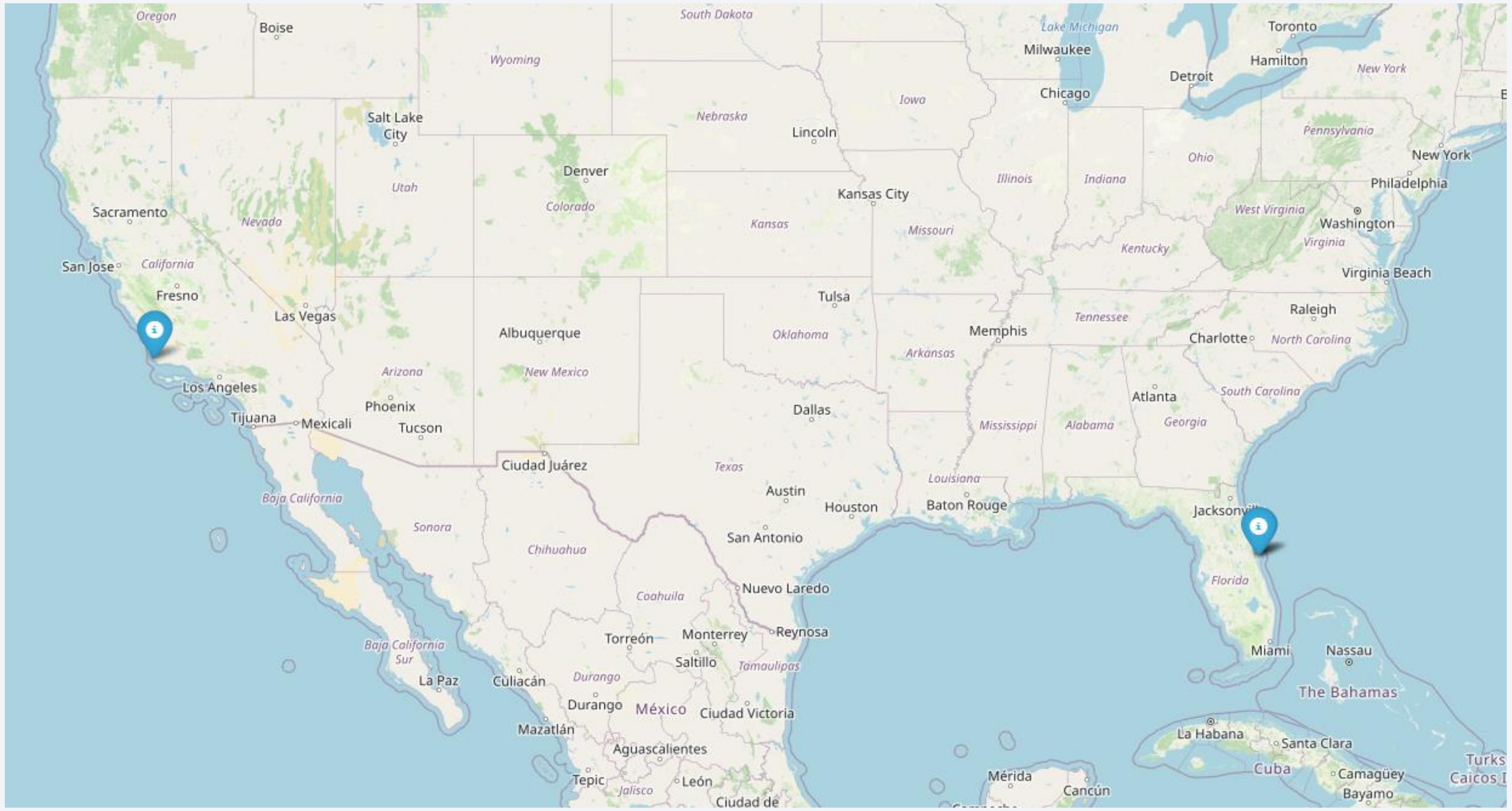


```python
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

The map shows the 4 total launch site locations for the SpaceX launch dataset.

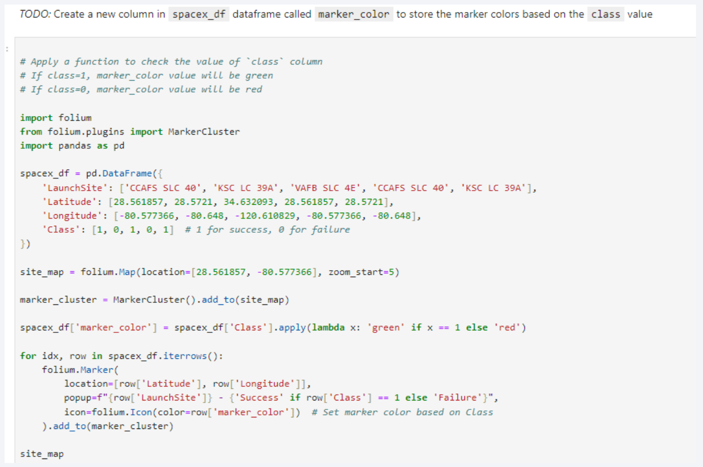
All launches are either found in California or Florida within the U.S.



