IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Svitlana Kokhan
23 April 2025

skokhan

Outline

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

Executive Summary

- Summary of methodologies

In this research, we have undertaken a comprehensive data science approach to predict the successful landing of the first stage of the SpaceX Falcon 9 rocket. As a new player in the market, SpaceY aims to establish itself as a strong competitor by adopting innovative, data-driven approaches to optimize rocket launch operations.

The methodologies employed in this project include data collection from publicly available sources, data wrangling to clean and preprocess the data, exploratory data analysis to uncover patterns, data visualization to illustrate key insights, machine learning model development to build a classification model for predictive analysis, model evaluation to ensure accuracy and reliability.

- Summary of all results

Our findings indicate that specific factors such as launch site, orbit type, payload mass, and booster version significantly influence the likelihood of a successful landing. The predictive model developed, utilizing Decision Tree Classifier, achieved training accuracy of 87,50% and test accuracy of over 94%, demonstrating its potential utility for strategic decision-making.

The insights obtained from this research can assist SpaceY in making competitive bids against SpaceX by accurately estimating launch costs and improving the cost-efficiency of reusable rocket technology.

Introduction

- Project background and context

The commercial space industry has entered a transformative era, with private companies revolutionizing space travel and satellite deployment. SpaceX, led by Elon Musk, has set new standards in cost efficiency by successfully reusing the first stage of its Falcon 9 rocket, significantly reducing launch costs. With launch costs as low as $62 million compared to competitors charging upwards of $165 million, SpaceX has established a dominant position in the market.

- Problems we want to find answers

To compete effectively in the space industry, SpaceY, a new entrant, must adopt a data-driven approach to optimize launch operations and improve cost efficiency. This research aims to address the following problems:

- What factors most significantly influence the success of a Falcon 9 first stage landing?
- How have mission success rates changed over time, and what trends can be identified?
- Can machine learning models accurately predict the success of first stage landings using publicly available data?
- What insights can be derived to improve strategic planning and decision-making during reusable rocket launches?

By answering these questions, this research will provide valuable insights to support SpaceY in its mission to establish itself as a key competitor in the commercial space sector.

Section 1

Methodology

Methodology

- Data collection methodology
  Perform a GET request to the SpaceX REST API, which provides structured JSON data. Utilize web scraping related Falcon 9 Wiki pages using the Python BeautifulSoup package to extract data from relevant HTML tables.
- Perform data wrangling
  We conduct exploratory data analysis to identify patterns in the data and determine appropriate labels for training classification models.
- Perform exploratory data analysis (EDA) using visualization and SQL
  Create and execute SQL queries to analyze SpaceX data. Utilize scatter plots and bar plots to visualize the data, determining which attributes correlate with successful landings, extracting meaningful patterns, and performing features engineering.
- Perform interactive visual analytics using Folium and Plotly Dash
  Develop an interactive dashboard to analyze launch records using Plotly Dash. Create interactive maps with Folium to examine the proximities of launch sites.
- Perform predictive analysis using classification models
  Preprocess and standardize data. Perform test/train split. Train Logistic Regression, Support Vector Machine, Decision Tree Classifier, and k-Nearest Neighbors. Tune models by finding the best hyperparameters. Evaluate accuracy and find the model that performs best.

Data Collection

- SpaceX REST API

The SpaceX launch data used in the analysis is gathered utilizing the GET request for the SpaceX REST API. This API provides data about past launches, including details about the rocket used, the payload delivered, launch specifications, landing specifications, and landing outcomes.

We use the API endpoints to access specific data for each launch. By performing a GET request with the requests library, we retrieve data, which is provided in a structured JSON format. To make the JSON data more usable for visualization and analysis, we normalize it into a flat table. We transform the raw data into a clean dataset. Filter the data focusing solely on Falcon 9 booster launches. Address any NULL values by replacing them with the mean values and applying a one-hot encoding algorithm. Finally, we convert the cleaned data into a Pandas DataFrame and export it to a CSV file.

- Web Scraping Falcon 9 Launch Records

Additionally, data is extracted from related Falcon 9 Wiki pages using the Python BeautifulSoup package. This process involves gathering information from relevant HTML tables that contain important records of Falcon 9 launches.

We then parse the data from these tables and convert it into a flat table format, specifically a Pandas DataFrame, ensuring that it is structured for effective analysis and visualization before exporting it to a CSV file.

Data Collection – SpaceX API

GitHub URL to Notebook

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

SpaceX API endpoint

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

Get API response

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

Normalize JSON

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome
```

Structure and fill data

```
# Create a data from launch_dict
data = pd.DataFrame(launch_dict)
```

Create DataFrame

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

Filter on DataFrame

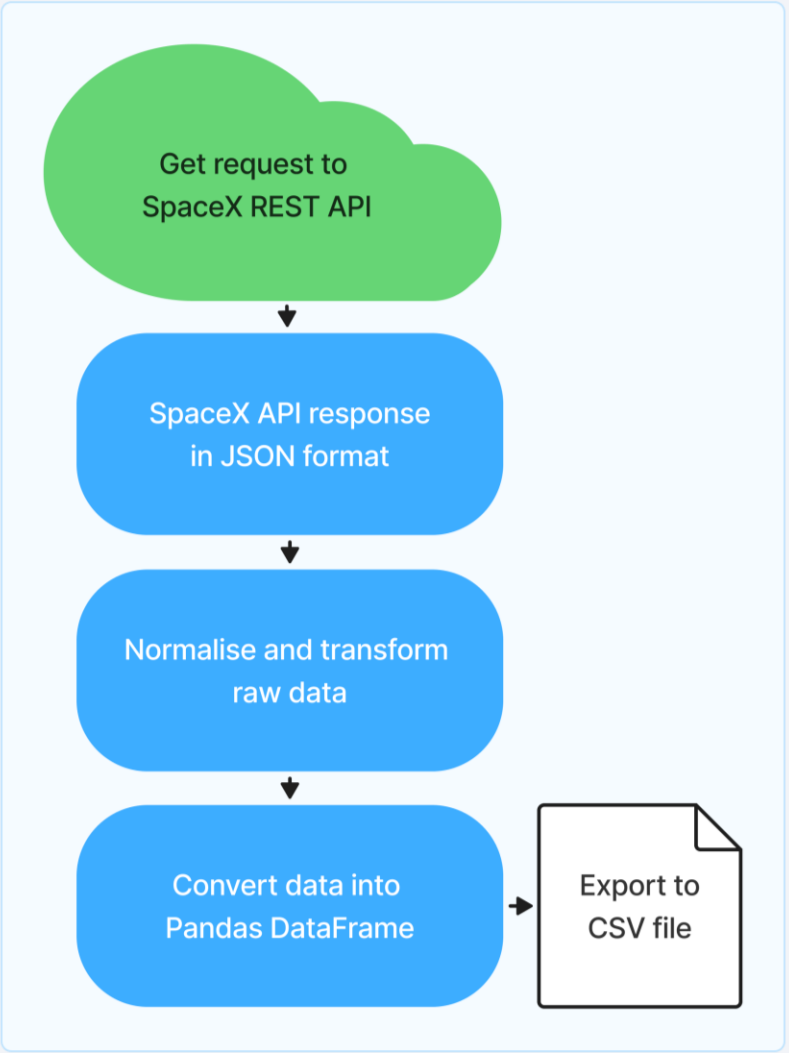
```
# Calculate the mean value of PayloadMass column
avg_payloadmass = data_falcon9['PayloadMass'].astype('float').mean(axis=0)

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, avg_payloadmass, inplace=True)
```

Replace missing values

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Export to CSV file



Data Collection - Scraping

GitHub URL to Notebook

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_
```

Falcon 9 Wiki URL

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

Get HTML response

```
# Use BeautifulSoup() to create a BeautifulSoup object from a resp
soup = BeautifulSoup(html, 'html.parser')
```

Use BeautifulSoup

```
# Use the find_all function in the BeautifulSoup object, with elem
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

Parse HTML

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
```

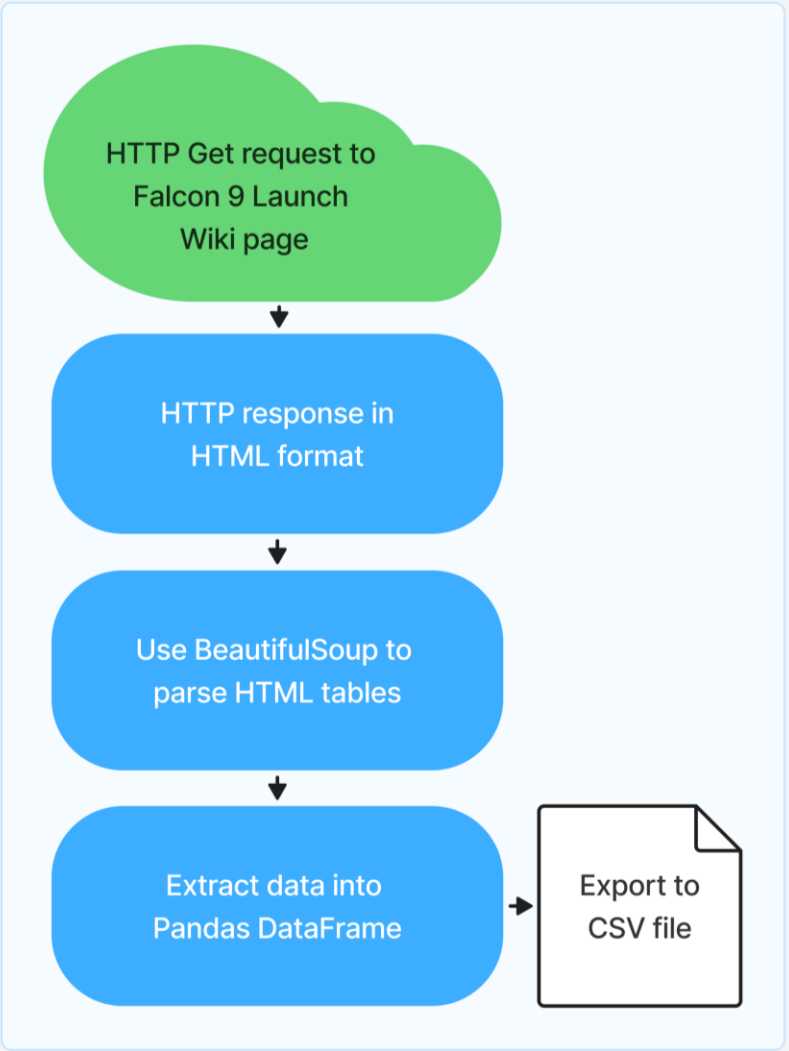
Structure and fill data

```
df = pd.DataFrame({key : pd.Series(value) for key, value in launch
```

Create DataFrame

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Export to CSV file

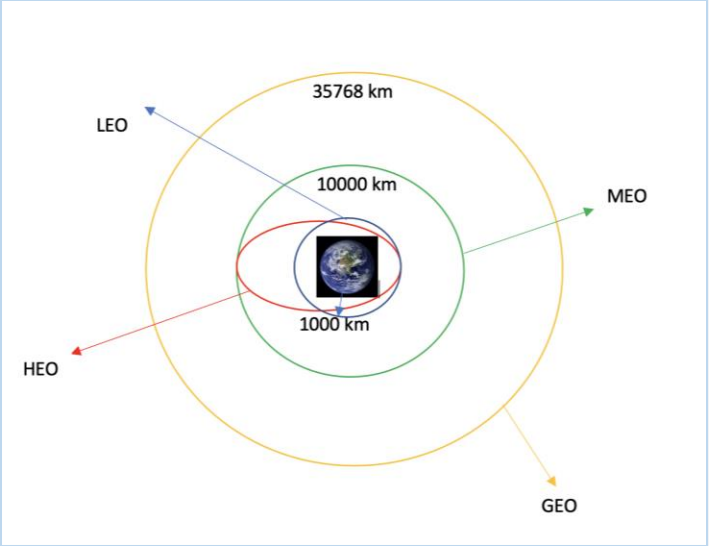


Data Wrangling

GitHub URL to Notebook

The SpaceX dataset includes data on launches from several launch facilities:

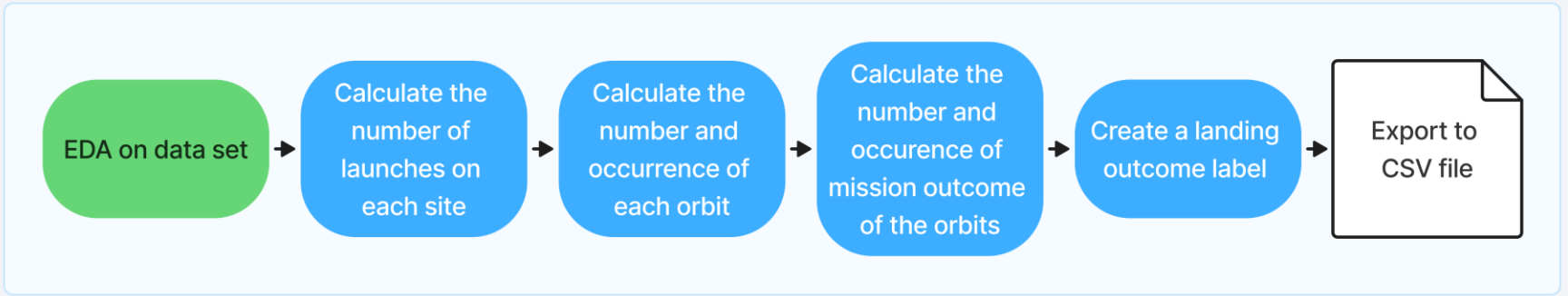
- Cape Canaveral Air Force Station Launch Complex 40 (CCAFS LC-40);
- Cape Canaveral Space Launch Complex 40 (CCAFS SLC 40);
- Kennedy Space Center Launch Complex 39A (KSC LC 39A);
- Vandenberg Air Force Base Space Launch Complex 4E (VAFB SLC 4E).



Each launch is aimed at reaching a specific orbit, and the various types of orbits used by SpaceX are illustrated in the accompanying plot.

The dataset also contains multiple instances where the booster did not land successfully. In some cases, a landing was attempted but failed due to an accident.

As part of our exploratory data analysis, we define both successful and unsuccessful landing outcomes and convert these outcomes into a classification variable. This variable will represent the result of each launch and will serve as the training labels. A label of 1 indicates a successful landing, while a label of 0 signifies an unsuccessful landing.



EDA with Data Visualization

GitHub URL to Notebook

Through exploratory data analysis (EDA) with visualization techniques, we use categorical scatter plots, bar plot, and line plot to examine the data. This helps us identify which attributes correlate with successful landings, extract meaningful patterns and trends, and conduct feature engineering.

- Categorical Scatter Plots
  - Flight Number vs. Payload Mass;
  - Flight Number vs. Launch Site;
  - Payload Mass vs. Launch Site;
  - Flight Number vs. Orbit Type;
  - Payload Mass vs. Orbit Type.

A scatter plot uses dots to represent values for two different numeric variables. We use categorical scatter plots to identify relationships between different attributes and successful landings.

- Bar Plot
  - Success Rate of Each Orbit Type.

A bar plot shows a comparison among categories. We use a bar plot to identify which orbits have the highest success rates.

- Line Plot
  - The Launch Success Yearly Trend.

Line plots are helpful for tracking changes over time. We use a line plot to visualize the average launch success trend.

EDA with SQL

GitHub URL to Notebook

By conducting exploratory data analysis (EDA) using SQL, we formulate and execute queries to analyze SpaceX data and address the following questions:

- Display the names of the unique launch sites in the space mission;
- Display 5 records where launch sites begin with the string "CCA";
- Display the total payload mass carried by boosters launched by NASA (CRS);
- Display average payload mass carried by booster version F9 v1.1;
- List the date when the first successful landing outcome in ground pad was achieved;
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000;
- List the total number of successful and failure mission outcomes;
- List the names of the booster versions which have carried the maximum payload mass;
- List the records which display the month names, failure landing outcomes in drone ship, booster versions, launch site for the months in year 2015;
- 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.

Build an Interactive Map with Folium

GitHub URL to PDF

GitHub URL to Notebook

The success rate of rocket launches may depend on the location and proximity of the launch site, particularly concerning the initial trajectories of the rockets. Identifying an optimal location for building a launch site involves many factors, and analyzing existing launch site locations may help us uncover some of these factors.

To illustrate our findings, we use Folium to create an interactive map, marking the locations of the launch sites and their nearby areas. We follow these key steps:

- Mark All Launch Sites
  We add the locations of each site on the map using their latitude and longitude coordinates, marking each site with a circle.
- Mark Success and Failed Launches for Each Site
  Each launch occurs at one of the four launch sites, which means that many launch records share the same coordinates. To simplify the map clutter caused by overlapping markers, we employ marker clusters. Successful launches are represented with green markers, while failed launches are indicated by red markers. This allows us to easily identify which sites have high success rates.
- Calculate Distances Between Launch Sites and Nearby Locations
  We explore the map for significant nearby proximities and use the Haversine formula to compute distances between locations. Polylines are added to the map to annotate proximities along with their respective distances.

Ultimately, our goal is to explain how to choose an optimal launch site.

Build a Dashboard with Plotly Dash

GitHub URL to PDF

GitHub URL to Plotly Dash

By utilizing an interactive analytics approach, we can quickly and effectively identify visual patterns. We develop a dashboard application using the Python Plotly Dash package. This dashboard enables us to extract insights from the SpaceX dataset more easily compared to traditional static graphs.

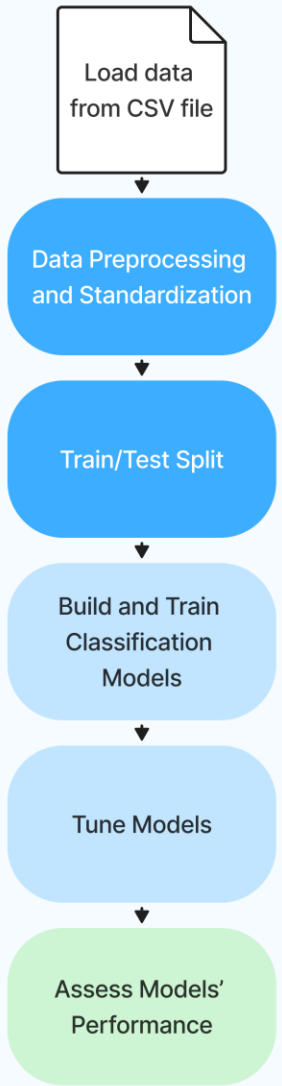
The dashboard features input components, including a dropdown list for launch sites and a slider for payload range, enabling interaction with a pie chart and a scatter plot.

- Launch Site Dropdown List
  There are four different launch sites, and we would like to first determine which one has the highest success count. After that, we can select a specific site to examine its detailed success rate.
- Payload Range Slider
  We explore whether the variable payload is correlated with mission outcomes. Using the dashboard, we can easily select different payload ranges to identify any visual patterns.
- Pie Chart
  We utilize a pie chart to visualize the total number of successful launches. If a specific launch site is selected, the pie chart displays both the success count and the failure count for that site.
- Scatter Plot
  The scatter plot allows us to visually observe the correlation between payload and mission outcomes for selected sites. Additionally, we incorporate color labels for each scatter point based on the booster version, enabling us to observe mission outcomes across different boosters.

Predictive Analysis (Classification)

GitHub URL to Notebook

- **Data Preprocessing and Standardization**
  We start by loading data from a CSV file. To effectively utilize machine learning, we begin by preprocessing and standardizing our data for classification models.
- **Train/Test Split**
  Next, we perform a train/test split to evaluate the models. This split allows us to use one portion of the data for training the model and another for analyzing its performance.
- **Build and Train Models**
  We employ various machine learning techniques to build predictive models, including Logistic Regression, Support Vector Machines, Decision Tree Classifiers, and k-Nearest Neighbors.
- **Tune Models**
  Additionally, we implement Grid Search to fine-tune these models and identify the optimal hyperparameters.
- **Assess the Performance and Effectiveness**
  Finally, we assess the accuracy of each model and plot confusion matrices to determine which one performs the best, ultimately enhancing the overall effectiveness of our predictive classification analysis.



