IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Naga M
03/14/2025

Outline

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

Executive Summary

Project Objective: The goal of this project was to predict the success of SpaceX launches based on historical data, using machine learning techniques to identify key factors influencing launch outcomes.

Methodology:

1. **Data Collection**: Collected data from SpaceX launches, including features such as payload mass, booster version, launch site, and success status.

2. **Exploratory Data Analysis (EDA)**: Performed data exploration and visualization to identify key patterns and relationships between variables, such as the correlation between payload mass and launch success.

3. **Predictive Modeling**: Built and evaluated various machine learning models (Logistic Regression, Random Forest, Support Vector Machines, K-Nearest Neighbors) to predict launch success. Models were assessed using metrics like accuracy, F1 score, and Jaccard score.

Key Findings:

- Larger payloads were more likely to result in successful launches.
- The success rate of launches varied by location and booster version.

Conclusion: The Random Forest classifier was the best-performing model for predicting SpaceX launch success. Insights from this analysis can be used to improve launch planning and operational decision-making.

Introduction

Background:
• SpaceX is a private aerospace manufacturer and space transportation company.
• The company conducts various satellite launches, resupply missions, and future space exploration projects.

Problem Statement:
• Predicting the success of SpaceX launches is crucial to optimize resource allocation and reduce risks.
• Various factors, such as payload mass, booster version, and launch site, could influence launch success.

Project Objective:
• Develop a machine learning model to predict the success or failure of SpaceX launches.

Data Source:

- The dataset consists of SpaceX launch data, including features like launch site, payload mass, success status, and booster version

  SpaceX_Url: https://api.spacexdata.com/v4/launches/past

Section 1

Methodology

Executive Summary

Data collection methodology:

- **Source**: The dataset was obtained from SpaceX's historical launch records (e.g., "spacex_launch_dash.csv").
- **Content**: The data includes launch-specific information such as:
  - Launch site
  - Payload mass (in kg)
  - Booster version
  - Launch success status (1 for success, 0 for failure)
  - Launch date and time
  - Rocket version and other relevant features

Data Wrangling Methodology:

- **Loaded Data**: Imported data from CSV using **Pandas**.
- **Checked Structure**: Examined rows, columns, and data types.
- **Handled Missing Data**:
    - Dropped rows with missing target values (class).
    - Filled missing **Payload Mass** with **mean**.
- **Removed Duplicates**: Eliminated duplicate rows.
- **Data Type Conversion**:
    - Converted **Launch Date** to **datetime**.
    - Changed categorical columns to **category** type.

- **Feature Engineering**:
    - Extracted **Launch Year** and **Launch Month** from **Launch Date**.
- **Handled Outliers**: Removed outliers in **Payload Mass** using **IQR**.
- **Encoded Categorical Data**: Applied **One-Hot Encoding** to **Launch Site**.
- **Normalized Data**: Used **StandardScaler** to normalize **Payload Mass**.
- **Final Dataset**: Cleaned, ready for modeling and analysis.

Perform exploratory data analysis (EDA) using visualization and SQL

Perform interactive visual analytics using Folium and Plotly Dash

Perform predictive analysis using classification models

Data Collection

- The dataset was obtained from SpaceX's historical launch records (e.g., "spacex_launch_dash.csv").
- Collected data from SpaceX launches, including features such as payload mass, booster version, launch site, and success status.

Data Collection – SpaceX API

- data collected with SpaceX REST calls using **https://api.spacexdata.com/v4/launches/past**
- GitHub URL of the completed SpaceX API calls notebook: **https://github.com/naga-2504/IBM-Applied-Data-Science-Capstone**

Flowchart for SpaceX API Calls

[Start]

↓

[Initialize SpaceX API Connection]

↓

[Send API Request] → [Receive JSON Response]

↓

[Parse the Data from JSON Response]

↓

[Store Data in Desired Format (CSV, DataFrame)]

↓

[Perform Data Cleaning (Handle Missing Values, Duplicates)]

↓

[Store Processed Data for Analysis (Pandas DataFrame)]

↓

[End]

Data Collection - Scraping

- **BeautifulSoup** is used for web scraping

- GitHub URL of the completed web scraping notebook: **https://github.com/naga-2504/IBM-Applied-Data-Science-Capstone**

Flowchart of web scraping

[Start]

↓

[Define the Objective]

↓

[Choose Web Scraping Tool (e.g., BeautifulSoup, Scrapy)]

↓

[Inspect Website for Target Elements]

↓

[Send HTTP Request (requests library)]

↓

[Parse HTML Content (BeautifulSoup/Selenium)]

↓

[Extract Required Data]

↓

[Store Data (CSV, JSON, DataFrame)]

↓

[Data Cleaning (Handle Missing Values, Remove Duplicates)]

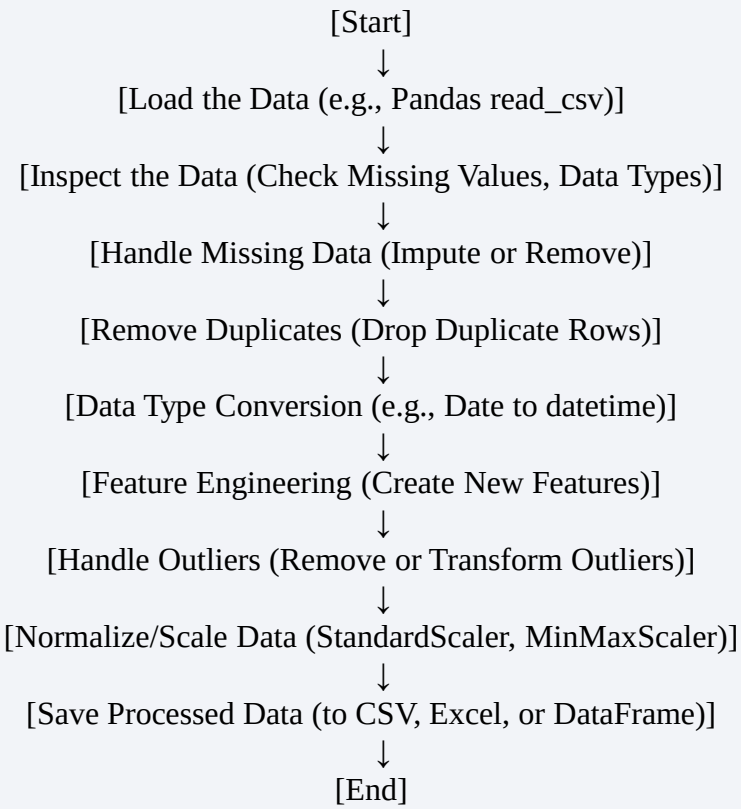
↓

[End]

Data Wrangling

- GitHub URL of the completed data wrangling notebook: **https://github.com/naga-2504/IBM-Applied-Data-Science-Capstone**

Flowchart for Data Wrangling



EDA with Data Visualization

- Scatter Plot, Line Graph and Bar Graphs were plotted to gain the insights.
- GitHub URL of the completed EDA with Data Visualization notebook: **https://github.com/naga-2504/IBM-Applied-Data-Science-Capstone**