<Color Marking for Launch Outcome>

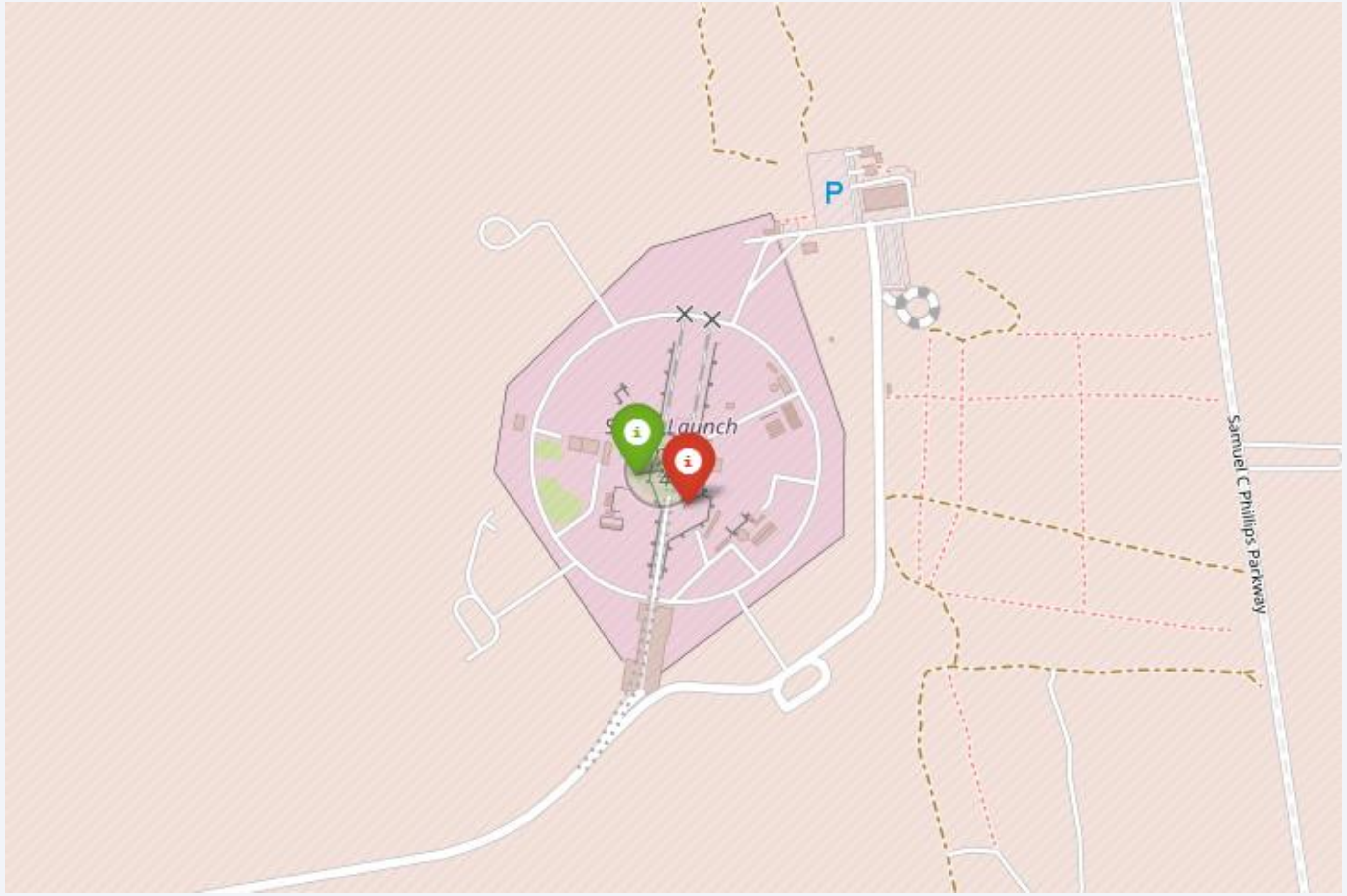
The map shows the color-labeled launch outcome.

Green sites are considered successful launches, whereas red sites are considered failed launches.