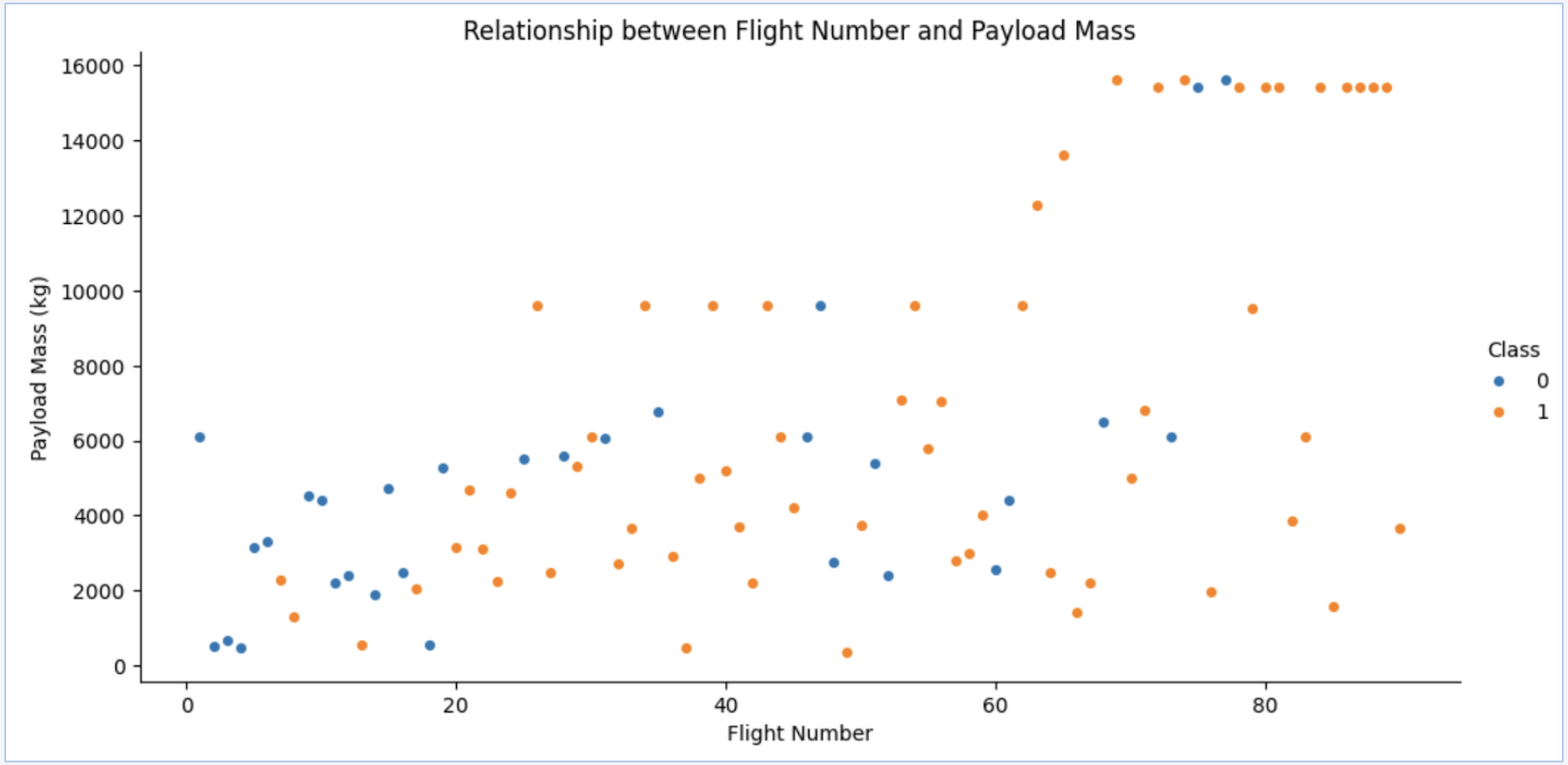
Results

- Exploratory data analysis results
  - Launch success has improved over time;
  - Launch site selection appears to have a greater impact on landing success than payload mass alone;
  - Orbit type does influence mission success, especially when combined with the complexity of the payload mass;
  - Orbits ES-L1, GEO, HEO and SSO have demonstrated a 100% success rate.
- Interactive analytics demo in screenshots
  - Most launch sites are located near the equator and positioned close to the coastline;
  - KSC LC-39A has the highest success rate among landing sites;
  - Newer boosters like FT, B4 and B5 show higher success rates, indicating enhancements in both payload capacity and launch reliability;
- Predictive analysis results
  - The Decision Tree Classifier achieved training accuracy of 87,50% and test accuracy of over 94%, making it the best-performing algorithm for predictive analysis on the SpaceX dataset.

Section 2

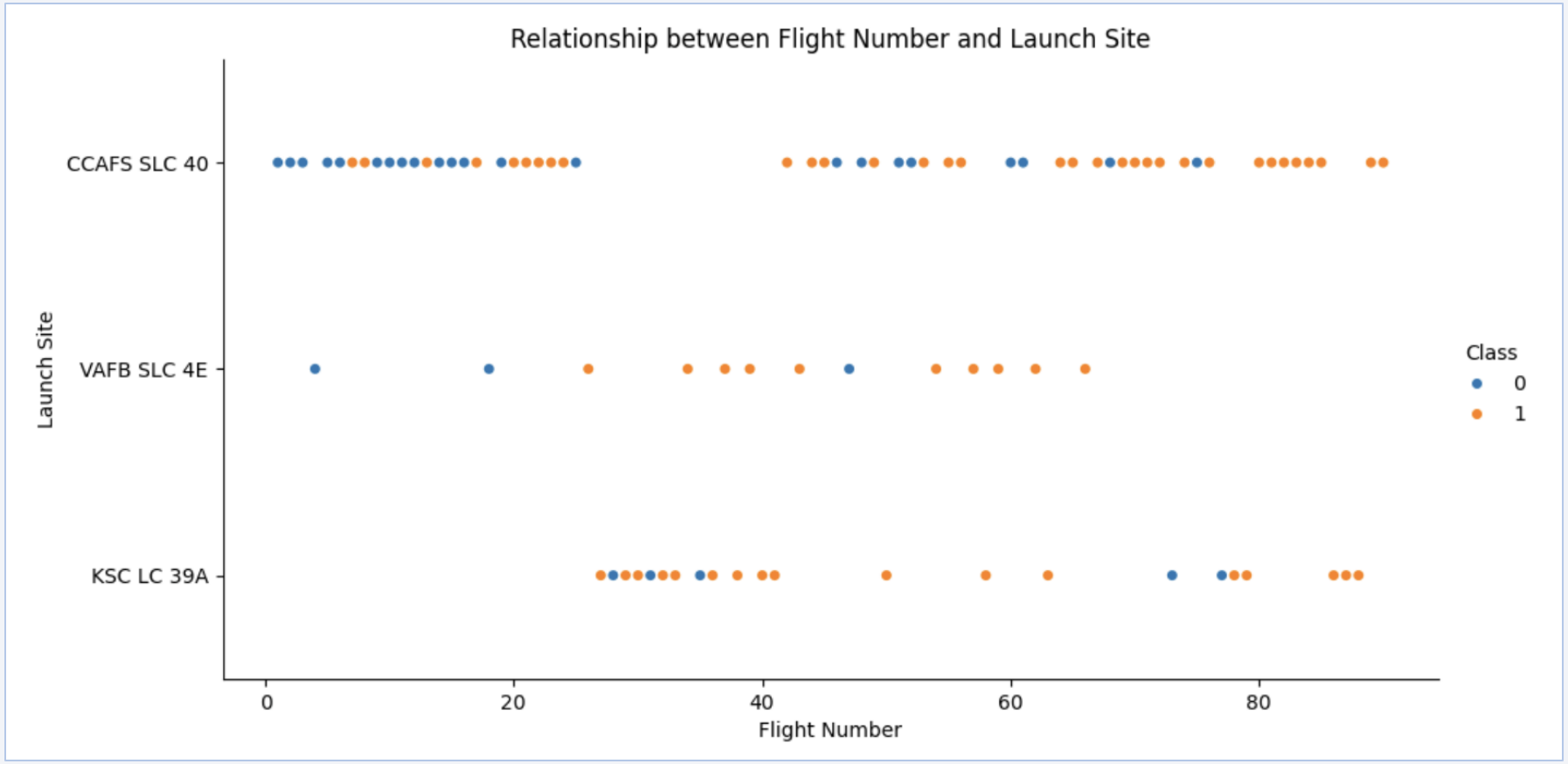
Insights drawn from EDA

Flight Number vs. Payload Mass



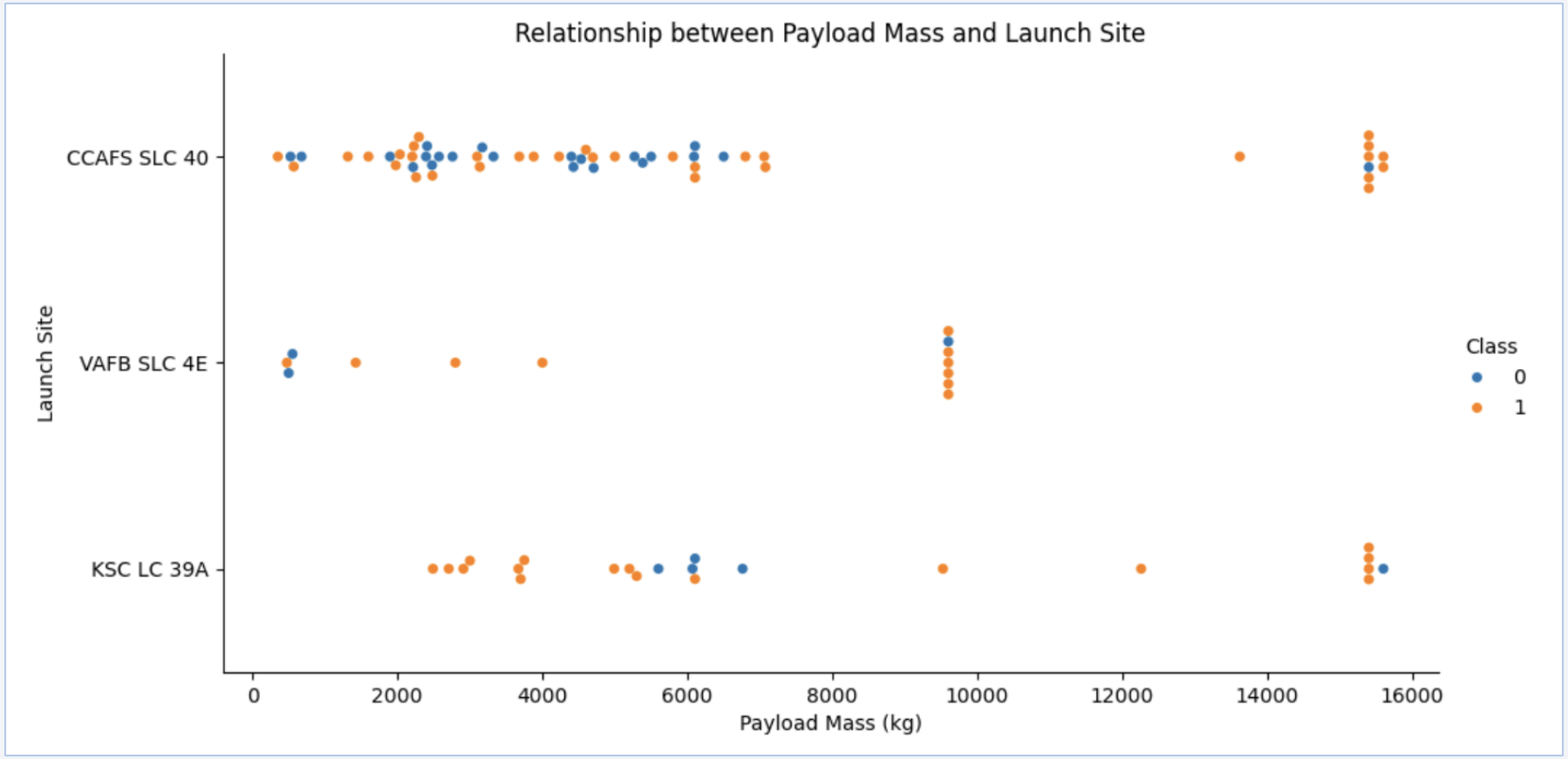
This plot suggests that SpaceX's reliability improved with time and experience. Earlier flights primarily carried lighter payloads and show a higher incidence of failures. In contrast, more recent flights have consistently carried heavier payloads and achieved greater success. There is a clear upward trend in the maximum payload mass as the flight number increases.

