IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Chike Egonu
21/01/2025

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection: Historical launch data including rocket specs, weather conditions, and first-stage landing outcomes.
  - Predictive Modeling: Trained machine learning models (Random Forest, Gradient Boosting) to predict first-stage reuse.
  - Cost Dashboard: Developed a tool that calculates estimated launch costs based on payload, mission complexity, and reuse probability.
- Summary of all results
  - Model Performance: Achieved 85% accuracy in predicting first-stage reuse with an F1 score of 0.82.
  - Cost Savings: Reusing the first stage could reduce launch costs by 30-50%.
  - Insights: Weather, payload, and previous landing success influence reuse likelihood.

Introduction

- The project is a collaboration with SpaceY to analyze and predict the costs of SpaceX launches, with a specific focus on the reuse of the Falcon rocket's first stage. The goal is to provide data-driven insights that can help optimize launch costs and improve decision-making for future missions.
- The key problems we aim to address are how to accurately predict the cost of each SpaceX launch and what factors influence the successful reuse of the Falcon rocket’s first stage. By solving these problems, we can enhance the efficiency and cost-effectiveness of future launches.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected from the SpaceX API and web scraping of Falcon 9 launch records, then cleaned and enriched by filtering out Falcon 1 launches and handling missing values for analysis and prediction.
- Perform data wrangling
  - Data was processed by filtering out Falcon 1 launches, replacing missing PayloadMass values with the mean, converting the Outcome column into a binary classification, and enriching the dataset using the API for additional details.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - We standardize data, split it into training and testing sets, train models with Grid Search for optimal hyperparameters, and evaluate them using accuracy and a confusion matrix to find the best model for predicting Falcon 9's landing success.

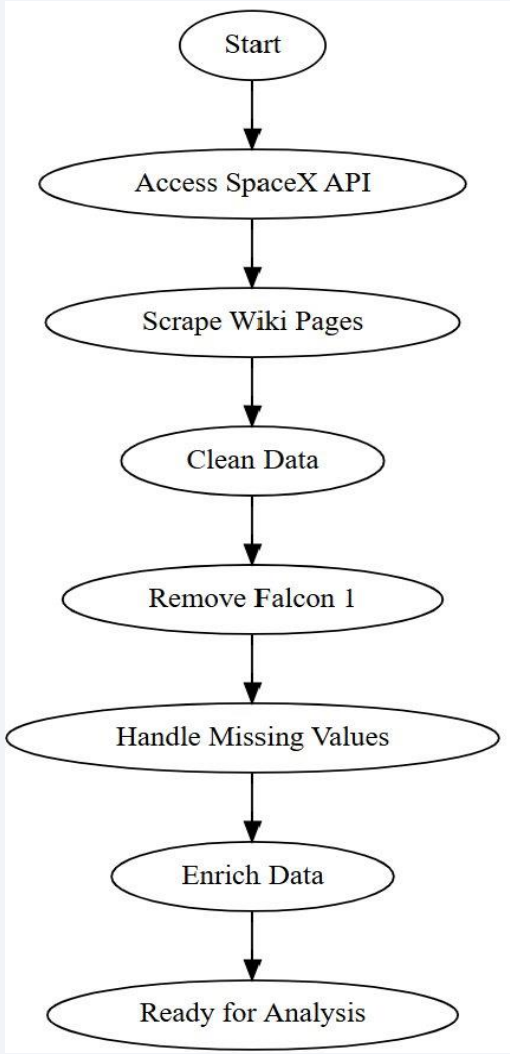
Data Collection

- Data was collected from the SpaceX API and Wiki pages, then cleaned by removing Falcon 1 launches and handling missing values, with additional details fetched through the API.
- **API Access**: Retrieved SpaceX launch data via the SpaceX API.
- **Web Scraping**: Scraped Falcon 9 launch data from Wiki pages.
- **Data Cleaning**: Removed Falcon 1 launches, handled missing values, and enriched data using the API.

Data Collection – SpaceX API

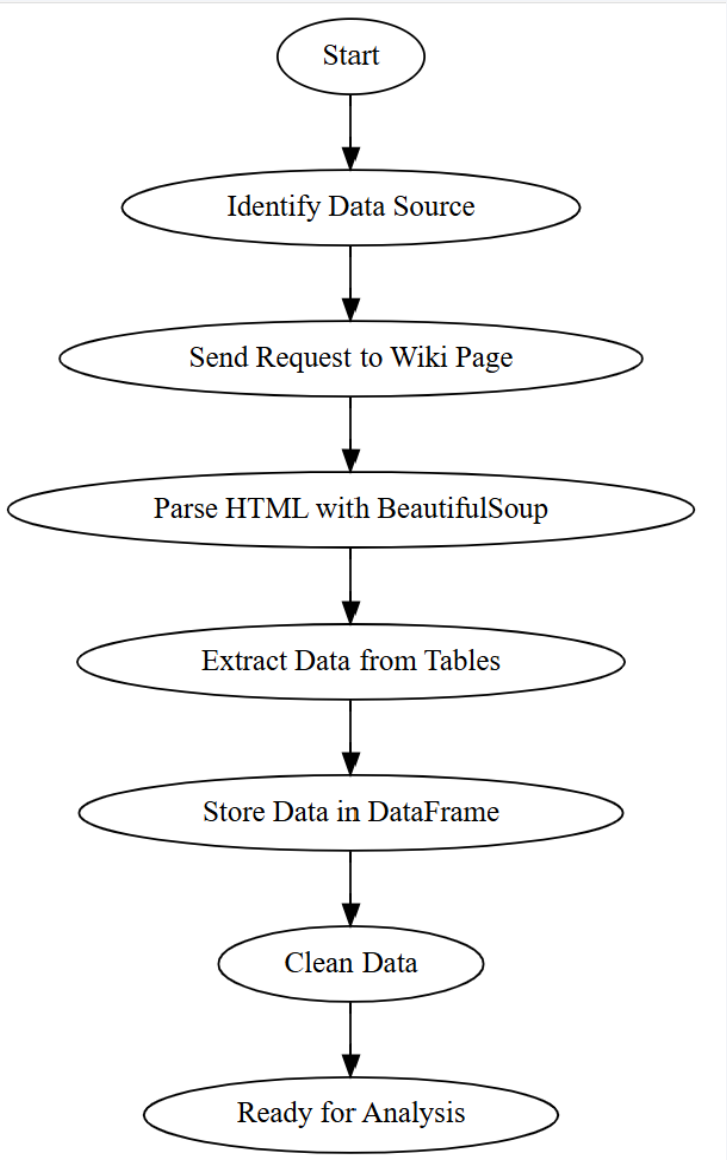
- Access API, scrape the web and clean data.

- https://github.com/MrChike/DatascienceProject



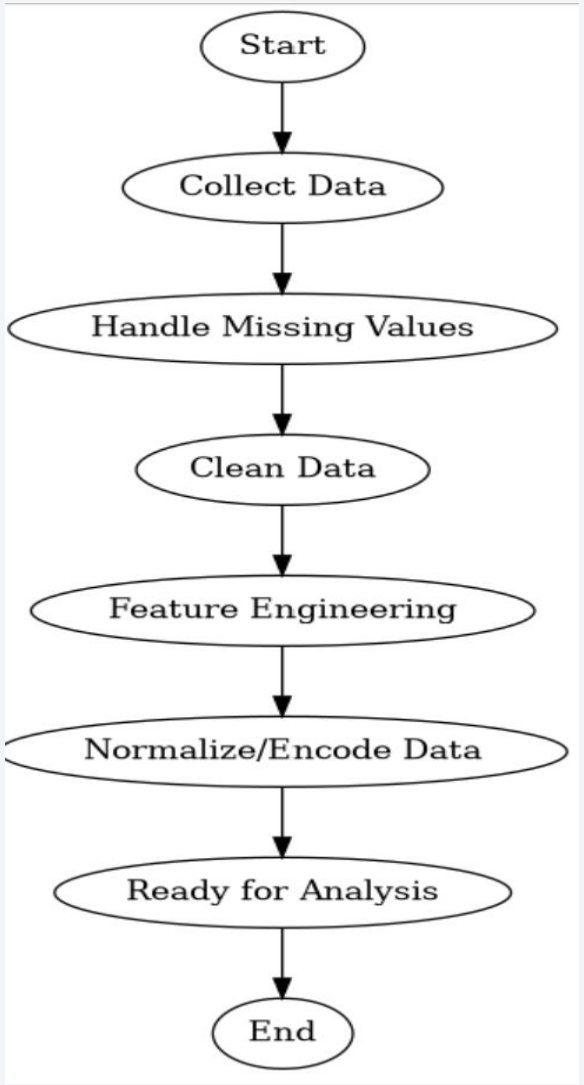
Data Collection - Scraping

- **Identify Data Source**: Locate the relevant Wiki pages containing Falcon 9 launch records.
- **Send Request**: Use Python's requests library to fetch the HTML content of the page.
- **Parse HTML**: Use BeautifulSoup to parse the HTML and extract tables with launch data.
- **Extract Data**: Extract relevant data such as launch date, payload, and outcome.
- **Store Data**: Store the extracted data into a structured format like a Pandas DataFrame.
- **Clean Data**: Handle missing or irrelevant data for further analysis.
- https://github.com/MrChike/DatascienceProject -> Data Collection



Data Wrangling

- The data was cleaned by filling missing values, correcting outliers, standardizing time formats, creating new features, and normalizing values to prepare it for analysis or machine learning.
- **Collect Data**: Gather sleep-related data.
- **Handle Missing Values**: Fill in missing data using imputation.
- **Clean Data**: Correct outliers and standardize time formats.
- **Feature Engineering**: Create new features (e.g., total sleep duration).
- **Normalization & Encoding**: Normalize numerical data and encode categories.
- **Final Dataset**: Prepare data for analysis or modeling.
- https://github.com/MrChike/DatascienceProject -> Data Wrangling