TODO: Create a new column in `spacex_df` dataframe called `marker_color` to store the marker colors based on the `class` value

```python
# Apply a function to check the value of `class` column
# If class=1, marker_color value will be green
# If class=0, marker_color value will be red

import folium
from folium.plugins import MarkerCluster
import pandas as pd

spacex_df = pd.DataFrame({
    'LaunchSite': ['CCAFS SLC 40', 'KSC LC 39A', 'VAFB SLC 4E', 'CCAFS SLC 40', 'KSC LC 39A'],
    'Latitude': [28.561857, 28.5721, 34.632093, 28.561857, 28.5721],
    'Longitude': [-80.577366, -80.648, -120.610829, -80.577366, -80.648],
    'Class': [1, 0, 1, 0, 1]  # 1 for success, 0 for failure
})

site_map = folium.Map(location=[28.561857, -80.577366], zoom_start=5)

marker_cluster = MarkerCluster().add_to(site_map)

spacex_df['marker_color'] = spacex_df['Class'].apply(lambda x: 'green' if x == 1 else 'red')

for idx, row in spacex_df.iterrows():
    folium.Marker(
        location=[row['Latitude'], row['Longitude']],
        popup=f"{row['LaunchSite']} - {'Success' if row['Class'] == 1 else 'Failure'}",