Flight Number vs. Launch Site



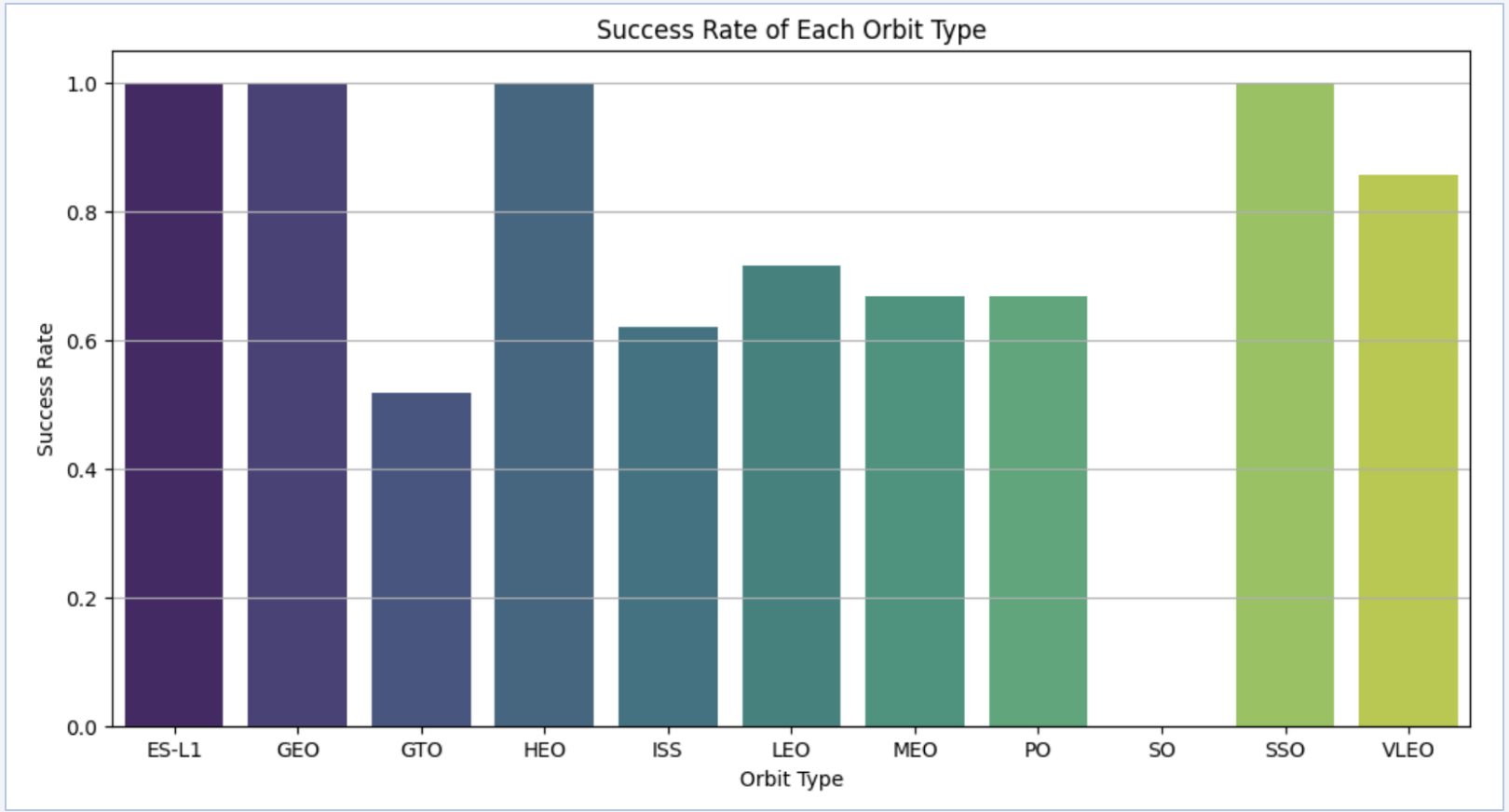
The CCAFS SLC 40 launch site has the highest number of launches, and the KSC LC 39A launch site was used later in the timeline. Across all launch sites, earlier flights had more failures. There is a clear improvement in success rate across all sites as the flight number increases.

Payload Mass vs. Launch Site



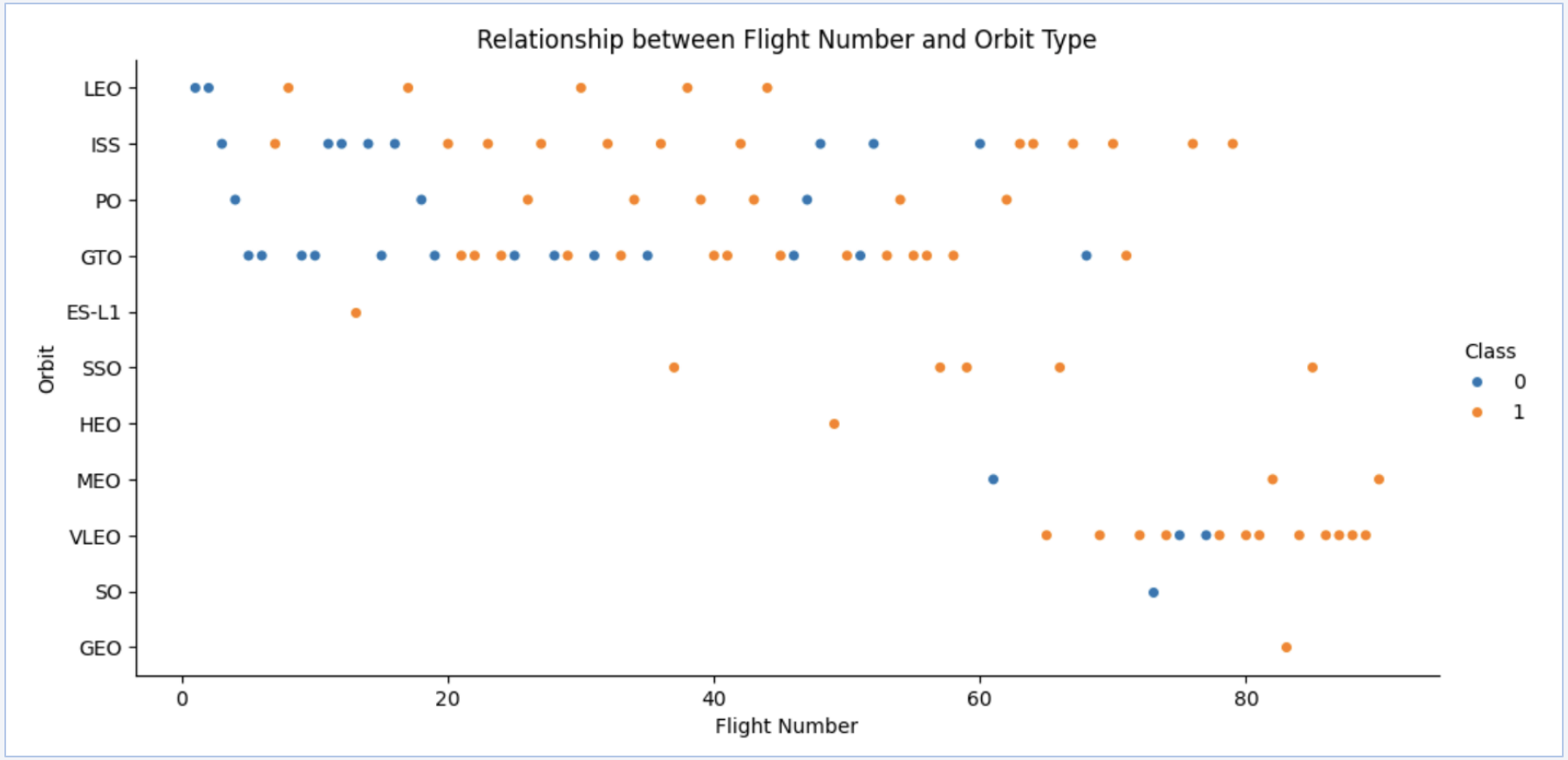
Payloads launched at VAFB SLC 4E are mostly in the lower to mid-weight range (0-10,000 kg). High payloads (>10,000 kg) are fewer but often successfully landed, particularly from CCAFS SLC 40 and KSC LC 39A. There is no strict correlation between payload mass and landing success, but launch site capabilities are likely influence success more than just the payload mass itself.

Success Rate of Each Orbit Type



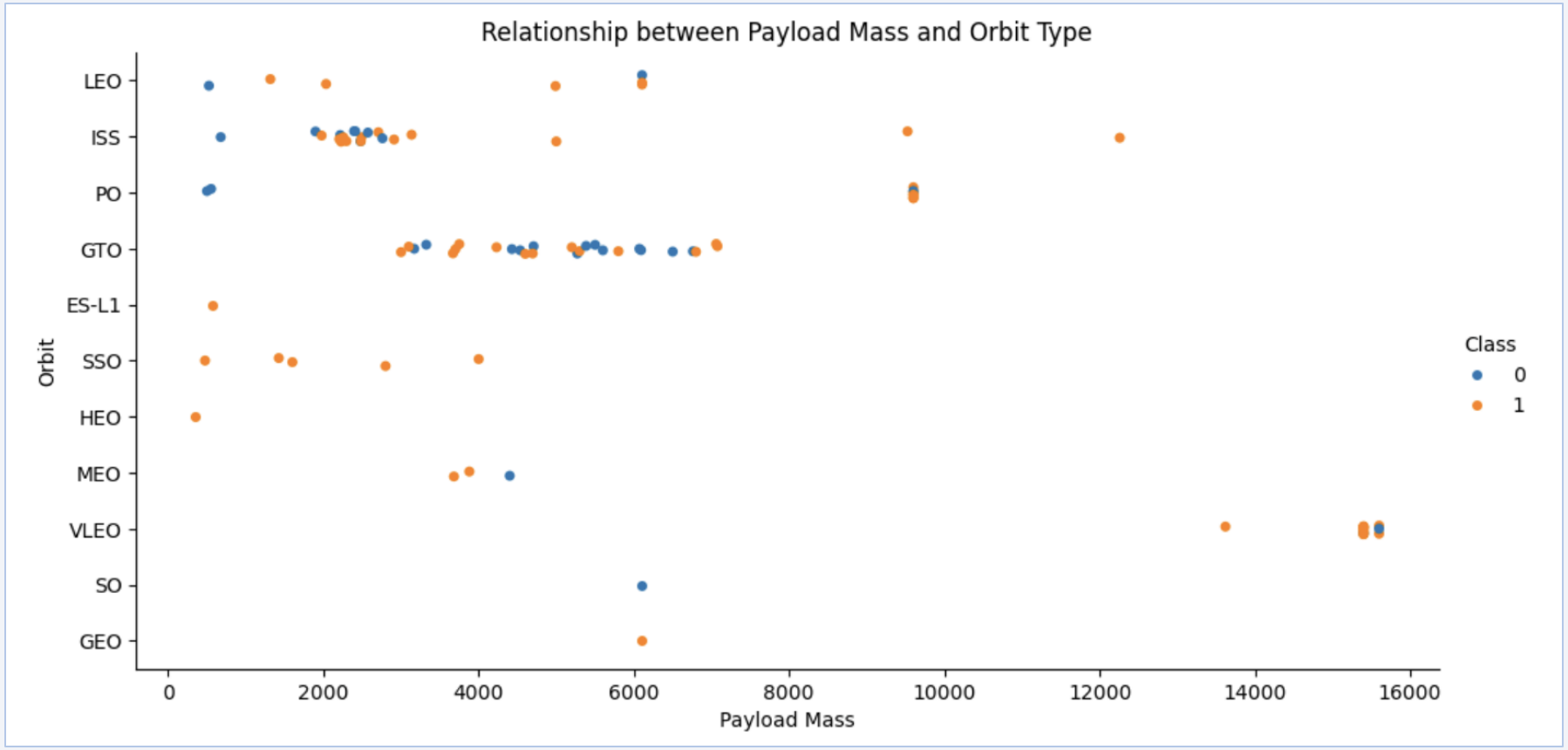
ES-L1, GEO, HEO, and SSO exhibit the highest success rates.

