IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

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

Executive Summary

Summary of methodologies

- Data Collection: Web scraping (BeautifulSoup), SpaceX API (requests), JSON to Pandas
- Data Wrangling : Cleaning, feature engineering (class, payload category), Pandas
- EDA: Visual analysis using Matplotlib, Seaborn; SQL queries in Db2
- Interactive Viz : Created maps with Folium, built dashboards with Plotly Dash
- Modeling : Trained classifiers (SVM, Logistic Regression, Decision Tree), hyperparameter tuning with GridSearchCV
- Evaluation: Used accuracy and confusion matrix to assess model performance

Summary of all results

- Successfully scraped and transformed Falcon 9 data from Wikipedia into a structured DataFrame.
- Defined clear binary labels for landing success (1 = successful, 0 = failed).
- Collected complete launch data from SpaceX API, structured and cleaned it.
- SQL revealed patterns: KSC LC-39A had highest success; F9 v1.1 had lower payload average.
- Created the class column, visualized key correlations between payload, site, and success.
- Folium maps showed all sites; proximity to coast, rail, and highways were calculated and labeled.
- Built ML models to predict landings; best model (SMV or KNN) achieved 83%+ accuracy.

Introduction

Project background and context

SpaceX has become a leader in commercial space launches. As part of their mission to improve cost-efficiency and reliability, analyzing past launch data can reveal patterns that help guide future decisions.

Problems we want to find answers among others

What factors influence the success of a launch?

Which launch sites perform best in terms of success rates?

Is there a relationship between payload mass and launch outcome?

Can we build a model to predict the success of a launch based on input features?

Section 1

Methodology

Methodology

Data collection:

- SpaceX API: Retrieved launch data in JSON format using Python's requests module.
- Key attributes included PayloadMass, LaunchSite, BoosterVersion, Longitude, Latitude and Outcome.
- Web Scraping: Used BeautifulSoup to extract additional booster and launch-related information from Wikipedia pages.

Data wrangling

- Initial Exploration (Inspection).
- Cleaned missing and null values from the dataset.
- Converted columns to appropriate data types (e.g., date, float).
- Renaming Columns or Standardizing Labels
- Created engineered features: class: 0 = failure, 1 = success

Methodology

Exploratory data analysis (EDA) using visualization tools like Matplotlib and Seaborn

Created scatter plots to visualize:

- Flight Number vs. Launch Site
- Payload Mass vs. Launch Site

Plotted bar charts showing:

- Success rate per Orbit type
- Useful for identifying which orbit types yield higher success frequencies.

Analyzed success rates across time:

- Built line chart of yearly average success
- Purpose: observe evolution of success performance over years.

Examined relationships involving Orbit:

- Scatter plots of Flight Number vs. Orbit
- Payload Mass vs. Orbit Type
- These plots showed which orbits are more commonly used with different payloads and mission stages

Methodology

- Exploratory data analysis (EDA) using SQL

SQL Analysis:

- Queried unique launch sites
- Identify boosters with the highest payloads
- Explored payload mass statistics by customer and booster version
- Count success/failure outcomes
- Group and filter launch records by site, year, and outcome

Methodology

Interactive visual analytics using Folium Maps

- Marked all SpaceX launch sites on an interactive map
- Color-coded success vs. failure outcomes
- Displayed mission outcomes by site
- Added polylines showing proximity to coastlines, railways, and highways
- Added labels and measurement markers

Methodology

Interactive visual analytics using Plotly Dash

- Enabled site selection via dropdown menu
- Used sliders to filter payload ranges dynamically
- Displayed pie charts and scatter plots for launch outcomes

Methodology

Perform predictive analysis using classification models

Prepared the dataset for modeling

- Create a column for the class
- Normalized numerical features using StandardScaler

Split the dataset into training and test sets

- Used an 80/20 train-test split to ensure robust evaluation

Trained multiple classification models

- Logistic Regression
- Support Vector Machine (SVM)
- Decision Tree

Methodology

Perform predictive analysis using classification models

Applied Grid Search for hyperparameter tuning

- Used GridSearchCV to find the best model parameters (e.g., C, kernel, max_depth)

Evaluated models

- Compared models using accuracy score
- Analyzed confusion matrices to assess prediction quality

Selected the best-performing model

- Identified the model with highest accuracy (over 83%) on test data
- Final model used for classification of landing success

Data Collection

How data sets were collected.

SpaceX launch data was collected using two primary methods:

- API calls to retrieve structured JSON data directly from SpaceX servers.
- Web scraping of Wikipedia pages using BeautifulSoup to supplement data such as booster version categories and mission details.

Both methods were implemented in Jupyter Notebooks using Python.

Data collection process.

API Endpoint (SpaceX) —▶ JSON Response —▶ Extract & Normalize —▶ DataFrame

Wikipedia HTML Page —▶ BeautifulSoup —▶ Parse Table —▶ DataFrame

Data Collection – SpaceX API

Data collection with SpaceX calls and key phrases

- SpaceX launch data was retrieved using Python's requests library.
- The /v4/launches endpoint returned structured JSON data including payload mass, launch site, booster version, and mission outcome.
- The JSON data was flattened and converted to a Pandas DataFrame for processing.

Key phrases:

- API, requests.get(),JSON normalization, Pandas, DataFrame

GitHub URL of the completed SpaceX API calls notebook

* https://github.com/YulianaKuri/Data-Driven-Insights-Project

[1] Send GET request to SpaceX API

↓

[2] Receive JSON response

↓

[3] Normalize JSON into flat structure

↓

[4] Load into Pandas DataFrame

Data Collection - Scraping

Web Scraping Process Overview

- Extracted additional launch data (such as booster version and orbit info) from Wikipedia using Python's requests and BeautifulSoup libraries.
- Parsed HTML tables and cleaned the extracted data before combining it with the API dataset.
- Ensured column consistency and aligned scraped data with API attributes.

Key Phrases:

- HTML parsing, requests.get(), BeautifulSoup, .find_all(), pandas.read_html()

GitHub URL of the completed web scraping notebook:

- https://github.com/YulianaKuri/Data-Driven-Insights-Project

[1] Wikipedia URL

↓

[2] Send GET Request

↓

[3] Parse HTML Table (with BeautifulSoup)

↓

[4] Clean and Format the Data

↓

[5] Load into Pandas DataFrame

Data Wrangling

How the Data Was Processed

- Merged API and scraped datasets into a unified DataFrame.
- Removed irrelevant or duplicate rows and handled missing values.
- Converted data types (e.g., dates, payload mass) to appropriate formats.
- Created new columns such as:
  - Class (binary: 0 = fail, 1 = success)
  - Payload Mass Category (grouped range of payloads)
  - Booster Version Category (e.g., F9 v1.1, FT, B4)
  - LandingOutcome

Key Phrases:

- Data cleaning, feature engineering, type conversion, column renaming, merging

GitHub URL of data wrangling notebook:

- https://github.com/YulianaKuri/Data-Driven-Insights-Project

Data Wrangling

Flowchart

Raw API Data + Scraped Data

↓

Merge & Align Columns

↓

Drop Nulls / Convert Data Types

↓

Feature Engineering

↓

Cleaned Analytical DataFrame

EDA with Data Visualization

Summary of Plotted Charts and Purpose

- Scatter Plots: To visualize relationships between variables (e.g., payload mass vs. launch success).
- Bar Charts: To analyze success rates across different orbit types.
- Line Charts: To track yearly trends in launch outcomes.

Each chart was selected to highlight correlations, success patterns, and mission outcomes visually.

GitHub URL of EDA with visualization notebook

- https://github.com/YulianaKuri/Data-Driven-Insights-Project

EDA with SQL

SQL queries performed

Retrieved all distinct launch sites from the database

- SELECT DISTINCT Launch_Site FROM SPACEX;

Filtered launch records by site prefix (e.g., 'KSC%')

- SELECT * FROM SPACEX WHERE Launch_Site LIKE 'KSC%';

Computed total payload mass for NASA customers

- SELECT SUM(Payload_Mass__KG_) FROM SPACEX WHERE Customer='NASA';

Calculated average payload for booster version 'F9 v1.1'

- SELECT AVG(Payload_Mass__KG_) FROM SPACEX WHERE Booster_Version='F9 v1.1';

Identified earliest successful ground landing

- SELECT MIN(Date) FROM SPACEX WHERE Landing_Outcome = 'Success (ground pad)';

EDA with SQL

SQL queries performed

Filtered missions with payload between 4000–6000kg and successful drone ship landings

- SELECT Booster_Version FROM SPACEX WHERE Landing_Outcome='Success (drone ship)' AND Payload_Mass__KG_ BETWEEN 4000 AND 6000;

Counted landing outcomes grouped by success/failure

- SELECT Landing_Outcome, COUNT(*) FROM SPACEX GROUP BY Landing_Outcome;

Ranked landing outcome frequencies in a date range

- SELECT Landing_Outcome, COUNT(*) FROM SPACEX WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY COUNT(*) DESC;

GitHub URL of the completed EDA with SQL notebook

- https://github.com/YulianaKuri/Data-Driven-Insights-Project

Build an Interactive Map with Folium

Map Objects Created and Added:

- Markers to represent each SpaceX launch site on the map with launch site name
- Circle Markers to highlight successful and failed landings at each site
- Lines (Polylines) connecting each site to nearby features like coastlines, railways, and highways
- Labels and Popups showing detailed info (e.g., distance, outcome, coordinates)

Build a Dashboard with Plotly Dash

Why These Objects Were Added

- To visualize launch site geography interactively
- To measure and display distances to nearby infrastructure
- To distinguish outcomes visually (e.g., success vs. failure)
- To enrich spatial analysis of potential launch site advantages

GitHub URL of Dashboard with Plotly Dash notebook

https://github.com/YulianaKuri/Data-Driven-Insights-Project

Build a Dashboard with Plotly Dash

Dashboard Components Added:

- Pie Chart Displays launch success vs. failure rate per site. Helps visualize overall mission outcomes clearly.
- Scatter Plot Shows correlation between payload mass and launch success, segmented by booster version category (e.g., FT, B4).
- Dropdown Filter Allows user to select a specific launch site (e.g., VAFB SLC-4E, CCAFS LC-40) and dynamically update all charts.
- Payload Range Slider Filters launch data by payload mass (0–10,000kg), letting users observe how success probability varies with payload.

Why These Components Were Included:

- To provide users with real-time interactive analysis of SpaceX mission data.
- To identify visual patterns between payload, booster version, and success outcomes.
- To enhance data exploration through user-friendly controls (dropdowns and sliders).

GitHub URL of Dashboard with Plotly Dash notebook

- https://github.com/YulianaKuri/Data-Driven-Insights-Project

Predictive Analysis (Classification)

1. Feature Selection and Preprocessing

- A new column Class was created to indicate landing success (1) or failure (0).
- Standardized numerical features using StandardScaler to normalize input ranges.

2. Data Splitting

Used train_test_split() to divide the dataset:

- 80% for training
- 20% for testing

Ensured the class distribution was balanced for fair model evaluation.

Predictive Analysis (Classification)

3. Model Training: Trained three different classifiers using sklearn:

*Logistic Regression – linear decision boundary

*Support Vector Machine (SVM) – effective in high-dimensional space

*Decision Tree Classifier – interpretable model with branching logic

*K-Nearest Neighbors (KNN) – classifies based on the majority label of the k closest data points

4. Hyperparameter Tuning with GridSearchCV: For each model, a range of hyperparameters were tested to optimize accuracy:

*SVM: kernel, C, gamma

*Decision Tree: max_depth, criterion, splitter, max_features, min_samples_leaf, min_samples_split

*Logistic Regression: penalty, C, solver

*K-Nearest Neighbors (KNN): n_neighbors, algorithm, p

Used 10-fold cross-validation to ensure robust hyperparameter selection.

Predictive Analysis (Classification)

5. Model Evaluation

Evaluated models using:

*Accuracy Score

*Confusion Matrix

6. Final Model Selection

Selected best model based on generalization performance on the test set.

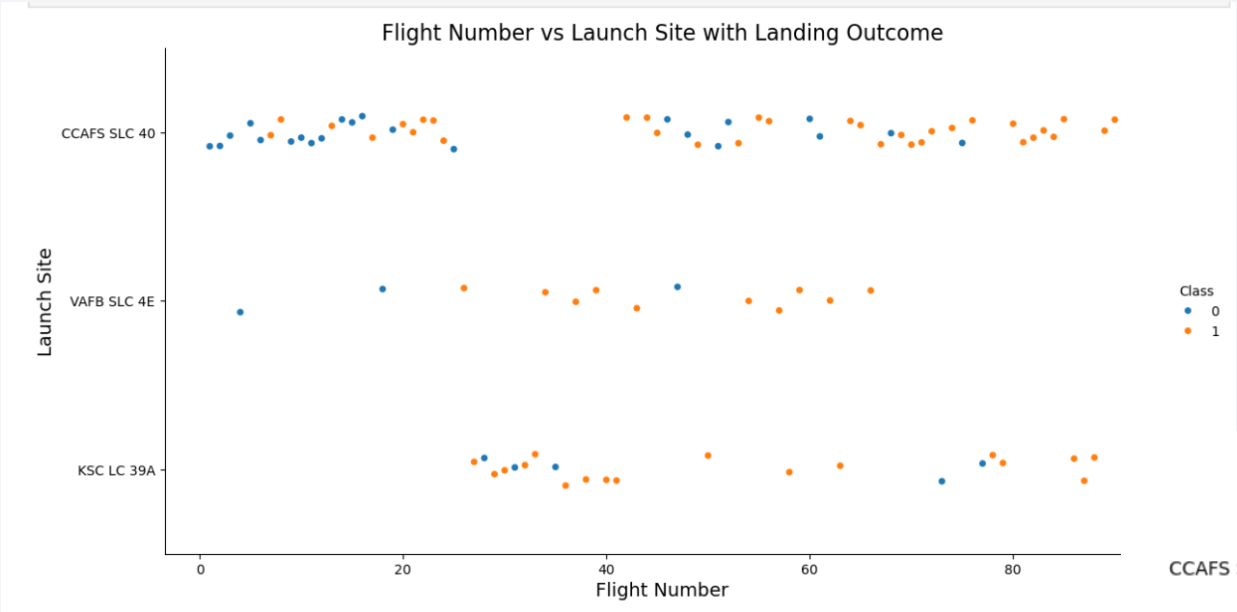
This model was used to predict successful first-stage landings of Falcon 9 rockets.

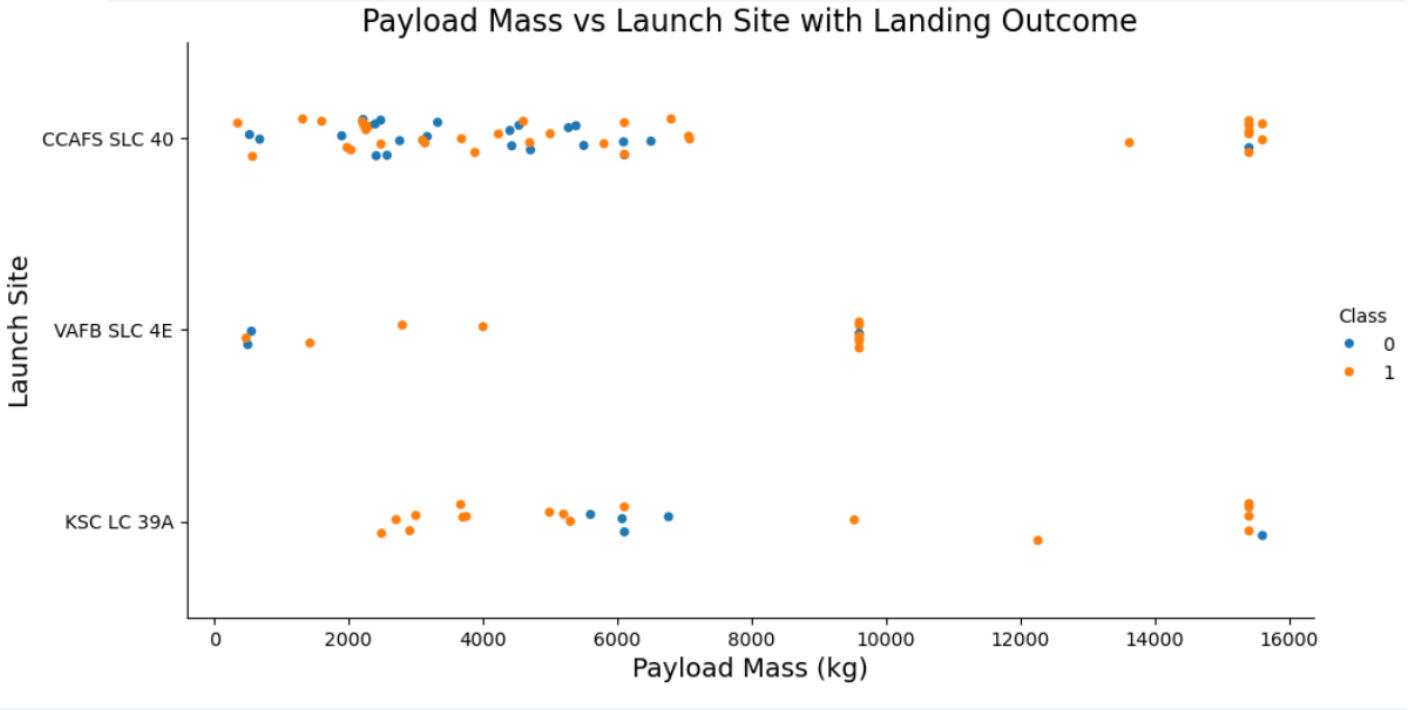
GitHub URL of Dashboard with Plotly Dash notebook