        icon=folium.Icon(color=row['marker_color'])  # Set marker color based on Class
    ).add_to(marker_cluster)

site_map
```



<Launch Sites vs. Closest Proximities>

Generated folium maps of selected launch site and its proximities to the coastline, highway, and railway.

Coastline Coordinates:
coastline_lat = 28.56167 coastline_lon = -80.56770

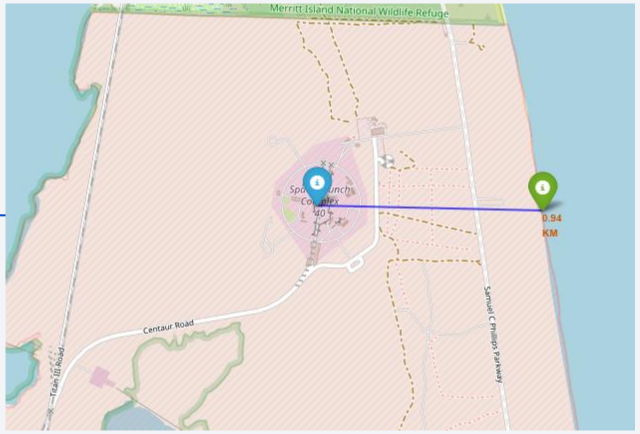
Distance to nearest coastline: 0.94 km

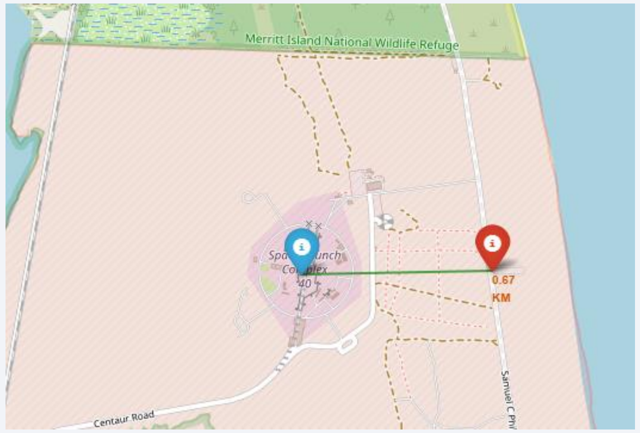
Highway Coordinates:
highway_lat = 28.5620 highway_lon = -80.5705

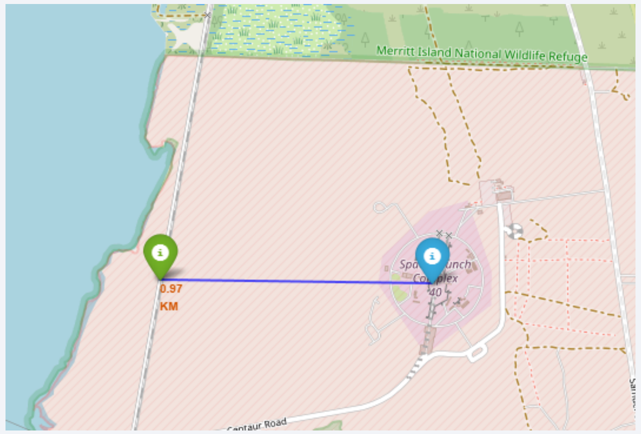
Distance to nearest highway: 0.67 km

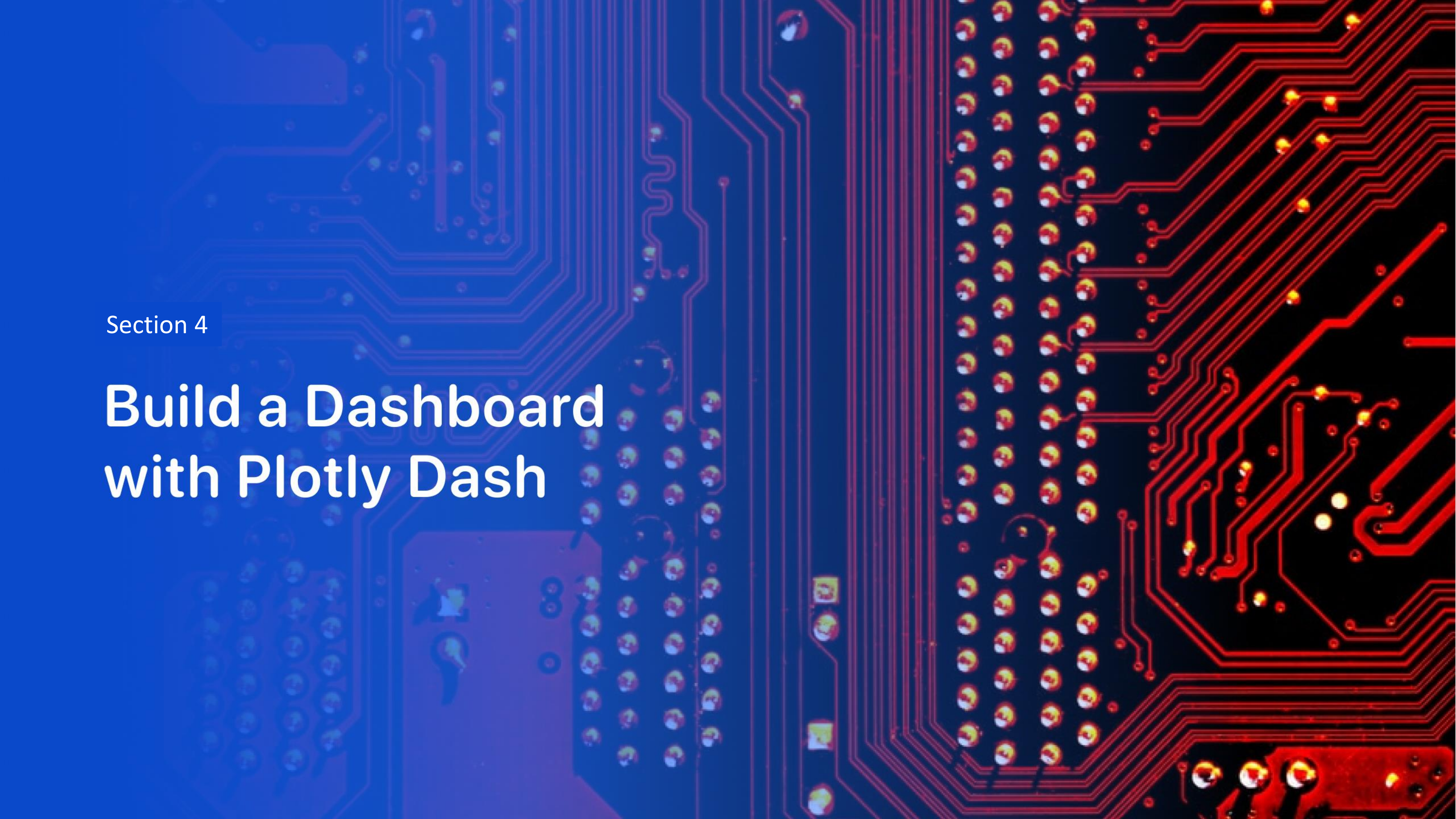
Railway Coordinates:
railway_lat = 28.5620 railway_lon = -80.5873

Distance to nearest railway: 0.97 km


Merritt Island National Wildlife Refuge
0.94 KM
Centaur Road
Samuel C Phillips Parkway







Section 4

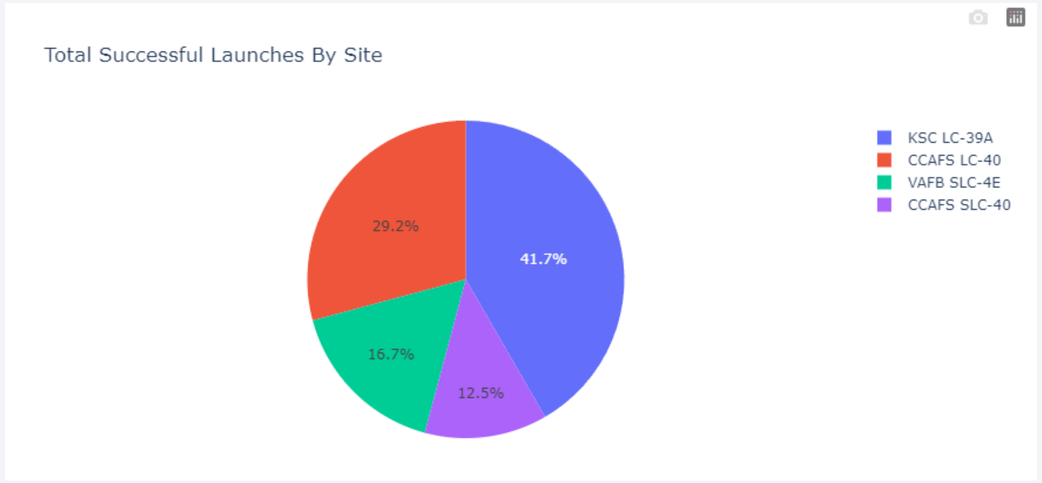
Build a Dashboard with Plotly Dash

<Total Successful Launches by Site>

This pie chart illustrates the distribution of total successful launches by site.

