Winning Space Race with Data Science

INSHA
17TH JANUARY 2025

Outline

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

Executive Summary

Summary of Methodologies

1. Data Collection:

   - Utilized the SpaceX REST API and web scraping to retrieve launch data.
   - Normalized JSON data and cleaned datasets by handling missing values and encoding categorical variables.

2. Data Wrangling:

   - Applied transformations, such as imputing missing values, handling outliers, and feature engineering.
   - Filtered data to focus on Falcon 9 launches and created binary labels for landing outcomes.

3. Exploratory Data Analysis (EDA):

   - Conducted SQL-based queries to extract insights, such as payload mass and launch site success rates.
   - Visualized relationships using scatter plots, bar charts, and line graphs to understand patterns.

4. Predictive Analysis:

   - Trained and evaluated multiple classification models, including SVM, logistic regression, decision trees, and KNN.
   - Performed hyperparameter tuning with GridSearchCV to optimize model performance.

Summary of Results

1. EDA Results:

- CCAFS SLC-40 had the most launches and the highest success rate (~70%).
- Total payload carried exceeded 549,446 kg, with a significant portion attributed to NASA missions.
- First successful ground landing occurred early in the Falcon 9 program, showcasing SpaceX's rapid technological progress.

2. Model Evaluation:

- Logistic regression achieved the highest accuracy, with a confusion matrix indicating minimal false positives.
- Other models performed well but were slightly less accurate, showing potential for future enhancements.

This comprehensive analysis highlights the importance of predictive analytics in reducing costs and improving operational efficiency for space launches.

Introduction: Capstone Overview

- **Objective**: Predict the successful landing of the Falcon 9 first stage.
- **Cost Comparison**:
  - SpaceX Falcon 9 launches cost **62 million dollars**.
  - Competing providers charge **165 million dollars** or more per launch.
- **Key Savings**:
  - SpaceX’s ability to **reuse the first stage** significantly reduces costs.
- **Impact of Prediction**:
  - By predicting whether the first stage will land successfully, we can estimate the **launch cost**.
- **Competitive Advantage**:
  - This information can help other companies **bid against SpaceX** for rocket launches.
- **Module Focus**:
  - You will be provided with an **overview of the problem** and the **tools** necessary to complete the course.
- Project background and context
- Problems you want to find answers