https://github.com/YulianaKuri/Data-Driven-Insights-Project

Results

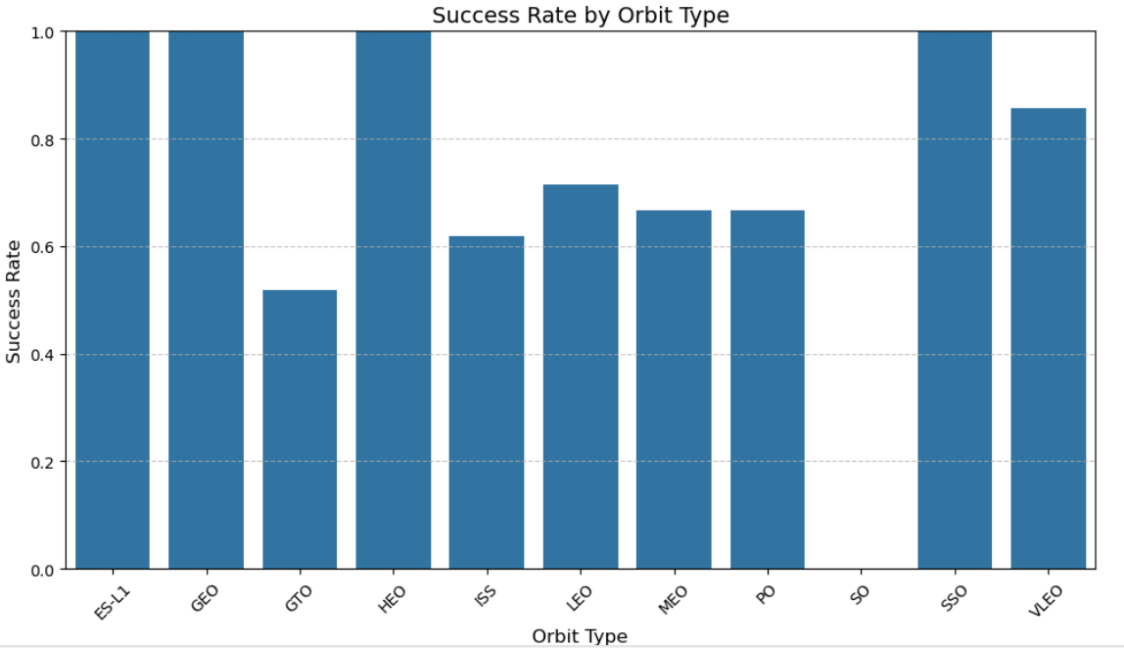
Exploratory data analysis results

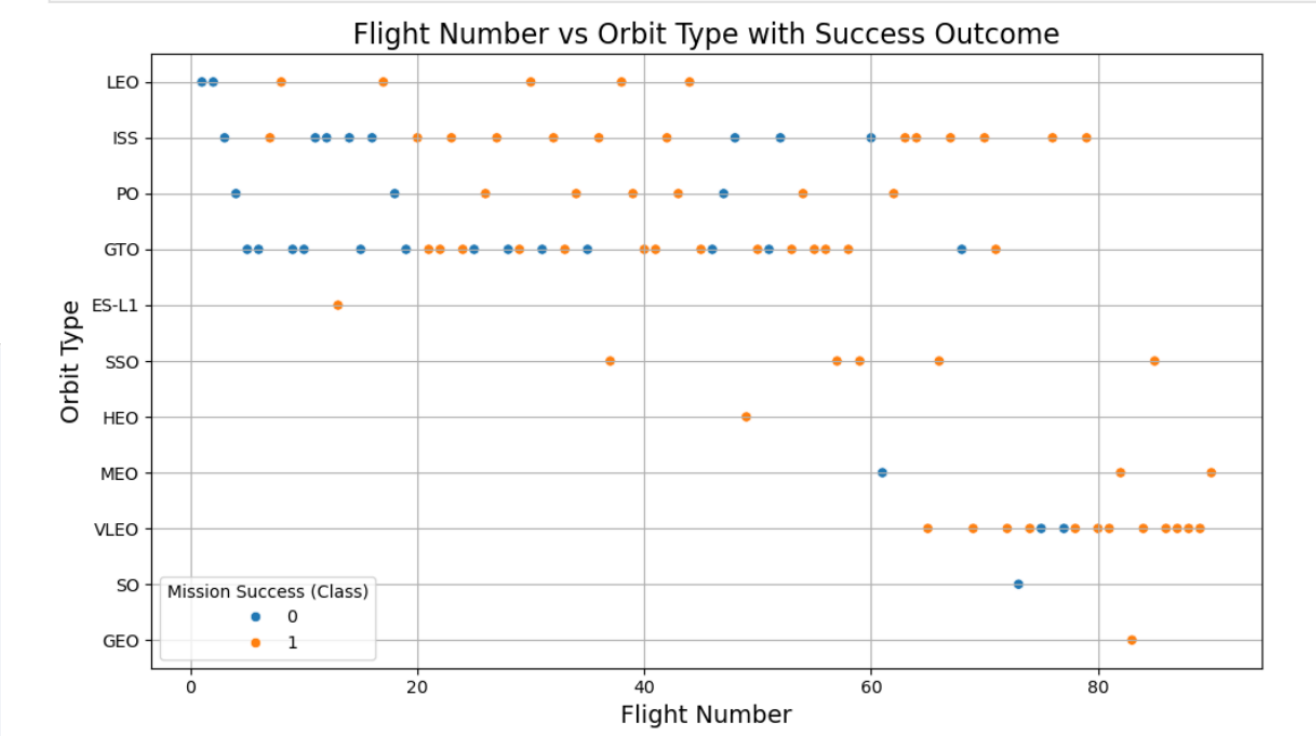




Results

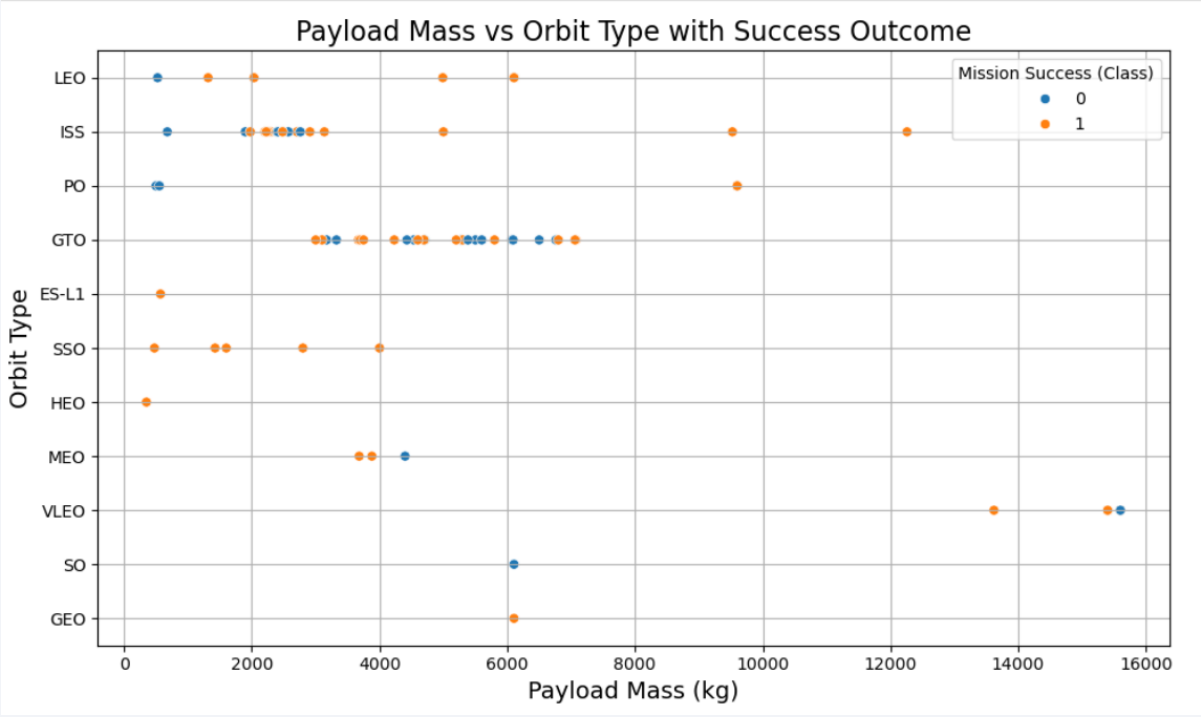
Exploratory data analysis results

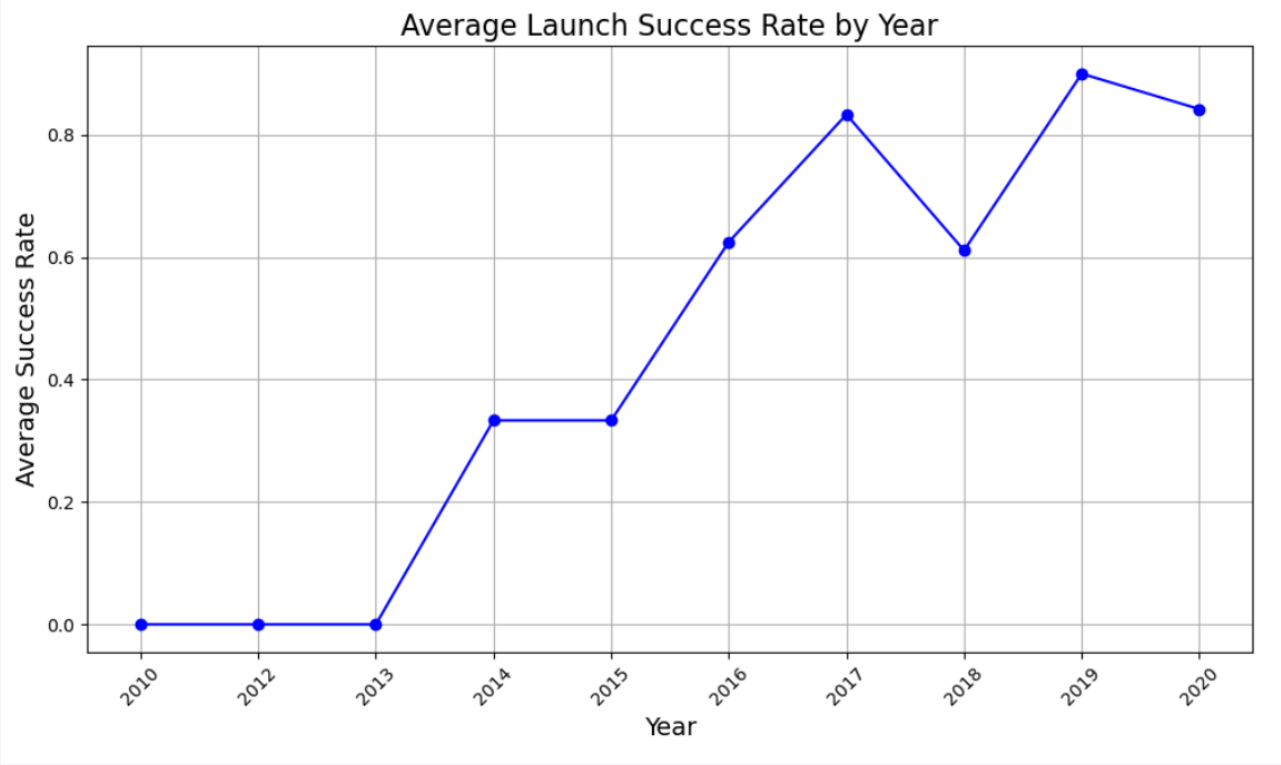




Results

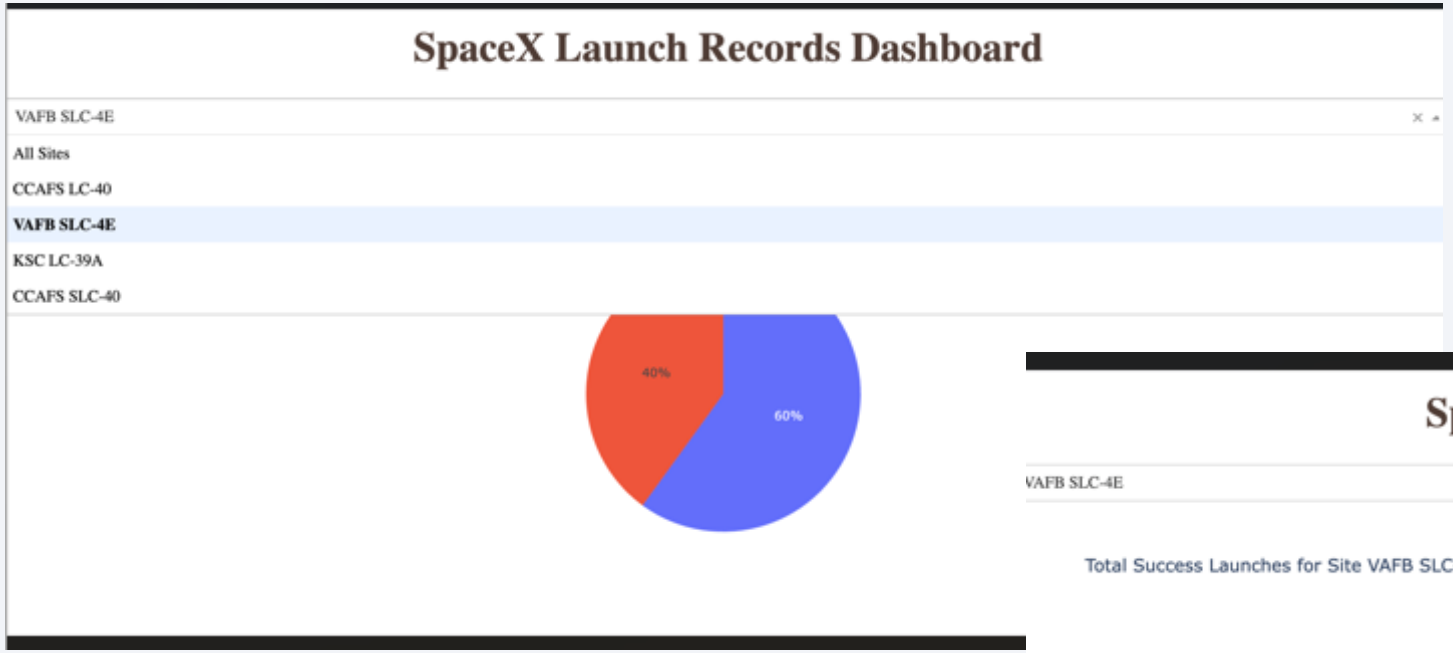
Exploratory data analysis results

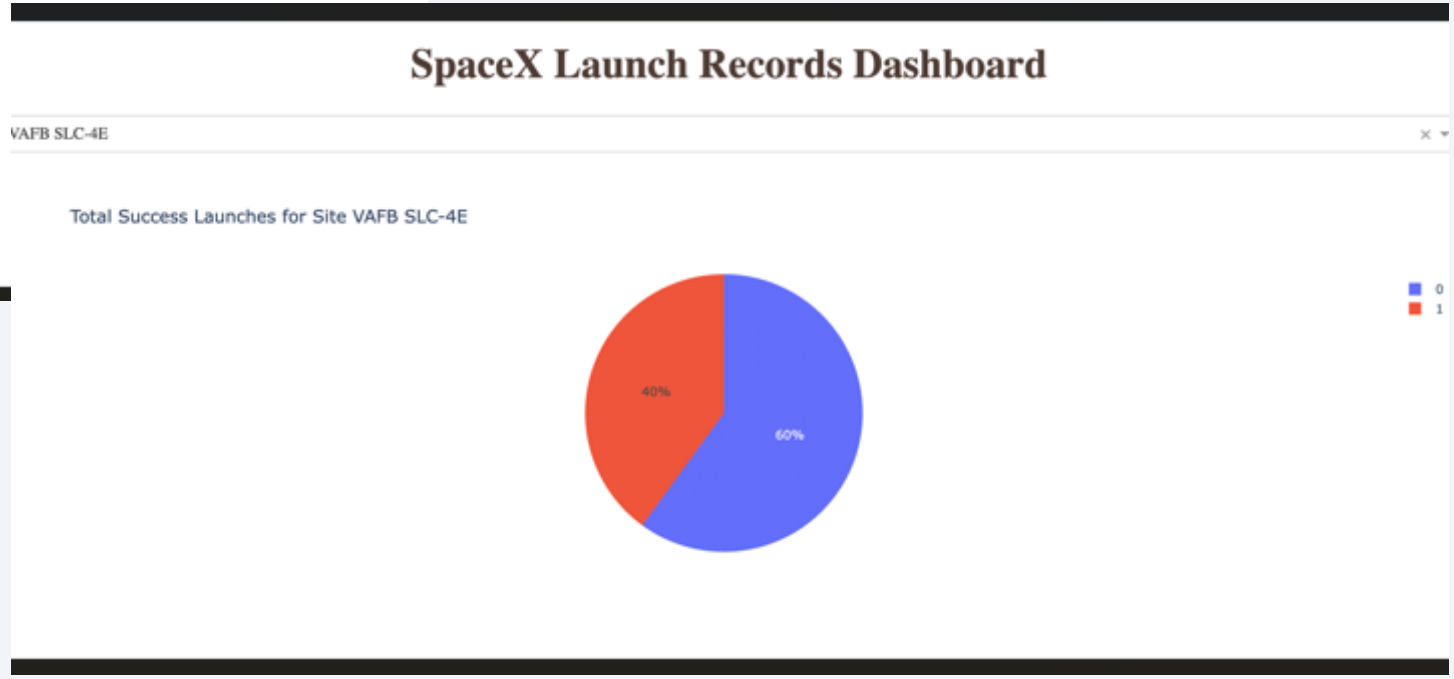




Results

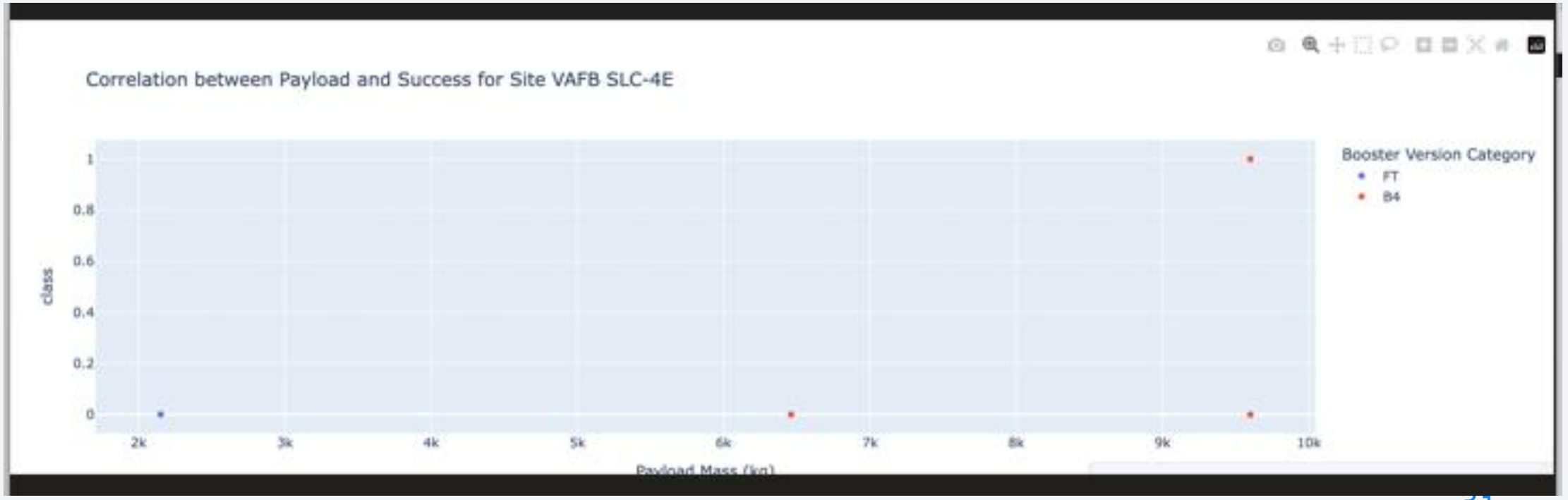
Interactive analytics demo in screenshots