The proportions for each site are as follows:

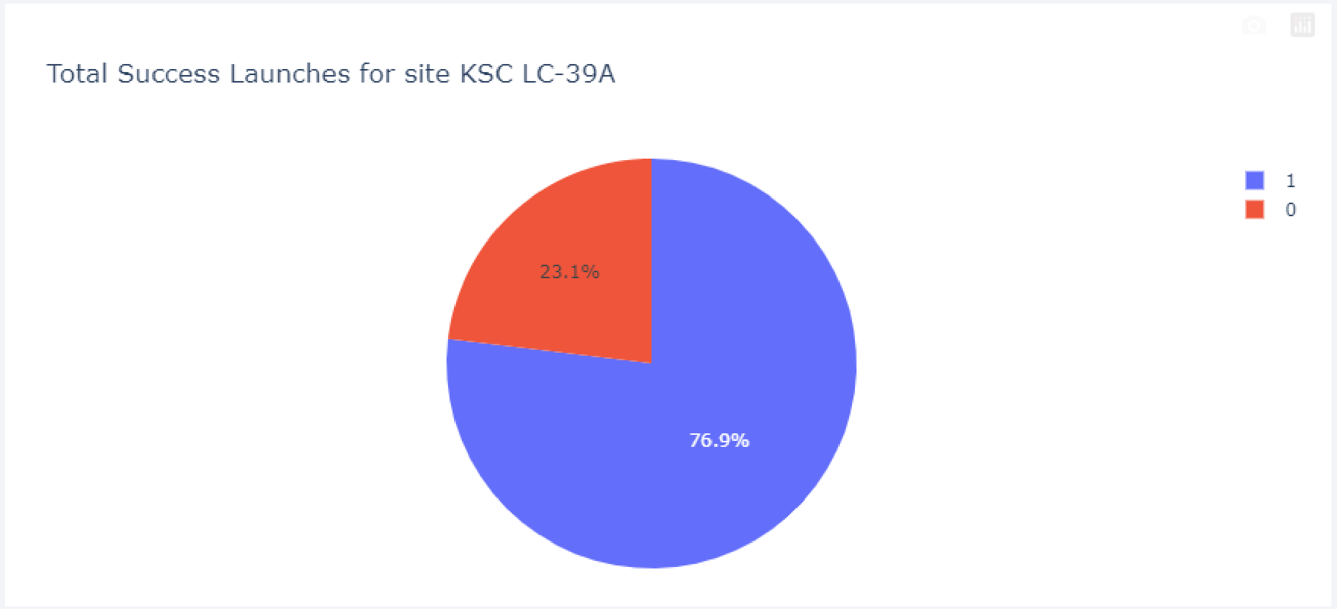
- KSC LC-39A: 41.7% of the total successful launches.
- CCAFS LC-40: 29.2% of the total successful launches.
- VAFB SLC-4E: 16.7% of the total successful launches.
- CCAFS SLC-40: 12.5% of the total successful launches.



The majority of successful launches were from KSC LC-39A, followed by CCAFS LC-40, VAFB SLC-4E, and lastly CCAFS SLC-40.

<Launch Site with Highest Launch Success>

The launch site with the highest launch success was for site “ KSC LC-39A” since it had a success rate of 76.9%.



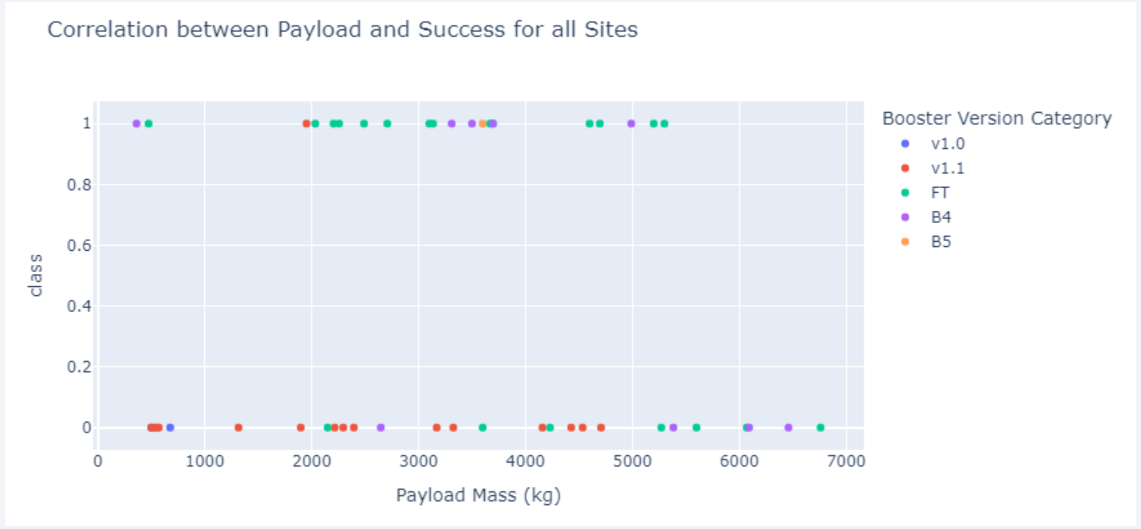
<Payload vs. Launch Outcome Scatter Plot >

This scatter plot shows the correlation between payload mass (kg) and the success of launches (class) across all sites, categorized by the booster version category.