Flight Number vs. Orbit Type



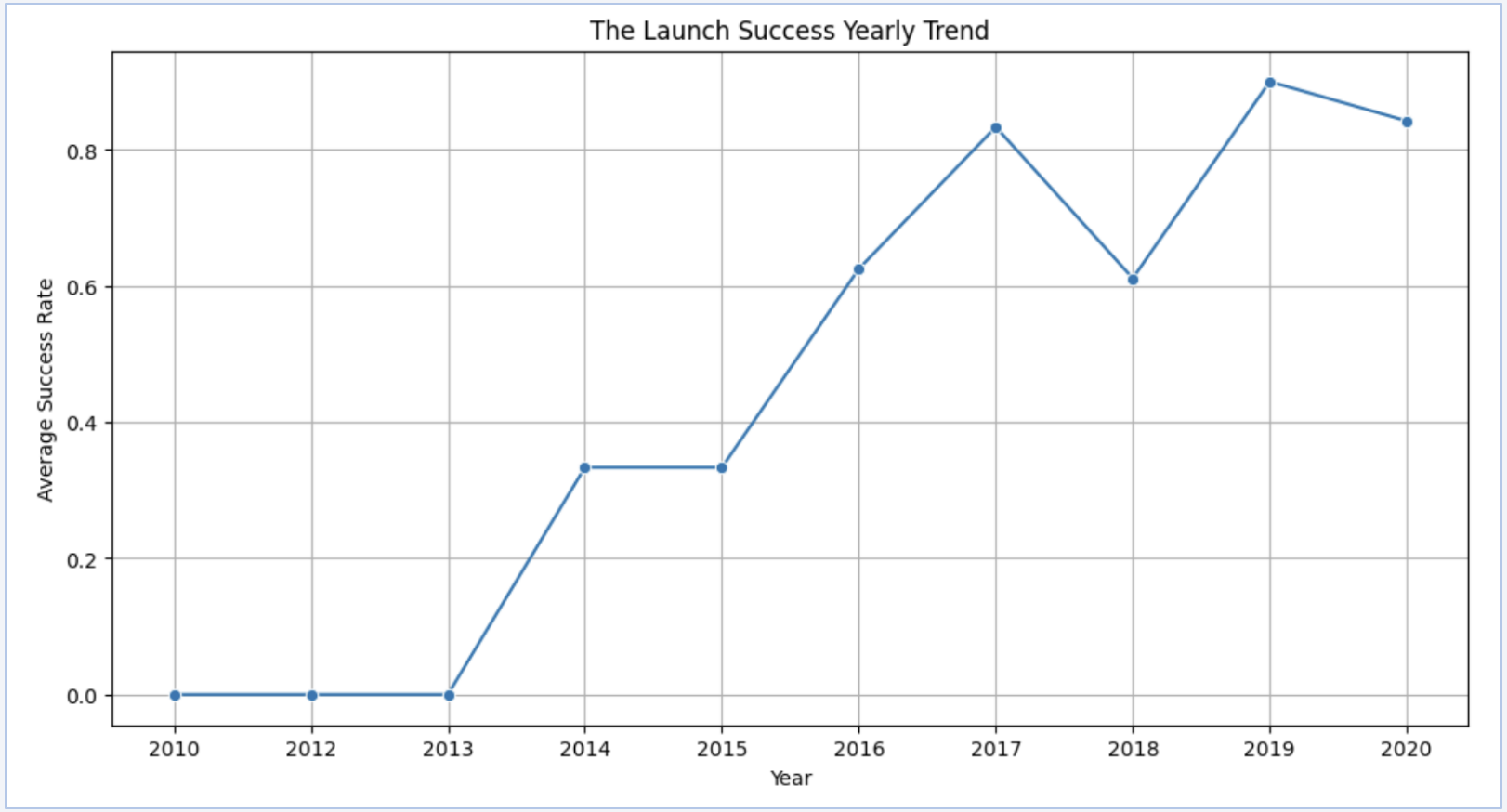
There is a noticeable increase in successful landings for LEO as the number of flights increases. In contrast, for GTO, ISS, PO, and VLEO, both successful and failed landings can be observed. Additionally, a higher proportion of successful landings occurs later in the launch schedule than earlier flights.

Payload Mass vs. Orbit Type



In LEO, PO, and ISS, there is a trend of increased successful landings as the payload mass increases. However, in GTO, the outcomes are mixed, making it difficult to differentiate between successful and failed landings. In GEO, HEO, and SSO, we observe successful outcomes. Orbit type does influence mission success, especially when combined with payload complexity.

Launch Success Yearly Trend



The overall trend shows a strong positive trajectory, with clear improvements in launch reliability over time. Consistent success begins to emerge in 2014, followed by a significant rise in performance from 2016 to 2020. This period is marked by high average success rates, peaking in 2019. Although there are minor dips in 2018 and 2020, they do not significantly impact the long-term upward trend.

All Launch Site Names

Query Explanation

This query retrieves the unique launch site names. The SELECT DISTINCT statement is used to eliminate duplicate rows from the result set and return unique Launch_Site values from the SPACEXTABLE.

```
select distinct "Launch_Site" from SPACEXTABLE order by "Launch_Site"
```

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

Launch Site Names Begin with 'CCA'

Query Explanation

This query retrieves the launch sites names that begin with "CCA". The LIKE operator is used in the WHERE clause to filter results based on a pattern. In this case, "CCA%", where the "%" wildcard allows for any sequence of characters following "CCA". The LIMIT clause is included to specify the number of records to return. In this case, only 5 records are returned from the SPACEXTABLE.

```
select * from SPACEXTABLE where "Launch_Site" like "CCA%" limit 5
```

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

Query Explanation

This query retrieves the total payload mass carried by boosters launched by NASA (CRS). The SUM() function returns the total of the numeric column PAYLOAD_MASS__KG_. The WHERE clause is used to filter records, focusing specifically on the NASA (CRS) customer. To ensure uniformity in text data retrieval, the LOWER() function is applied to the text in Customer.

```
select sum("PAYLOAD_MASS__KG_") as "Total_Payload_Mass" from SPACEXTABLE
where lower("Customer") = "nasa (crs)"
```

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

Query Explanation

This query retrieves the average payload mass carried by booster version F9 v1.1. The AVG() function returns the average value of the numeric column PAYLOAD_MASS__KG_. The WHERE clause is used to filter records, focusing specifically on the F9 v1.1booster version. To ensure uniformity in text data retrieval, the LOWER() function is applied to the text in Booster_Version.

```
select avg("PAYLOAD_MASS__KG_") as "Average_Payload_Mass" from SPACEXTABLE
where lower("Booster_Version") = "f9 v1.1"
```

Average_Payload_Mass
2928.4

First Successful Ground Landing Date

Query Explanation

This query retrieves the date of the first successful landing on a ground pad. The MIN() function is used to return the earliest value from the Date column. The WHERE clause is used to filter records, focusing specifically on successful landing outcomes on ground pad. To ensure uniformity in text data retrieval, the LOWER() function is applied to the text in Landing_Outcome.

```
select min("Date") as "First_Succesful_Landing" from SPACEXTABLE
where lower("Landing_Outcome") = "success (ground pad)"
```

First_Succesful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Query Explanation

This query retrieves the names of booster versions that successfully landed on a drone ship and carried a payload mass between 4000 and 6000 kg. The WHERE clause is used to filter records based on landing success, landing type, and payload range. To ensure uniformity in text data retrieval, the LOWER() function is applied to the text in Landing_Outcome.

```
select "Booster_Version" from SPACEXTABLE
where lower("Landing_Outcome") = "success (drone ship)" and
"PAYLOAD_MASS__KG_" between 4000 and 6000
```

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

Total Number of Successful and Failure Mission Outcomes

Query Explanation

This query retrieves the total number of successful and failed mission outcomes. It uses two subqueries within the SELECT statement, each applying the COUNT() function to count matching rows. The LIKE "%Success%" and LIKE "%Failure%" clauses capture mission outcomes that include these keywords.

```
select
(select count(*) from SPACEXTABLE where "Mission_Outcome" like "%Success%") as
"Total_Succesful",
(select count(*) from SPACEXTABLE where "Mission_Outcome" like "%Failure%") as
"Total_Failure"
```

Total_Succesful	Total_Failure
100	1

Boosters Carried Maximum Payload

Query Explanation

This query retrieves the names of booster versions that carried the maximum payload mass. It uses a subquery within the WHERE clause to identify the highest payload value by applying the MAX() function to the PAYLOAD_MASS__KG_ column. Only records matching this maximum value are returned. Finally, the results are sorted in ascending order based on the Booster_Version column using the ORDER BY clause.

```
select "Booster_Version" from SPACEXTABLE
where "PAYLOAD_MASS__KG_" = (
  select max("PAYLOAD_MASS__KG_") from SPACEXTABLE
) order by "Booster_Version"
```

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

2015 Launch Records

Query Explanation

This query retrieves a list of failed landing outcomes on drone ship, along with the corresponding booster versions and launch site names, including launch month names, for launches that occurred during the year 2015. The CASE statement is used to obtain month names based on the corresponding month numbers.

```
select
substr("Date", 6, 2) as "Month_Number",
case
when substr("Date", 6, 2) = "01" then "January"
when substr("Date", 6, 2) = '02' then 'February'
when substr("Date", 6, 2) = '03' then 'March'
when substr("Date", 6, 2) = '04' then 'April'
when substr("Date", 6, 2) = '05' then 'May'
when substr("Date", 6, 2) = '06' then 'June'
when substr("Date", 6, 2) = '07' then 'July'
when substr("Date", 6, 2) = '08' then 'August'
when substr("Date", 6, 2) = '09' then 'September'
when substr("Date", 6, 2) = '10' then 'October'
when substr("Date", 6, 2) = '11' then 'November'
when substr("Date", 6, 2) = '12' then 'December'
end as "Month_Name", "Landing_Outcome", "Booster_Version", "Launch_Site"
from SPACEXTABLE where lower("Landing_Outcome") like "failure%drone%ship%" and substr("Date", 0, 5) = '2015'
```

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

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

Query Explanation

This query retrieves a list of landing outcomes along with their occurrence counts and corresponding ranks. It uses the COUNT() function to calculate how often each outcome occurs and the RANK() function to assign a rank based on frequency. The data is filtered to include only records between 2010-06-04 and 2017-03-20, and the results are displayed in descending order of count.

```
select "Landing_Outcome", count(*) as "Outcome_Count",
rank() over (order by count(*) desc) as "Outcome_Rank" from SPACEXTABLE
where "Date" between "2010-06-04" and "2017-03-20" group by "Landing_Outcome"
```

Landing_Outcome	Outcome_Count	Outcome_Rank
No attempt	10	1
Success (drone ship)	5	2
Failure (drone ship)	5	2
Success (ground pad)	3	4
Controlled (ocean)	3	4
Uncontrolled (ocean)	2	6
Failure (parachute)	2	6
Precluded (drone ship)	1	8