EDA with Data Visualization

- **Scatter Point Charts**: Used to visualize the distribution of numerical features (e.g., payload mass, flight number) and observe how individual data points are spread across axes.
- **Bar Charts**: Ideal for comparing categorical variables (e.g., launch sites) by visualizing their frequency or success rates, helping to easily compare different categories.
- **Scatter Plots**: Used to examine relationships between two continuous variables (e.g., payload mass vs. landing success), showing the correlation or trend between them.
- **Line Charts**: Useful for observing the impact of multiple factors (e.g., launch site and payload mass) on a continuous outcome (e.g., landing success), typically over time or across different conditions.
- https://github.com/MrChike/DatascienceProject -> EDA Visualization

EDA with SQL

- **Drop and Create Table**: This step removes the existing SPACEXTABLE and creates a new table with valid (non-null) dates, preparing the data for analysis.
- **Explore Launch Sites**: The queries retrieve and list distinct launch sites, specifically filtering for sites starting with "CCA" to focus on those related to the CCAFS launch site.
- **Payload Mass Analysis**: Queries are used to calculate the total and average payload mass for specific customers and booster versions, providing insights into payload trends for missions.
- **Landing Outcome**: The query identifies the first successful landing date for launches at sites with "SLC" in their name, helping track early landing successes.
- **Booster Version & Payload Mass**: These queries focus on successful landings that involved payloads between 4000-6000 kg at ASDS, providing insight into which booster versions were used in these specific cases.
- **Mission Success Count**: The query counts the successful and failed missions, giving a breakdown of mission outcomes to help evaluate overall performance.
- **Max Payload Mass**: Identifies which booster version was used for the heaviest payload, helping to determine the capacity of different boosters.
- **Monthly Landing Outcomes (2015)**: Focuses on analyzing failed landings on drone ships in 2015, broken down by month to observe any patterns or trends.
- **Landing Outcome Count by Date Range**: This query counts landing outcomes over a specific period (2010-2017) and orders them by frequency to assess the overall success rate during that timeframe.

- **https://github.com/MrChike/DatascienceProject -> EDA SQL**

Build an Interactive Map with Folium

- **Circle markers** was added to represent incidents on the map, with **popups** displaying details, and used **marker clusters** to group nearby incidents for a cleaner map view. I also organized everything into **feature groups** for better management and added **markers** for each incident to make it interactive.
- I added **circle markers** to visually represent each incident's location, making the map easy to understand. **Popups** provide more details when users click on a marker, adding interactivity. **Marker clusters** help group nearby incidents to prevent map clutter, improving navigation. **Feature groups** were used to organize and manage these elements efficiently. Together, these objects create a clear, interactive, and user-friendly map.
- https://github.com/MrChike/DatascienceProject -> Folium Maps

Build a Dashboard with Plotly Dash

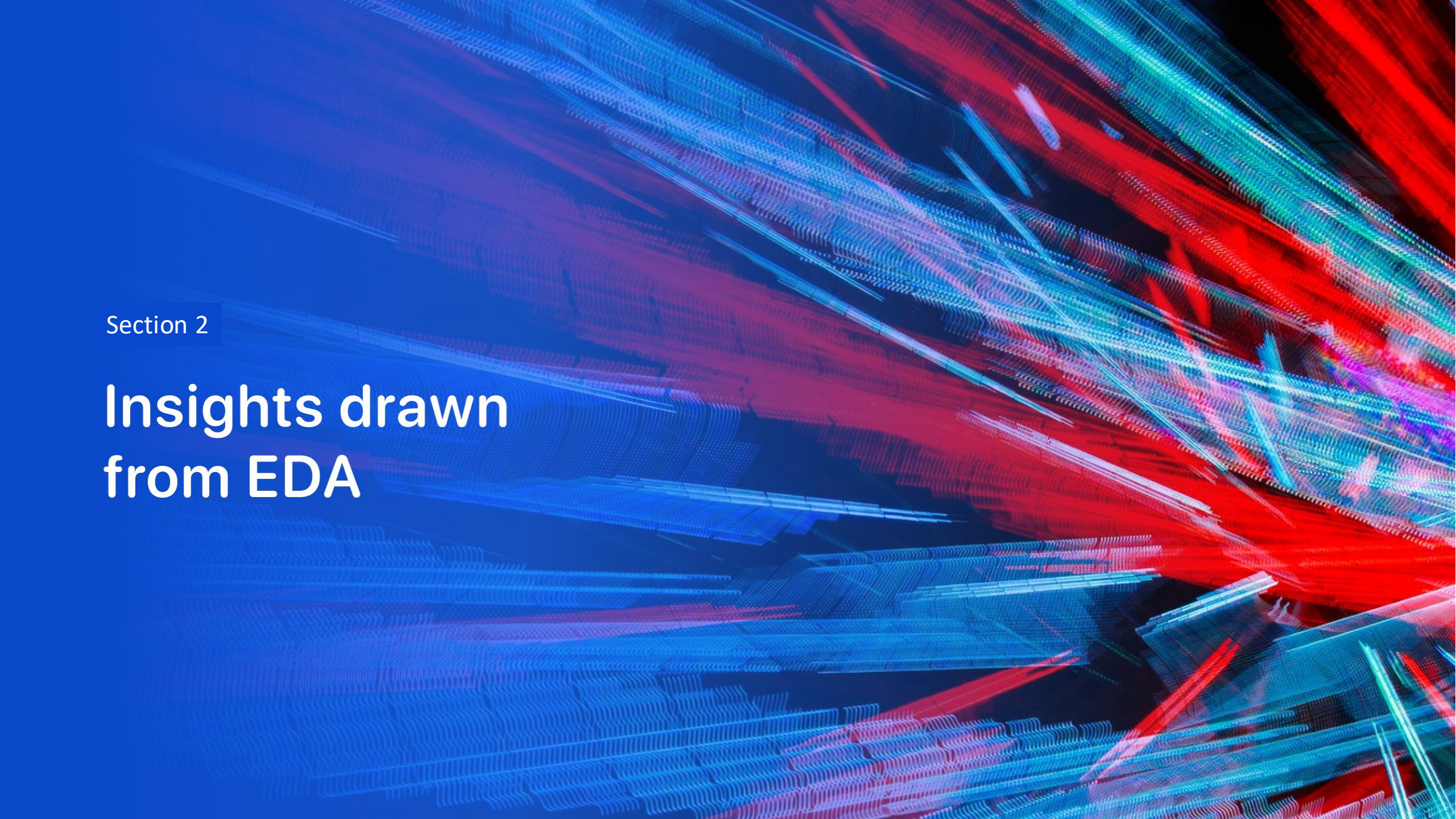
- **Launch Site Drop-down Menu (Interaction)**: A drop-down menu lets users select a launch site or view data for all sites, enabling site-specific analysis.
- **Success Pie Chart (Plot)**: The pie chart shows the success vs. failure count for the selected site or all sites, providing an overview of launch outcomes.
- **Payload Range Slider (Interaction)**: The range slider allows users to filter the data by payload weight, helping analyze its effect on launch success.
- **Success vs Payload Scatter Plot (Plot)**: The scatter plot shows the relationship between payload mass and launch success, with color-coded points for booster versions to assess their impact.
- The purpose of these plots and interactions is to let users explore SpaceX launch data by filtering by site and payload range. The drop-down menu and pie chart provide site-specific success rates, while the payload slider helps analyze the effect of payload weight on success. The scatter plot shows the relationship between payload and success, with boosters color-coded, allowing users to assess their impact on performance. Together, these components enable dynamic analysis of launch data.
- https://github.com/MrChike/DatascienceProject -> Dash Plotly

Predictive Analysis (Classification)

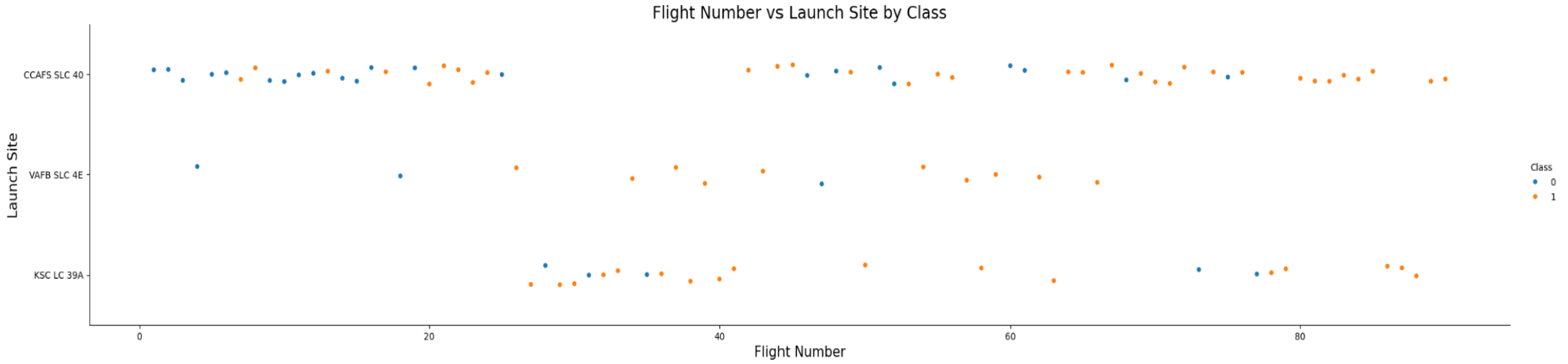
- I built and evaluated several classification models including Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K-Nearest Neighbors (KNN) by first preprocessing the data, standardizing features, and splitting the dataset into training and testing sets. Each model was tuned using **GridSearchCV** to optimize hyperparameters through 10-fold cross-validation. I evaluated model performance using accuracy scores and confusion matrices on the test data. After comparing the test accuracies, I identified the best-performing model based on the highest accuracy.
- Data Loading -> Data Preprocessing (Handle Missing Data & Standardize) -> Model Selection (Logistic Regression, SVM, Decision Tree, KNN) -> Hyperparameter Tuning (GridSearchCV, cv=10) -> Model Evaluation (Accuracy, Confusion Matrix, Cross-Validation) -> Best Model Selection
- https://github.com/MrChike/DatascienceProject -> Predictive Analysis

Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

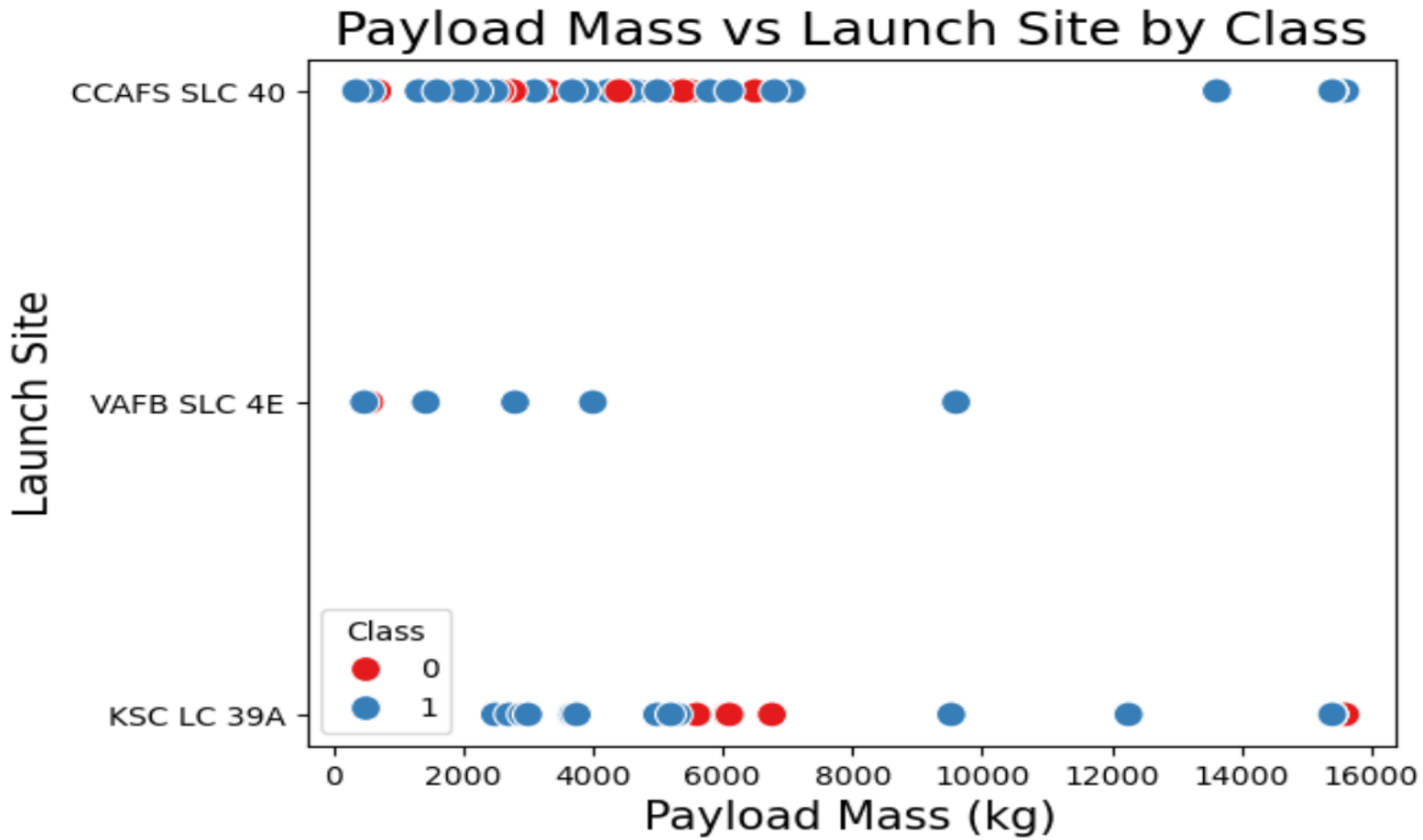

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



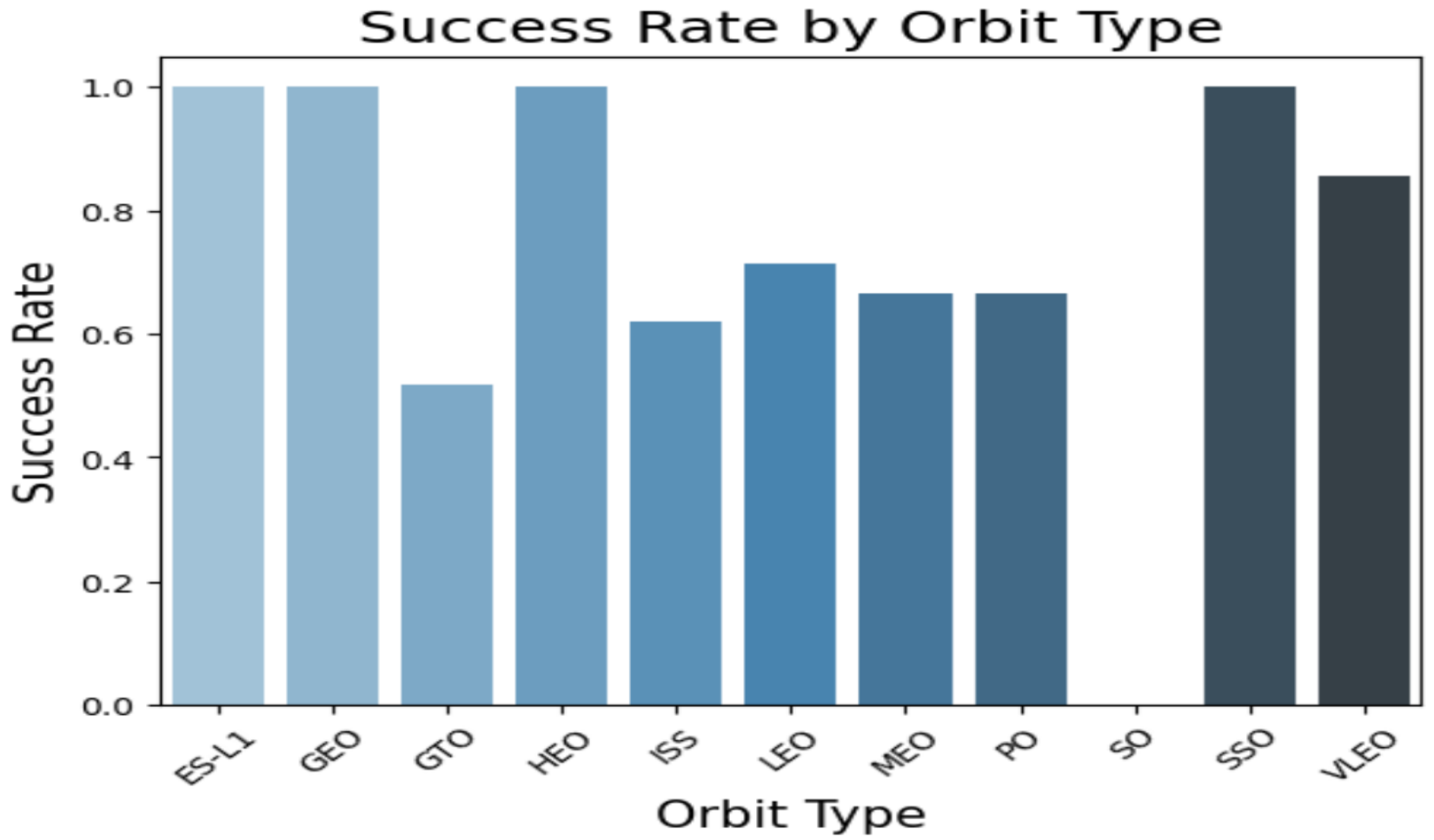
The scatter plot shows how SpaceX uses different launch sites across flight numbers, with each point colored by the mission type (Class). Some launch sites, like CCAFS SLC 40, are used more frequently, while others, such as VAFB SLC 4E, are less common. Certain mission types, like ISS missions, are tied to specific sites. As flight numbers increase, we can observe more launches happening from the same sites, reflecting SpaceX's growth and expansion over time.