Results

Interactive analytics demo in screenshots



Results

Predictive analysis results

Modelo	Train Accuracy	Test Accuracy
Logistic Regression	0.8464	**0.8333**
SVM	0.8482	**0.8333**
Decision Tree	0.8625	0.7222
KNN	0.8482	**0.8333**

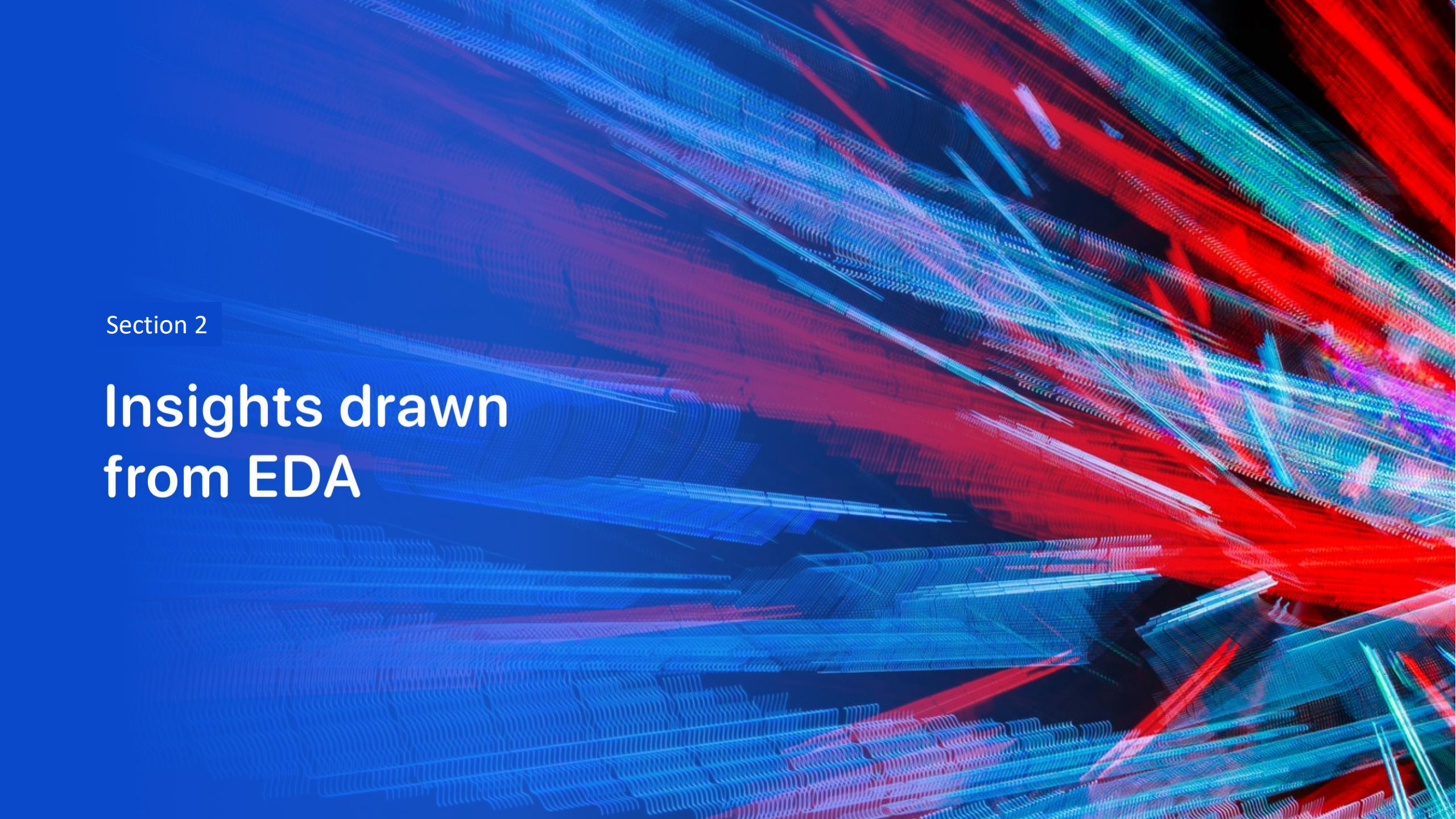
Model Selection Summary

Based on the results:

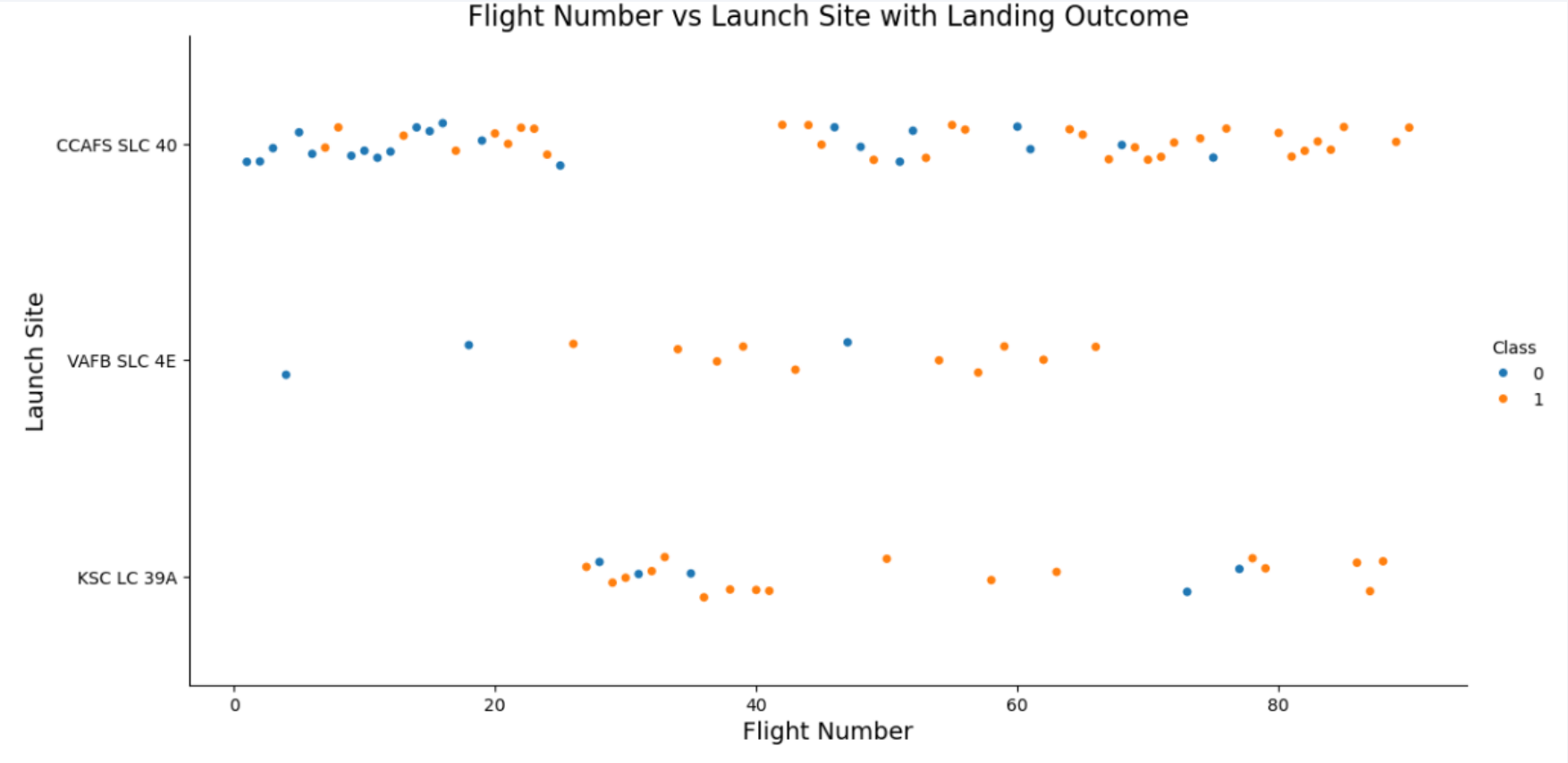
- Logistic Regression, SVM, and KNN all achieved a high test accuracy of 0.8333.
- Decision Tree had the highest training accuracy (0.8625) but significantly lower test accuracy (0.7222), indicating overfitting.

Support Vector Machine (SVM) is recommended as the best model due to:

- Strong performance in high-dimensional spaces
- Consistent training and test accuracy
- Effective tuning using GridSearchCV


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



Flight Number vs. Launch Site

Explanation

This scatter plot displays the relationship between flight number and launch site, along with the landing outcome (success or failure).

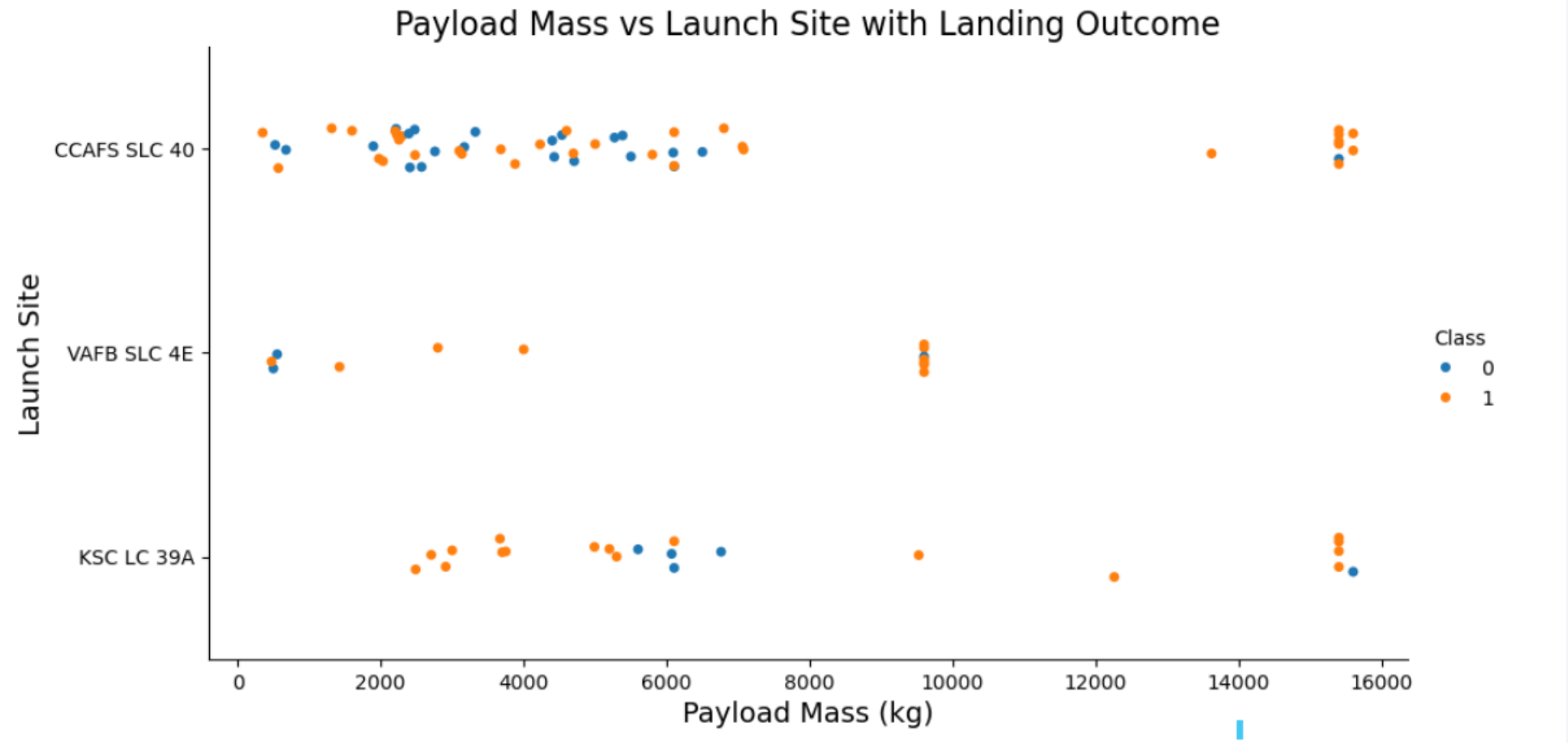
- The Y-axis shows the three different SpaceX launch sites:
  - CCAFS SLC 40
  - VAFB SLC 4E
  - KSC LC 39A
- The X-axis represents the flight number (chronological order of launches).
- The color of each dot indicates the landing outcome:
  - Orange (1): Successful landing
  - Blue (0): Failed landing

Flight Number vs. Launch Site

General Observation:

- Flight number is loosely correlated with landing success. Earlier flights across all sites had more failed landings, while later flights show a trend toward higher success rates.
- This implies SpaceX's landing technology and mission execution have significantly improved over time, especially at high-frequency launch sites like CCAFS SLC 40.

Payload vs. Launch Site



Payload vs. Launch Site

Explanation

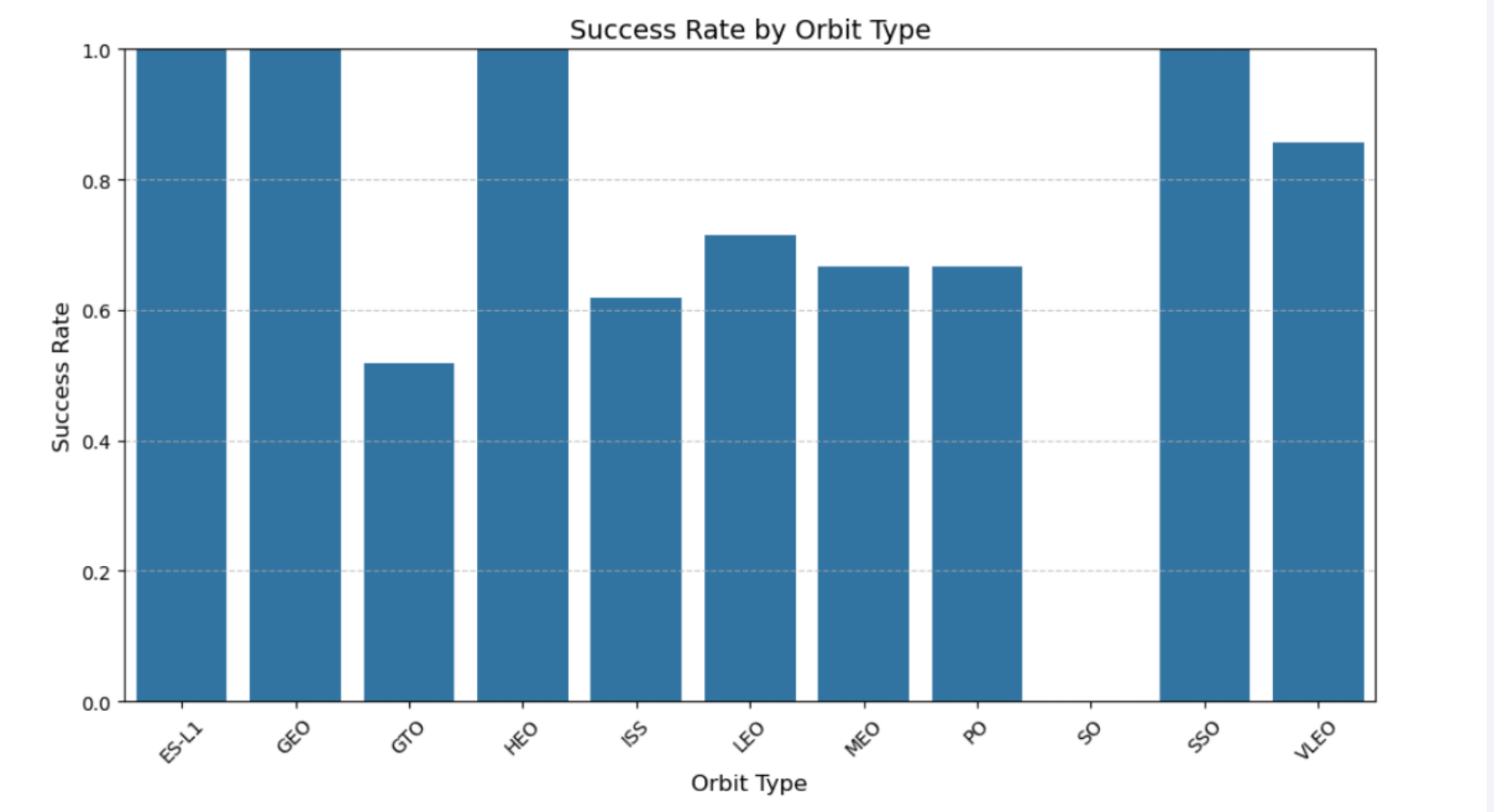
This scatter plot shows the relationship between payload mass and launch site, along with the landing outcome of Falcon 9 launches.

- The X-axis represents the payload mass (in kilograms).
- The Y-axis shows the three SpaceX launch sites:
  - CCAFS SLC 40
  - VAFB SLC 4E
  - KSC LC 39A
- Each point represents a launch, and the color indicates whether the landing was:
  - Class 0 = Unsuccessful
  - Class 1 = Successful

Insights

- Across all launch sites, there is a mix of successful and failed landings, but successful landings ( ) appear more frequently for payloads under 6000 kg.
- CCAFS SLC 40 has the highest number of launches across a wide payload range.
- At higher payload masses (above 10,000 kg), most launches seem successful, especially at CCAFS SLC 40.

Success Rate vs. Orbit Type



Success Rate vs. Orbit Type

Explanation

This bar chart illustrates the success rate of Falcon 9 landings for various orbital destinations.