Section 3

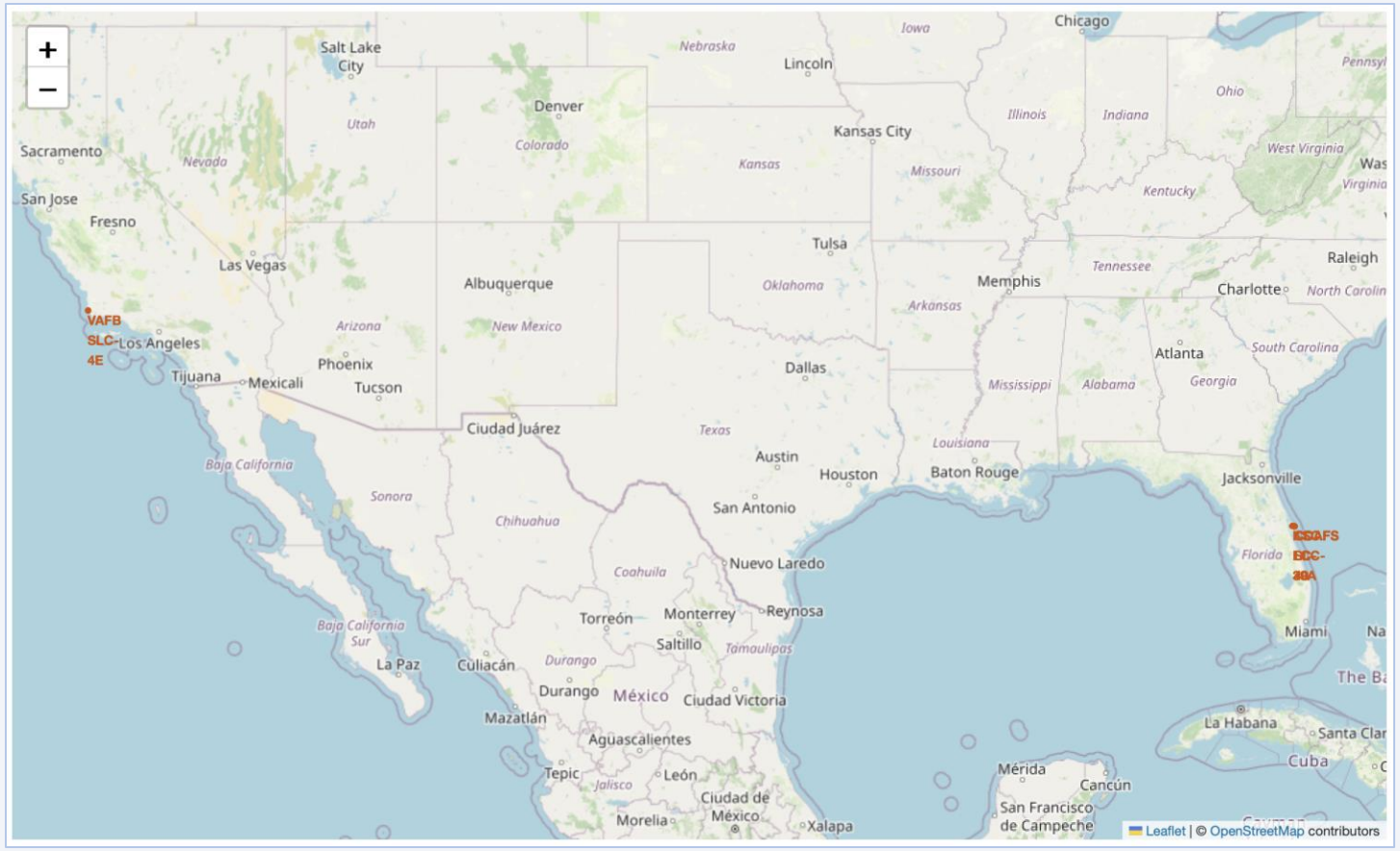
Launch Sites Proximities Analysis

All Launch Sites

All launch sites are based in the U.S., and located close to coastlines for safety and ocean overflight, and chosen to support a range of orbital destinations.

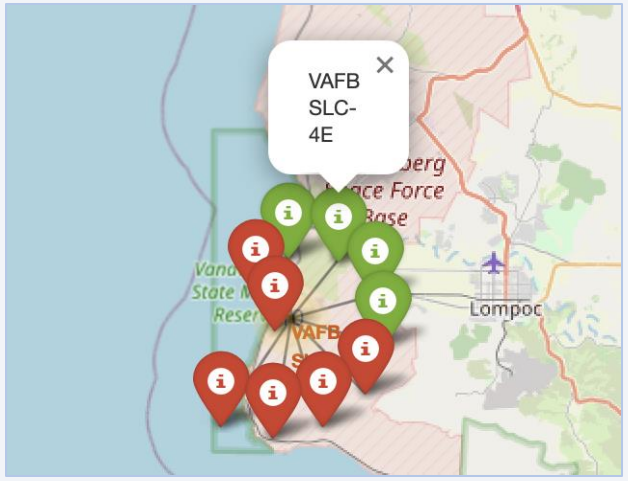
CCAFS LC-40, **CCAFS SLC-40**, and **KSC LC-39A** are located on the east coast of Florida. These launch sites benefit from proximity to the equator, which allows rockets to take full advantage of Earth's rotational speed for more efficient launches.

VAFB SLC-4E is located in California, supports southward launch trajectories over the Pacific Ocean. This positioning ensures safety and provides mission flexibility, particularly for PO and SSO orbit launches.



Successful and Failed Launches for Each Site

Launches for VAFB SLC-4E, located in California



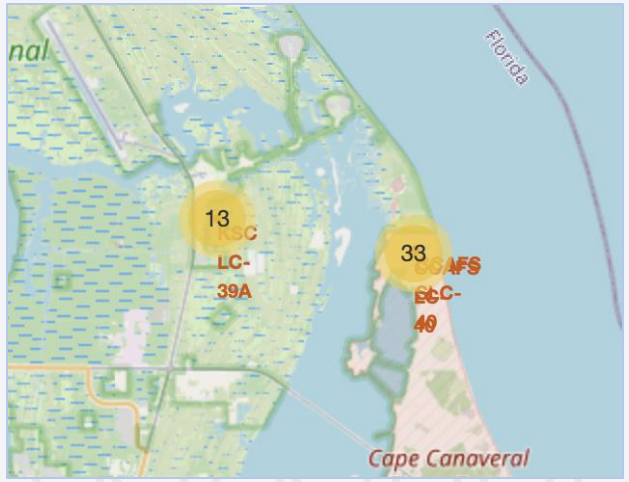
Color-labeled markers are used to display launch outcomes on the map.

Successful launches are marked with green markers.

Failed launches are marked with red markers.

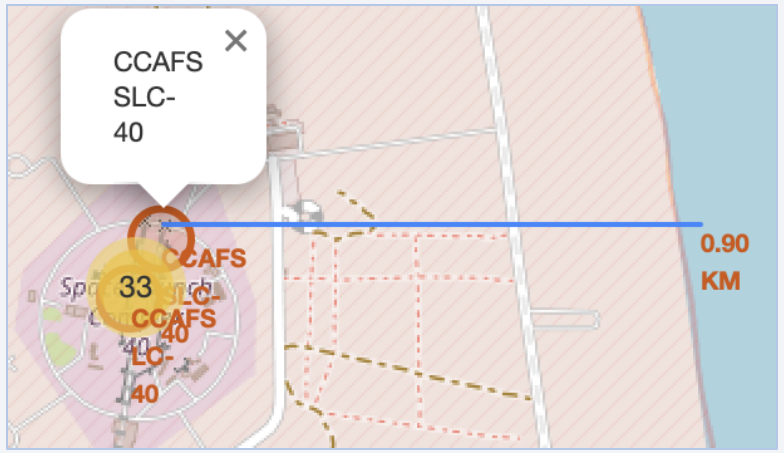
KSC LC-39A has the highest success rate among landing sites.

Launches for KSC LC-39A, CCAFS SLC-40, and CCAFS LC-40, located in Florida



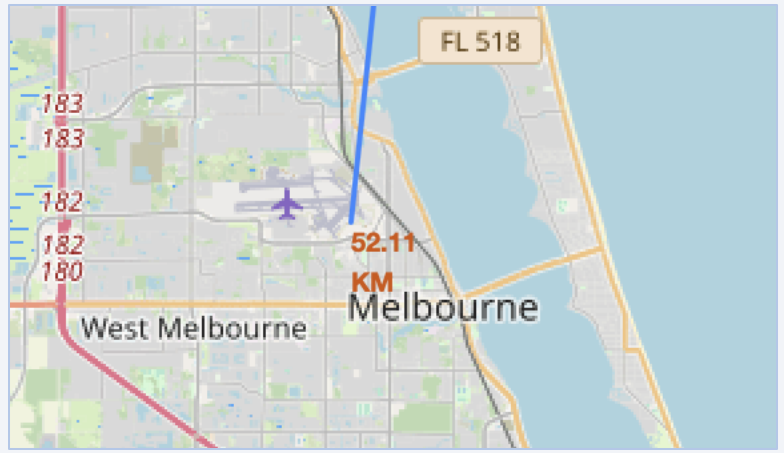
Proximity of the CCAFS SLC-40 Launch Site

Atlantic coastline point, FL

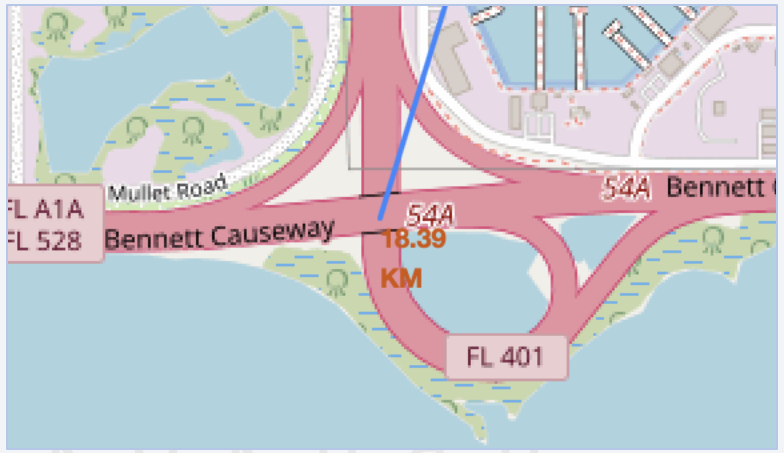


Lynx Central railway station, Orlando, FL

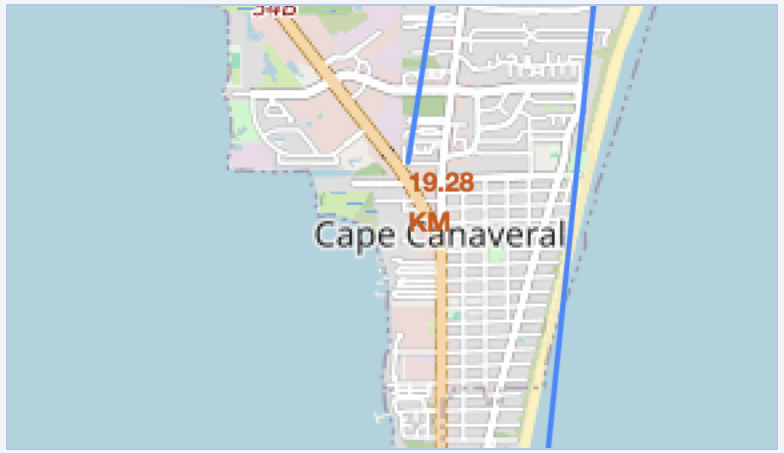
Melbourne Orlando International Airport, FL