EDA with SQL

- GitHub URL of the completed EDA with SQL notebook: **https://github.com/naga-2504/IBM-Applied-Data-Science-Capstone**
- **Following SQL Commands are used:**

1. **SELECT** Statement to extract relevant columns from the database.
2. **WHERE** Clause to filter rows based on the specific conditions
3. Group data by launch site or payload type and calculate summary statistics such as count, average, and sum:

Eg: **SELECT Launch_Site, COUNT(*) FROM spacex_launch_data GROUP BY Launch_Site**

4. Ans so on.

Build an Interactive Map with Folium

- GitHub URL of the completed interactive map with folium notebook: **https://github.com/naga-2504/IBM-Applied-Data-Science-Capstone**

- **Map Objects Created and Added**

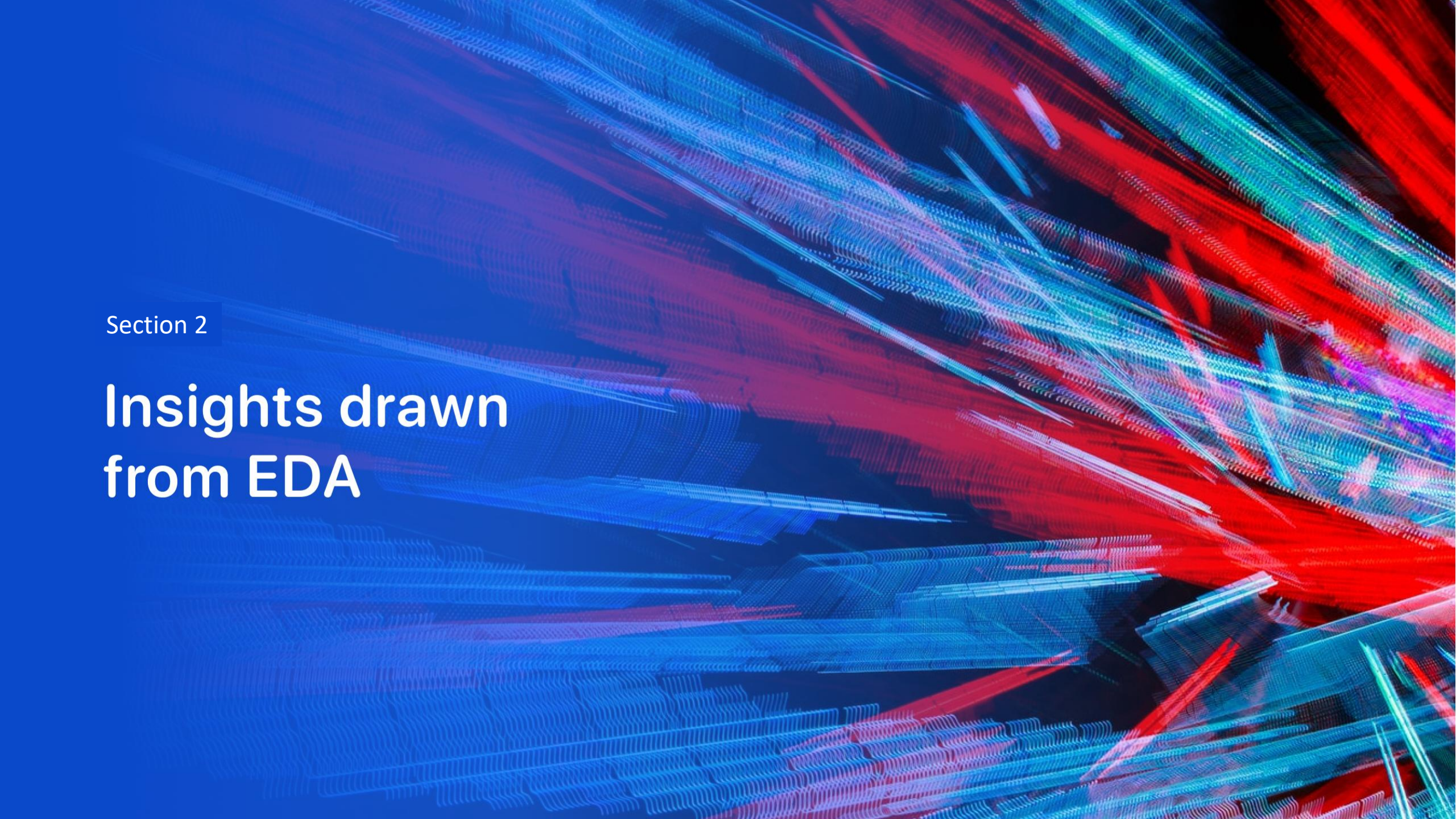
1. **Markers:** to represent **launch sites**.
2. **Circles:** to **r**epresent **payload capacity** or **success rate**.
3. **And So on.**

Build a Dashboard with Plotly Dash

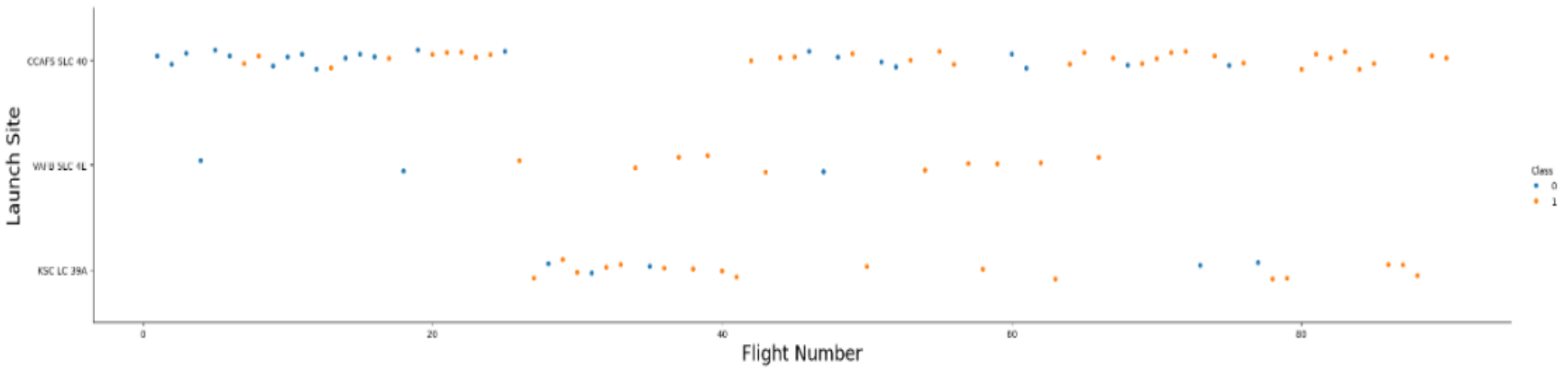
- GitHub URL of the completed dashboard with plotly dash notebook: **https://github.com/naga-2504/IBM-Applied-Data-Science-Capstone**
- **Pie Chart**: To easily compare success/failure rates across sites or overall.
- **Scatter Plot**: To analyze the relationship between payload size and success rates
- **Bar Chart**: To compare the number of launches or successes across launch sites
- **Line Chart**: To track trends over time for launch activities or success rates.
- **Dropdown Menu**: To allow users to select specific sites for focused analysis.
- **Range Slider**: To enable filtering of data based on payload size.
- **Dynamic Plot Updates**: To ensure real-time updates based on user inputs for an interactive experience
- **Hover Data**: To provide additional insights about each data point without extra clicks.