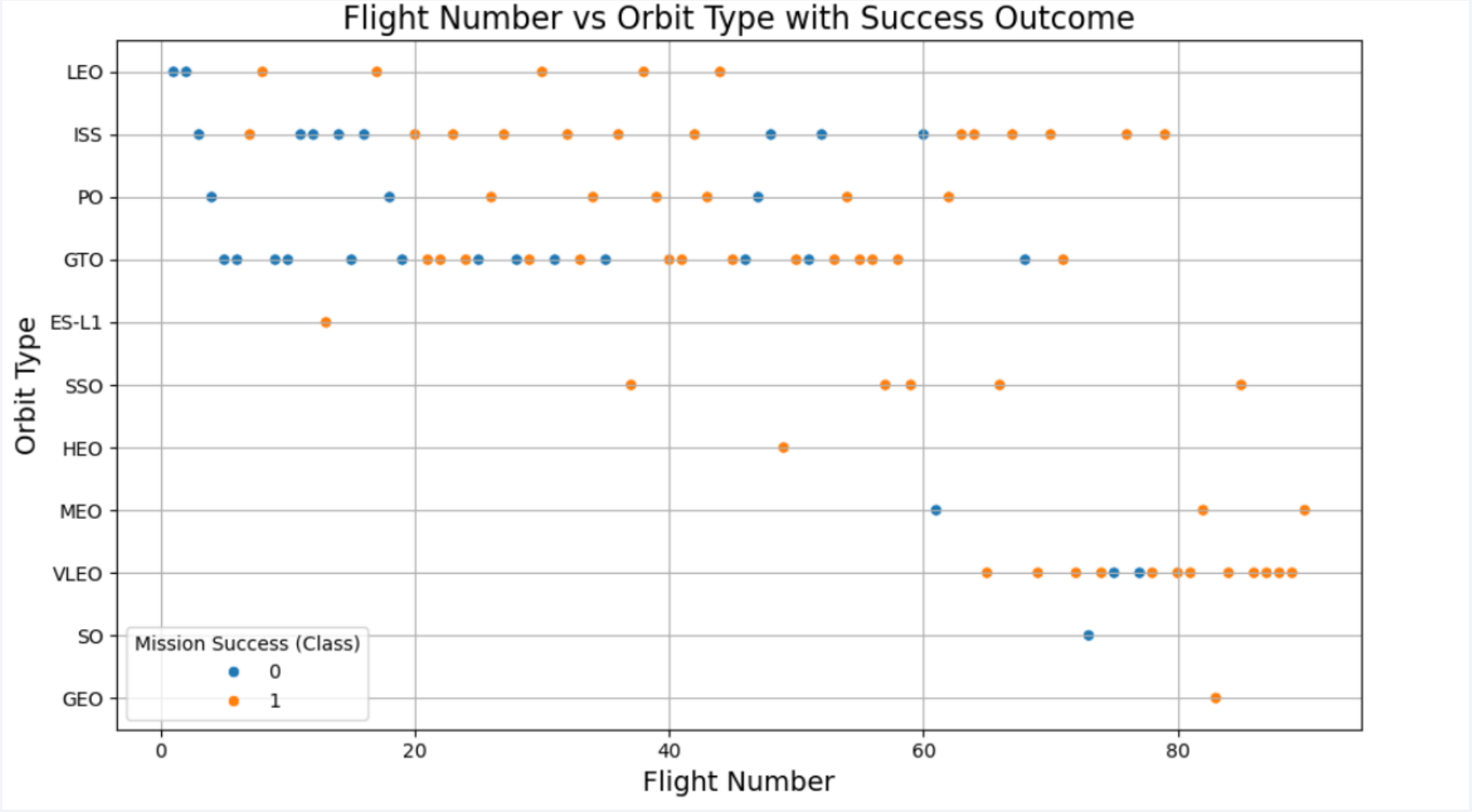
- The X-axis represents the different orbit types, such as GEO (Geostationary), ISS (International Space Station), LEO (Low Earth Orbit), etc.
- The Y-axis shows the success rate, ranging from 0 to 1 (i.e., 0% to 100%).

Insights:

- Orbit types like GEO, ES-L1, HEO, and SSO show a 100% success rate, indicating highly reliable landings when targeting these orbits.
- GTO (Geostationary Transfer Orbit) has a lower success rate (~0.52), suggesting more challenging missions.
- Most other orbits (LEO, ISS, MEO, VLEO, etc.) have moderate success rates between 60% and 85%.

This chart helps identify which orbital missions are more likely to result in successful first-stage landings, providing valuable input for mission planning and risk assessment.

Flight Number vs. Orbit Type



Flight Number vs. Orbit Type

Explanation

This scatter plot visualizes the relationship between flight number, orbit type, and mission success for Falcon 9 launches.

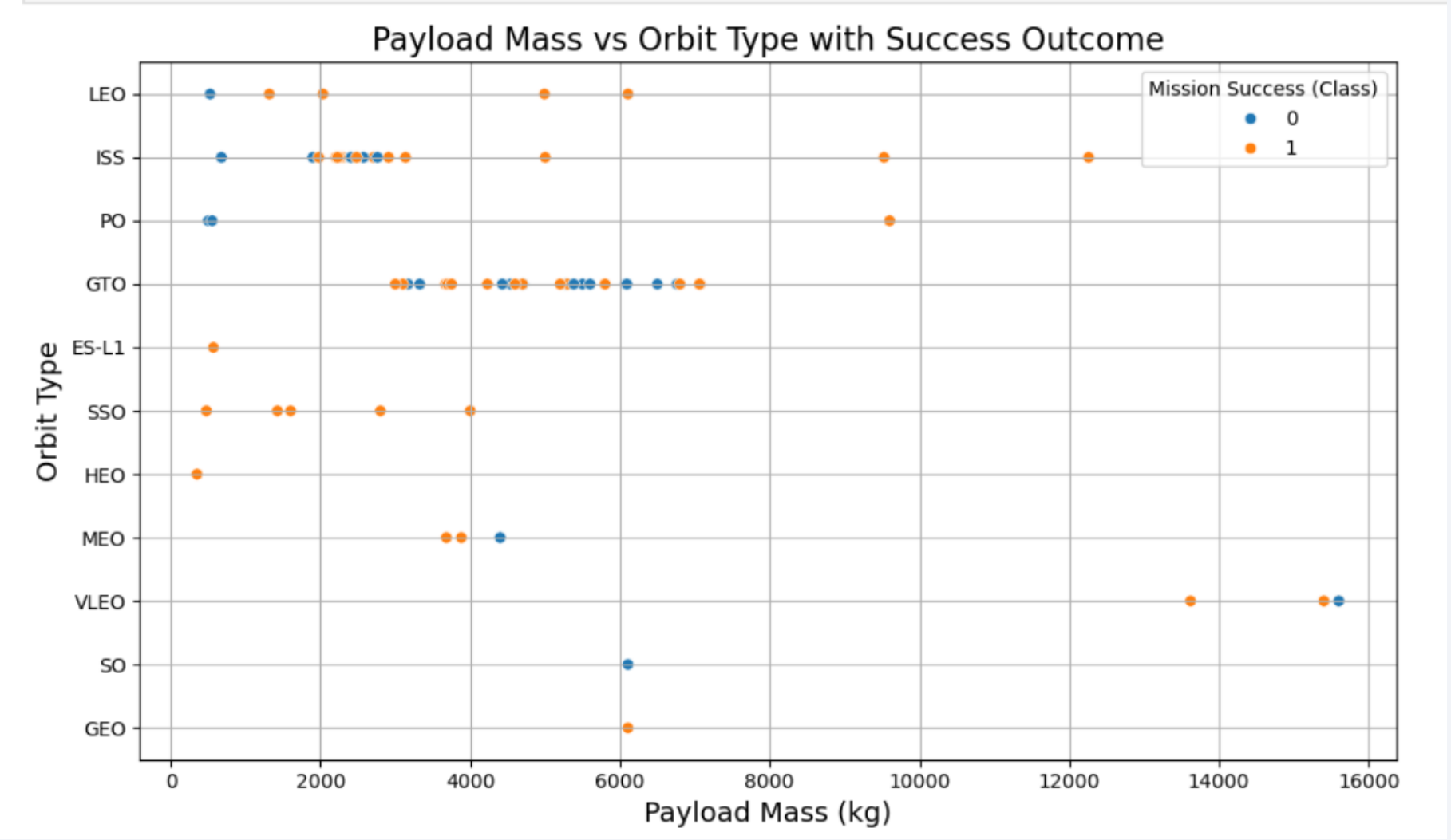
- The X-axis represents the flight number (chronological order of launches).
- The Y-axis shows different orbit types targeted by the missions (e.g., LEO, GTO, ISS).
- Each dot represents a launch:
  - Orange dots (1) indicate successful landings.
  - Blue dots (0) indicate failed landings.

Key Observations

- Over time, there's a visible trend toward more successful launches (more orange dots appear at higher flight numbers).
- Some orbits like SSO, GEO, SE-L1 and HEO have only successful landings.
- GTO, VLEO and the others show a mix of both successful and failed attempts, indicating higher variability.

This chart helps analyze how success rates vary by orbit type and whether experience (number of flights) improves performance over time.

Payload vs. Orbit Type



Payload vs. Orbit Type

Explanation

This scatter plot illustrates the relationship between payload mass, orbit type, and the landing success outcome of SpaceX Falcon 9 missions.

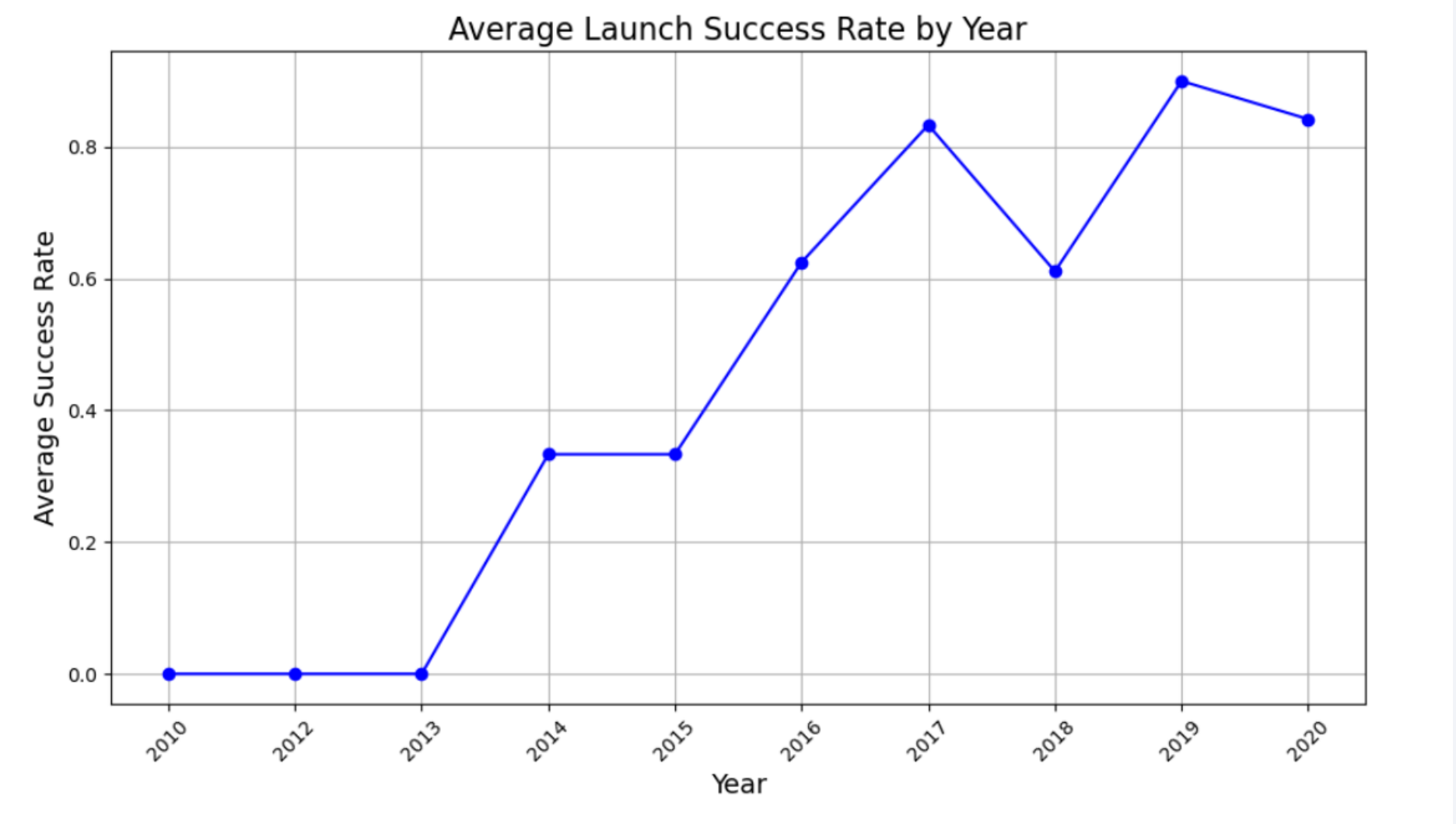
- The X-axis shows the payload mass in kilograms.
- The Y-axis displays the type of orbit targeted (e.g., LEO, GTO, ISS).
- Each dot represents a launch:
  - Orange dots (1):Successful landings.
  - Blue dots (0): Unsuccessful landings.

Key Insights

- GTO and LEO orbits have a wide range of payload masses and show a mix of both successful and failed missions.
- SSO and HEO orbits show mostly successful missions, even with smaller payloads.
- For higher payloads (above ~10,000 kg), the number of successful missions decreases slightly, indicating greater difficulty.

This chart helps identify whether certain orbit types or payload masses influence the likelihood of a successful booster landing.

Launch Success Yearly Trend



Launch Success Yearly Trend

Explanation

This line chart shows the average success rate of SpaceX Falcon 9 launches from 2010 to 2020.

- X-axis: Years from 2010 to 2020
- Y-axis: Average success rate of launches (from 0 to 1)

Key Insights

- From 2010 to 2013, SpaceX had no successful landings (success rate = 0).
- Starting 2014, success rates began to rise, indicating improved performance.
- The rate peaked in 2019, nearing 0.9, showing high reliability in launches.
- A slight drop is seen in 2020, but the success rate still remained above 0.8.

Overall, the graph highlights SpaceX's continuous improvement in launch success over the decade.

All Launch Site Names

Unique Launch Sites:

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

All Launch Site Names

Query result

```
import sqlite3
import pandas as pd
# Connect to the SQLite database
conn = sqlite3.connect("my_data1.db")
# Execute the SQL query
query = 'SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE'
df_result = pd.read_sql_query(query, conn)
# Display the resultprint("Unique Launch Sites:")
print(df_result)
# Close the connection
conn.close()
```

Explanation

This Python code connects to an SQLite database called my_data1.db and runs an SQL query to retrieve all unique launch sites from the table SPACEXTABLE. It uses the SELECT DISTINCT statement to ensure only distinct values from the "Launch_Site" column are returned. The result is stored in a Pandas DataFrame and then printed. Finally, the database connection is closed to free up resources

Launch Site Names Begin with 'KSC'

- 5 records where launch sites' names start with `KSC`

Date	Time (UTC)	Booster Version	Launch Site	Payload	Payload Mass (KG)	Orbit	Customer	Mission Outcome	Landing Outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Launch Site Names Begin with 'KSC'

Query presentation

```
# Connect to the SQLite database
conn = sqlite3.connect("my_data1.db")
# Execute the SQL query to get 5 records where Launch_Site begins with 'KSC’
query = '’’
SELECT *
FROM SPACEXTABLE
WHERE "Launch_Site" LIKE 'KSC%’
LIMIT 5'’’
df_result = pd.read_sql_query(query, conn)
# Display the resultprint("Records where Launch_Site begins with 'KSC’: »)
print(df_result)
# Close the connectionconn.close()
```

Explanation

This SQL query is designed to extract data from an SQLite database called my_data1.db, specifically from the table named SPACEXTABLE. The query filters and retrieves the first 5 records where the Launch_Site column starts with the prefix 'KSC' (e.g., KSC LC-39A). The result is then read into a Pandas DataFrame named df_result, which allows further analysis or display using Python. This type of query is useful when you want to analyze launch site-specific information (e.g., for Kennedy Space Center) from a larger dataset.

Total Payload Mass

Query presentation

```
import sqlite3
import pandas as pd
# Connect to the SQLite
databaseconn = sqlite3.connect("my_data1.db")
# SQL query to get the total payload mass for NASA (CRS) launches
query = "'
SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass
FROM SPACEXTABLE
WHERE Customer = 'NASA (CRS)"'
# Run the query and load the result into a DataFrame
df_result = pd.read_sql_query(query, conn)
# Display the resultprint("Total Payload Mass Carried by NASA (CRS):")
print(df_result)
# Close the connection
conn.close()
```

Total Payload Mass

Explanation

This SQL query connects to a SQLite database (my_data1.db) and calculates the total payload mass launched specifically for NASA (CRS) missions. It does this by using the SUM() function on the "PAYLOAD_MASS__KG_" column in the SPACEXTABLE. The result is stored in a Pandas DataFrame called df_result, which is then printed to display the total mass. This query is useful for analyzing how much payload NASA has transported in collaboration with SpaceX.

Total payload carried by boosters from NASA

Total Payload Mass Carried by NASA (CRS): 45596

Average Payload Mass by F9 v1.1

Query presentation

```
# Connect to the SQLite database
conn = sqlite3.connect("my_data1.db")
# SQL query to calculate the average payload mass for booster version F9 v1.1
query = '‘’
SELECT AVG("PAYLOAD_MASS__KG_") AS Average_Payload_Mass
FROM SPACEXTABLEWHERE "Booster_Version" = 'F9 v1.1'"‘’
# Run the query and load the result into a DataFrame
df_result = pd.read_sql_query(query, conn)
# Display the resultprint("Average Payload Mass for Booster Version F9 v1.1:“
)print(df_result)
# Close the connection
conn.close()
```

Average Payload Mass by F9 v1.1

Explanation

This query connects to the SQLite database my_data1.db and calculates the average payload mass for launches that used the F9 v1.1 booster version. It uses the AVG() function on the "PAYLOAD_MASS__KG_" column from the SPACEXTABLE, filtered by rows where "Booster_Version" equals 'F9 v1.1'. The result is loaded into a Pandas DataFrame and printed. This query is useful for analyzing the typical payload weight carried by a specific version of the SpaceX Falcon 9 rocket.

Average payload mass carried by booster version F9 v1.1

Average Payload Mass for Booster Version F9 v1.1: **2928.4**

First Successful Ground Landing Date

Query presentation

```
import sqlite3
import pandas as pd
# Connect to the SQLite database
conn = sqlite3.connect("my_data1.db")
# SQL query to find the earliest date of successful landing on a drone ship
query = '''
SELECT MIN(Date) AS First_Successful_Drone_Landing
FROM SPACEXTABLEWHERE "Landing_Outcome" = 'Success (drone ship)'''
# Run the query and load the result into a DataFrame
df_result = pd.read_sql_query(query, conn)
# Display the resultprint("First successful drone ship landing date:")
print(df_result)
# Close the connection
conn.close()
```

First Successful Ground Landing Date

Explanation

This script connects to the my_data1.db SQLite database and runs a SQL query that retrieves the earliest date on which a Falcon 9 first stage successfully landed on a drone ship. It does this by selecting the minimum value of the Date column from SPACEXTABLE filtered where "Landing_Outcome" equals 'Success (drone ship)'. The result is loaded into a Pandas DataFrame and printed. This query is useful for identifying the very first time SpaceX achieved a successful drone-ship recovery.