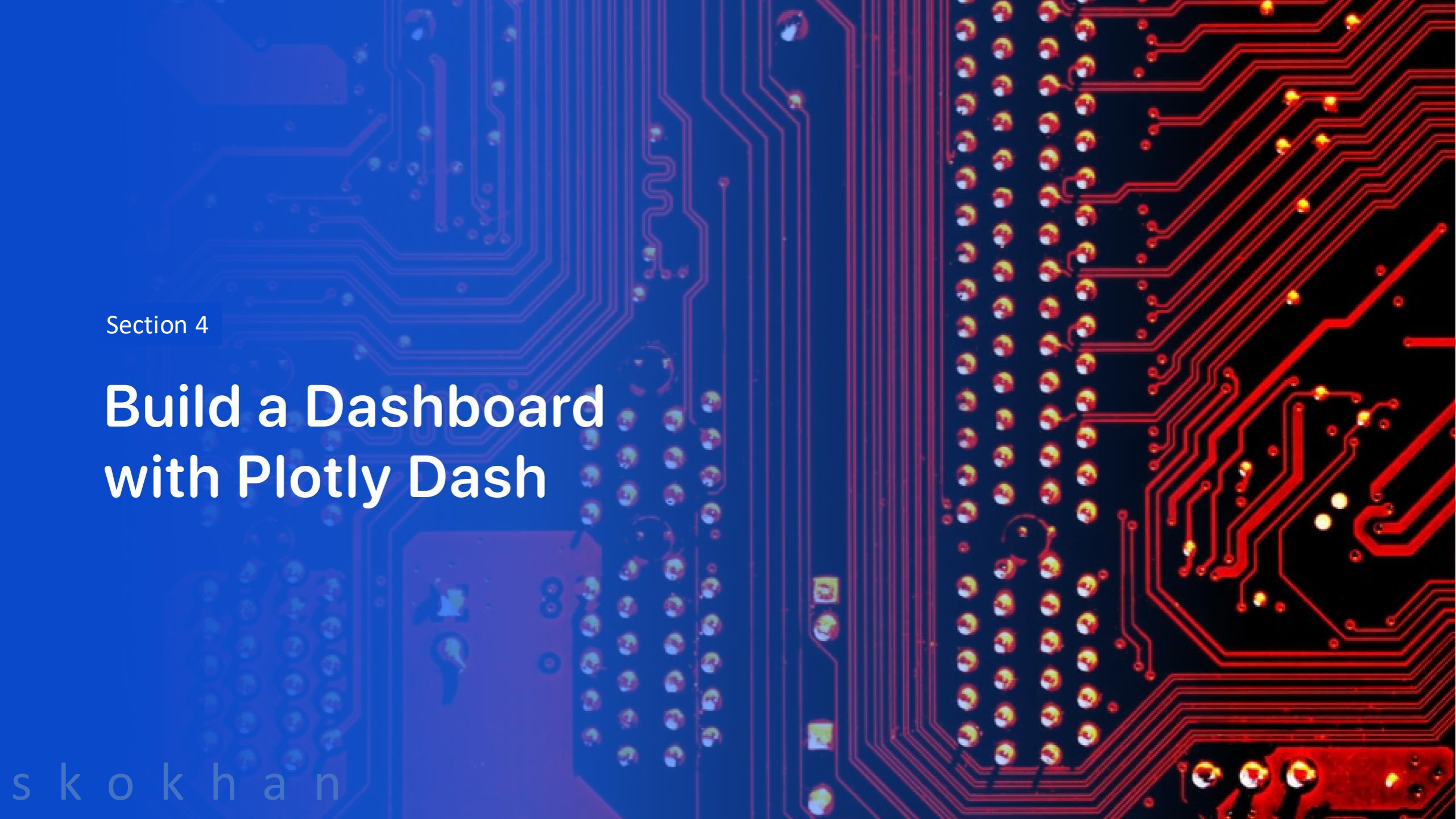
Highway, Cape Canaveral, FL



Cape Canaveral city, FL



CCAFS SLC-40 launch site location ensures safe launch paths over the Atlantic Ocean, safe drop zones and minimal risk to populated areas.

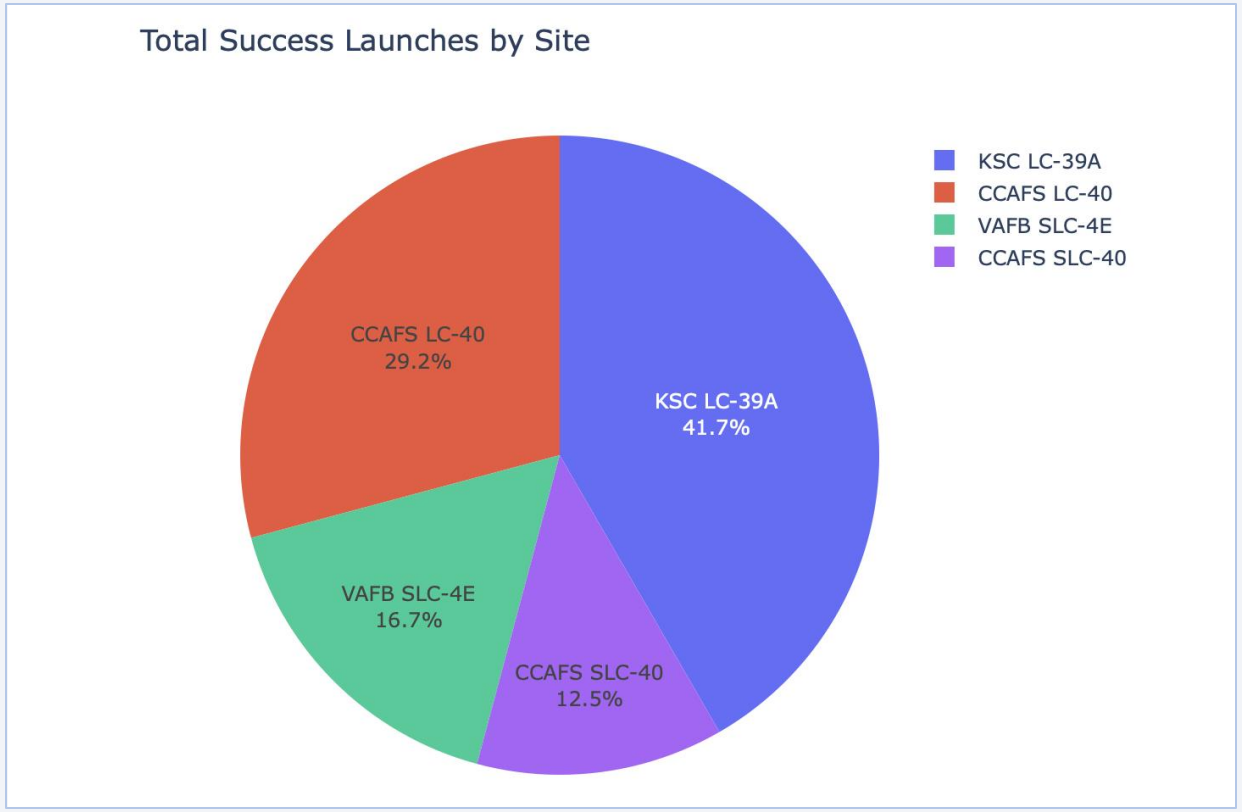


Section 4

Build a Dashboard with Plotly Dash

skokhan

Total Launch Success for All Sites

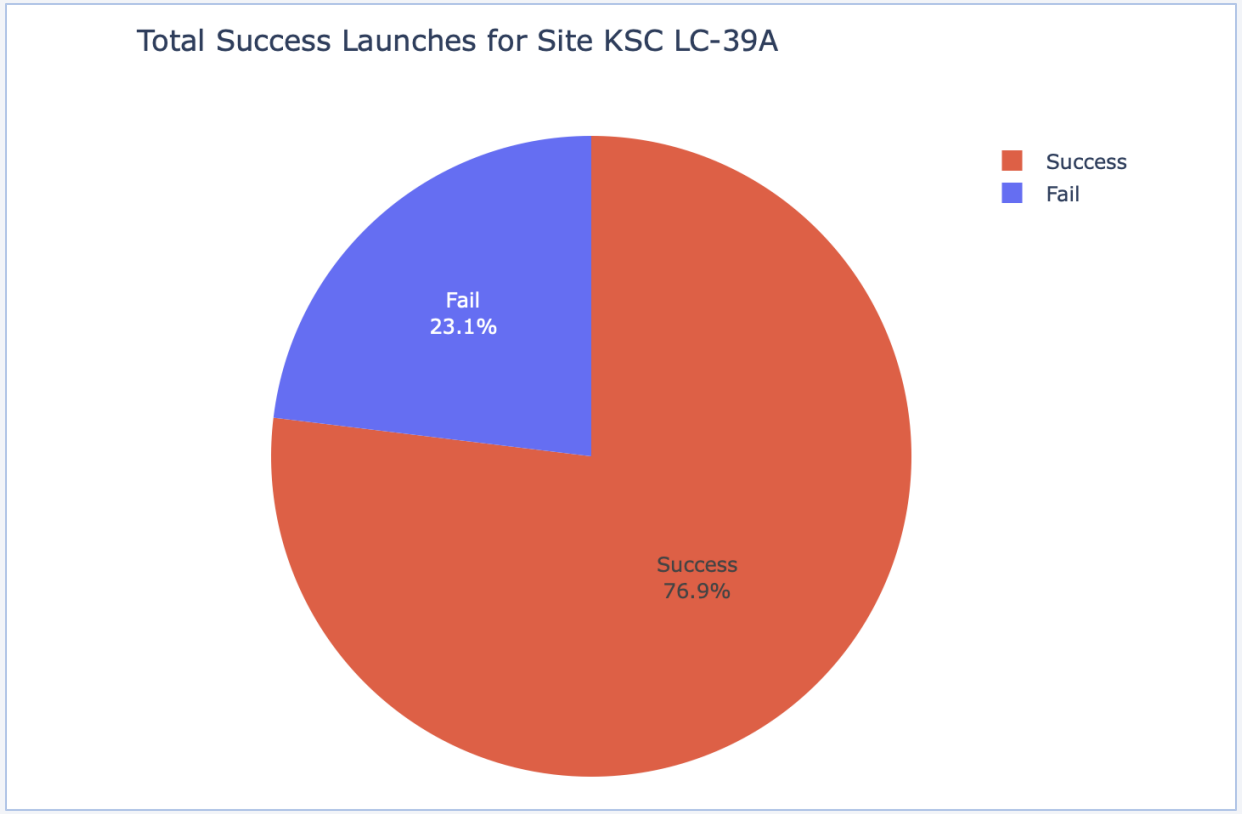


This pie chart presents the proportion of successful SpaceX launches from four different launch sites:

- KSC LC-39A leads with the highest percentage of successful launches at 41.7%, underscoring its key role in SpaceX's launch activities;
- CCAFS LC-40 follows with 29.2%, also a significant contributor, showing that Florida's east coast is a major hub;
- VAFB SLC-4E in California contributes 16.7%, likely due to its use for specific mission types;
- CCAFS SLC-40 holds the smallest portion at 12.5%, possibly indicating a lower mission frequency or a shorter operational timeframe.

This data highlights the strategic importance of Florida's launch sites, especially KSC LC-39A, in SpaceX's success story.

KSC LC-39A – High Success Rate



This pie chart presents the proportion of successful and failed launch outcomes at the KSC LC-39A launch site.

- A high success rate is observed, with 76.9% of launches achieving their intended objectives;
- Failures account for 23.1%, suggesting that while performance is strong, there remains potential for further improvement.

This data highlights KSC LC-39A as not only the most frequently used site, but also one demonstrating a consistently high success rate, making it a key asset in launch reliability and operational planning.