Success Distribution (Class = 1): Most payloads, regardless of mass or booster type, resulted in successful launches.

Failure Distribution (Class = 0): Failures are scattered across payload masses, with no clear pattern tied to specific booster versions.

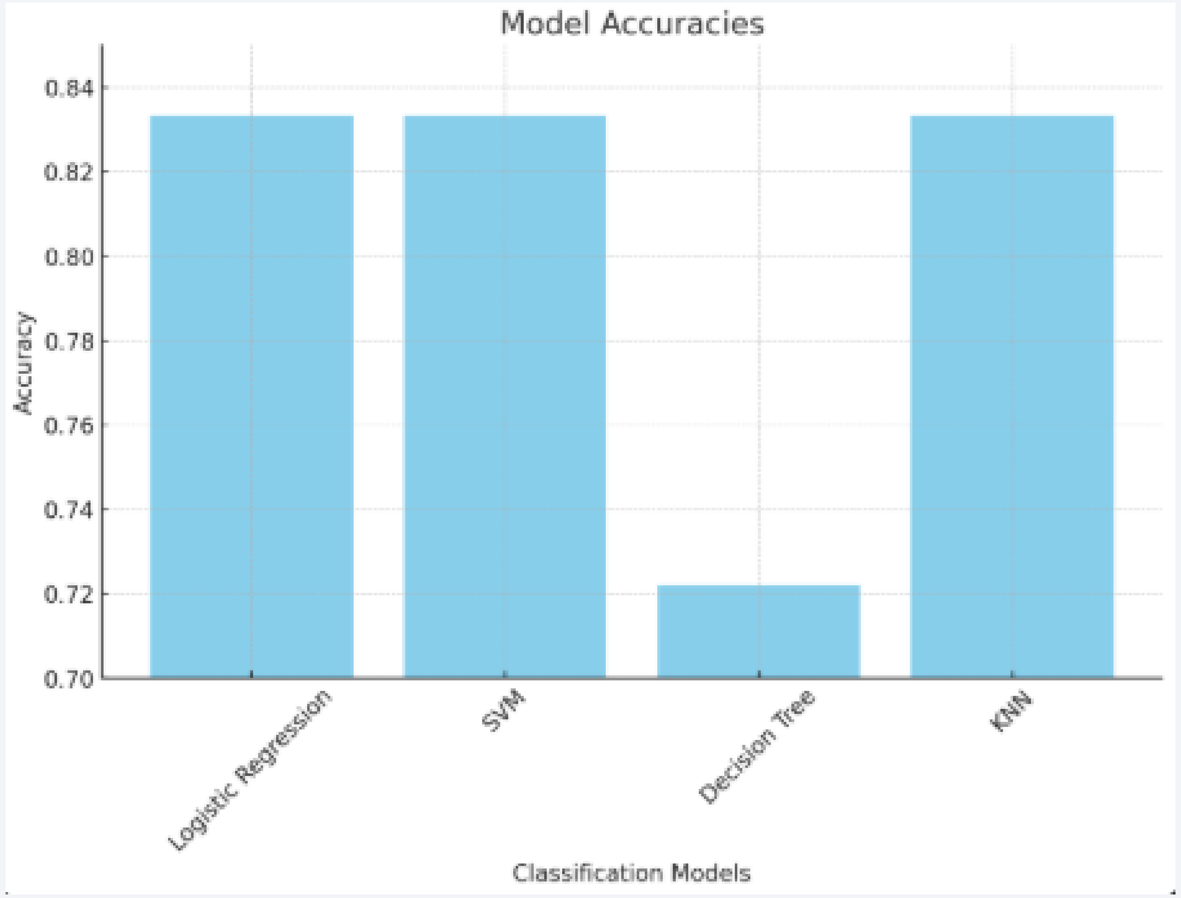
Payload Mass Range: Payloads with masses up to 7000 kg have been successfully launched across multiple booster versions.