First successful drone ship landing date:

First successful drone ship landing date: 2016-04-08

Successful Drone Ship Landing with Payload between 4000 and 6000

Query presentation

```
import sqlite3
import pandas as pd
# Connect to the SQLite database
conn = sqlite3.connect("my_data1.db")
# SQL query to find booster names with successful ground pad landing and payload between 4000 and 6000 kg
query = "'
SELECT DISTINCT "Booster_Version"
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (ground pad)'
AND "PAYLOAD_MASS__KG_" > 4000
AND "PAYLOAD_MASS__KG_" < 6000"'
# Run the query and load the result into a DataFrame
df_result = pd.read_sql_query(query, conn)
# Display the result
print("Booster versions with successful ground pad landing and payload between 4000 and 6000 kg:")
print(df_result)
# Close the connection
conn.close()
```

Successful Drone Ship Landing with Payload between 4000 and 6000

Explanation

This query connects to the my_data1.db SQLite database and retrieves a list of distinct booster versions that had a successful landing on a ground pad and carried a payload mass between 4000 and 6000 kilograms. It filters the data from the SPACEXTABLE table using the conditions for landing outcome and payload range. The result is displayed in a Pandas DataFrame. This query is useful for identifying which booster models successfully performed medium-weight missions with precise landings on solid ground.

Booster versions with successful ground pad landing and payload between 4000 and 6000 kg:

Booster_Version

* F9 FT B1032.1
* F9 B4 B1040.1
* F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

Query presentation

```
import sqlite3
import pandas as pd
# Connect to the SQLite database
conn = sqlite3.connect("my_data1.db")
# SQL query to count each type of mission outcome (success and failure)
query = "'
SELECT "Mission_Outcome", COUNT(*) AS Outcome_Count
FROM SPACEXTABLEGROUP BY "Mission_Outcome""'
# Run the query and load the result into a DataFrame
df_result = pd.read_sql_query(query, conn)
# Display the result
print("Total number of mission outcomes (success and failure):")
print(df_result)
# Close the connection
conn.close()
```

Total Number of Successful and Failure Mission Outcomes

Explanation

This query connects to the my_data1.db SQLite database and counts how many times each mission outcome (such as success or failure) appears in the SPACEXTABLE. It uses the SQL GROUP BY clause to group the results by the Mission_Outcome column and then applies COUNT(*) to get the number of occurrences for each type. The result is loaded into a Pandas DataFrame and printed. This query is useful for summarizing the overall performance of space missions by showing how many were successful and how many failed.

Total number of mission outcomes (success and failure):

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

Boosters Carried Maximum Payload

Query presentation

```
import sqlite3
import pandas as pd
# Connect to the SQLite database
conn = sqlite3.connect("my_data1.db")
# SQL query using a subquery to find booster versions with the maximum payload mass
query = "'
SELECT DISTINCT "Booster_Version"
FROM SPACEXTABLE
WHERE "PAYLOAD_MASS__KG_" = (
        SELECT MAX("PAYLOAD_MASS__KG_")
        FROM SPACEXTABLE)"'
# Run the query and load the result into a DataFrame
df_result = pd.read_sql_query(query, conn)
# Display the result
print("Booster versions that carried the maximum payload mass:")
print(df_result)
# Close the connectionconn.close()
```

Boosters Carried Maximum Payload

Explanation

his query connects to the my_data1.db SQLite database and retrieves the booster versions that carried the maximum payload mass. It uses a subquery to first find the maximum value of "PAYLOAD_MASS__KG_" in the SPACEXTABLE, then selects the distinct "Booster_Version" entries that match that maximum value. The result is loaded into a Pandas DataFrame and displayed. This query is useful for identifying which booster(s) handled the heaviest payloads in SpaceX's launch history.

Booster versions that carried the maximum payload mass:

* 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

2017 Launch Records

Query presentation

```
import sqlite3
import pandas as pd
# Connect to the SQLite database
conn = sqlite3.connect("my_data1.db")
# SQL query using substr() to filter for year 2017 and extract month number
query = "'
SELECT     substr(Date, 6, 2) AS Month,    "Landing_Outcome",    "Booster_Version",    "Launch_Site"
FROM SPACEXTABLE
WHERE substr(Date, 1, 4) = '2017'  AND "Landing_Outcome" = 'Success (ground pad)"''
# Execute querydf_result = pd.read_sql_query(query, conn)
# Display the resultprint("Records for 2017 with successful ground pad landings:")
print(df_result)
# Close the connection
conn.close()
```

2017 Launch Records

Explanation

This query connects to the my_data1.db SQLite database and retrieves launch records from the year 2017 where the landing outcome was a successful landing on the ground pad. It uses the substr() function to extract the month from the "Date" column and selects relevant fields such as "Landing_Outcome", "Booster_Version", and "Launch_Site". The result is loaded into a Pandas DataFrame and displayed. This query is useful for analyzing successful ground landings during a specific year and identifying seasonal or monthly trends.

Records for 2017 with successful ground pad landings:

Month	Landing_Outcome	Booster_Version	Launch_Site
02	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
05	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
06	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
08	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
09	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
12	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40

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

Query presentation

```
import sqlite3
import pandas as pd
# SQL query to count and rank landing outcomes between two dates
query = '''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‘’
# Execute querydf_result = pd.read_sql_query(query, conn)
# Display the resultprint("Landing outcomes ranked by count between 2010-06-04 and 2017-03-20:")
print(df_result)
# Close the connection
conn.close()
```

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

Explanation

This query connects to the SQLite database my_data1.db and retrieves the count of each landing outcome from the SPACEXTABLE between the dates June 4, 2010 and March 20, 2017. It groups the results by the "Landing_Outcome" column and counts how many times each type of landing occurred in that period. The results are then sorted in descending order based on the count, showing the most frequent landing outcomes first. This is useful for analyzing and ranking the performance or frequency of different landing outcomes within a specific historical time range.

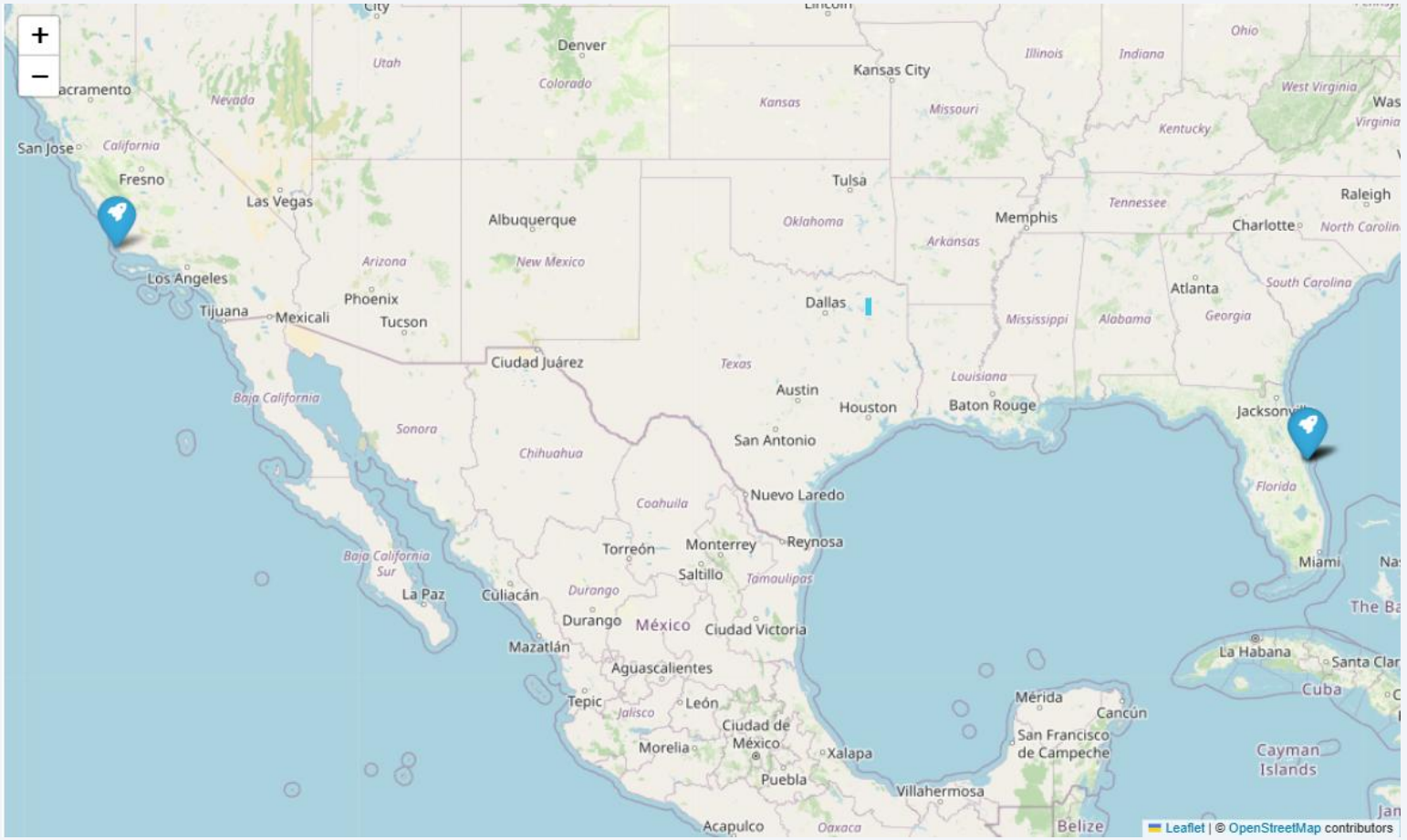
Landing outcomes ranked by count between 2010-06-04 and 2017-03-20:

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

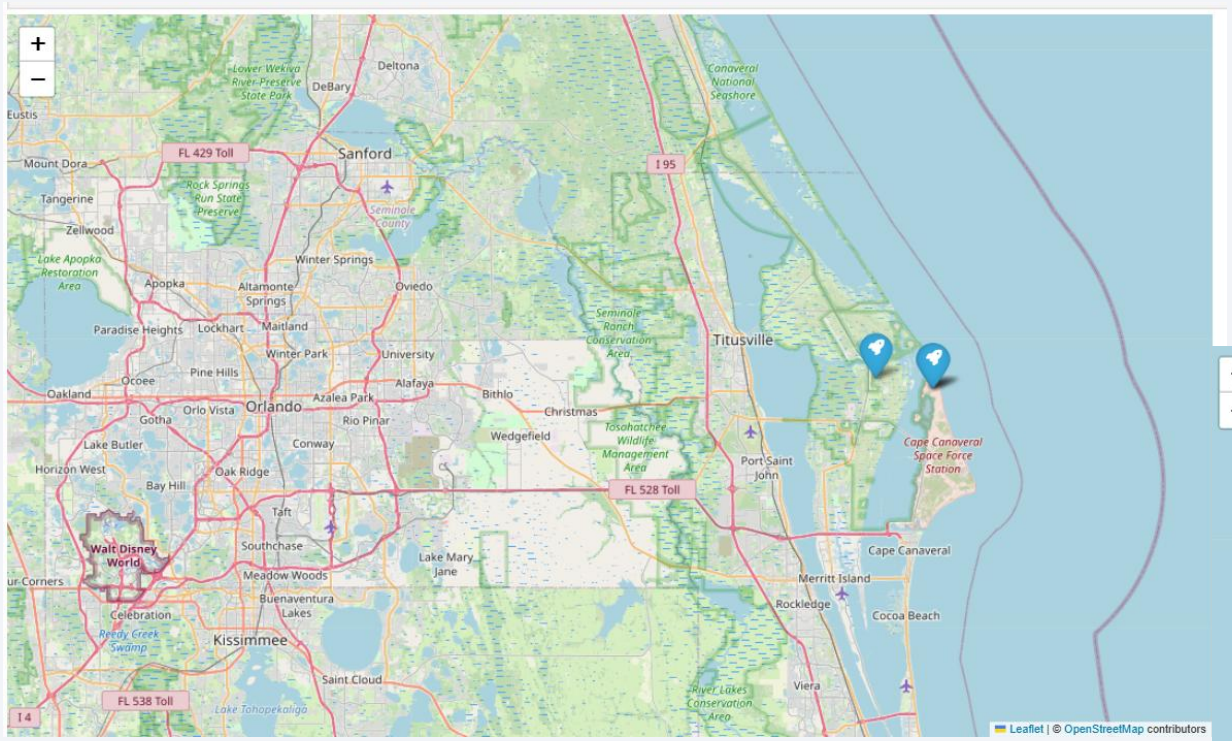
Section 3

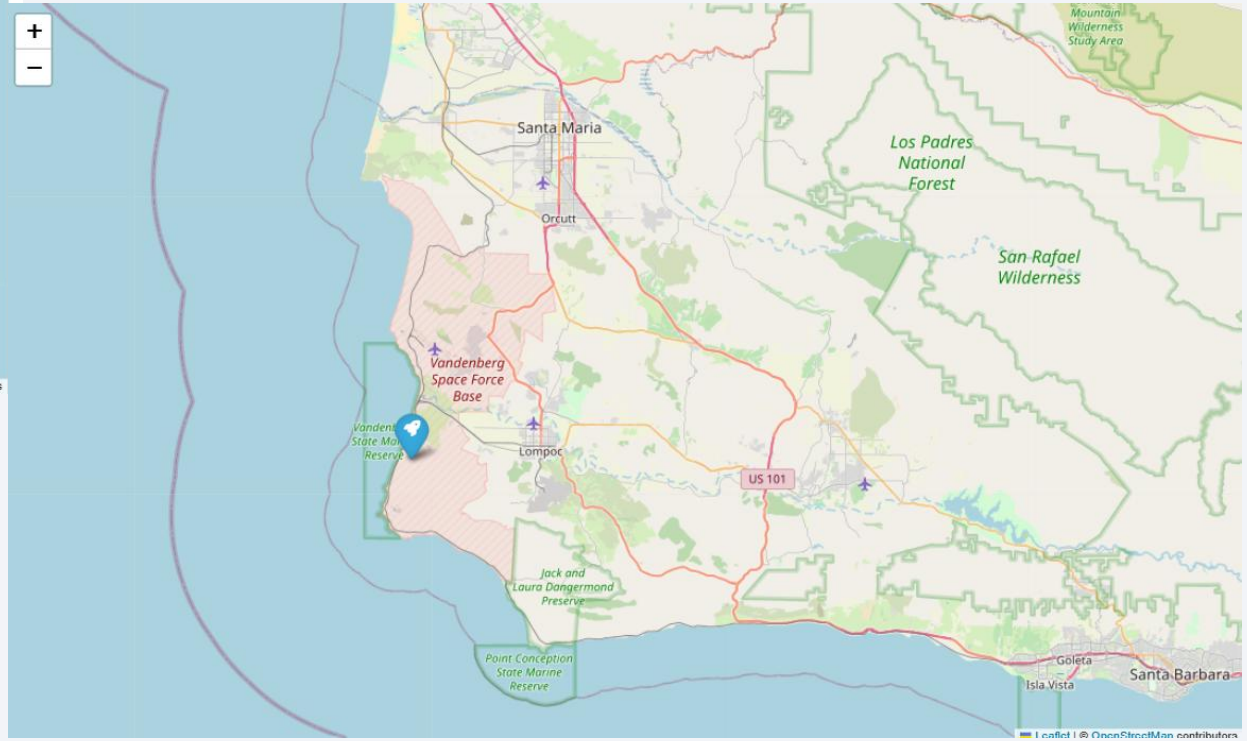
Launch Sites Proximities Analysis

Location of SpaceX Launch Sites in North America



Location of SpaceX Launch Sites in North America





Location of SpaceX Launch Sites in North America

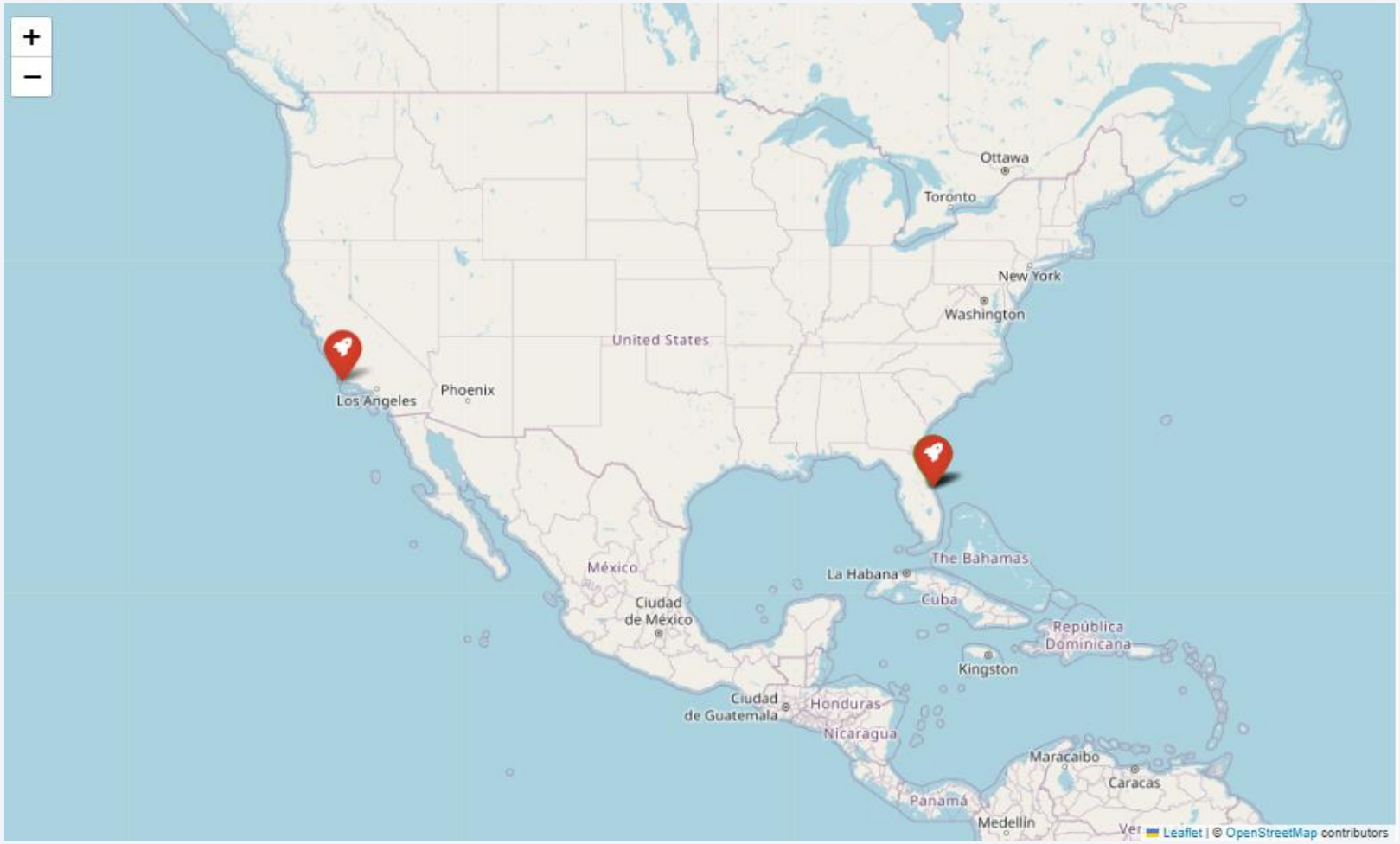
Description:

This interactive map displays the main launch sites used by SpaceX, marked using their geographic coordinates with the help of Python's folium library. Each marker represents a specific launch site and shows its name when clicked. The visualization helps identify the geographic distribution of these key facilities, which include:

1. CCAFS LC-40 (Cape Canaveral Air Force Station Launch Complex 40) located in Florida, EE. UU.
2. CCAFS SLC-40 (Space Launch Complex 40), located in Florida, EE. UU.
3. KSC LC-39A (Kennedy Space Center Launch Complex 39A) located in Merritt Island, Florida, EE. UU.
4. VAFB SLC-4E (Vandenberg Air Force Base Space Launch Complex 4E) Located in Base Area Vandenberg, California, EE. UU.