Section 1

Methodology

Methodology

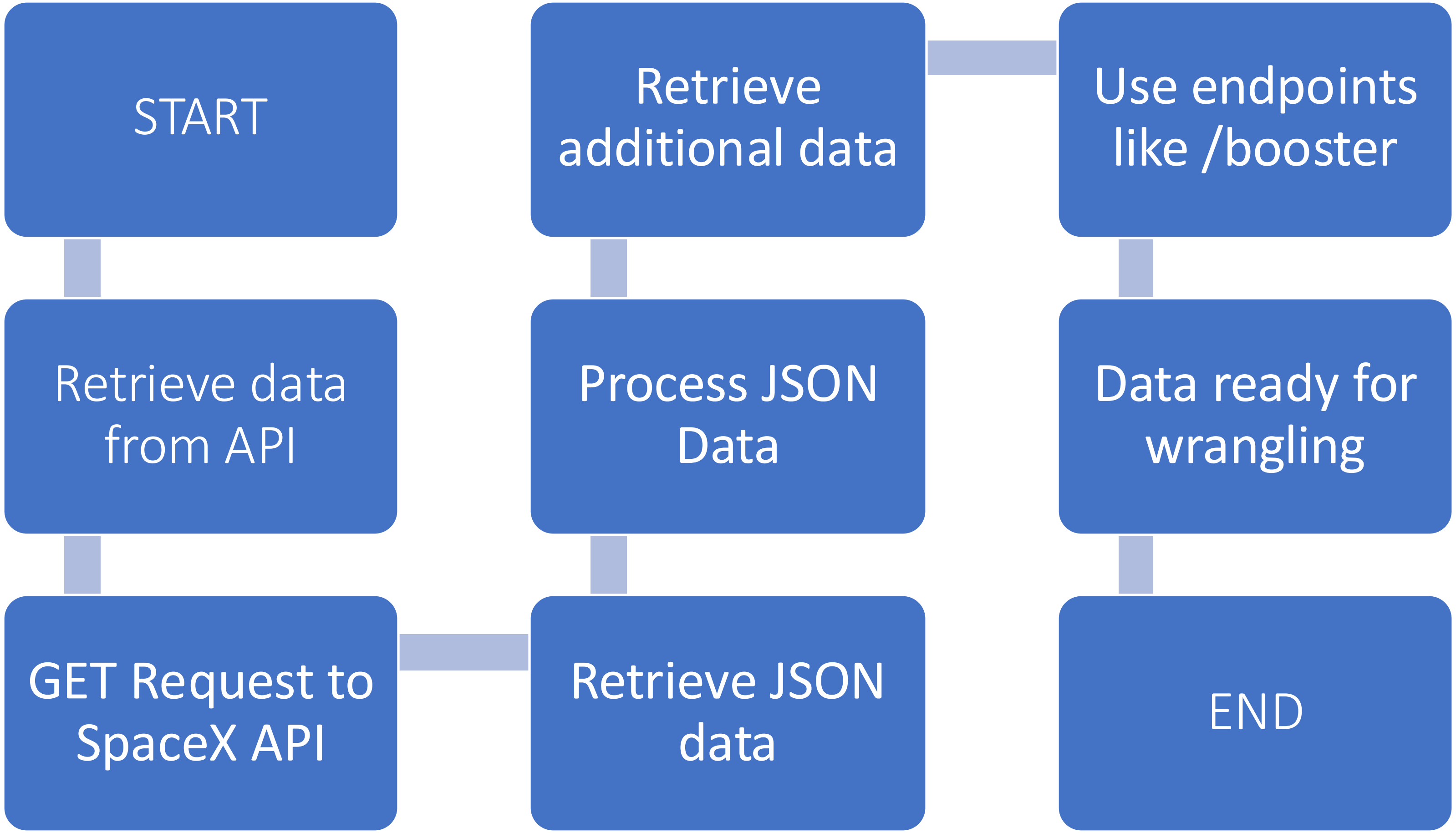
Executive Summary

- Data collection methodology:
  - The JSON data was normalized into a flat table using `json_normalize`, with Falcon 1 data filtered out to focus on Falcon 9. NULL values in columns like `PayloadMass` were replaced with the column mean. The `Outcome` column was binary encoded (1 for success, 0 for failure), and columns such as `LaunchSite` and `Orbit` were processed for further analysis.
- Perform data wrangling
  - The data was cleaned by normalizing the JSON data using `json_normalize`, removing Falcon 1 data, and replacing NULL values in `PayloadMass` with the column mean. For transformation, the `Outcome` column was binary encoded (1 for success, 0 for failure), and columns like `LaunchSite` and `Orbit` were processed for 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