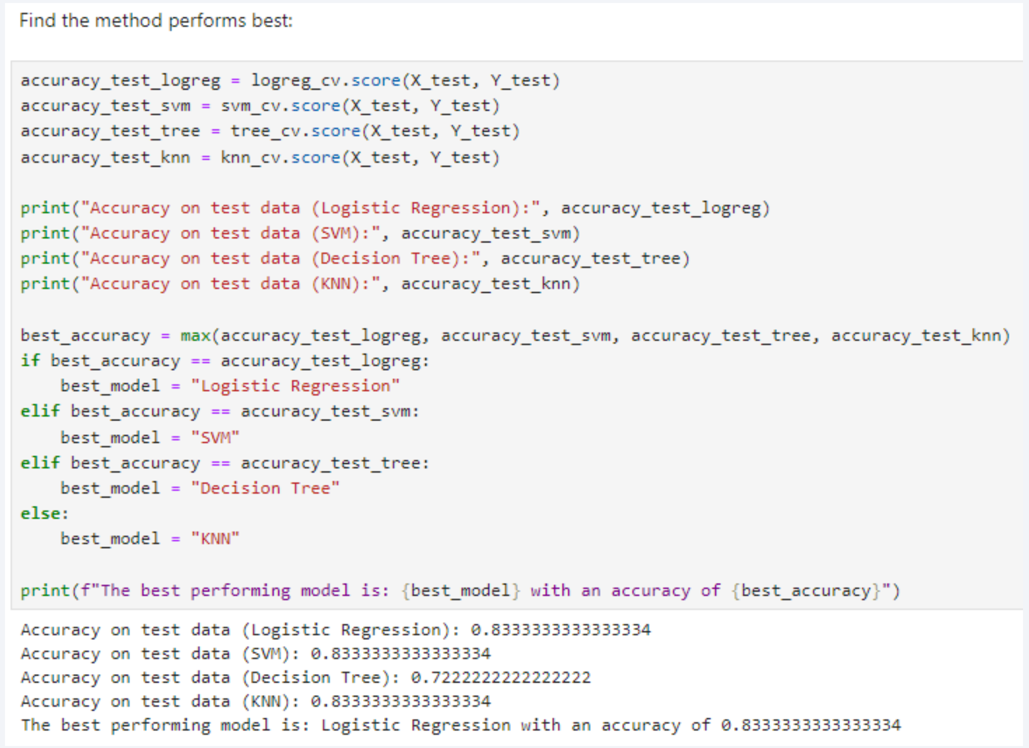


Section 5

Predictive Analysis (Classification)

Classification Accuracy




Find the method performs best:

```
accuracy_test_logreg = logreg_cv.score(X_test, Y_test)
accuracy_test_svm = svm_cv.score(X_test, Y_test)
accuracy_test_tree = tree_cv.score(X_test, Y_test)
accuracy_test_knn = knn_cv.score(X_test, Y_test)

print("Accuracy on test data (Logistic Regression):", accuracy_test_logreg)
print("Accuracy on test data (SVM):", accuracy_test_svm)
print("Accuracy on test data (Decision Tree):", accuracy_test_tree)
print("Accuracy on test data (KNN):", accuracy_test_knn)

best_accuracy = max(accuracy_test_logreg, accuracy_test_svm, accuracy_test_tree, accuracy_test_knn)
if best_accuracy == accuracy_test_logreg:
    best_model = "Logistic Regression"
elif best_accuracy == accuracy_test_svm:
    best_model = "SVM"
elif best_accuracy == accuracy_test_tree:
    best_model = "Decision Tree"
else:
    best_model = "KNN"

print(f"The best performing model is: {best_model} with an accuracy of {best_accuracy}")
```

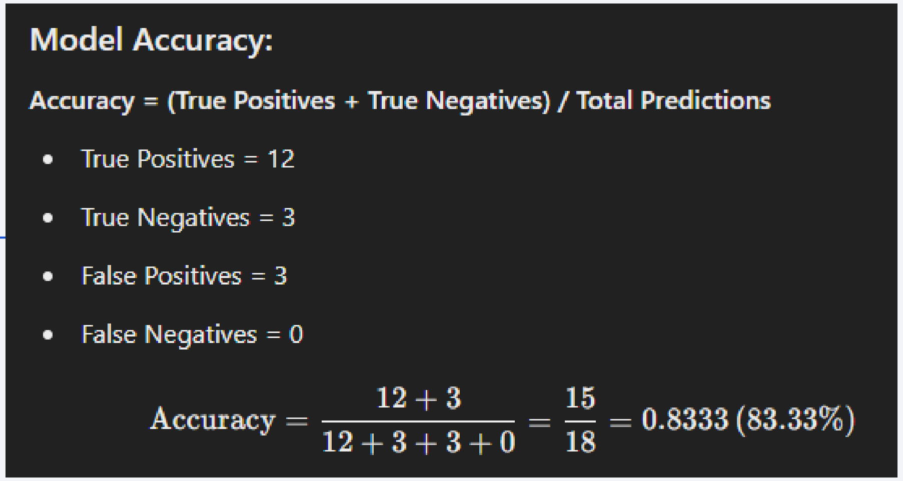
```
Accuracy on test data (Logistic Regression): 0.8333333333333334
Accuracy on test data (SVM): 0.8333333333333334
Accuracy on test data (Decision Tree): 0.7222222222222222
Accuracy on test data (KNN): 0.8333333333333334
The best performing model is: Logistic Regression with an accuracy of 0.8333333333333334
```

Confusion Matrix

- **Rows:** Represent the true labels (actual outcomes).