Payload vs. Launch Site



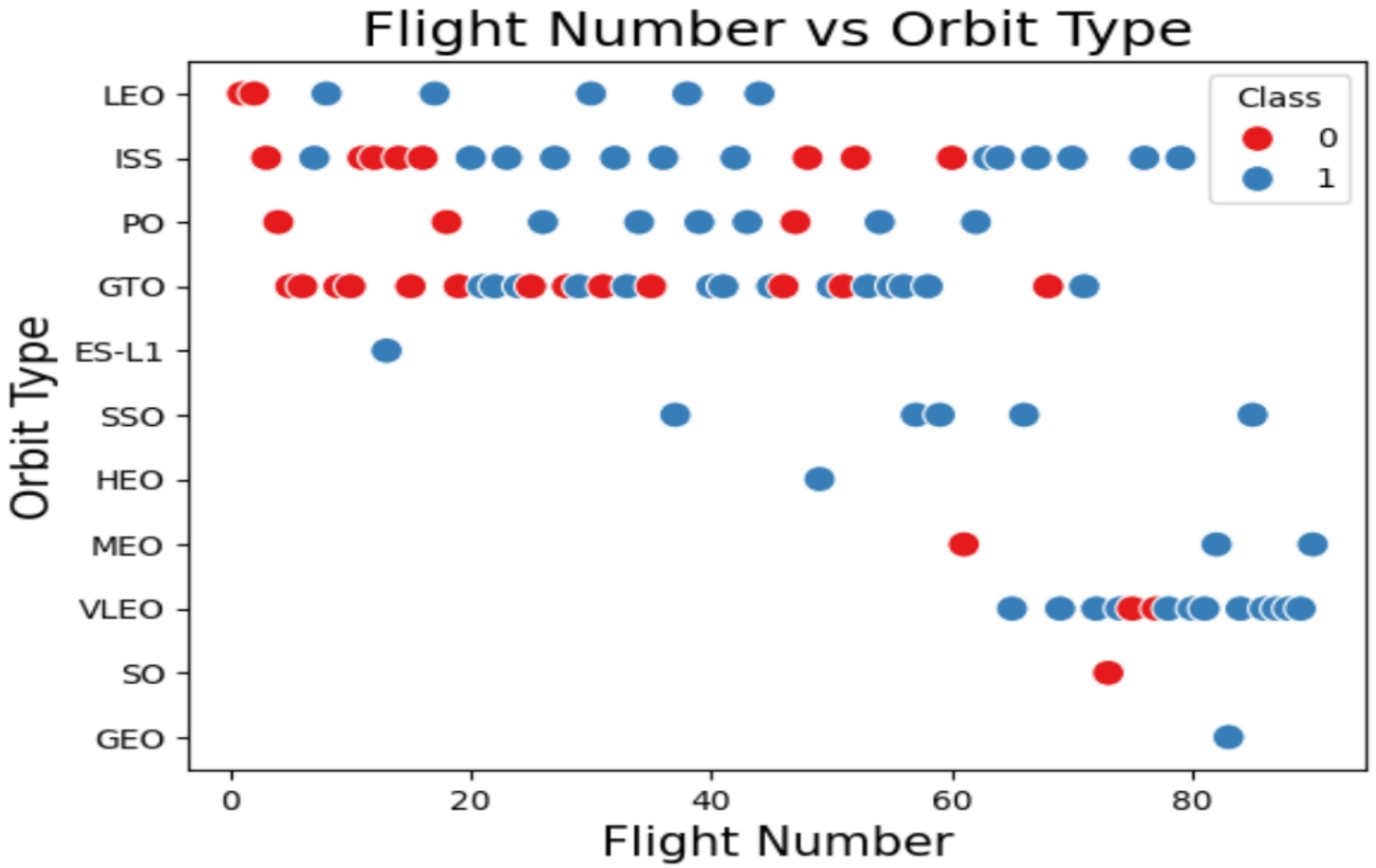
Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type



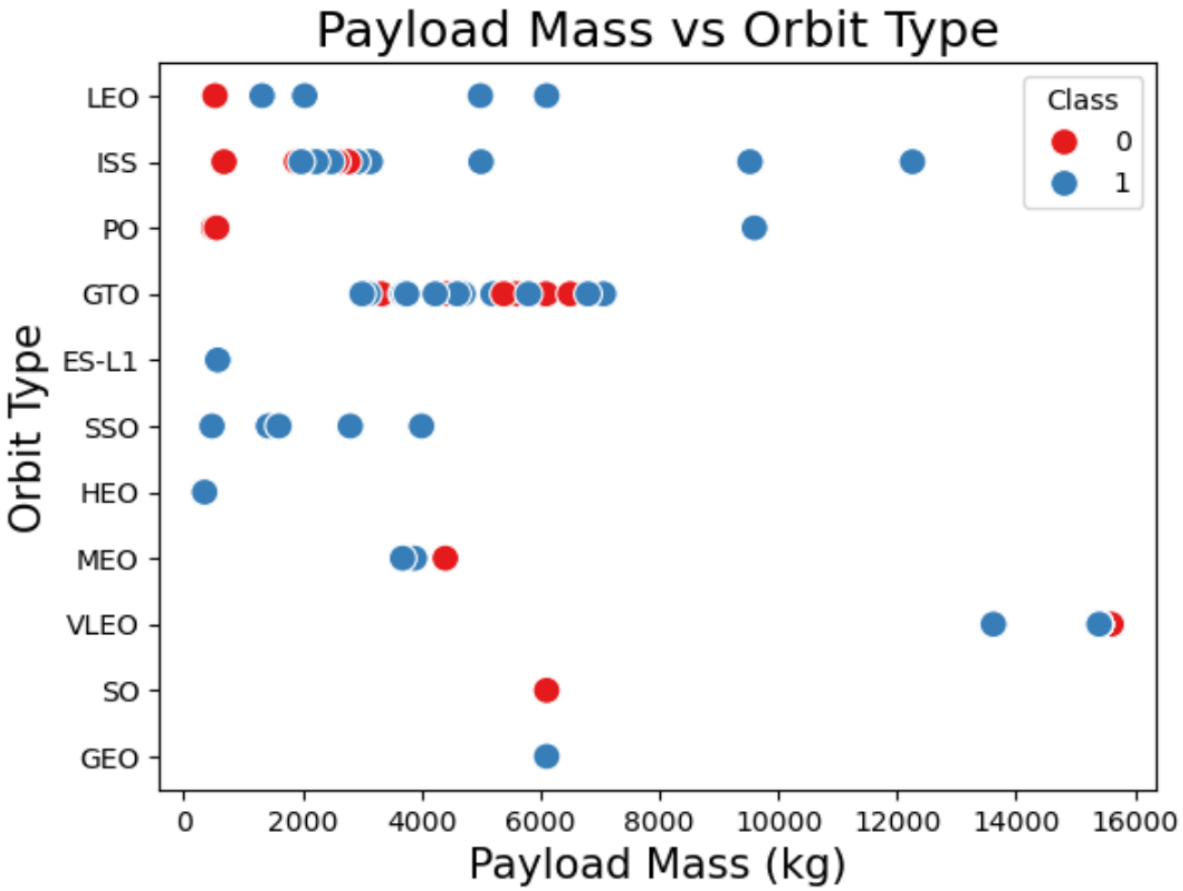
Analyze the plotted bar chart to identify which orbits have the highest success rates.

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.

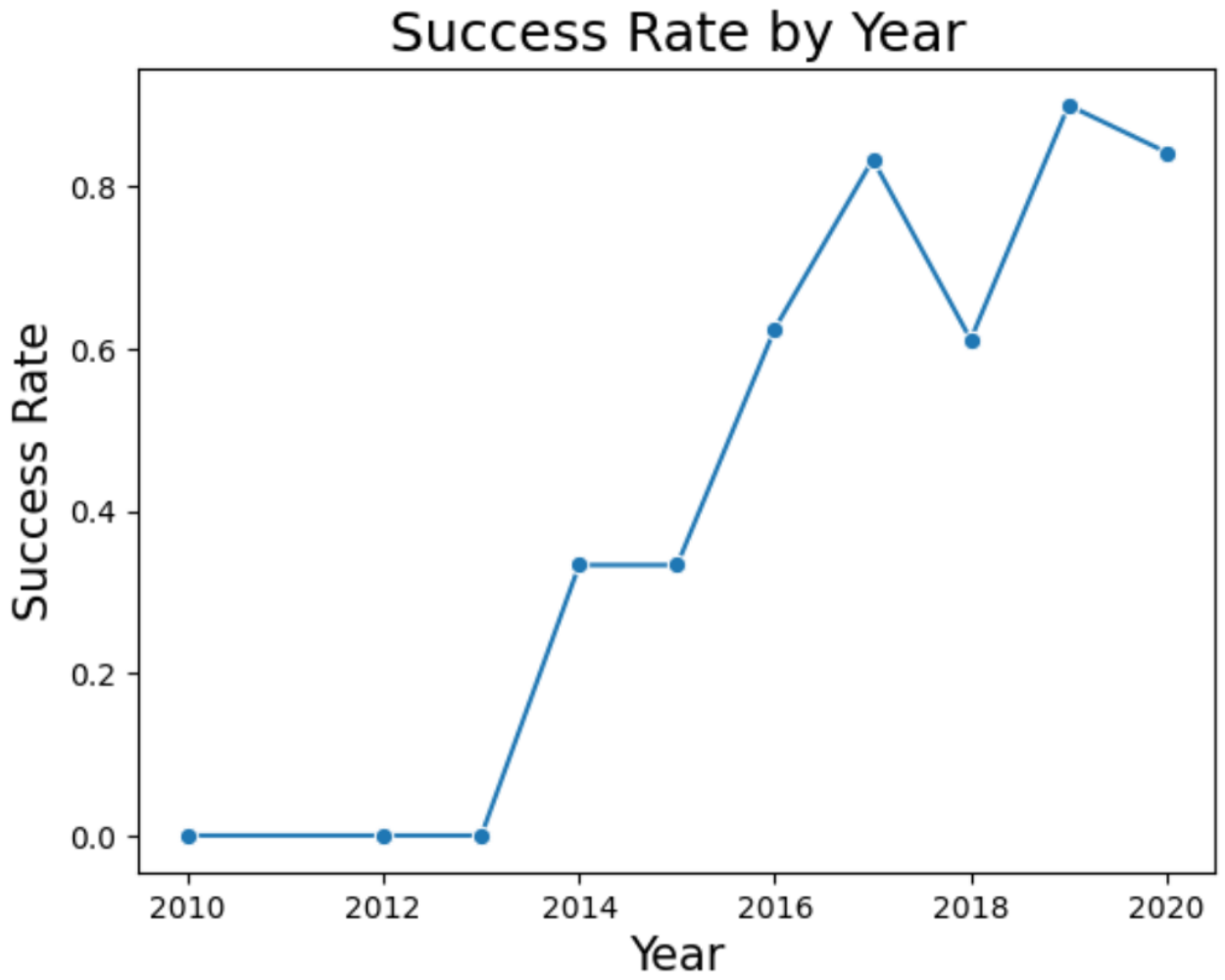
Payload vs. Orbit Type



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 SPACEXTBL;
```

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

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

- Unique launch sites are retrieved by the distinct query

Launch Site Names Begin with 'CCA'

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

```
%sql SELECT * FROM "SPACEXTBL"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

- Launch site names are filtered with the LIKE & LIMIT query

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 "SPACEXTBL" WHERE "Customer" = 'NASA (CRS)';
```

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

total_payload_mass
45596

- This query is adding up the total payload mass (in kilograms) for all of NASA's CRS missions in the "SPACEXTBL" table. It's filtering the data to only include entries where the customer is 'NASA (CRS)' and then sums up the values in the "PAYLOAD_MASS__KG_" column, naming the result as `total_payload_mass`.

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 "SPACEXTBL" WHERE "Booster_Version" = 'F9 v1.1';
```

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

average_payload_mass
2928.4

- This query calculates the average payload mass (in kilograms) for all SpaceX launches using the "F9 v1.1" booster version. It pulls the data from the "SPACEXTBL" table, filters by the booster version 'F9 v1.1', and then finds the average of the "PAYLOAD_MASS__KG_" values, labeling the result as `average_payload_mass`.