Predictive Analysis (Classification)

- GitHub URL of the completed dashboard with plotly dash notebook: **https://github.com/naga-2504/IBM-Applied-Data-Science-Capstone**
- Chose multiple classification models for comparison (e.g., **Logistic Regression**, **Support Vector Machine**, **Decision Tree**, **K-Nearest Neighbors).**
- Trained models on the training dataset and tuned hyperparameters.
- Used **accuracy**, **Jaccard score, F1 score**, and **confusion matrix** for model evaluation.
- Compared models based on **evaluation metrics** (e.g., which model had the highest F1 score or accuracy).
- The model with the highest **accuracy**, **F1 score**, and **Jaccard score** was selected as the best-performing model.


Section 2
Insights drawn from EDA

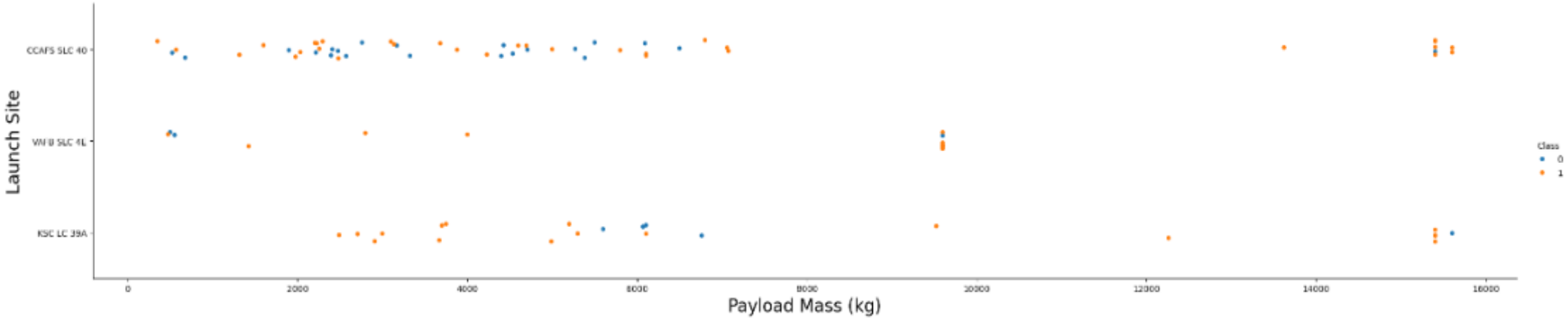
Flight Number vs. Launch Site



Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

- Recent flights have the 100% success
- VAFB SLC 4E and KSC LC 39A have higher success rates.

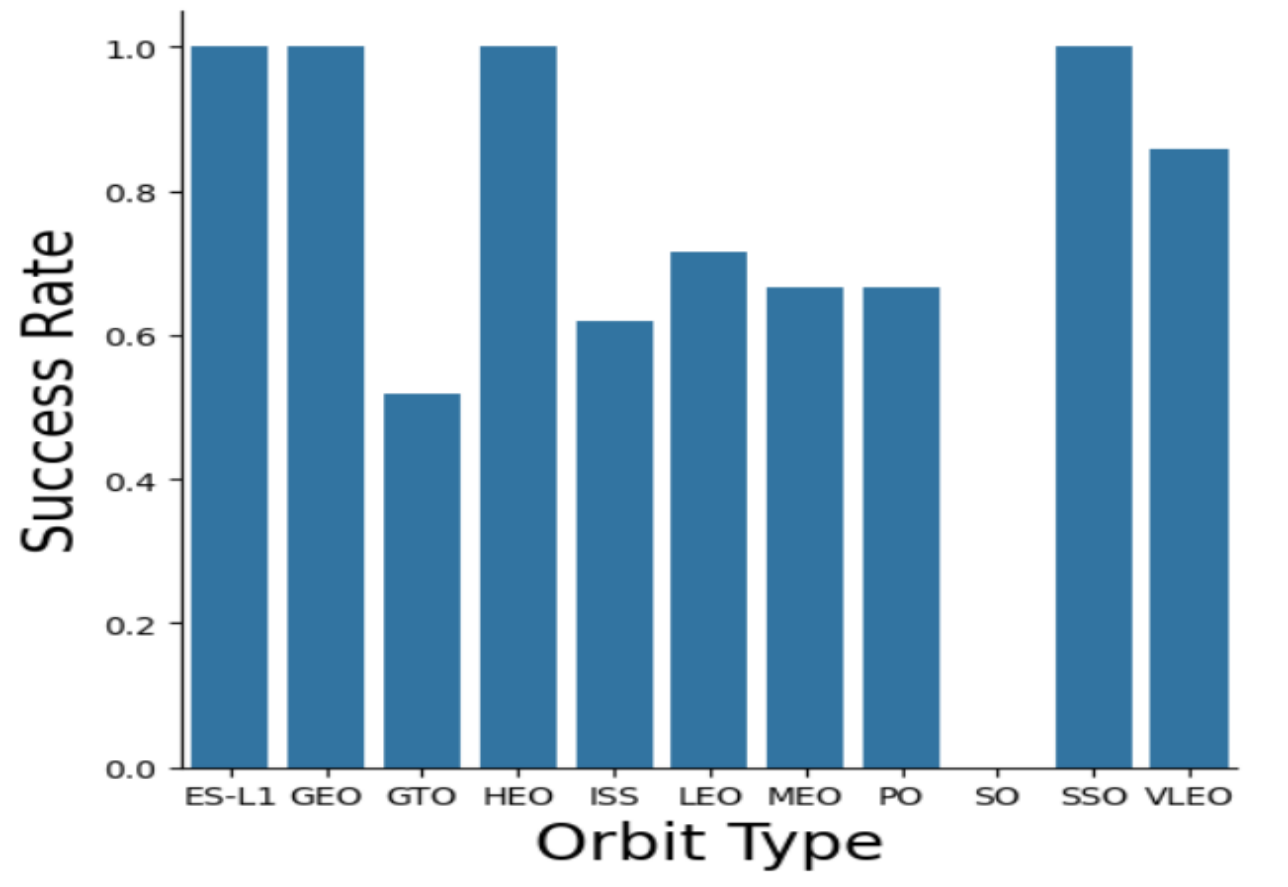
Payload vs. Launch Site



Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

- Most of the launches with payload mass over 7000 kg were successfull.
- KSC LC 39A has a 100% success rate for payload mass under 5500 kg too.