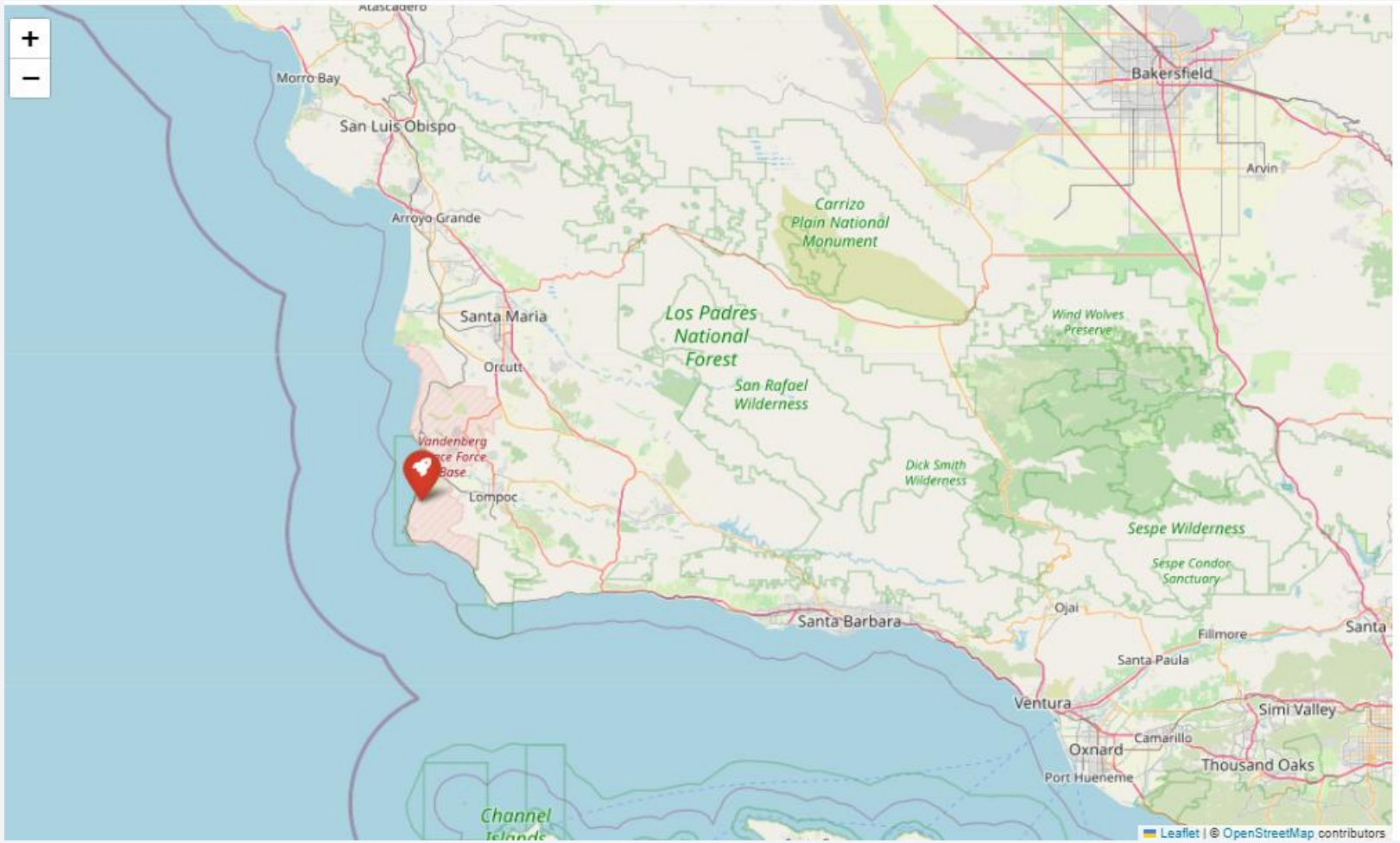
These locations are plotted using their latitude and longitude, allowing users to better understand where they are situated, even if they're not familiar with the coordinate values.

Launch Outcomes by Site – Successful and Failed SpaceX Missions



Launch Outcomes by Site – Successful and Failed SpaceX Missions

Launch Outcomes by Site – Successful and Failed SpaceX Missions

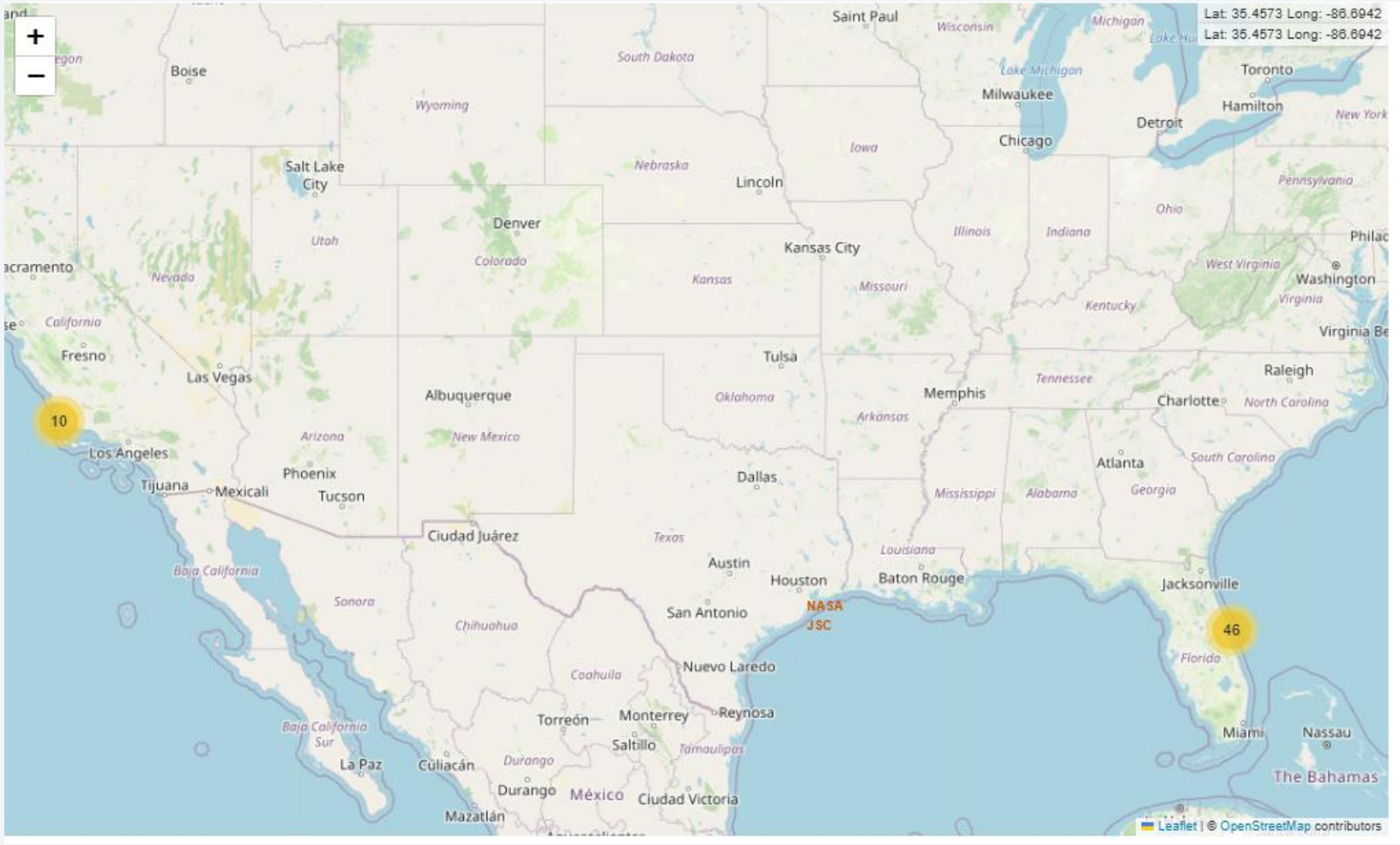


Launch Outcomes by Site – Successful and Failed SpaceX Missions

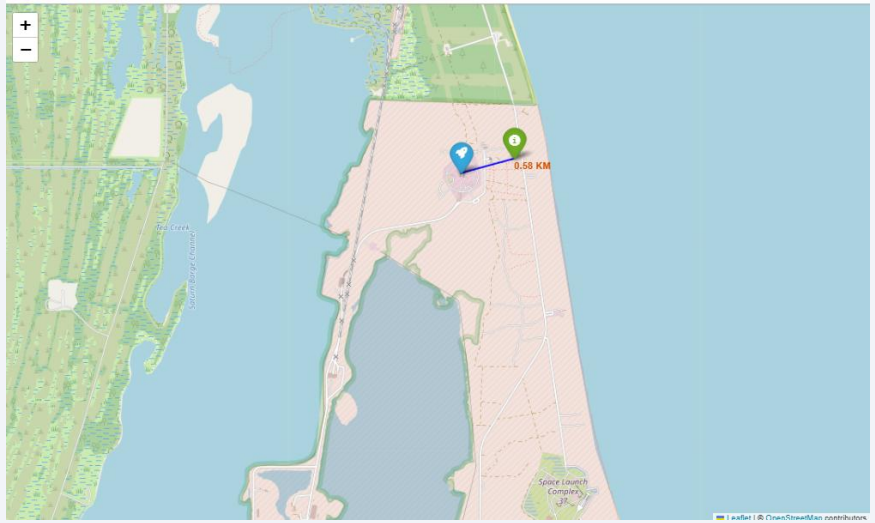
Description:

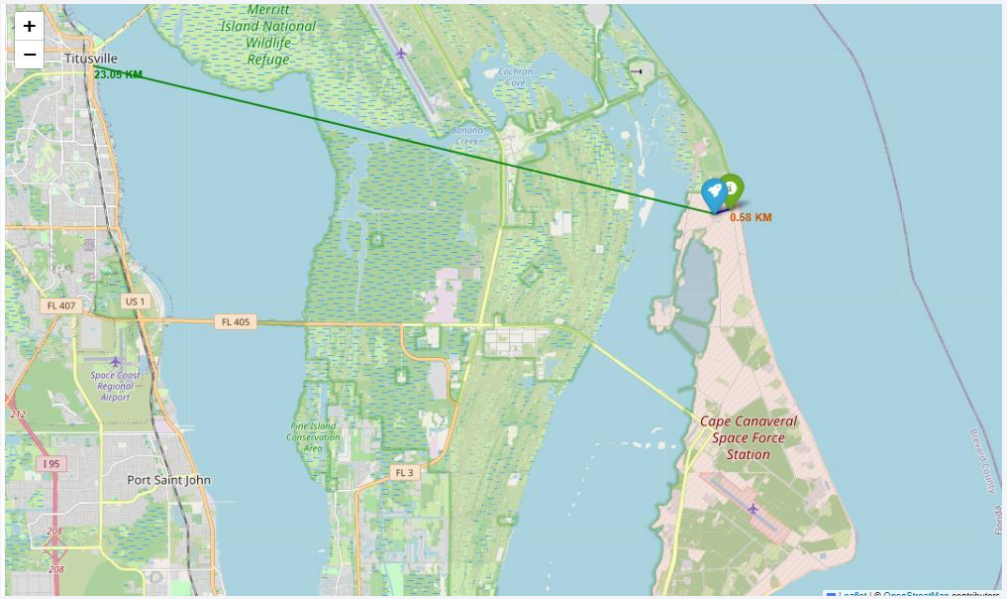
- This map displays the outcomes of individual SpaceX launches based on their geographic locations. Each marker represents a launch event, colored according to its result:
- Green marker: Successful launch (class = 1)
- Red marker: Failed launch (class = 0)
- The popup associated with each marker includes the launch site and whether the outcome was a success or failure. This visual representation allows us to quickly assess performance patterns across different launch sites and identify where failures or successes occurred more frequently.
- In the current view, the red marker indicates at least one failed launch in Florida, specifically from the Cape Canaveral or Kennedy Space Center area.

Measuring the Distance from Launch Sites to Nearby Landmarks



Measuring the Distance from Launch Sites to Nearby Landmarks

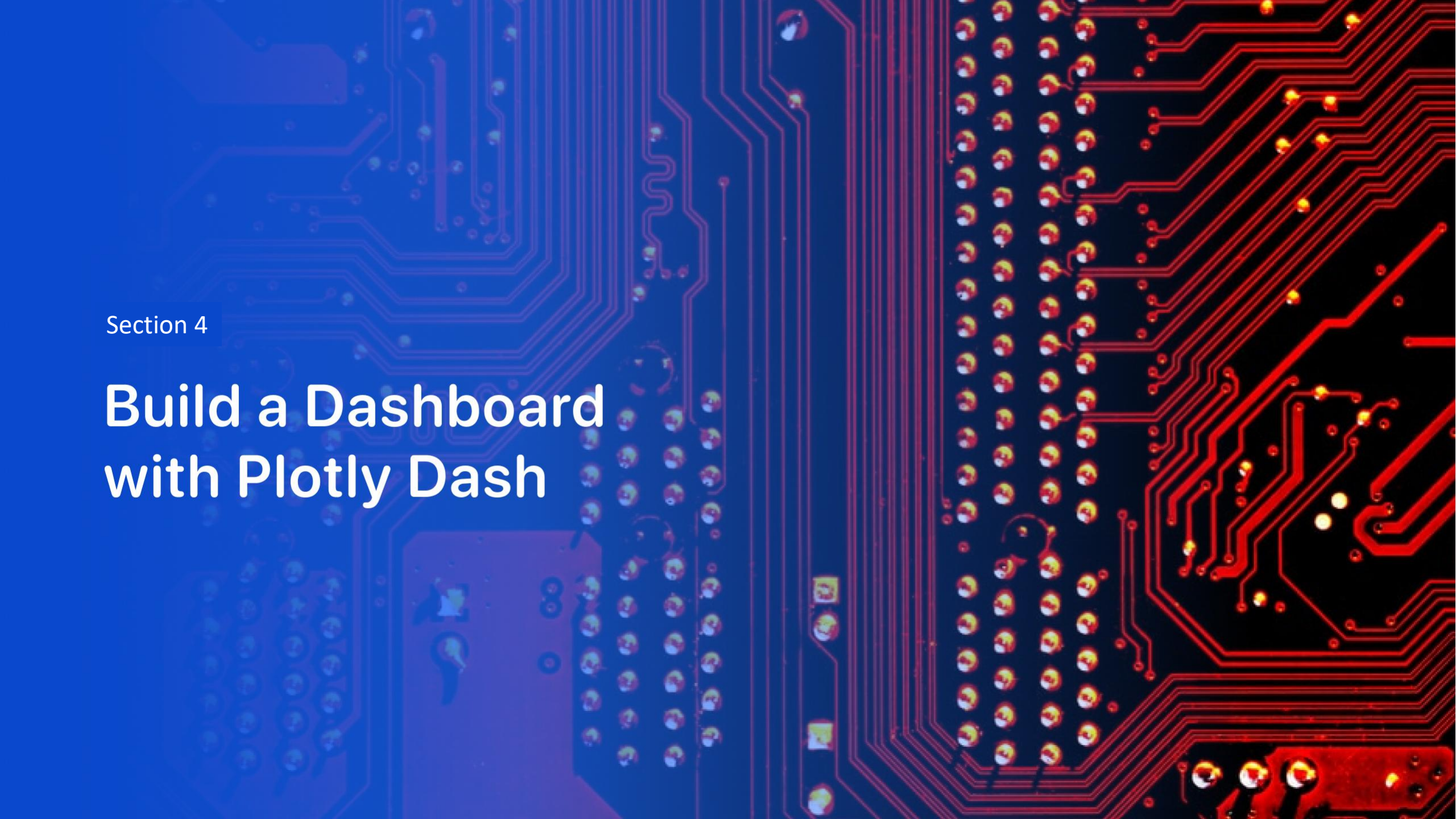




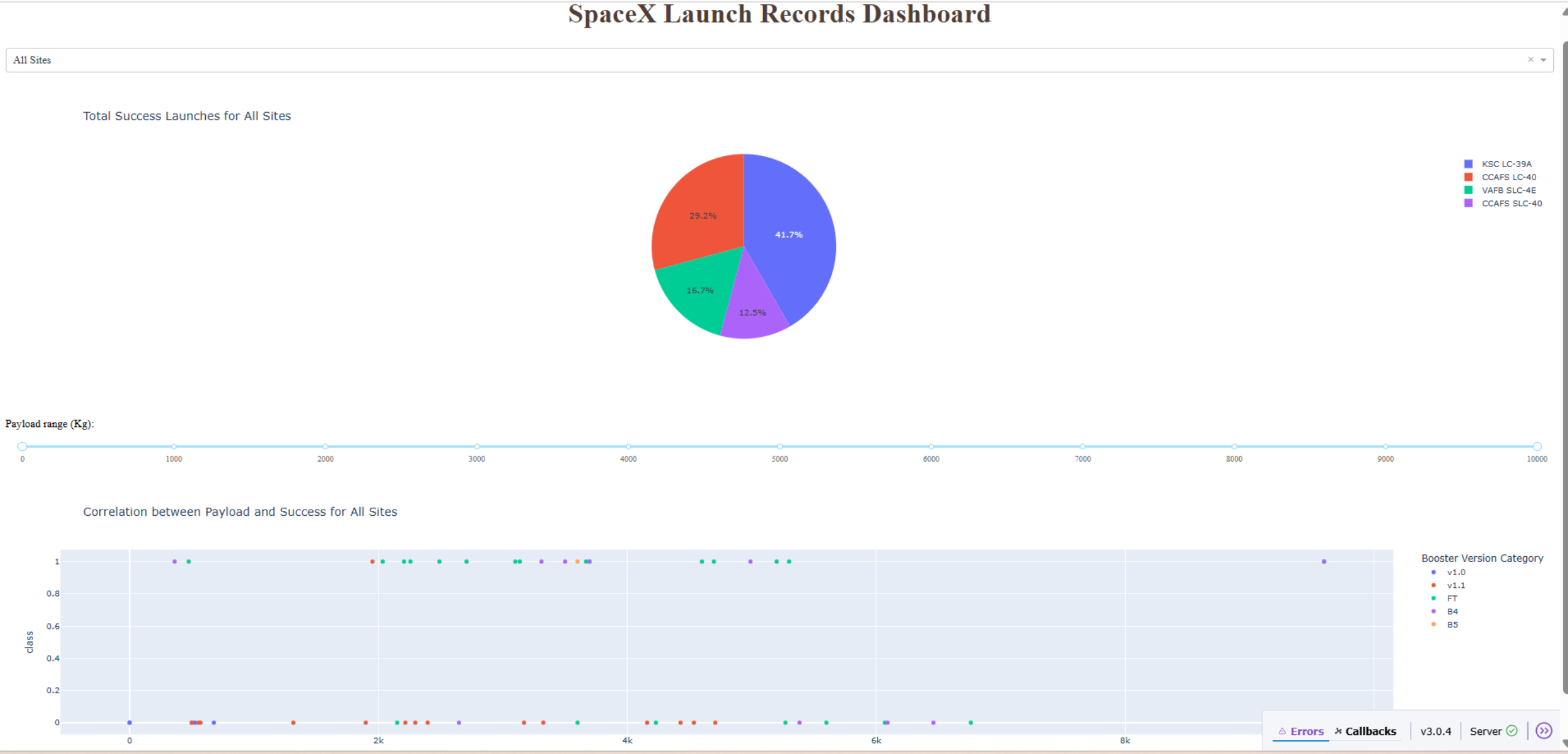
Measuring the Distance from Launch Sites to Nearby Landmarks