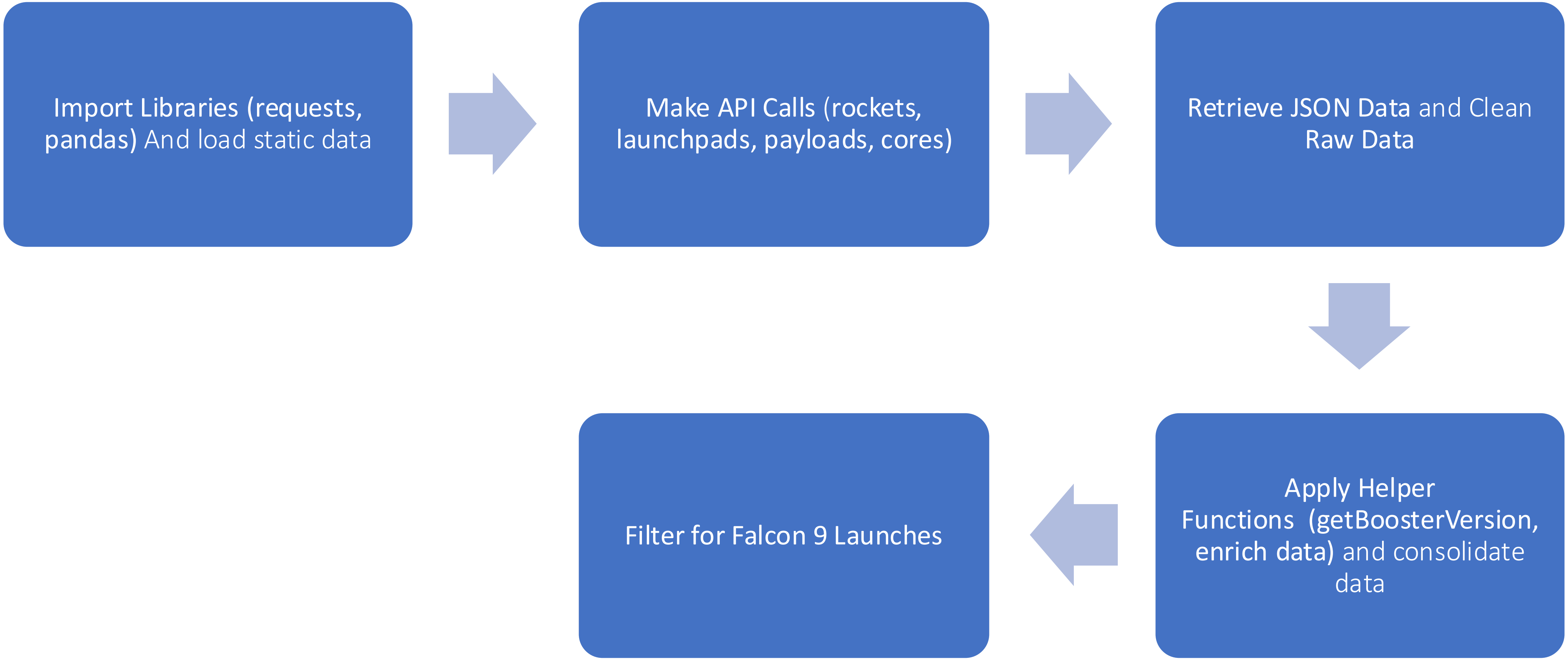
- **Data Collection Process**
- **Sources**:
  - **SpaceX REST API**: Used to retrieve detailed launch data.
  - **Web Scraping**: Web scraping was performed on Falcon 9 launch records from Wiki pages using **BeautifulSoup**.
- **Data Retrieval**:
  - **GET Request to API**: A GET request was made to the SpaceX REST API, specifically to the endpoint `api.spacexdata.com/v4/launches/past`.
  - **Additional Data**: For missing information, additional API endpoints (e.g., `/booster`, `/launchpad`) were used.
- **JSON Response**:
  - The API returned data in **JSON format**, containing a list of JSON objects representing individual launches.

FLOWCHART


START
Retrieve data from API
GET Request to SpaceX API
Retrieve JSON data
Process JSON Data
Retrieve additional data
Use endpoints like /booster
Data ready for wrangling
END

Data Collection – SpaceX API

- **Libraries Import**: Import necessary libraries like `requests` and `pandas`.
- **Load Static Data**: Load any static data required for the analysis.
- **API Calls**: Use `requests.get()` to make API calls to endpoints like rockets, launchpads, payloads, and cores.
    - Retrieve relevant JSON data from these endpoints.
- **Data Cleaning**: Remove rows with multiple cores or payloads.
    - Extract specific fields from the raw data.
    - Convert timestamps to readable dates.
    - Replace missing values (e.g., calculate the mean for `PayloadMass`).
- **Helper Functions**: Use functions like `getBoosterVersion` to add details such as booster version, launch site name, and payload metadata.
- **Data Consolidation**: Consolidate the information into a structured dictionary.
    - Convert the dictionary into a Pandas DataFrame.
- **Filter for Falcon 9 Launches**: Focus the DataFrame on Falcon 9 launches to ensure the dataset is focused.
- **Result**: The process yields a clean, comprehensive, and up-to-date dataset ready for analysis.