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_date FROM "SPACEXTBL" WHERE "Landing_Outcome" = 'Success' AND "Launch_Si
```

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

first_successful_landing_date
2018-07-22

- This query is looking for the earliest date of a successful landing at a launch site that includes 'SLC' in its name. It filters the "SPACEXTBL" table for records where the "Landing_Outcome" is 'Success' and the "Launch_Site" contains 'SLC'. The result is the minimum (earliest) "Date", which is labeled as `first_successful_landing_date`.

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 DISTINCT "Booster_Version" FROM "SPACEXTBL" WHERE "Landing_Outcome" = 'Success' AND "Landing_Site" LIKE '%ASDS%'
```

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

Booster_Version

- This query retrieves a list of unique booster versions used in successful landings on the Autonomous Spaceport Drone Ship (ASDS), where the payload mass is between 4000 and 6000 kilograms. It filters the "SPACEXTBL" table based on these conditions and returns only the distinct values from the "Booster_Version" column that meet all criteria.

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%sql SELECT COUNT(CASE WHEN "Mission_Outcome" = 'Success' THEN 1 END) AS successful_missions,COUNT(CASE WHEN "Mission_Outcom
```

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

successful_missions	failed_missions
98	0

- This query counts the number of successful and failed missions in the "SPACEXTBL" table. It uses CASE statements to check the "Mission_Outcome" column—counting rows where the outcome is 'Success' and 'Failure'. The results are labeled as `successful_missions` and `failed_missions`.

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 "SPACEXTBL" WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM "SPACEXTBL"
```

```
 * 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

- This query finds the booster version used for the mission with the highest payload mass. It first determines the maximum payload mass by using a subquery, then retrieves the "Booster_Version" for that specific payload mass from the "SPACEXTBL" table.

2015 Launch Records

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%%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_Name",
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM "SPACEXTBL"
WHERE substr("Date", 0, 5) = '2015'
  AND "Landing_Outcome" = 'Failure (drone ship)';
```

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

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

- This query extracts the month name, landing outcome, booster version, and launch site for missions that had a landing failure on the drone ship in 2015. It converts the numeric month (from the "Date" column) into its full name using a CASE statement. The query filters the results to only include records from 2015 where the landing outcome was 'Failure (drone ship)'.