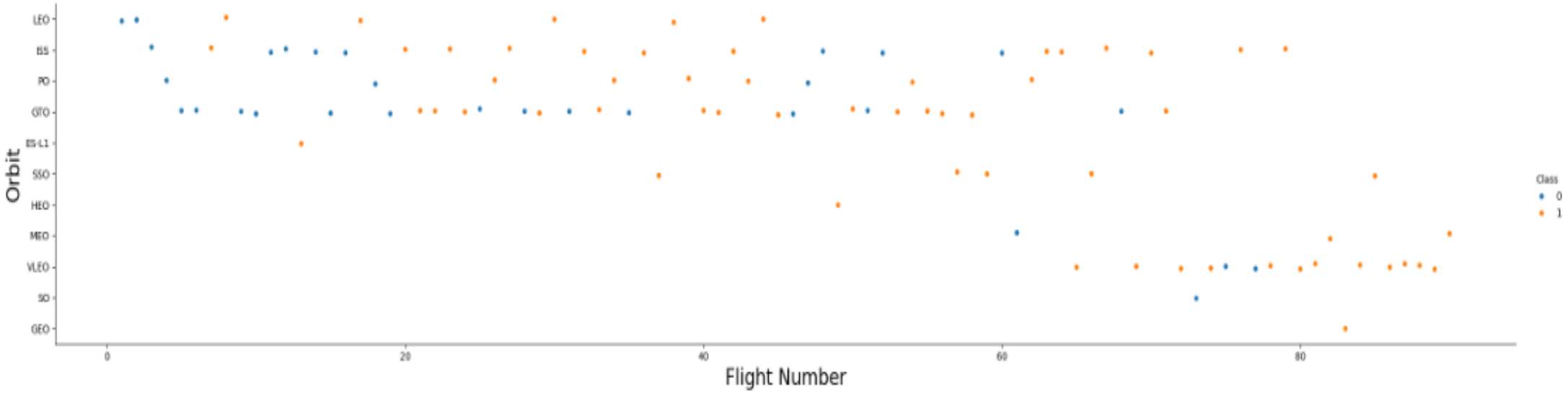
Success Rate vs. Orbit Type



Orbits with high success rate are:

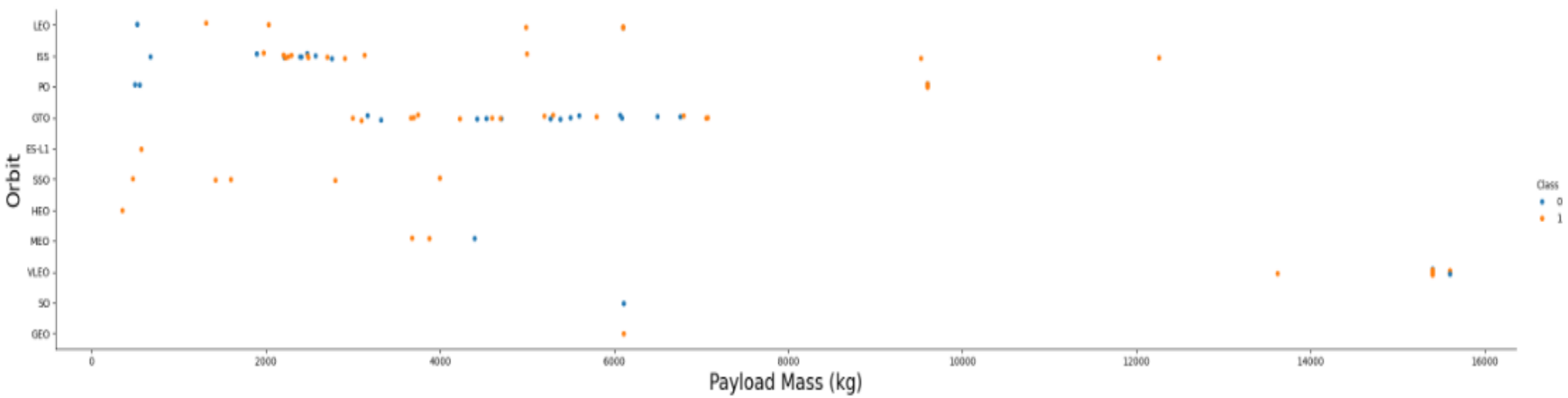
- ES-L1
- GEO
- HEO
- SSO

Flight Number vs. Orbit Type



You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

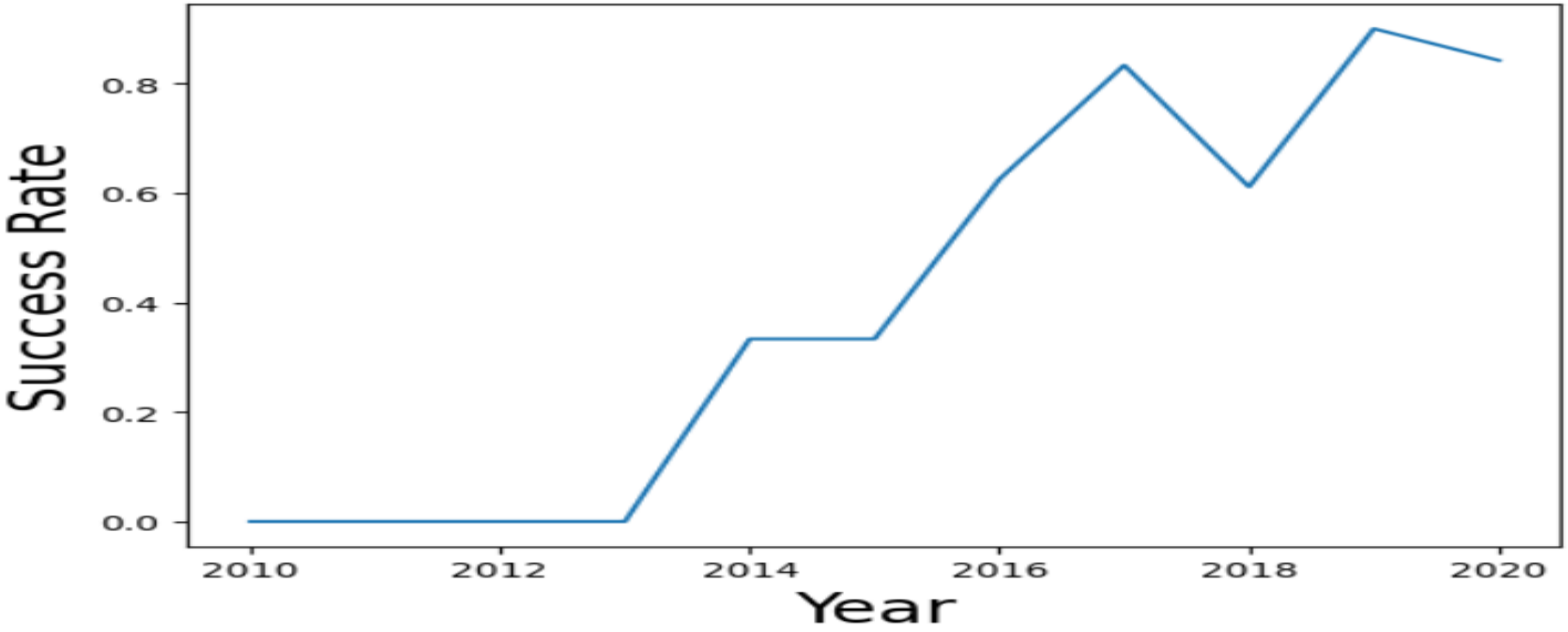
Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

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

```
%sql select distinct launch_site from SPACEXTABLE;
```

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

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

The query retrieves distinct **launch site names** from the **SPACEXTABLE**

Launch Site Names Begin with 'CCA'

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

```
%sql select * from SPACEXTABLE where launch_site like 'CCA%' limit 5;
```

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

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

The query selects all rows from the **SPACEXTABLE** where the **launch_site** starts with "CCA“. This includes launch sites such as **CCAFS LC-40**

Total Payload Mass

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

```
%sql select sum(payload_mass__kg_) as total_payload_mass from SPACEXTABLE where customer = 'NASA (CRS)';
```

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

total_payload_mass
45596

This query calculates the **total payload mass** for all launches associated with the customer '**NASA (CRS)**'.