Link to GitHub for this part: [InshaRizwan/data-cleaning-and-wrangling]

Data Collection - Scraping

- We installed the required packages, `beautifulsoup4` and `requests`, to perform web scraping. After requesting the webpage using `requests.get()`, we created a BeautifulSoup object by parsing the response content. We then located all the tables with the class "wikitable" and extracted the column names from the table headers. A dictionary was prepared to store the column names as keys and empty lists as values. We iterated through each row, extracting the relevant data, and used helper functions to clean and format the values. The data was then stored in the dictionary. Finally, we converted the dictionary into a Pandas DataFrame for further analysis.

Install Required Packages (beautifulsoup4, requests)

Request Web Page (GET) (requests.get())

Create BeautifulSoup Object (soup = BeautifulSoup())

Find All Tables and Extract Column Names

Prepare Data Dictionary and Iterate Through Rows

Store Data in Dictionary and Convert to DataFrame

URL for Github: [InshaRizwan/data-scraping]

Data Wrangling

- **Load the Dataset**: Use pandas to load the dataset into a DataFrame and inspect the first few rows with `head()` and `info()` to understand its structure and data types.
- **Check for Missing Values**: Use `isnull().sum()` to identify columns or rows with missing data.
- Handle missing data by either:
  - Imputing missing values using mean or median.
  - Dropping rows/columns with excessive missing data.
- Encode categorical variables into numerical values using label encoding or one-hot encoding.
- Create new features or labels based on existing columns (e.g., success/failure of booster landing).
- **Handle Outliers**: Identify outliers using methods like IQR or Z-score, remove or adjust outliers as necessary.
- **Normalize or Scale Data**: Ensure numerical features are scaled or normalized if required for machine learning models.
- **Export Processed Data**: After cleaning and transformation, export the processed data into a new CSV file for further analysis or modeling.

URL for GitHub: InshaRizwan/Data-Wrangling

Start

Load Data into df

Inspect Data (head, info)

Are there missing values? --> Yes --> Handle Missing Data

-------> Fill/Drop

No

Check for Outliers ---> No ---> Proceed to Feature Engineering

Handle Outliers ---> Yes ---> Proceed to Feature Engineering

Feature Engineering (Create New Columns, Encode Categorical Features) AND Normalize Data (if required)

Export Cleaned Data (df.to_csv())

EDA with Data Visualization

Here's a table summarizing the plotted charts and their purposes:

Task	Chart Type	Purpose	Key Insights
1	Scatter Plot: Flight Number vs. Launch Site	Observe how flight frequency at various sites correlates with successful outcomes.	Increasing flight numbers show higher success rates, indicating operational improvements.
2	Scatter Plot: Payload Mass vs. Launch Site	Analyze the impact of payload mass on landing success at different sites.	Success varies by site, even for heavier payloads.
3	Bar Chart: Success Rate by Orbit Type	Explore success rate variations across different orbit types.	Certain orbits show higher success rates due to trajectory or payload factors.
4	Scatter Plot: Flight Number vs. Orbit Type	Evaluate the relationship between repeated missions and success for specific orbit types.	Higher flight numbers improve success for some orbits, indicating experience benefits.