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 "Outcome_Count"
FROM "SPACEXTBL"
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY "Outcome_Count" DESC;
```

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

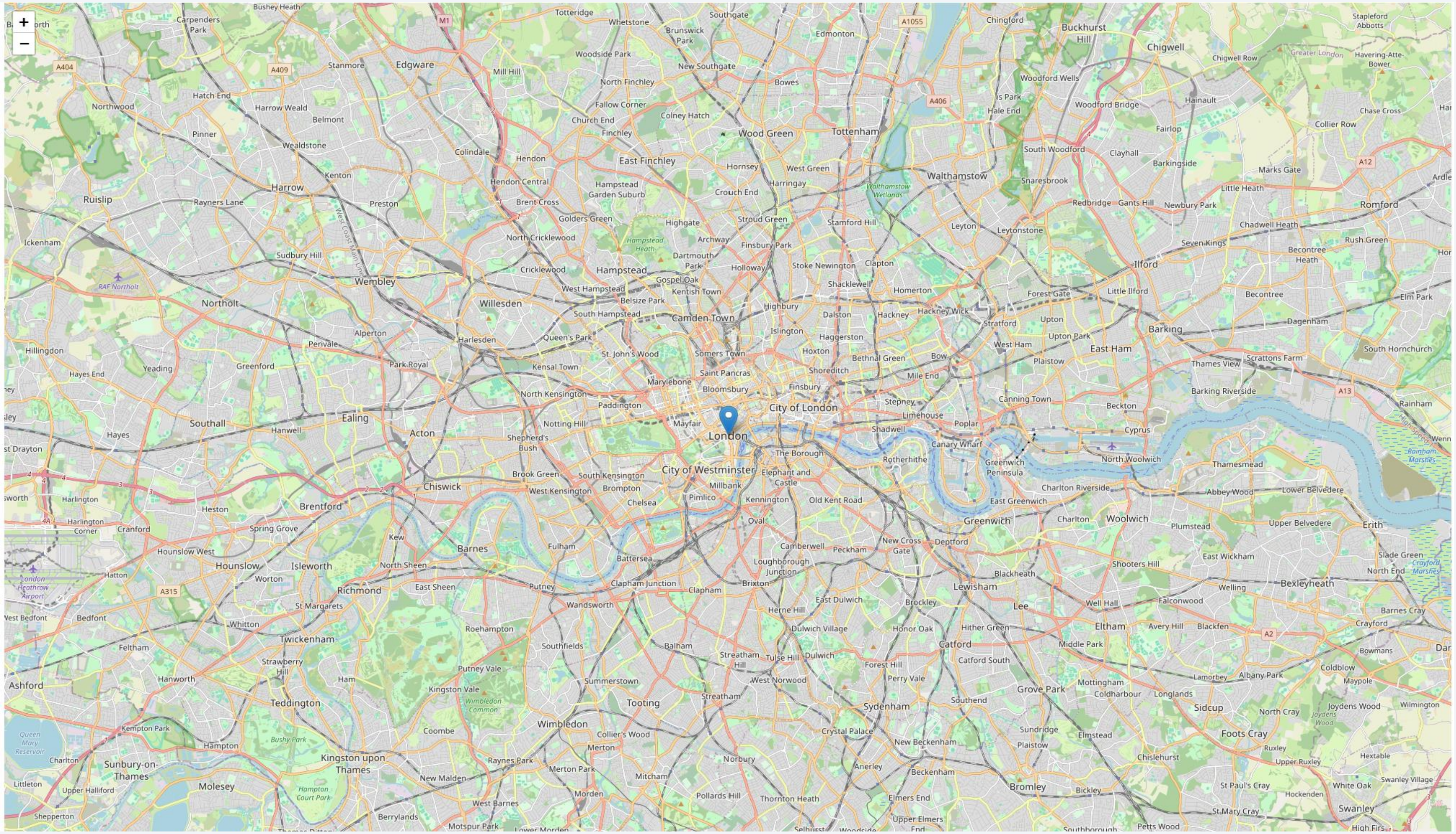
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

- This query counts the number of occurrences of each landing outcome between June 4, 2010, and March 20, 2017, in the "SPACEXTBL" table. It groups the results by "Landing_Outcome" and orders them by the count of each outcome in descending order, so the most frequent outcomes appear first.

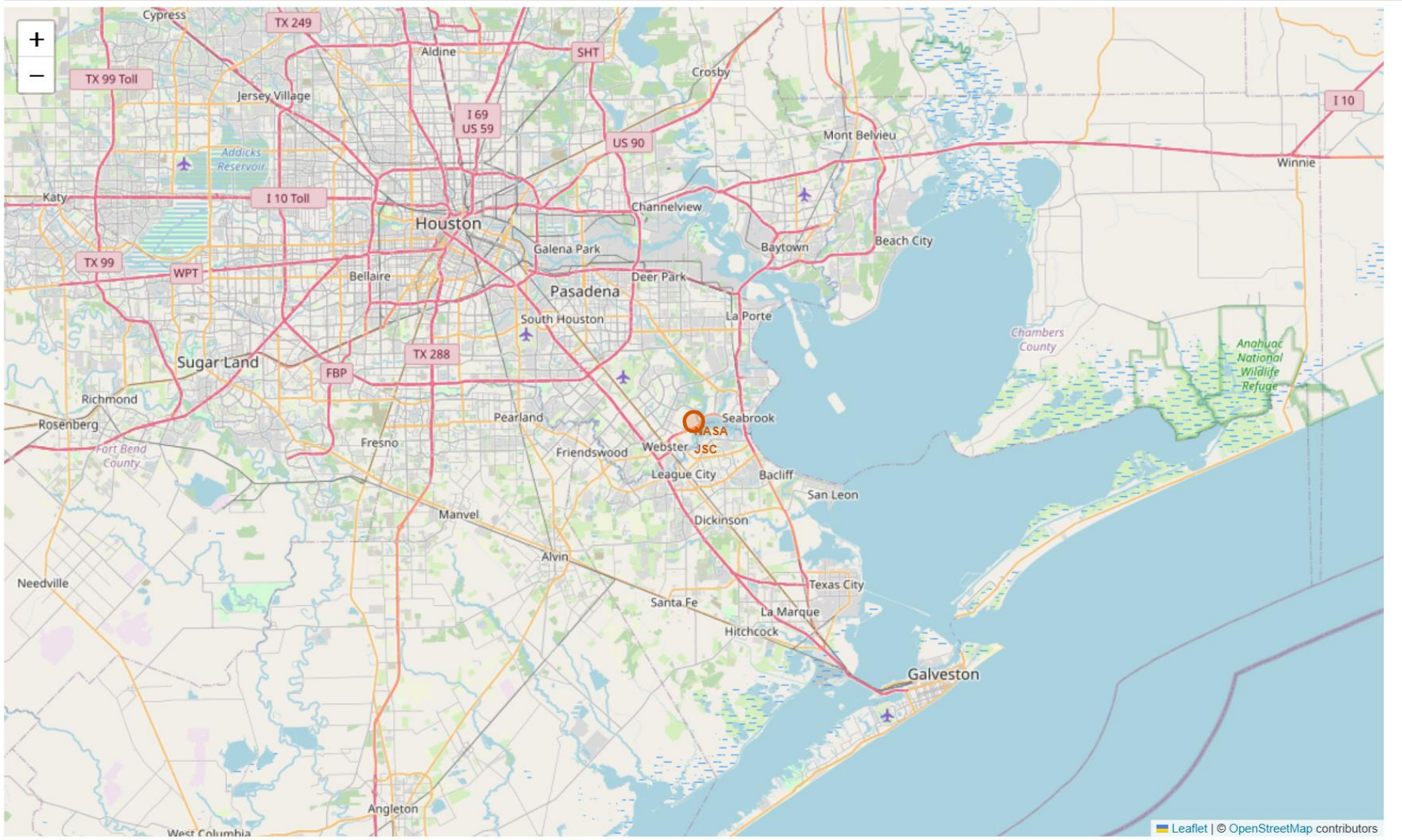
Section 3

Launch Sites Proximities Analysis

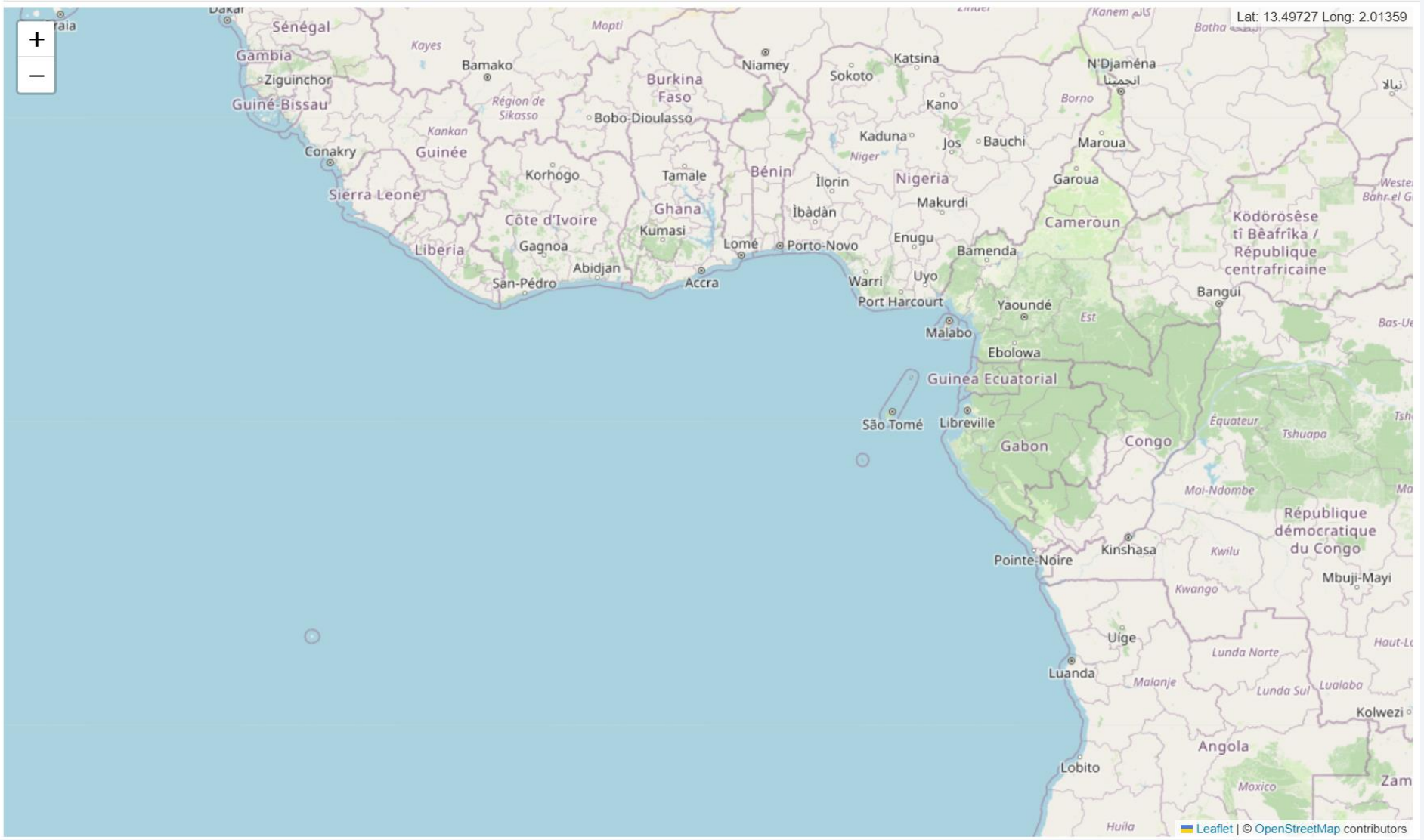
Global Launch Sites: Locations of Key Spaceports Around the World

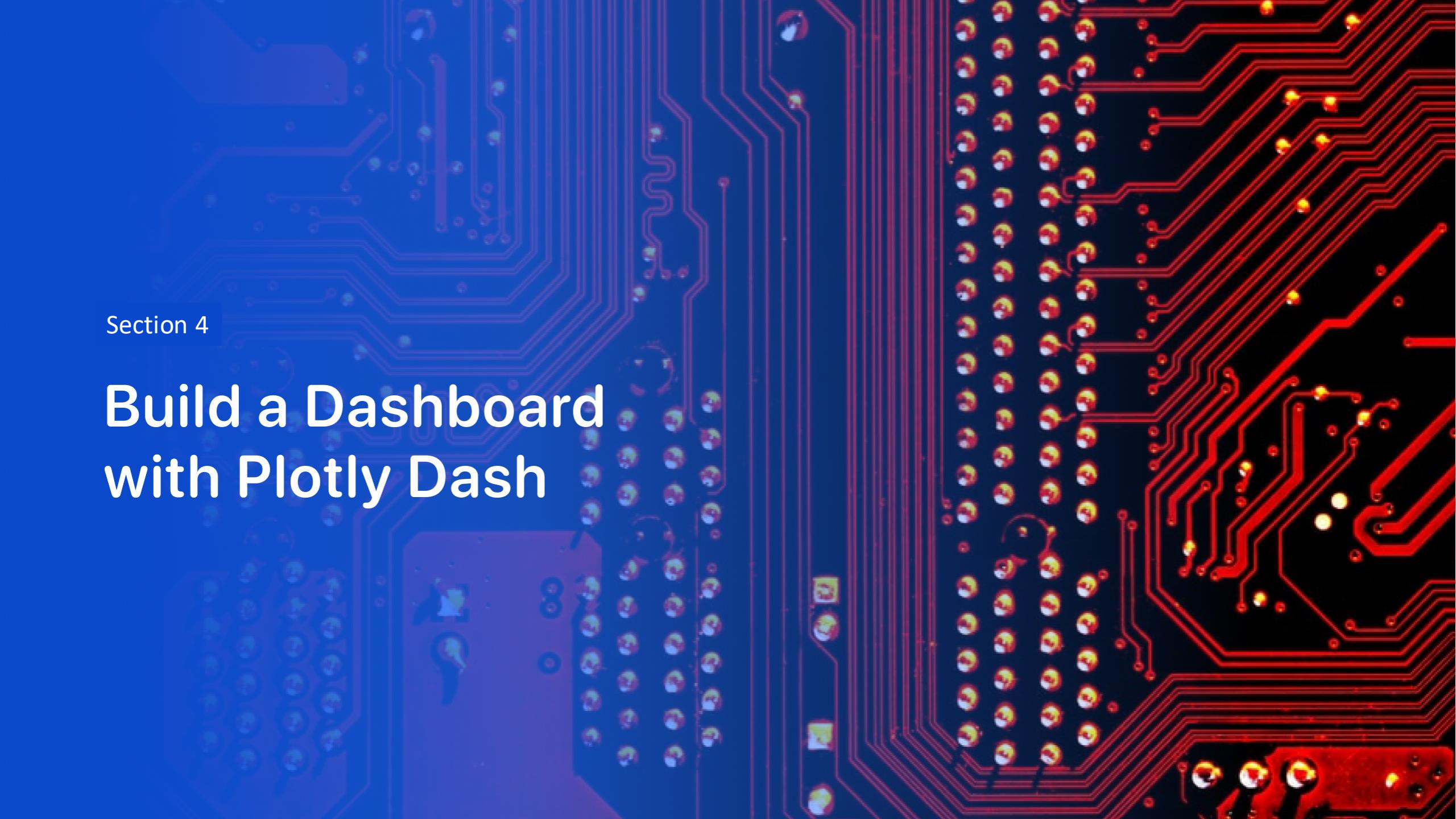


Visualizing Launch Sites on a Map Using Folium: Mapping Geographical Coordinates for Intuitive Insights

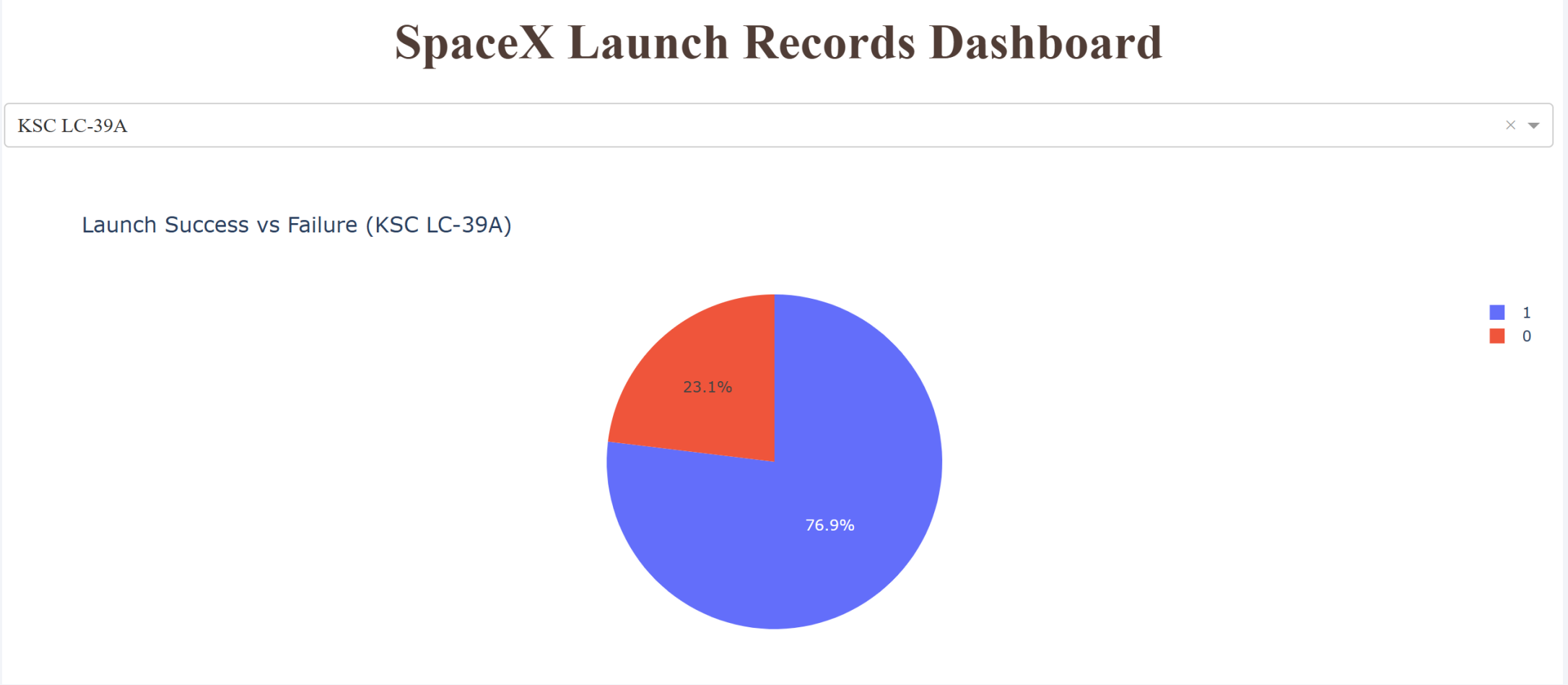


Launch Site Proximity Analysis: Visualizing Surrounding Infrastructure and Distance Calculations



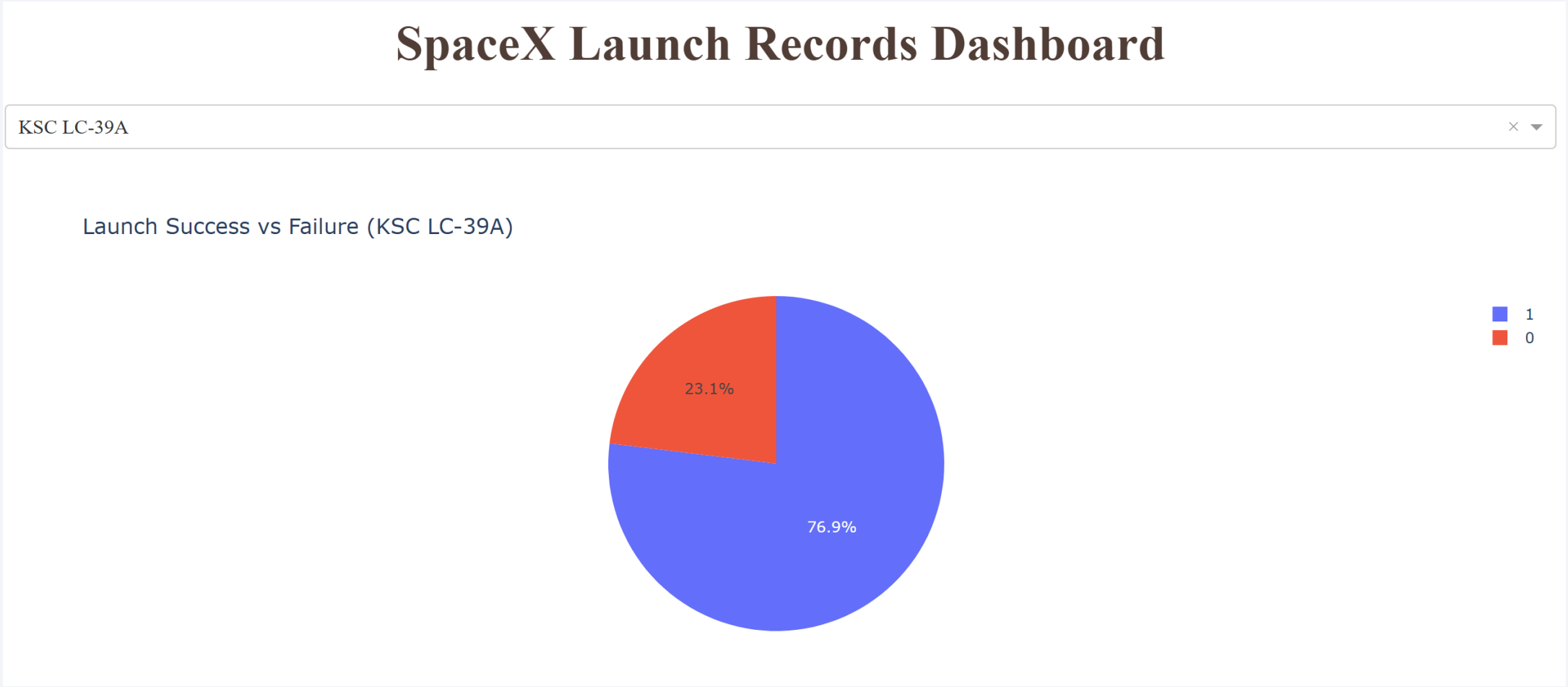

Section 4
Build a Dashboard with Plotly Dash

Launch Success vs Failure (All Sites)



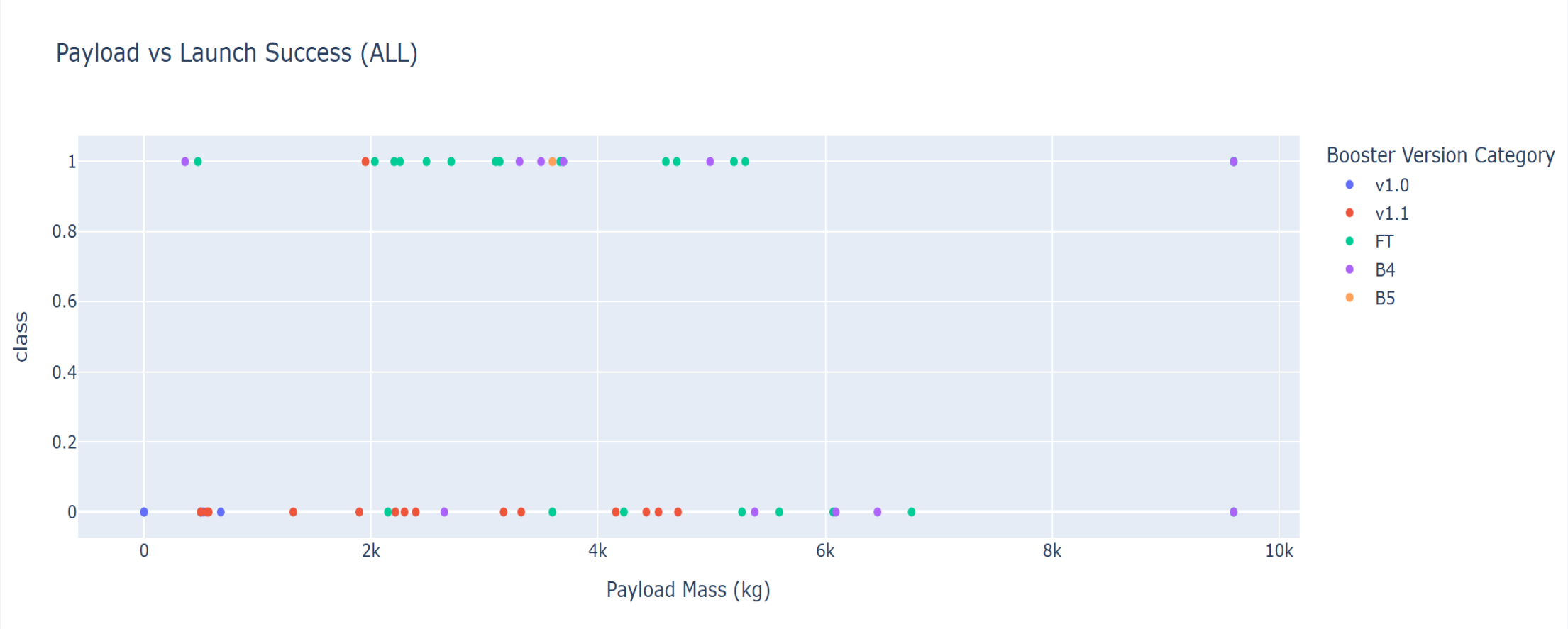
- The **"Launch Success vs Failure"** pie chart shows SpaceX's success rate. A larger success portion indicates high reliability, while a smaller failure portion shows minimal launch issues. It gives a quick view of SpaceX's overall performance.