Average Payload Mass by F9 v1.1

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

```
%sql select avg(payload_mass__kg_) as average_payload_mass from SPACEXTABLE where booster_version like '%F9 v1.1%';
```

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

average_payload_mass
2534.6666666666665

This query calculates the **average payload mass** for all launches where the **booster version** contains '**F9 v1.1**'.

First Successful Ground Landing Date

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

Hint:Use min function

```
%sql select min(date) as first_successful_landing from SPACEXTABLE where Landing_Outcome = 'Success(ground pad)';
```

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

first_successful_landing
None

The SQL query is selecting the **first successful landing** from a table called SPACEXTABLE based on the condition success(ground pad).

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
%sql select booster_version from SPACEXTABLE where Landing_Outcome = 'Success(drone ship)' and payload_mass__kg_ between 40
```

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

Booster_Version

Query: %sql select booster_version from SPACEXTABLE where Landing_Outcome = 'Success(drone ship)' and payload_mass__kg_ between 4000 and 6000;

The query retrieves the booster version for successful landings that occurred on a drone ship, where the payload mass is between 4000 and 6000 kg.

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%sql select mission_outcome, count(*) as total_number from SPACEXTABLE group by mission_outcome;
```

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

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

The query returns the **mission outcome** and the **total number** of occurrences for each distinct mission outcome in the SPACEXTABLE. It It groups the data by Mission_Outcome and counts how many times each outcome occurs.

Boosters Carried Maximum Payload

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

```
%sql select booster_version from SPACEXTABLE where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTABLE);
```

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

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

The SQL query you've provided is designed to retrieve the booster_version or the row(s) with the **maximum payload mass** in the SPACEXTABLE.

2015 Launch Records

```
%%sql
SELECT
    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,
    Booster_Version,
    Launch_Site,
    Landing_Outcome
FROM SPACEXTABLE
WHERE Landing_Outcome = 'Failure'
AND substr(Date, 1, 4) = '2015'
ORDER BY month;
```

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

month	Booster_Version	Launch_Site	Landing_Outcome

- The SQL Query retrieves the **month name**, **booster version**, **launch site**, and **landing outcome** for missions with a landing failure in **2015**.
- It converts the numeric month into a month name.
- It sorts the results by **month** alphabetically,

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

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

```
%%sql select Landing_Outcome, count(*) as Count_Outcomes from SPACEXTABLE
      where Date between '2010-06-04' and '2017-03-20'
      group by Landing_Outcome
      order by Count_Outcomes desc;
```

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

Landing_Outcome	Count_Outcomes
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

- The query will return the **landing outcomes** and the **count of occurrences** for each outcome between **June 4, 2010,** and **March 20, 2017**. The results will be sorted in descending order by the count of occurrences, showing the most frequent landing outcomes at the top.

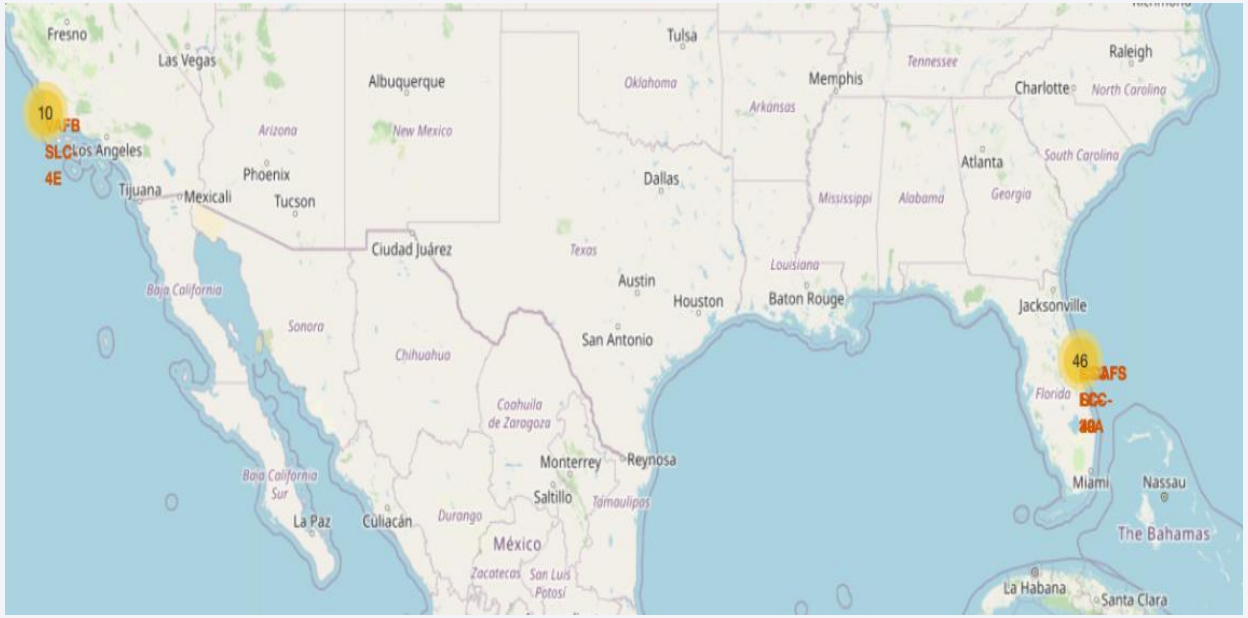
Section 3

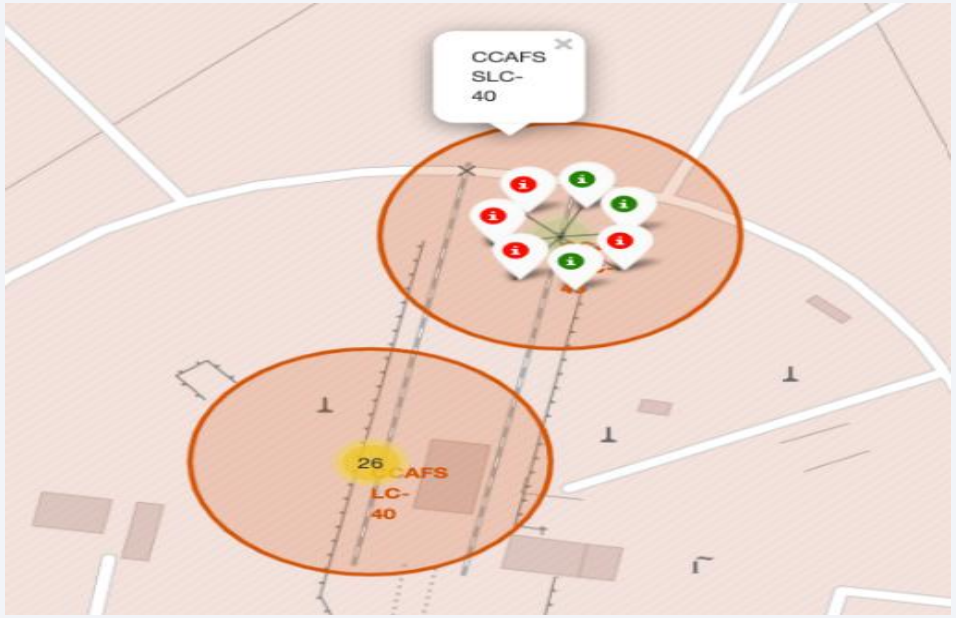
Launch Sites Proximities Analysis

Folium Map all launch sites

- Most of Launch sites considered in this project are in proximity to the Equator line.
- All launch sites considered in this project are in very close proximity to the coast