5	Scatter Plot: Payload Mass vs. Orbit Type	Determine how payload mass affects success rates for different orbits.	Heavier payloads perform differently depending on the orbit type.
6	Line Chart: Yearly Success Trends	Visualize the trend of average success rates over the years.	Positive trends indicate advancements in technology and operations over time.
7	One-Hot Encoding for Categorical Features	Prepare categorical data for machine learning by converting it into numerical format.	Features like orbit, launch site, landing pad, and serial were encoded for model compatibility.
8	Casting Numeric Columns to Float64	Standardize numeric data format for computational processing.	Ensured compatibility with machine learning algorithms.

URL to GitHub:[InshaRizwan/EDA-with-visualization]

EDA with SQL

- We performed several SQL queries the summary of which are:
- **Task 1**: Displayed the unique launch site names in the SpaceX mission.
  - SQL Query: `SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;`
- **Task 2**: Retrieved 5 records where launch sites start with "CCA".
  - SQL Query: `SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;`
- **Task 3**: Calculated the total payload mass carried by NASA (CRS).
  - SQL Query: `SELECT SUM("Payload_Mass_(kg)") FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)';`
- **Task 4**: Computed the average payload mass carried by booster version **F9 v1.1**.
  - SQL Query: `SELECT AVG("Payload_Mass_(kg)") FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';`
- **Task 5**: Identified the date of the first successful ground pad landing.
  - SQL Query: `SELECT MIN("Date") FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)';`

- **Task 6**: Listed booster versions with successful drone ship landings and payload mass between 4000 and 6000 kg.
  - SQL Query: `SELECT DISTINCT "Booster_Version" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (drone ship)' AND "Payload_Mass_(kg)" BETWEEN 4000 AND 6000;`
- **Task 7**: Counted the total number of successful and failed mission outcomes.
  - SQL Query: `SELECT "Landing_Outcome", COUNT(*) FROM SPACEXTABLE GROUP BY "Landing_Outcome";`
- **Task 8**: Listed booster versions carrying the maximum payload mass.
  - SQL Query: `SELECT "Booster_Version" FROM SPACEXTABLE WHERE "Payload_Mass_(kg)" = (SELECT MAX("Payload_Mass_(kg)") FROM SPACEXTABLE);`
- **Task 9**: Retrieved failure outcomes on drone ships in 2015, with month, booster version, and launch site details.
  - SQL Query: `SELECT strftime('%m', "Date") AS "Month", "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Failure (drone ship)' AND substr("Date", 1, 4) = '2015';`
- **Task 10**: Ranked the count of landing outcomes between 2010-06-04 and 2017-03-20 in descending order.
  - SQL Query: `SELECT "Landing_Outcome", COUNT(*) FROM SPACEXTABLE WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY "Landing_Outcome" ORDER BY COUNT(*) DESC;`

URL of GitHub:[InshaRizwan/EDA-with-SQL]()

Build an Interactive Map with Folium

Object	Purpose	Details
Circles	Highlighted key locations on the map (e.g., launch sites, NASA Johnson Space Center)	- folium.Circle was used to represent key launch sites with a radius of 1000 meters.- A Popup containing site details (e.g., "NASA Johnson Space Center") was added to give more information upon hovering or clicking the circle.- This visually highlights the area of the site.
Static Markers	Indicated the exact locations of key sites with labels	- folium.map.Marker was used with a custom DivIcon to add text labels in a styled format (e.g., orange color, 12 font size).- Each marker had a popup that displayed additional details like the name of the site and its status.- Used for static locations like NASA JSC for better site identification.
Dynamic Markers	Represented the success/failure of individual launches across different sites	- folium.Marker was used for dynamic launch data.- Green markers were used for successful launches (class=1) and red markers for failed launches (class=0).- Each marker had a popup showing the launch site and success/failure status. This allowed users to track individual mission outcomes.