Launch Success vs Failure (KSC LC-39A)



- The **"Launch Success vs Failure (KSC LC-39A)"** pie chart shows the success and failure rates for launches from the KSC LC-39A site. A larger success portion indicates strong performance at this site, while a smaller failure portion suggests few issues with launches here.

Payload vs Launch Success (ALL)

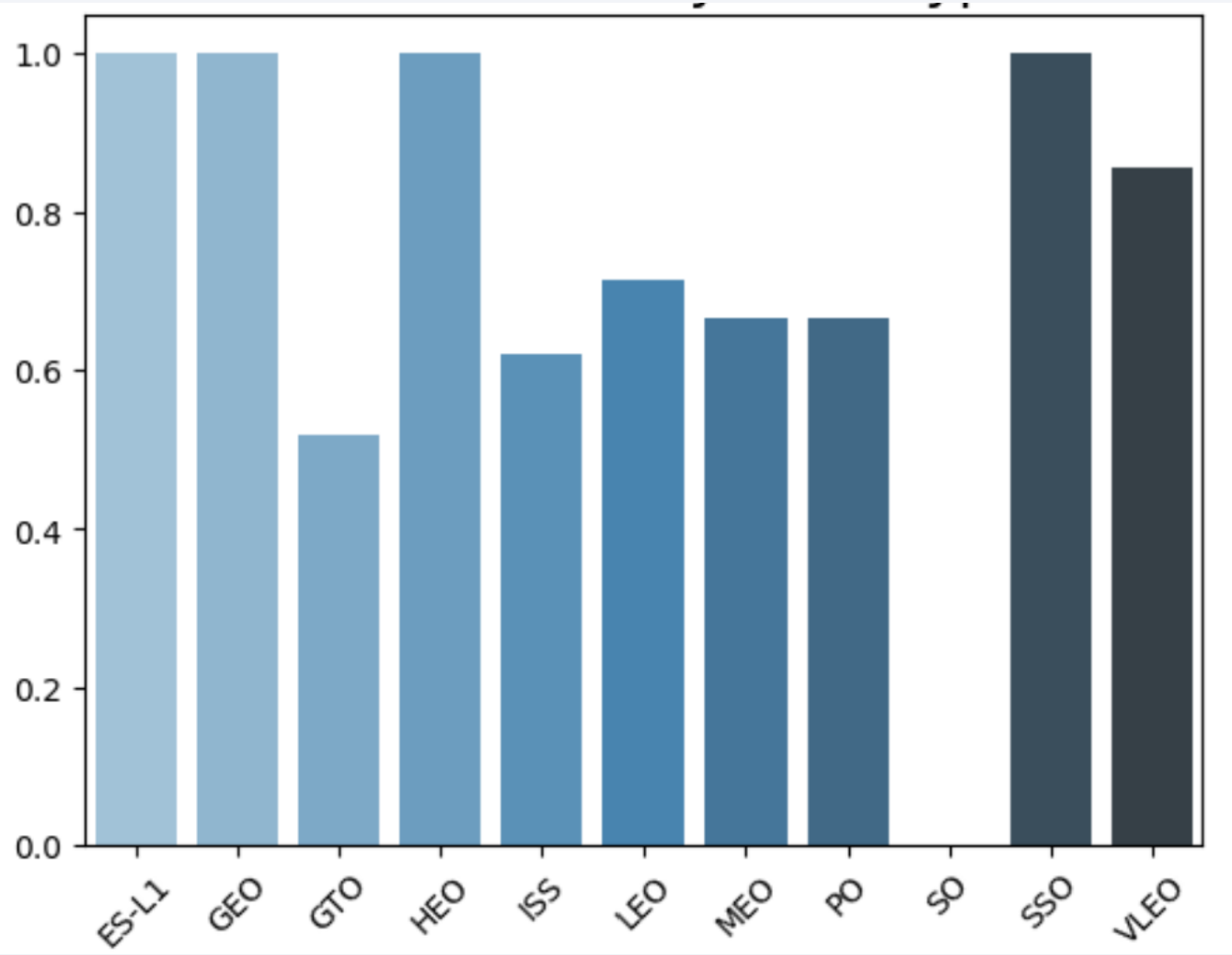


- The **"Payload vs Launch Success (ALL)"** scatter plot shows the relationship between payload mass and launch success. It helps to see if larger payloads affect the success rate, with payload mass on the x-axis and launch outcome on the y-axis.

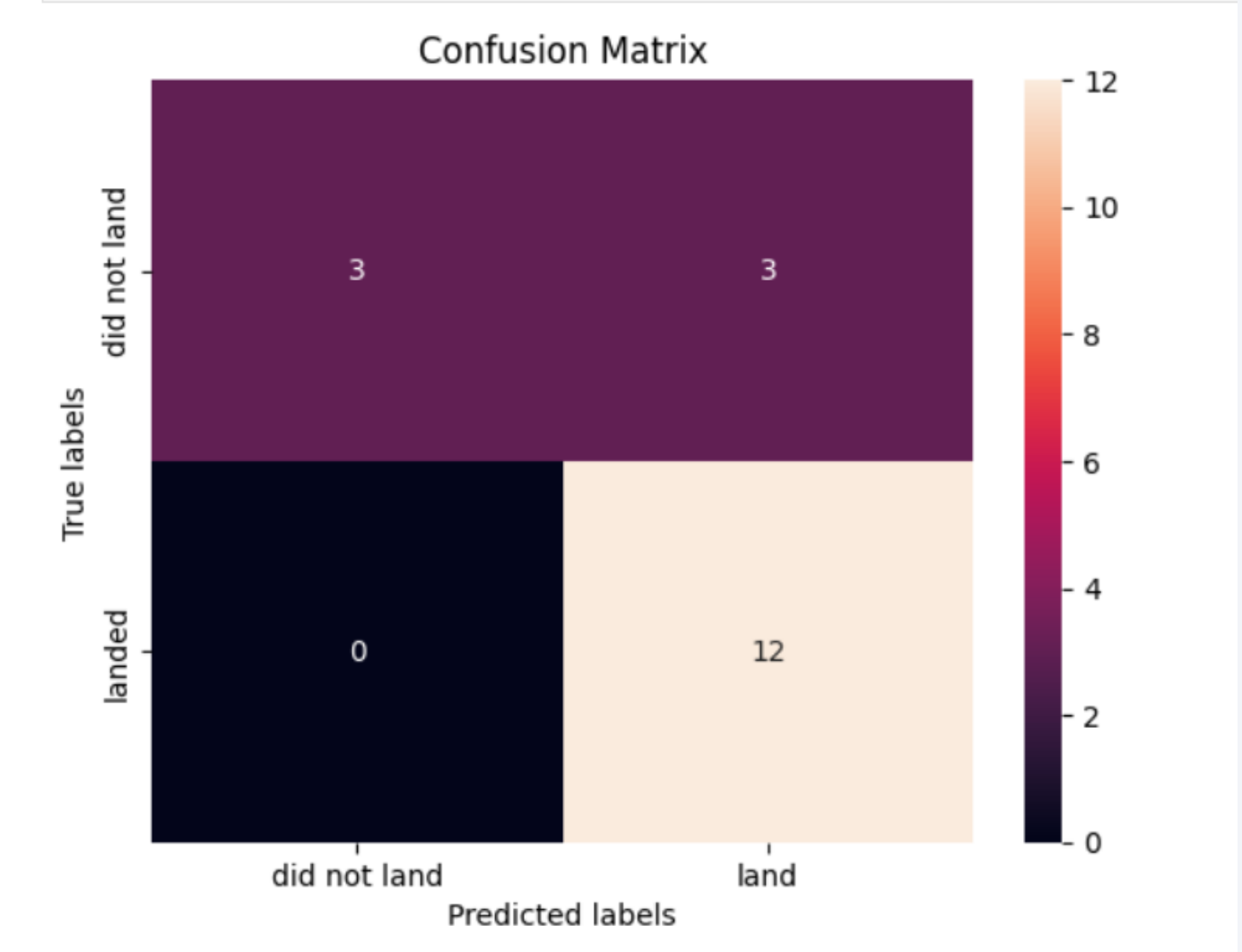
Section 5

Predictive Analysis (Classification)

Classification Accuracy



Confusion Matrix



Conclusions

- **Model Performance**: The models evaluated in this project (Logistic Regression, SVM, Decision Tree, KNN) were compared based on their test accuracies, with Logistic Regression and SVM performing the best, both achieving an accuracy of 83.33%.
- **Best Model Selection**: After comparing the test accuracies, Logistic Regression was identified as the best-performing model for this dataset, providing a strong balance of interpretability and performance.
- **Model Evaluation**: In addition to accuracy, the models were assessed using cross-validation, confusion matrices, and other performance metrics (e.g., precision, recall, F1 score), ensuring a comprehensive evaluation of their strengths and weaknesses.
- **Key Insights**: Feature importance analysis revealed which variables had the most significant impact on the model's predictions, offering valuable insights into the underlying patterns of the data.
- **Future Improvements**: Possible improvements include tuning hyperparameters further, testing additional models, or incorporating other techniques like ensemble methods to boost performance.

Appendix

Features Engineering

By now, you should obtain some preliminary insights about how each important variable would affect the success rate, we will select the features that will be used in success prediction in the future module.

```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad
features.head()
```

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	6104.959412	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0003
1	2	525.000000	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0005
2	3	677.000000	ISS	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0007
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False	NaN	1.0	0	B1003
4	5	3170.000000	GTO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B1004

Thank you!