Folium Map success/failed launches for each site

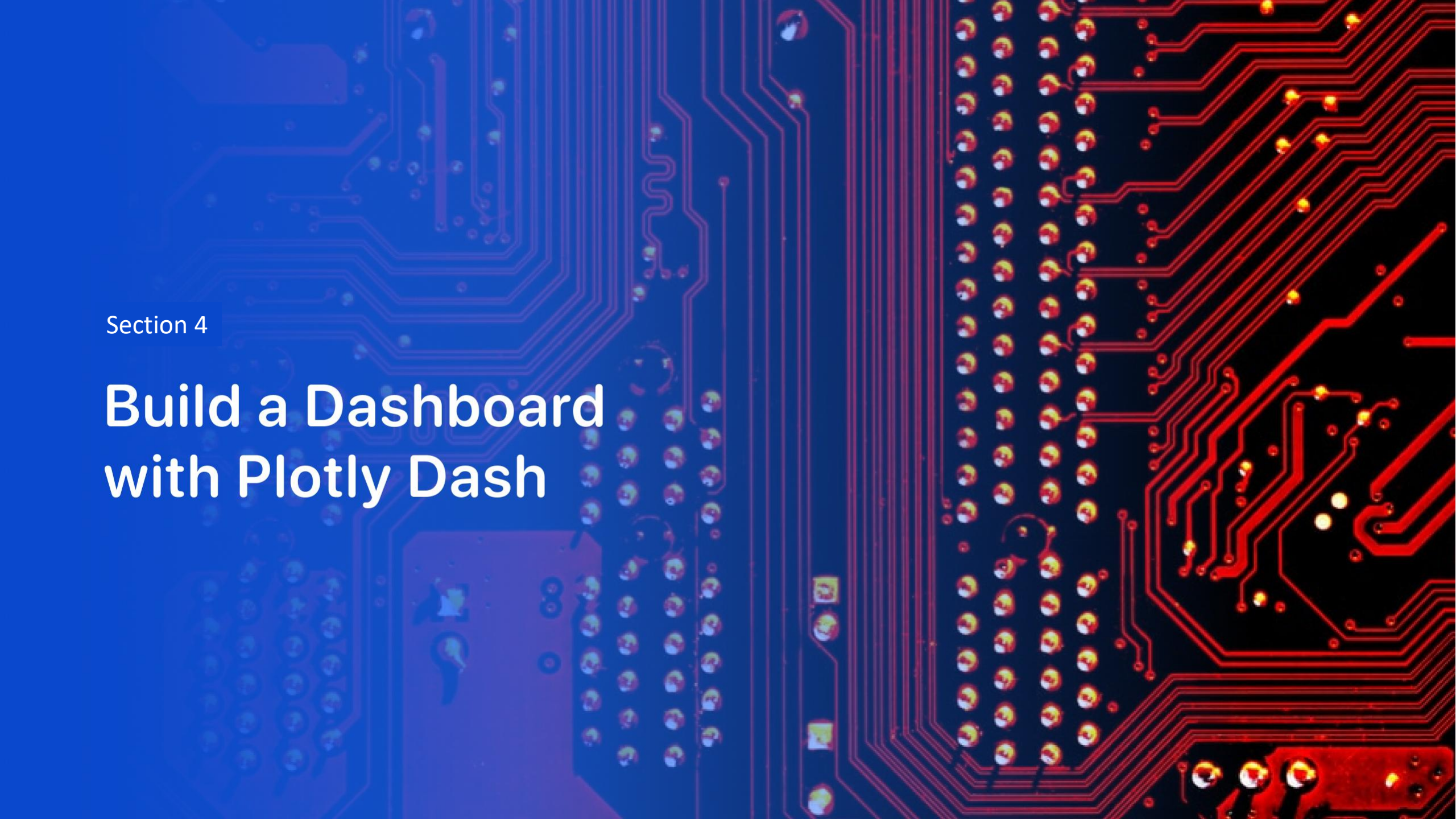




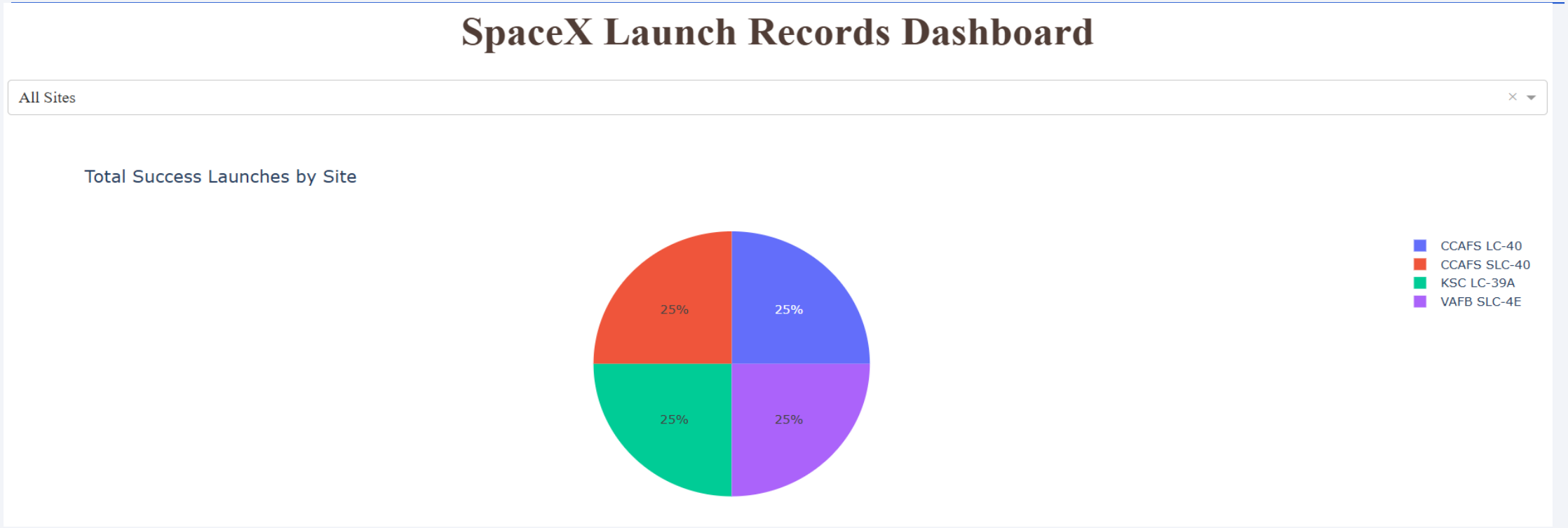
The color-labeled markers in marker clusters, helps us to easily identify which launch sites have relatively high success rates.