Marker Clusters	Grouped markers with identical coordinates to reduce map clutter and make the map easier to interpret	- MarkerCluster was applied to group markers that had identical coordinates (since many launches occur at the same location).- This approach improves the readability of the map by clustering nearby points and reduces marker overlap. The markers could be expanded for more details when clicked.
Mouse Position Tool	Enabled interactive exploration and coordinate identification on the map	- MousePosition was added to display the real-time coordinates (latitude, longitude) of the mouse cursor on the map.- Users could interactively hover over different points on the map to get their coordinates.- This tool helped identify proximity to specific points of interest (e.g., railways, nearby facilities).

URL to GitHub: [InshaRizwan/Building-an-Interactive-Map-with-Folium-]()

Build a Dashboard with Plotly Dash

- The Summary of plots/graphs and interactions we have added to the dashboard are:

Plot/Interaction	Description	Purpose
Dropdown for Launch Site Selection	A dropdown input allowing users to select a launch site or view all sites.	Enables users to focus on a specific launch site or see data for all sites, allowing for site-specific comparison or overall trends.
Range Slider for Payload Mass	A slider allowing users to select a range of payload masses (0 Kg to 10,000 Kg).	Filters launch data based on payload mass, enabling users to identify patterns between payload size and mission outcomes.
Pie Chart for Launch Success Rate	A pie chart showing the distribution of successful vs. failed launches.	Provides a quick visual of success rates, which can be viewed for all sites or filtered by specific launch site, allowing comparison of success rates.

Scatter Plot for Success vs Payload Mass	A scatter plot where the x-axis is payload mass, the y-axis is launch outcome (success/failure), and points are colored by booster version.	Visualizes how payload mass correlates with mission success, with additional information on booster version performance.
Callback for Interactive Plot Updates	Callback functions that update the pie chart and scatter plot based on user input.	Ensures the dashboard dynamically updates visualizations based on selected launch site and payload range, enhancing user interaction.
Integration of Multiple Inputs (Dropdown & Slider)	Dropdown and range slider work together to filter and visualize data.	Provides a customized user experience where the user can filter data based on both launch site and payload range, enabling focused analysis.