  - Row 1: "did not land" (negative class).
  - Row 2: "landed" (positive class).
- **Columns:** Represent the predicted labels (model's outcomes).
  - Column 1: "did not land.“
  - Column 2: "land.“

The cells contain the following values:

- True Negative (Top Left - 3): The model correctly predicted "did not land" when it actually did not land.
- False Positive (Top Right - 3): The model incorrectly predicted "landed" when it actually did not land.
- False Negative (Bottom Left - 0): The model incorrectly predicted "did not land" when it actually landed. (Here, the model had no such mistakes.)
- True Positive (Bottom Right - 12): The model correctly predicted "landed" when it actually landed.

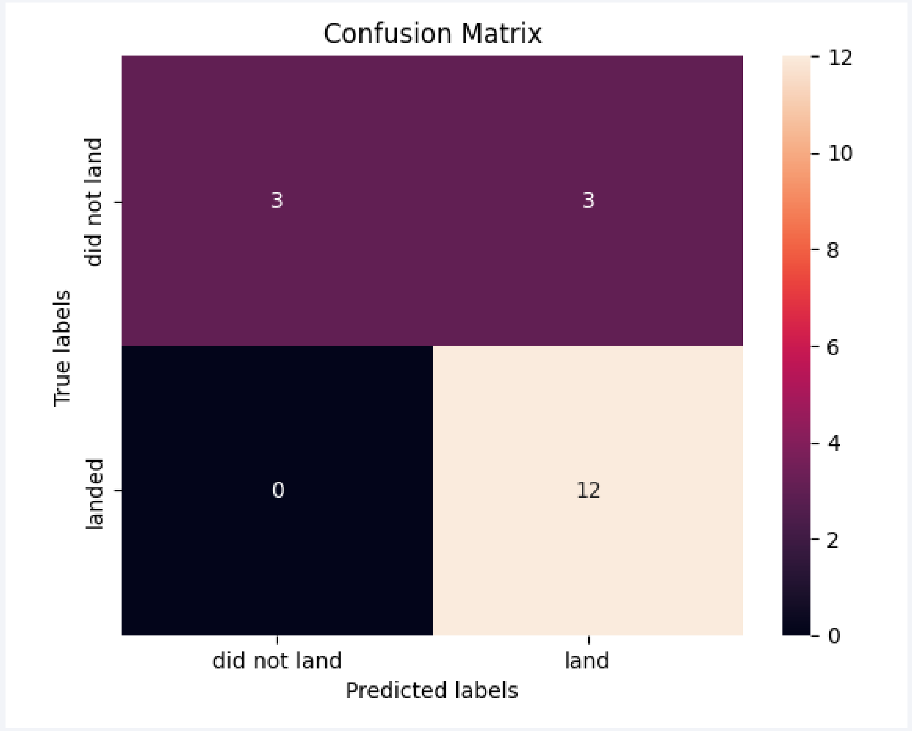


Model Accuracy:

Accuracy = (True Positives + True Negatives) / Total Predictions

- True Positives = 12
- True Negatives = 3
- False Positives = 3
- False Negatives = 0

$$\text{Accuracy} = \frac{12+3}{12+3+3+0} = \frac{15}{18} = 0.8333\,(83.33\%)$$



Conclusions

- **Predictive Model Success:**
  - **Logistic Regression emerged as the best-performing model with an accuracy of 83.33%.**
  - SVM and KNN models performed similarly but were not prioritized based on decision logic.
- **False Positive Issue:**
  - The confusion matrix revealed that **false positives (incorrectly predicting "landed") are the primary issue for the Logistic Regression model.**
  - This highlights areas for improvement, especially in real-world applications where minimizing false positives is critical.
- **EDA Insights:**
  - Launch site and payload mass significantly impact landing outcomes.
  - Specific **orbit types and increasing flight numbers correlate with higher success rates**.
- **Model Evaluation Techniques:**
  - Cross-validation and hyperparameter tuning were crucial in selecting the best model and optimizing its performance.
- **Practical Applications:**
  - Predictive models can help SpaceX optimize resource allocation and bid competitively in the aerospace market by improving landing success forecasts.

Appendix

All of the following data highlights can be found in the public GitHub:

https://github.com/mst141umiami/CAPSTONE.git

Python Code Snippets: Example of Logistic Regression model training. Code used for confusion matrix generation.

SQL Queries: Queries used to extract launch data and analyze outcomes, such as:Count of successful launches per site.Average payload mass for successful and failed landings.

Charts: Bar chart of model accuracies and Confusion matrix for Logistic Regression.

Notebook Outputs: Screenshots of key EDA results, such as correlation heatmaps or box plots of payload weight vs. landing outcome.

Data Sets: Cleaned dataset used for model training and testing, derived from SpaceX launch records (API and web scraping).

Thank you!