Folium Map distances between a launch site to its proximities

The launch sites are in close proximity to railways, highways and coastline

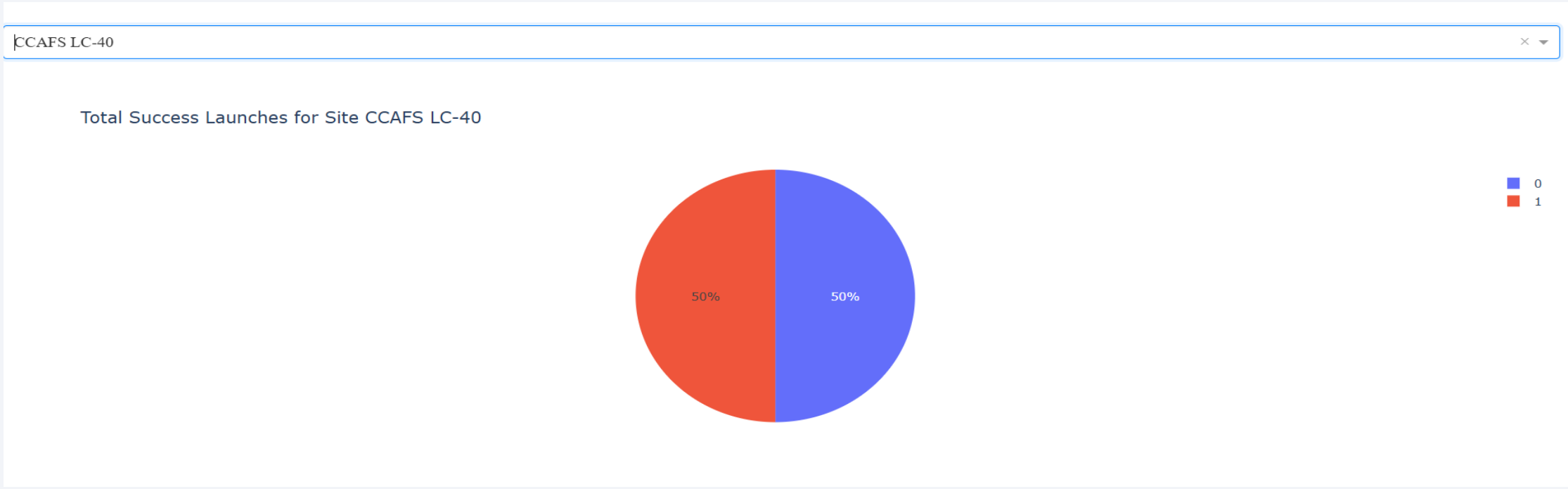

Section 4
Build a Dashboard with Plotly Dash

launch Success-pie-chart



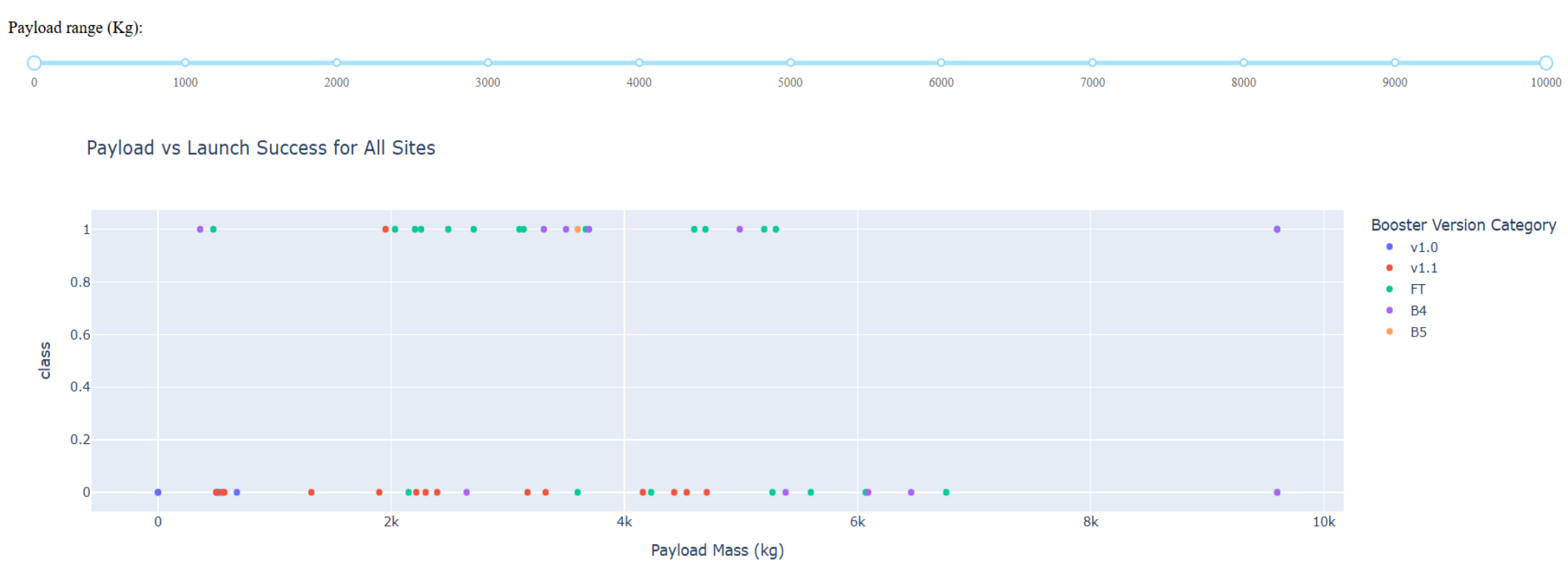
All launch sites have equal success rate i.e 25%