Payload vs. Launch Outcome

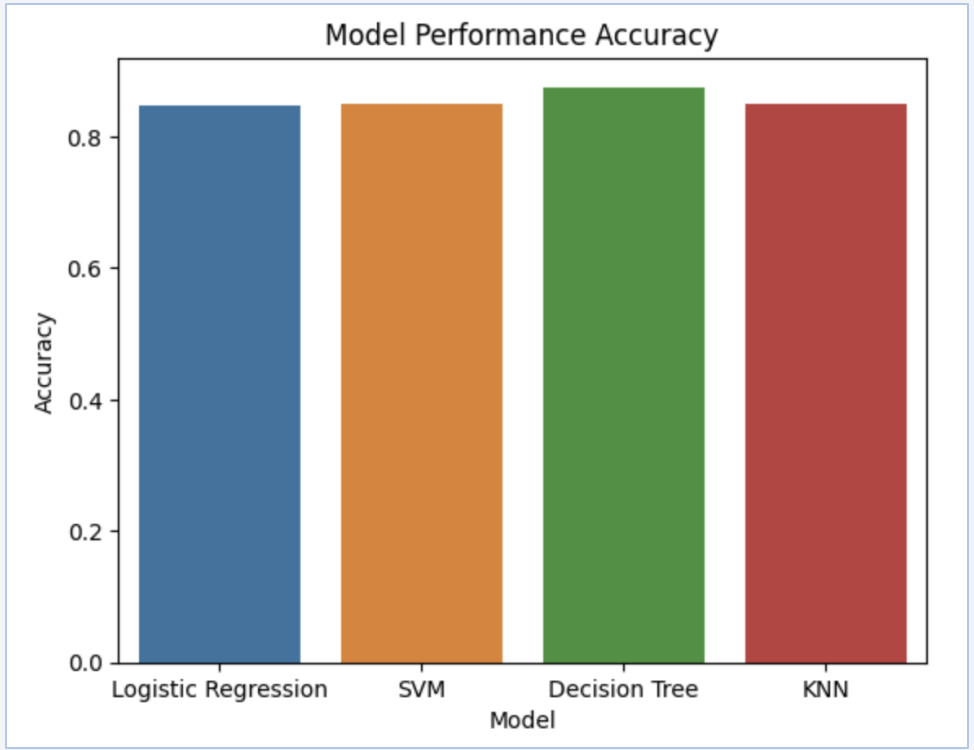
Over time, a clear upward trend is observed, where both payload capacity and reliability improve with newer booster versions.

- Low-range payloads (under 3,000 kg): earlier booster versions v1.0 and v1.1, show more variability in success;
- Mid-range payloads (3,000–7,000 kg): modern boosters FT, B4, and B5 display improved success rates; notably the FT booster shows successful launches around the 5,000 kg range;
- High-range payloads (above 7,000 kg): successful launches indicating the reliability of the newer booster versions in handling heavier payloads.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



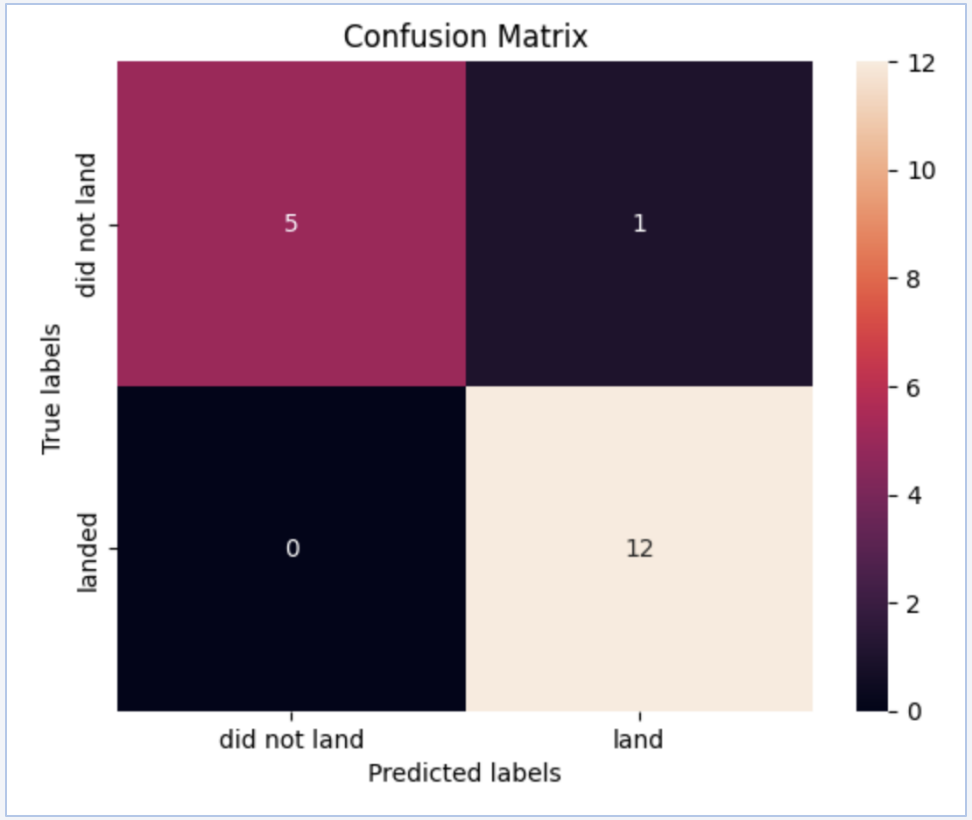
The bar plot illustrates performance accuracy scores of four machine learning models used to predict SpaceX launch success:

- Logistic Regression, SVM (Support Vector Machine), and KNN (K-Nearest Neighbors) show similar performance, with training accuracy scores ranging from 84.64% to 84.82%, and a consistent test accuracy of 83.33%, indicating stable and reliable performance among these classifiers on the SpaceX dataset;
- Decision Tree Classifier stands out with the highest training accuracy of 87.5% and a significantly higher test accuracy of 94.44%, indicating it not only learned well from the training data but also generalized effectively to unseen data, making it the best-performing model for predicting launch success.

Model	Performance Accuracy
Logistic Regression	0.846429
SVM	0.848214
Decision Tree	0.875000
KNN	0.848214

Model	Test Accuracy
Logistic Regression	0.833333
SVM	0.833333
Decision Tree	0.944444
KNN	0.833333

Confusion Matrix – Decision Tree Classifier



This confusion matrix illustrates the performance of the Decision Tree Classifier in predicting the SpaceX launch success:

- The model correctly classified 12 successful landings and 5 failed landings;
- Only 1 misclassification occurred — a false positive, where a failed landing was mistakenly predicted as successful;
- No false negatives, indicating the model never missed a successful landing.

The model is very effective at capturing all actual landings. It has a slight imperfection in predicting failures, but overall this is a strong and well-performing model for predicting launch success.

Conclusions

- The overall trend shows a strong positive improvements in mission reliability over time;
- There is a notable improvement in landing success rates, as the launch number increases;
- Both payload capacity and mission reliability have improved;
- Low to mid-weight payloads have higher landing success;
- Newer booster models (FT, B4, B5) show strong reliability, even with heavier payloads;
- Missions at orbits ES-L1, GEO, HEO, and SSO are more successful;
- The KSC LC-39A launch site shows top successful landing rates;
- Machine learning models demonstrate strong predictive capability on the SpaceX dataset;
- The Decision Tree Classifier achieved the highest performance, accurately capturing all successful landings;
- Publicly available SpaceX dataset is sufficiently informative for predictive modeling.

Appendix – A

- SQL query

  Since SQLite does not provide built-in functions to convert month numbers into month names, the CASE statement is used to perform this conversion manually.

```
select
substr("Date", 6, 2) as "Month_Number",
case
when substr("Date", 6, 2) = "01" then "January"
when substr("Date", 6, 2) = '02' then 'February'
when substr("Date", 6, 2) = '03' then 'March'
when substr("Date", 6, 2) = '04' then 'April'
when substr("Date", 6, 2) = '05' then 'May'
when substr("Date", 6, 2) = '06' then 'June'
when substr("Date", 6, 2) = '07' then 'July'
when substr("Date", 6, 2) = '08' then 'August'
when substr("Date", 6, 2) = '09' then 'September'
when substr("Date", 6, 2) = '10' then 'October'
when substr("Date", 6, 2) = '11' then 'November'
when substr("Date", 6, 2) = '12' then 'December'
end as "Month_Name", "Landing_Outcome", "Booster_Version", "Launch_Site"
from SPACEXTABLE where lower("Landing_Outcome") like "failure%drone%ship%" and
substr("Date", 0, 5) = '2015'
```

Appendix – B

- SQL query

  This query retrieves the total count of successful and failed mission outcomes by analyzing detailed entries in the Mission_Outcome column.

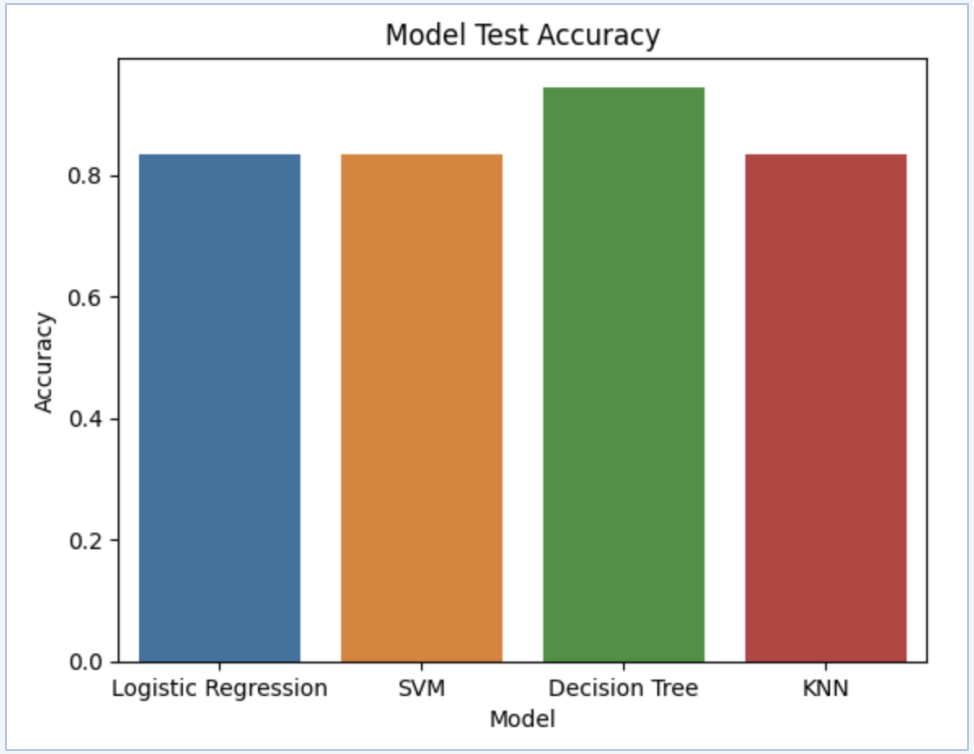
```
select trim("Mission_Outcome") as "Mission_Outcome",
count(*) as "Total_Count" from SPACEXTABLE group by trim("Mission_Outcome")
```

Mission_Outcome	Total_Count
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Appendix - C

- Bar Plot

  This graph highlights that the Decision Tree Classifier outperforms the other models, achieving the highest test accuracy of 94.44%.



Appendix - D

- Python code

  The classification_report() function is used to assess the performance of a classification model. It provides a clear and comprehensive summary of key metrics — such as precision, recall, F1-score, and support — for each class, offering valuable insights into how well the model performs across different categories.

```
from sklearn.metrics import classification_report

# Logistic Regression
y_pred_lr = logreg_cv.predict(X_test)
print("Logistic Regression Report:\n", classification_report(Y_test, y_pred_lr))

# Support Vector Machine
y_pred_svm = svm_cv.predict(X_test)
print("SVM Report:\n", classification_report(Y_test, y_pred_svm))

# Decision Tree Classifier
y_pred_tree = tree_cv.predict(X_test)
print("Decision Tree Report:\n", classification_report(Y_test, y_pred_tree))

# k-Nearest Neighbors
y_pred_knn = knn_cv.predict(X_test)
print("kNN Report:\n", classification_report(Y_test, y_pred_knn))
```

Thank you!

skokhan