Description:

- We calculate the geographic distance between a SpaceX launch site and its nearby points of interest (e.g., coastline, railways, highways, or city centers). Using Python libraries like geopy and folium, we:
- Obtain the coordinates (latitude and longitude) of both the launch site and a nearby feature.
- Calculate the distance between these points using the Haversine formula or geodesic method from geopy.
- Visualize the result on an interactive map with:
  - Markers for both the launch site and the nearby location.
  - A label displaying the distance in kilometers (e.g., "0.58 KM").
- This visualization helps to analyze the strategic positioning of launch sites:
  - Most are within 0.5–2 km from the coastline (to minimize risk in case of failure).
  - Typically 1–3 km from highways and railways (for logistical access).
  - Located far from urban areas (to comply with safety regulations).This spatial analysis supports the understanding of why SpaceX selects specific sites for rocket launches based on proximity to essential infrastructure and safety requirements.


Section 4
Build a Dashboard with Plotly Dash

Dashboard Summary: Launch Success Rate and Payload Correlation Across SpaceX Sites



Dashboard Summary: Launch Success Rate and Payload Correlation Across SpaceX Sites

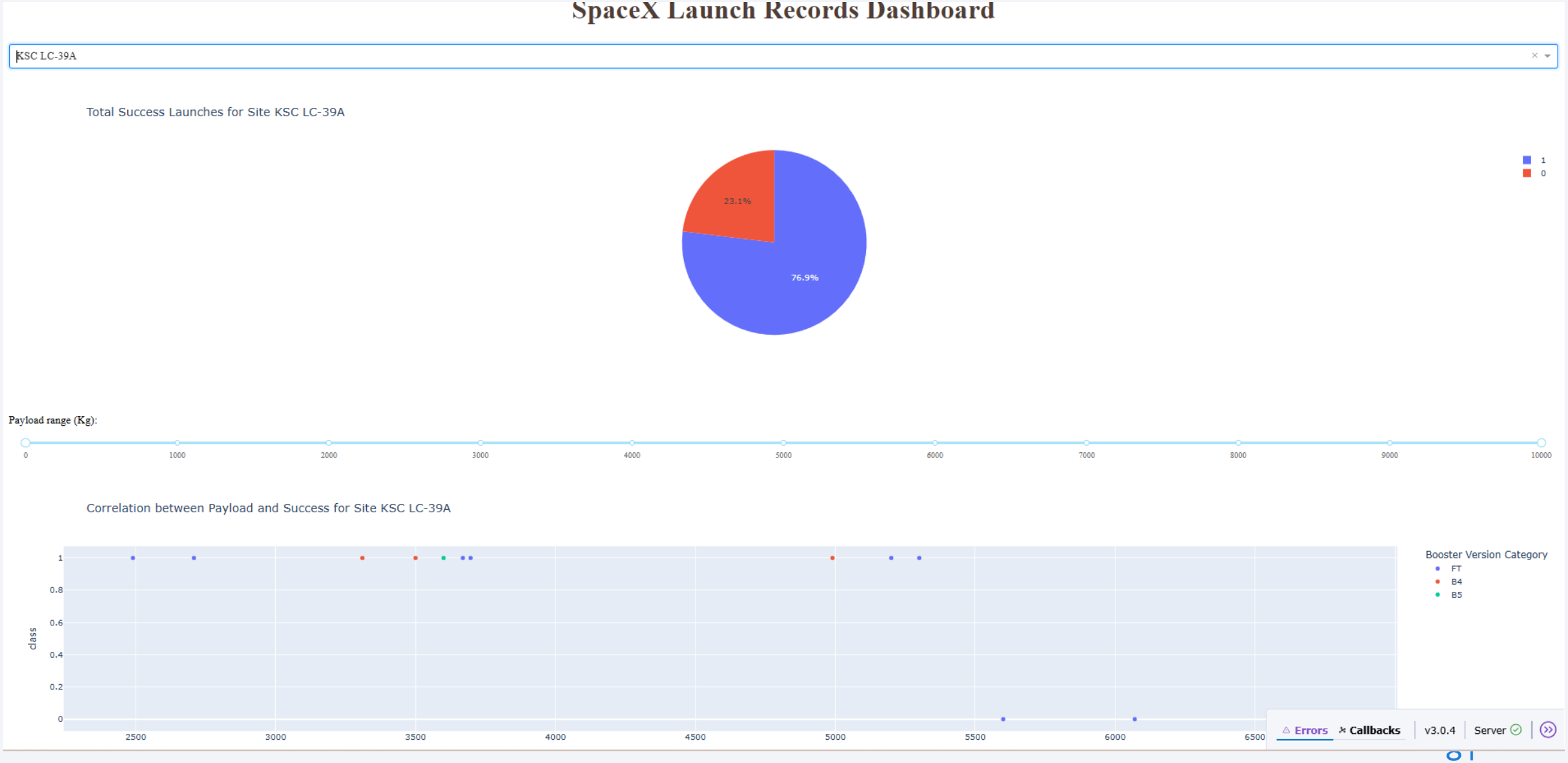
Pie Chart (Top):

- This chart displays the overall distribution of successful launches across all SpaceX launch sites.
- KSC LC-39A has the largest share of successes (41.7%), followed by CCAFS LC-40 (29.2%), VAFB SLC-4E (16.7%), and CCAFS SLC-40 (12.5%).
- This suggests that KSC LC-39A is the most active or most successful site in the dataset.

Scatter Plot (Bottom):

- The scatter plot shows the correlation between payload mass and launch success for all sites.
- The class value on the y-axis (0 = failure, 1 = success) allows you to visually assess success rates across payload weights.
- Successes (class = 1) dominate across most payload ranges, especially for medium and high payloads.
- The color coding by Booster Version Category adds another layer, suggesting that some boosters (like FT, B5) might be linked to higher success rates.

Launch Performance and Payload Analysis for KSC LC-39A



Launch Performance and Payload Analysis for KSC LC-39A

KSC LC-39A

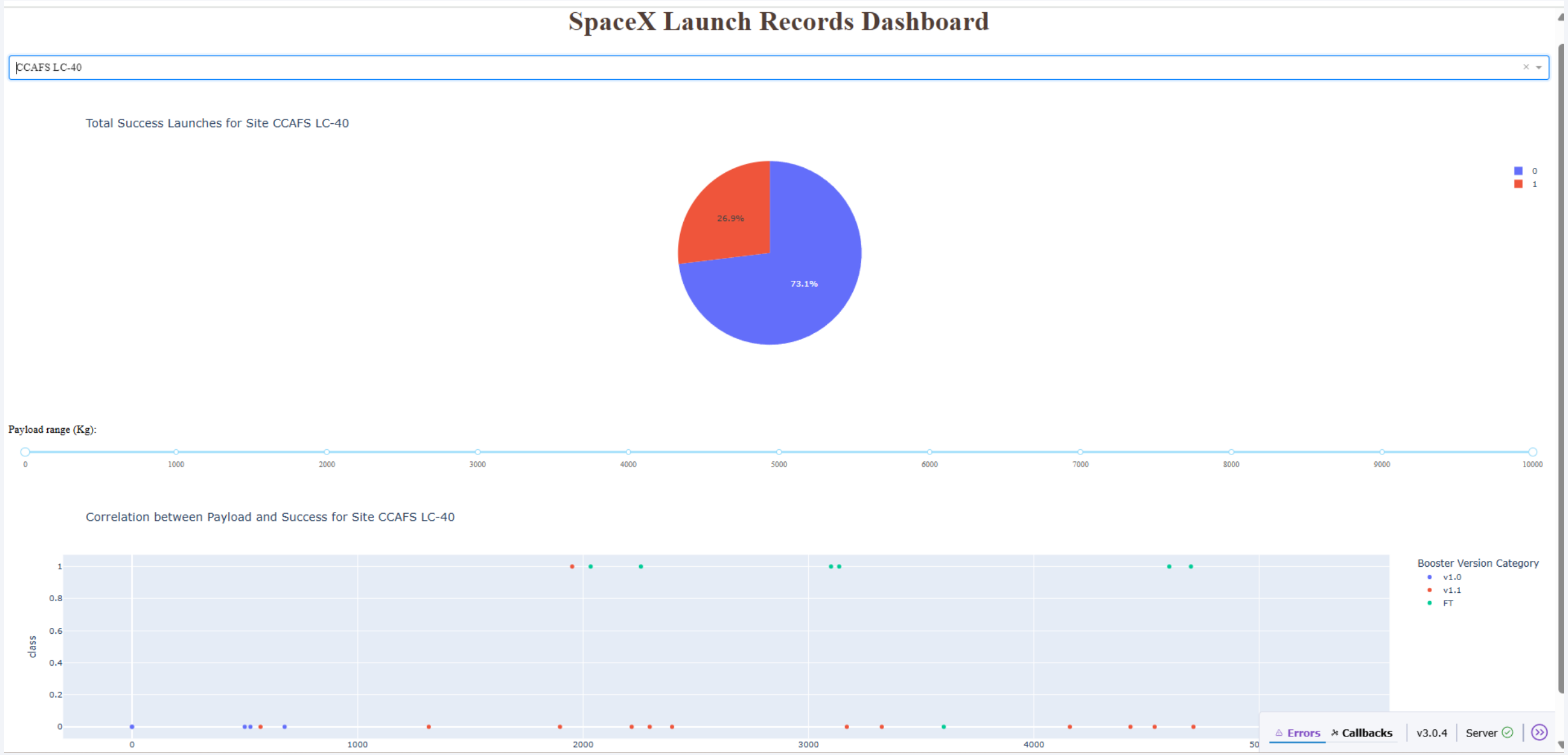
Pie Chart Analysis:

- ✓ The success rate for this site is high, with 76.9% successful launches and only 23.1% failures.
- ✓ This indicates strong reliability and operational efficiency at this location.

Scatter Plot Analysis:

- ✓ The majority of payloads, regardless of weight, result in successful launches (class = 1).
- ✓ Most booster versions used here appear to perform reliably.
- ✓ This suggests that KSC LC-39A is one of the most effective launch sites.

Success Rate and Payload Correlation for Launch Site CCAFS LC-40



Success Rate and Payload Correlation for Launch Site CCAFS LC-40

CCAFS LC-40

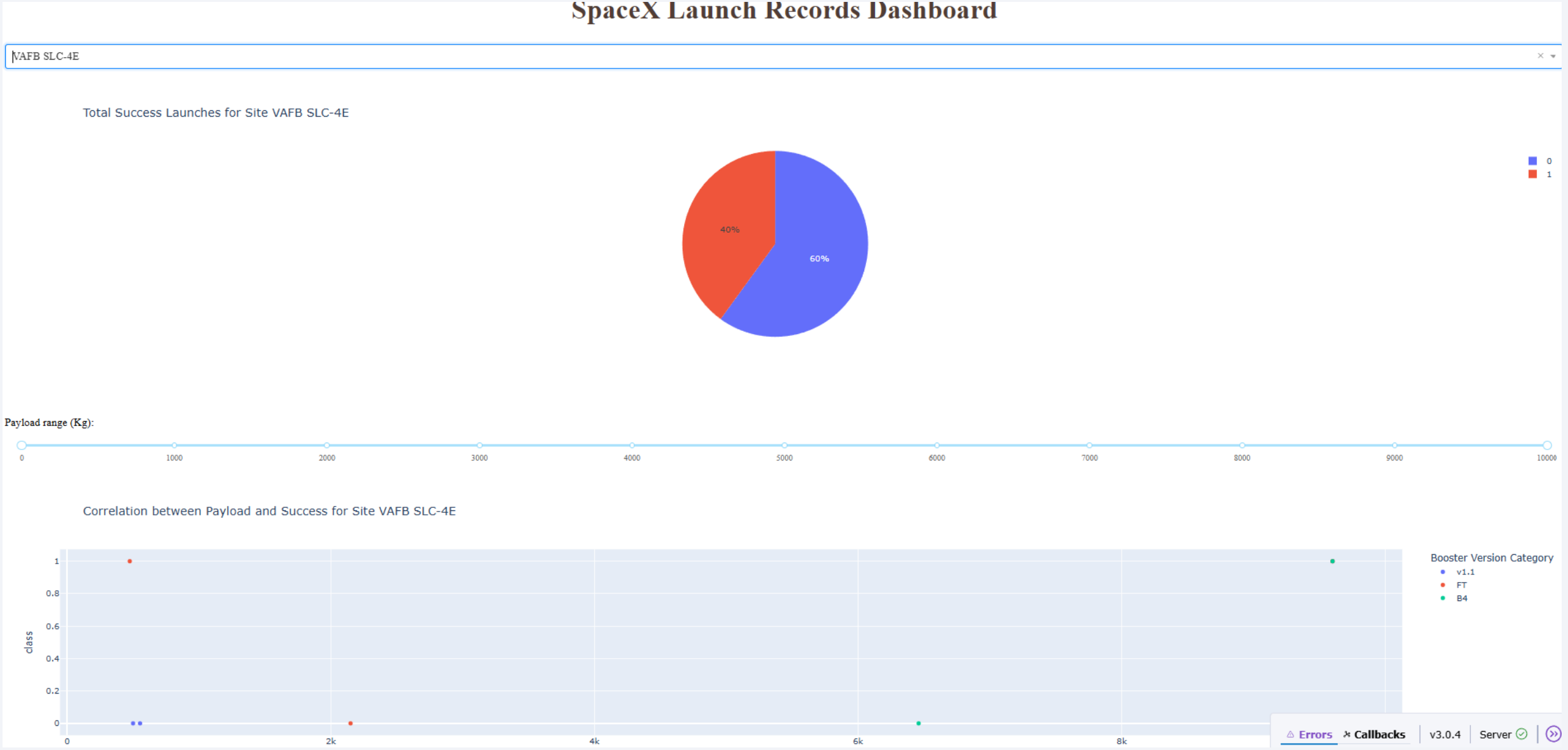
Pie Chart Analysis:

- This site shows a **73.1% success rate**, which is quite strong.
- Only **26.9% of launches resulted in failure**, indicating good reliability.

Scatter Plot Analysis:

- Launches span a broad range of payload weights.
- Most failures are clustered at lower payloads, while heavier payloads have higher success rates.
- This may imply better engineering/testing for larger payloads or more conservative booster selection.

Launch Performance and Payload Success Analysis – VAFB SLC-4E



Launch Performance and Payload Success Analysis – VAFB SLC-4E

VAFB SLC-4E

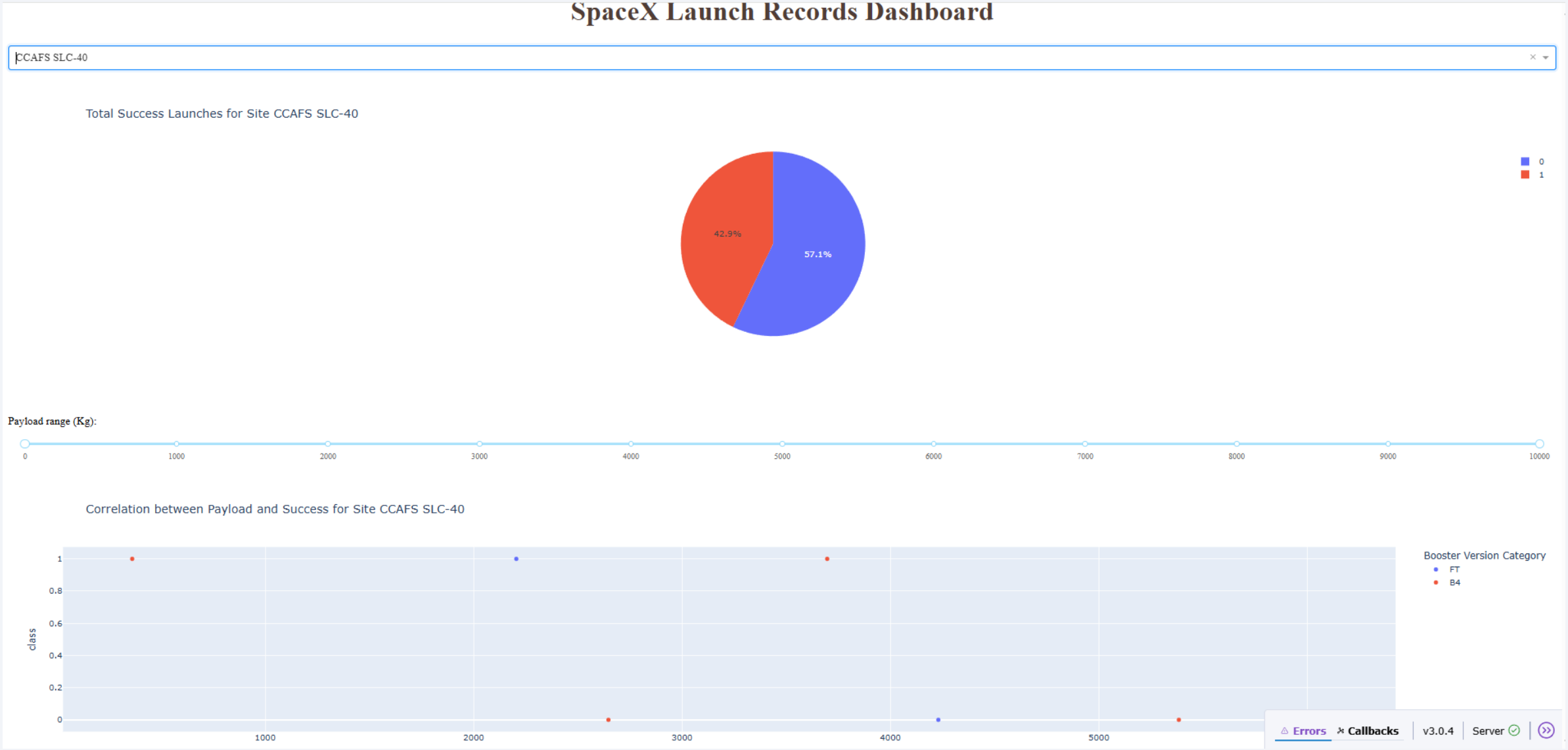
Pie Chart Analysis:

- The success rate is 60%, with 40% failures.
- It's slightly better than CCAFS SLC-40 but still not optimal.