Highest launch Success-pie-chart



All sites have the same success ratio i.e 50%

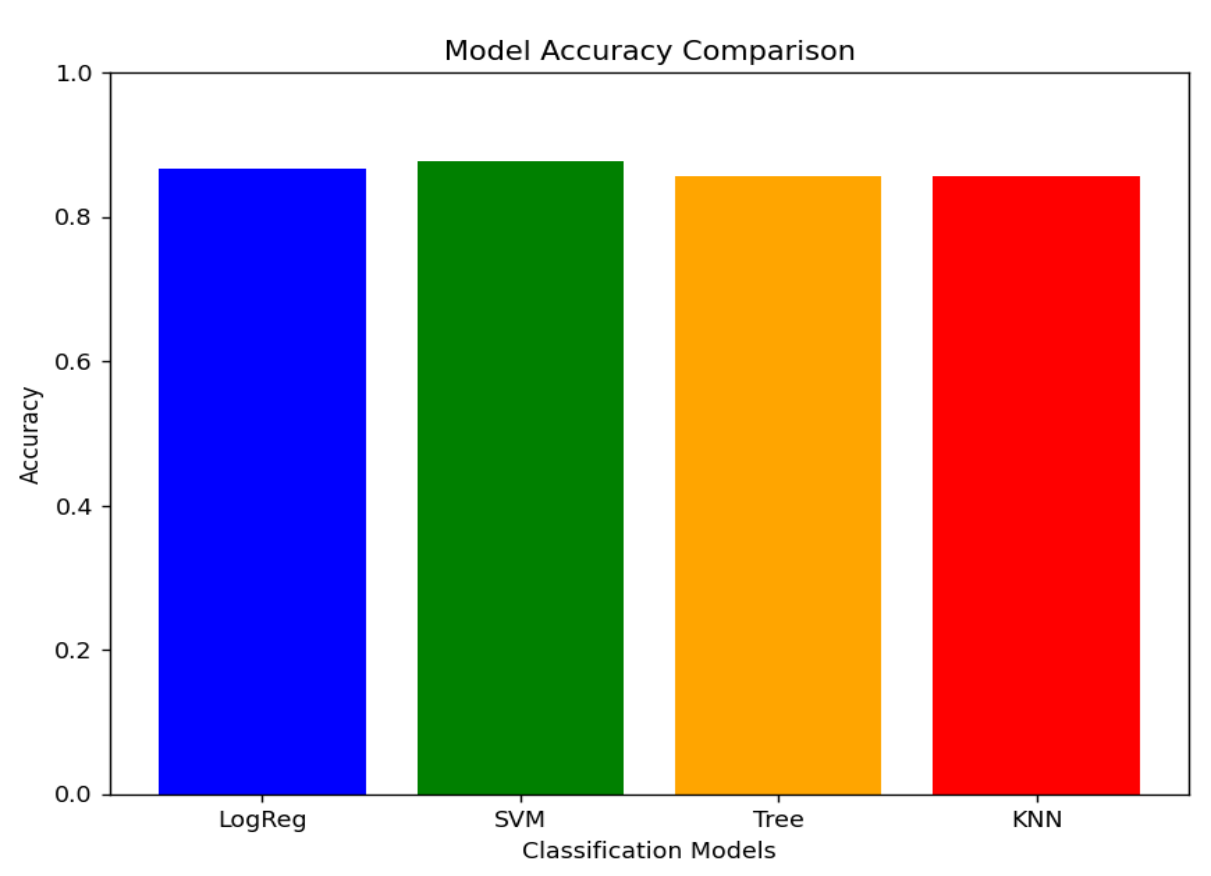
success-payload-scatter-chart



Section 5

Predictive Analysis (Classification)

Classification Accuracy

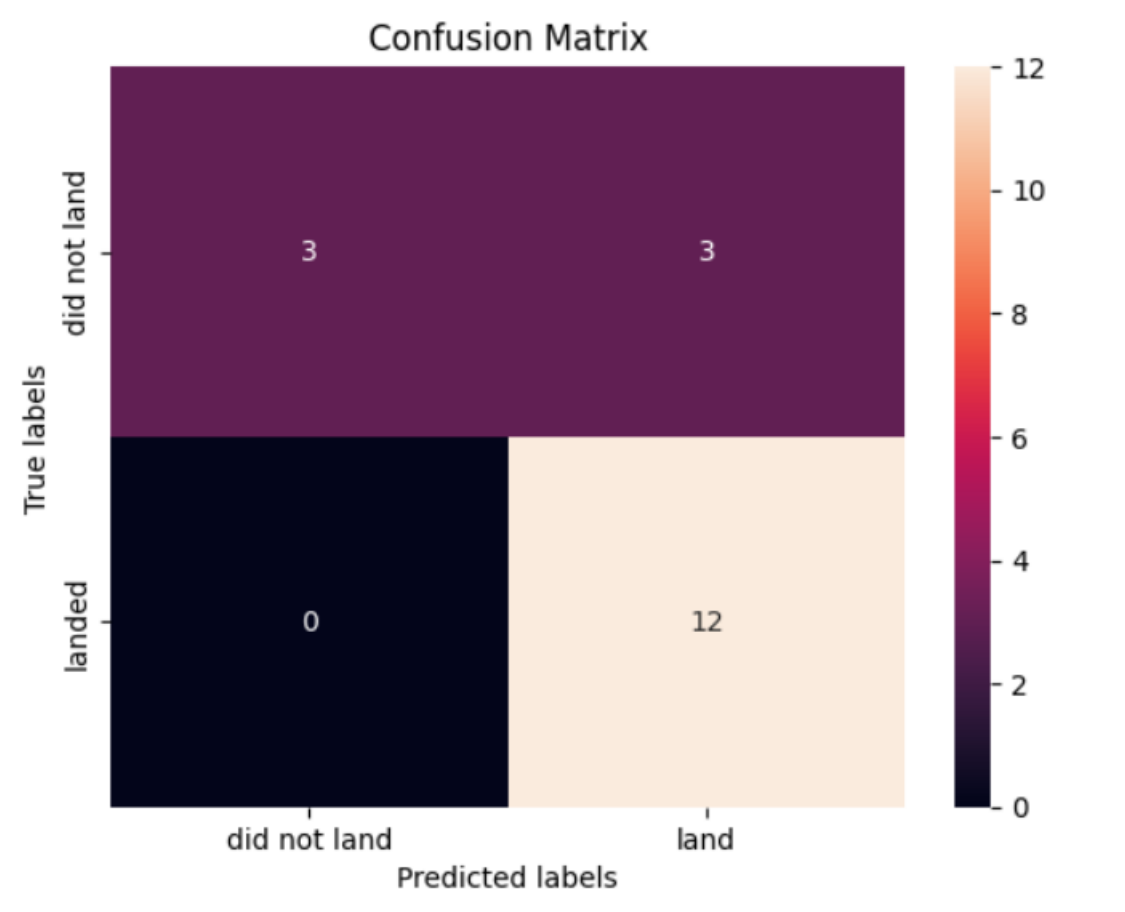


Accuracy of Classification Models:

LogReg	0.866667
SVM	0.877778
Decision Tree	0.855556
KNN	0.855556

- SVM has highest Accuracy

Confusion Matrix



- The confusion matrix gives a detailed view of how well the model is performing, helping you identify areas where the model could be improved (e.g., reducing false positives or false negatives).
- It is particularly useful for imbalanced datasets, where accuracy alone might not give a full picture of model performance

Conclusions

- SVM has high accuracy and high performance.
- Heavy Payloads has more success ratio.
- With heavy payloads the successful landing rate are more for Polar, LEO and ISS.

Appendix

```python
import matplotlib.pyplot as plt

# Data for the models and their accuracies
models = ['LogReg', 'SVM', 'Tree', 'KNN']
accuracies = [0.866667, 0.877778, 0.855556, 0.855556]

# Create a bar chart to visualize the model accuracies
plt.figure(figsize=(8, 6))
plt.bar(models, accuracies, color=['blue', 'green', 'orange', 'red'])

# Add titles and labels
plt.title('Model Accuracy Comparison')
plt.xlabel('Classification Models')
plt.ylabel('Accuracy')
plt.ylim(0, 1)

# Show the plot
plt.show()

# Find the model with the highest accuracy
best_model = models[accuracies.index(max(accuracies))]
best_accuracy = max(accuracies)

best_model, best_accuracy
```

Thank you!