Scatter Plot Analysis:

- Fewer data points suggest that fewer launches took place at this site.
- Successful and failed launches are interspersed, but a noticeable number of successes occur with mid-to-high payload ranges.

Launch Performance and Payload Success Analysis – CCAFS SLC-40



Launch Performance and Payload Success Analysis – CCAFS SLC-40

CAFS SLC-40

Pie Chart Analysis:

- This site has a moderate success rate: 57.1% success and 42.9% failure.
- It's the least successful site among those visualized.

Scatter Plot Analysis:

- Launch failures appear across a range of payload masses.
- The distribution is more scattered, indicating inconsistent outcomes.
- This might suggest issues related to payload configurations, booster versions, or external launch conditions.

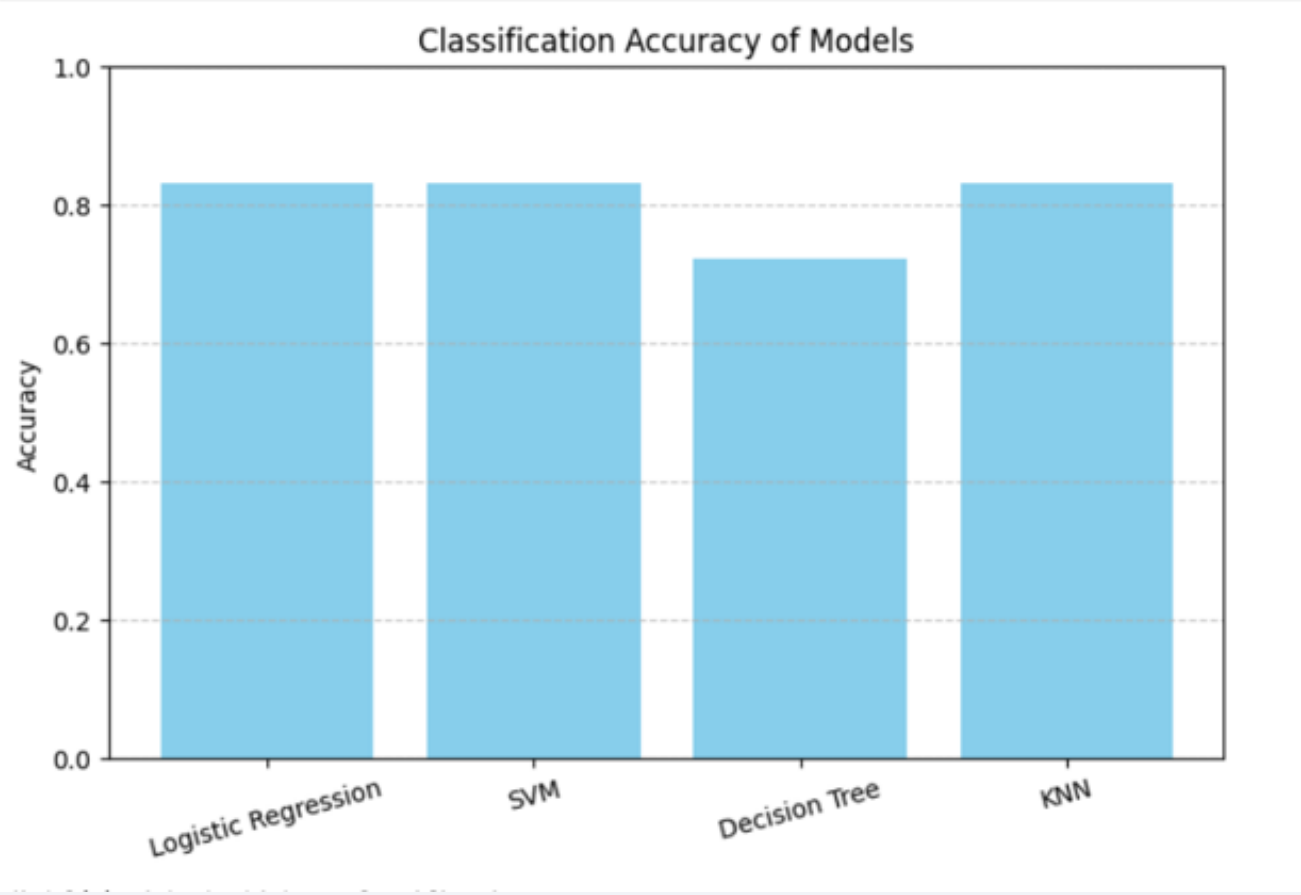
Overall Observations

KSC LC-39A and CCAFS LC-40 are the most successful sites based on visual evidence. There is a positive trend between payload mass and success—heavier payloads are often associated with successful launches. The booster version category also appears to influence launch outcomes, suggesting that newer or more advanced boosters might improve success rates. Sites with lower performance (like CCAFS SLC-40) might benefit from operational reviews or targeted improvements.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



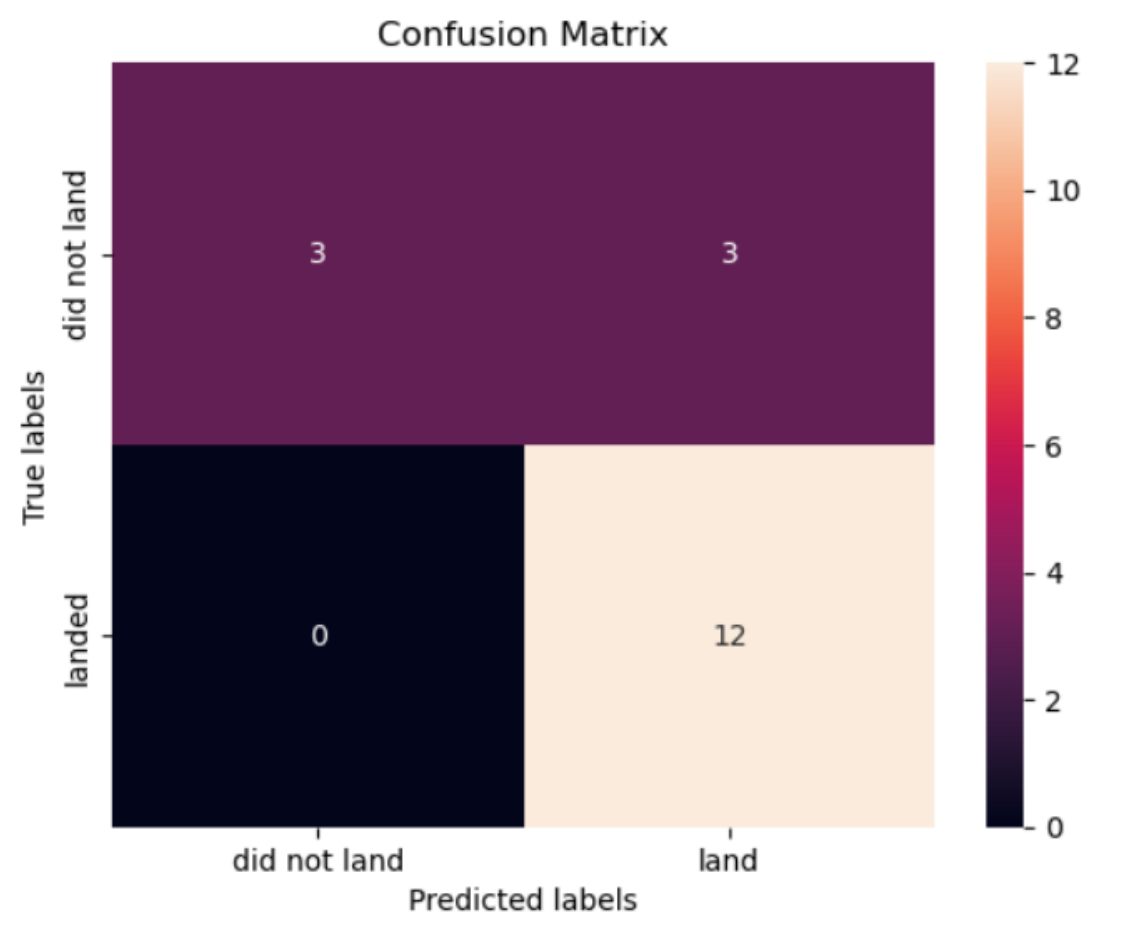
Here is the visualization of the classification accuracy for four models: Logistic Regression, SVM, Decision Tree, and KNN.

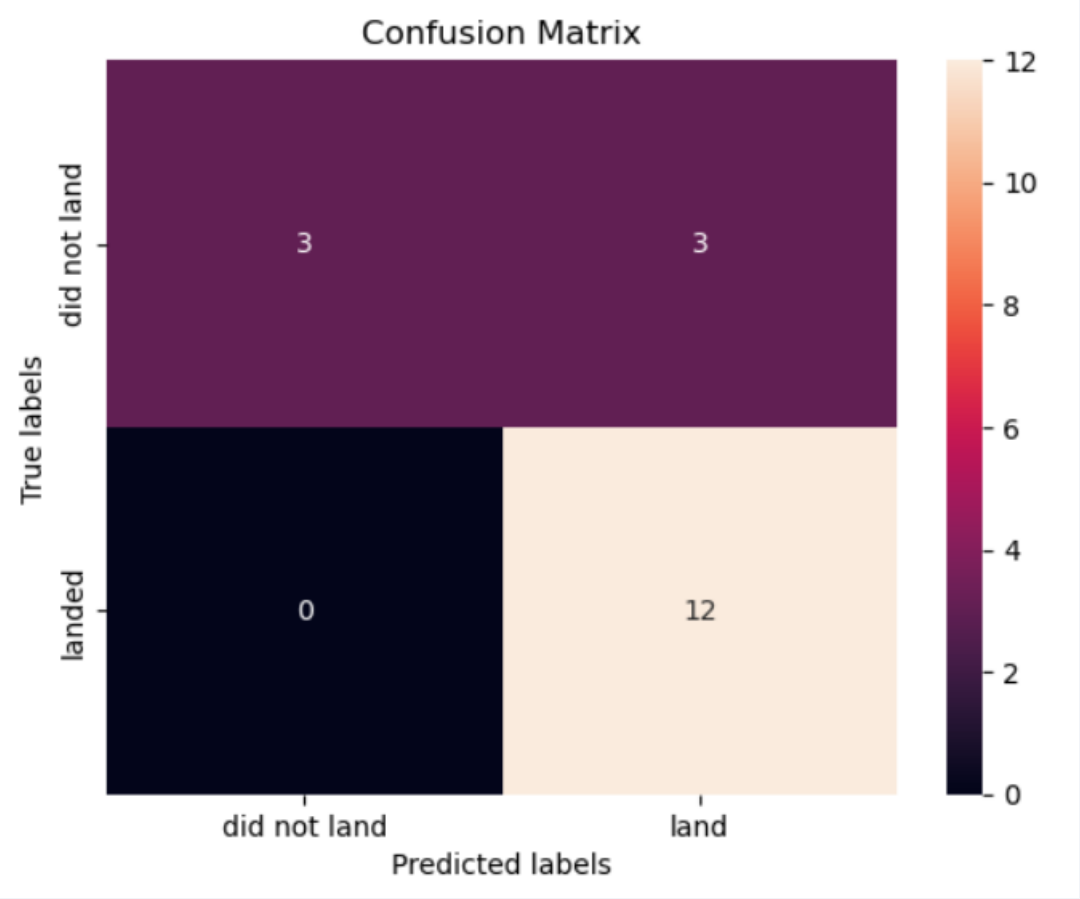
All models except Decision Tree achieved the same highest accuracy of 0.833. ▽ Decision Tree had the lowest accuracy at 0.722.

Conclusion: The models with the highest classification accuracy are:

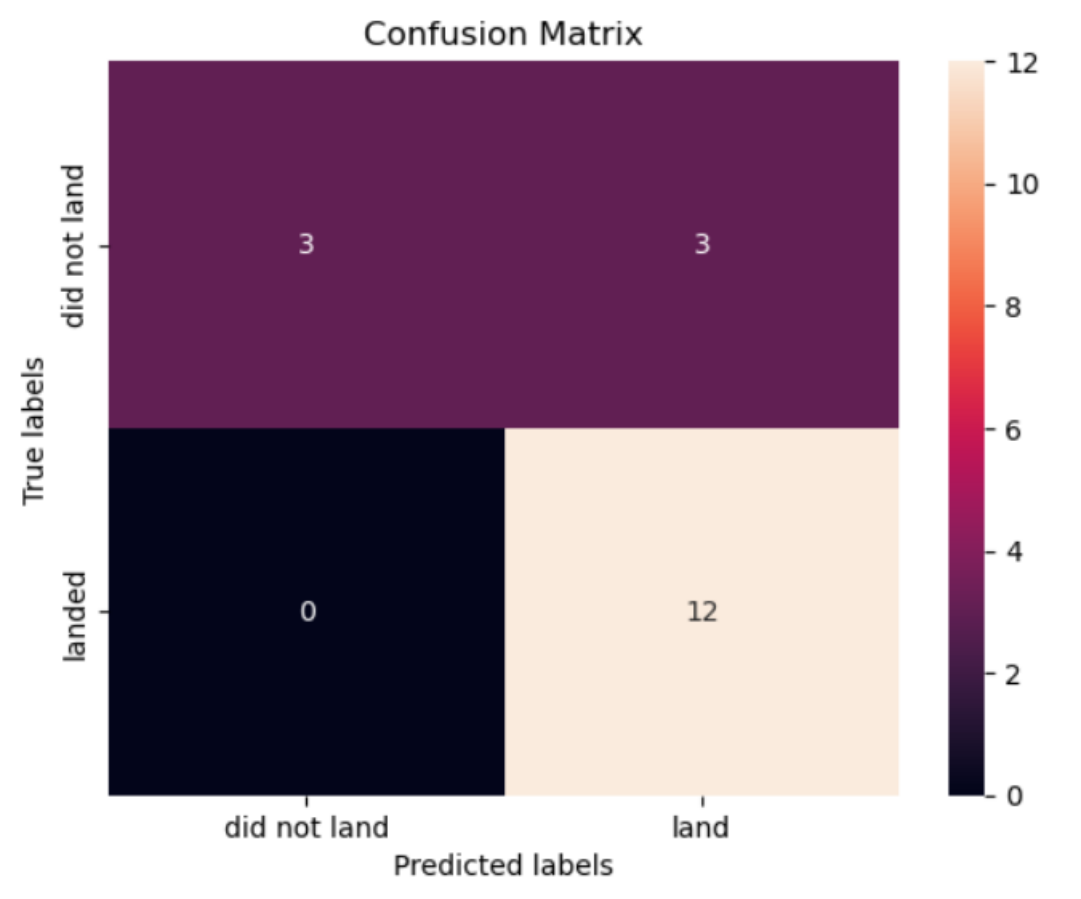
- Logistic Regression
- SVM
- KNN

Confusion Matrix Of Logistic Regression, SVM and KNN





Confusion Matrix of Logistic Regression, SVM and KNN



This **exact same confusion matrix** and classification performance was observed across **all three models**: **Logistic Regression**, **Support Vector Machine (SVM)**, and **K-Nearest Neighbors (KNN)**, showing they all performed identically on this dataset.

The confusion matrix shown here represents the classification performance of the model in predicting whether a rocket landed or did not land:

True Positives (TP): 12→ The model correctly predicted that the rocket landed.

True Negatives (TN): 3→ The model correctly predicted that the rocket did not land.

False Positives (FP): 3→ The model incorrectly predicted landed when it actually did not land.

False Negatives (FN): 0→ The model never incorrectly predicted did not land when it actually landed.

Summary: This means the model performs very well at identifying landings (100% sensitivity), but has some difficulty distinguishing failed landings (accuracy of 50% for the “did not land” class).

Conclusions

- **Launch Data Integration Achieved**
  Successfully combined structured SpaceX API data with web-scraped information from Wikipedia into a clean and unified DataFrame.

- **Launch Success Factors Identified**
  Launch site, payload mass, and orbit type were found to be strong indicators of landing success probability

.

- **KSC LC-39A**
  had the **highest launch success rate** among all SpaceX launch sites.

- **Orbit Types Influence Success Rates**
  Orbits like **GEO**, **SSO**, **HEO** and **ES-L1** had the highest landing success rates, while **GTO** showed lower reliability.

- **Payload Insights**
  Missions with **payloads under 6000 kg** had higher success rates. Very high payloads (>10,000 kg) showed slightly more failures.

Conclusions

- **Geographical Insights via Folium Maps**
  Most launch sites are near **coasts**, **highways**, and **railways**, with calculated distances confirming proximity.

- **Machine Learning Classification**
  Three models (Logistic Regression, SVM, and KNN) achieved the **same high accuracy (83%)**, while Decision Tree showed signs of **overfitting**.

- **Confusion Matrix Interpretation**
  The models predicted **landings correctly** with 100% sensitivity but struggled to detect failed landings, revealing **class imbalance**.

- **Dashboard Development with Plotly Dash**
  A fully functional interactive dashboard allowed dynamic filtering by site and payload mass, enabling real-time data analysis.

- **End-to-End Pipeline Executed**
  From data collection to web scraping, cleaning, EDA, spatial mapping, machine learning modeling, and dashboard creation—an entire data science workflow was completed.

Thank you!