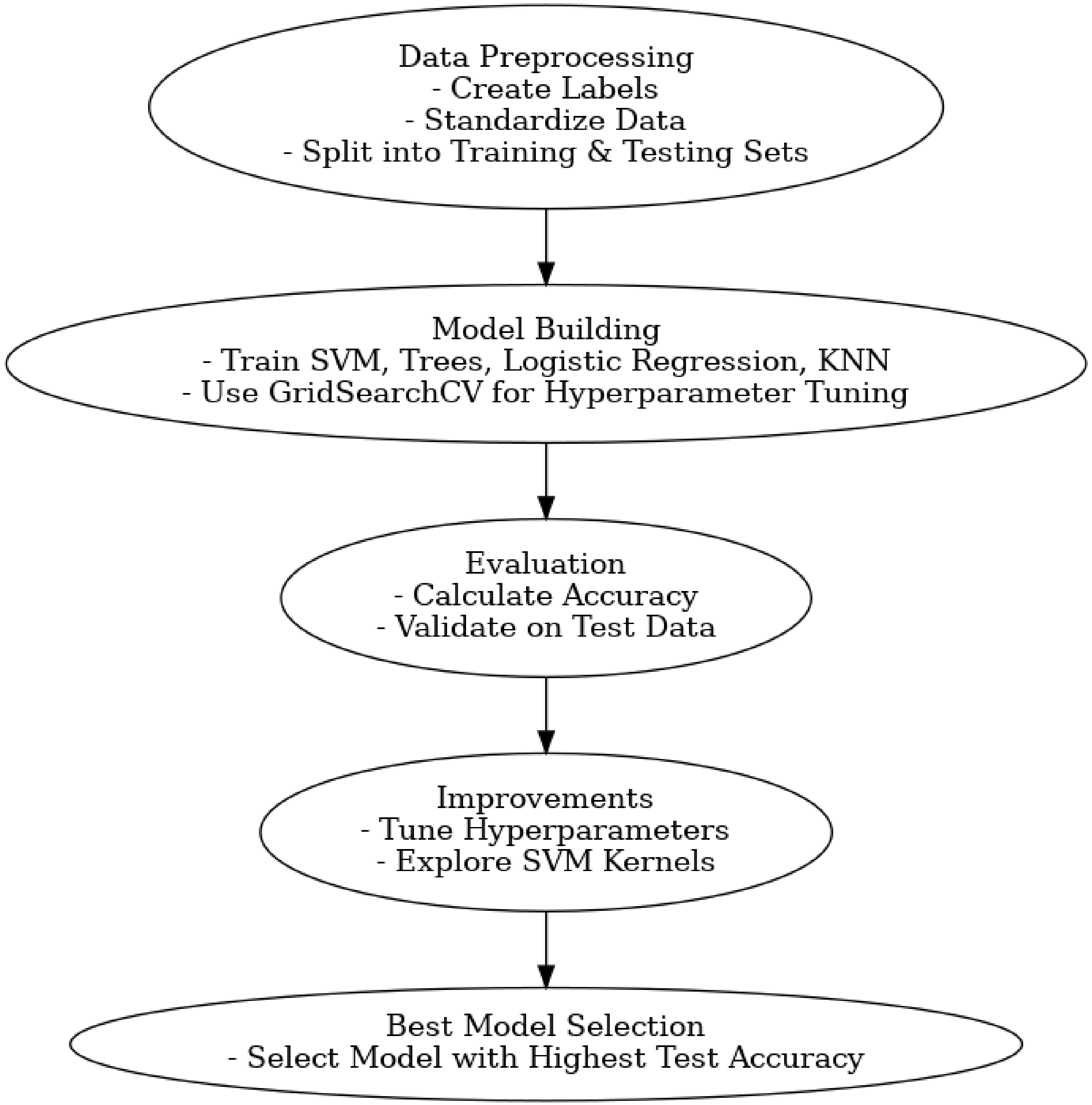
URL to Github: [InshaRizwan/Dashboard]()

Predictive Analysis (Classification)

The following steps were involved in building, evaluating, improving, and identifying the best classification model:

- **Data Preprocessing**:
  - Created labels for the target variable (class column).
  - Standardized the dataset for consistency in model input.
  - Split the dataset into training and testing sets using `train_test_split`.
- **Model Building**:
  - Trained multiple models including SVM, classification trees, logistic regression, and k-nearest neighbors (KNN).
  - Used `GridSearchCV` to optimize hyperparameters for each model.
  - Identified the best-performing configurations for each model.
- **Evaluation**:
  - Assessed model accuracy using validation and test datasets.
  - Calculated the accuracy of each model with test data to determine performance.
- **Improvements**:
  - Performed hyperparameter tuning with `GridSearchCV` to enhance model performance.
  - Explored different kernels for SVM and other parameter adjustments.
- **Best Performing Model**:
  - Selected the model with the highest accuracy on the test dataset.

The Model Development Flowchart



GitHub URL: InshaRizwan/The-Model-Development

Results

- Exploratory data analysis results

Aspect	Details
Dataset Size	18 test samples.
Evaluation Metrics	Accuracy, confusion matrix, and hyperparameter tuning were used to evaluate performance.
Key Hyperparameters	- Logistic Regression: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
	- SVM: {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
	- KNN: {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
Accuracy Scores	- Logistic Regression: 0.846 on training, 0.833 on test data.
	- SVM: 0.848 on training, 0.833 on test data.
	- Decision Tree: 0.611 on test data.
	- KNN: 0.848 on training, 0.833 on test data.
Confusion Matrix Insights	- True Positives: 12 (landed and correctly predicted landed).
	- False Positives: 3 (not landed but incorrectly predicted as landed).
Best Performing Model	Logistic Regression.
Best SVM Kernel	Sigmoid.

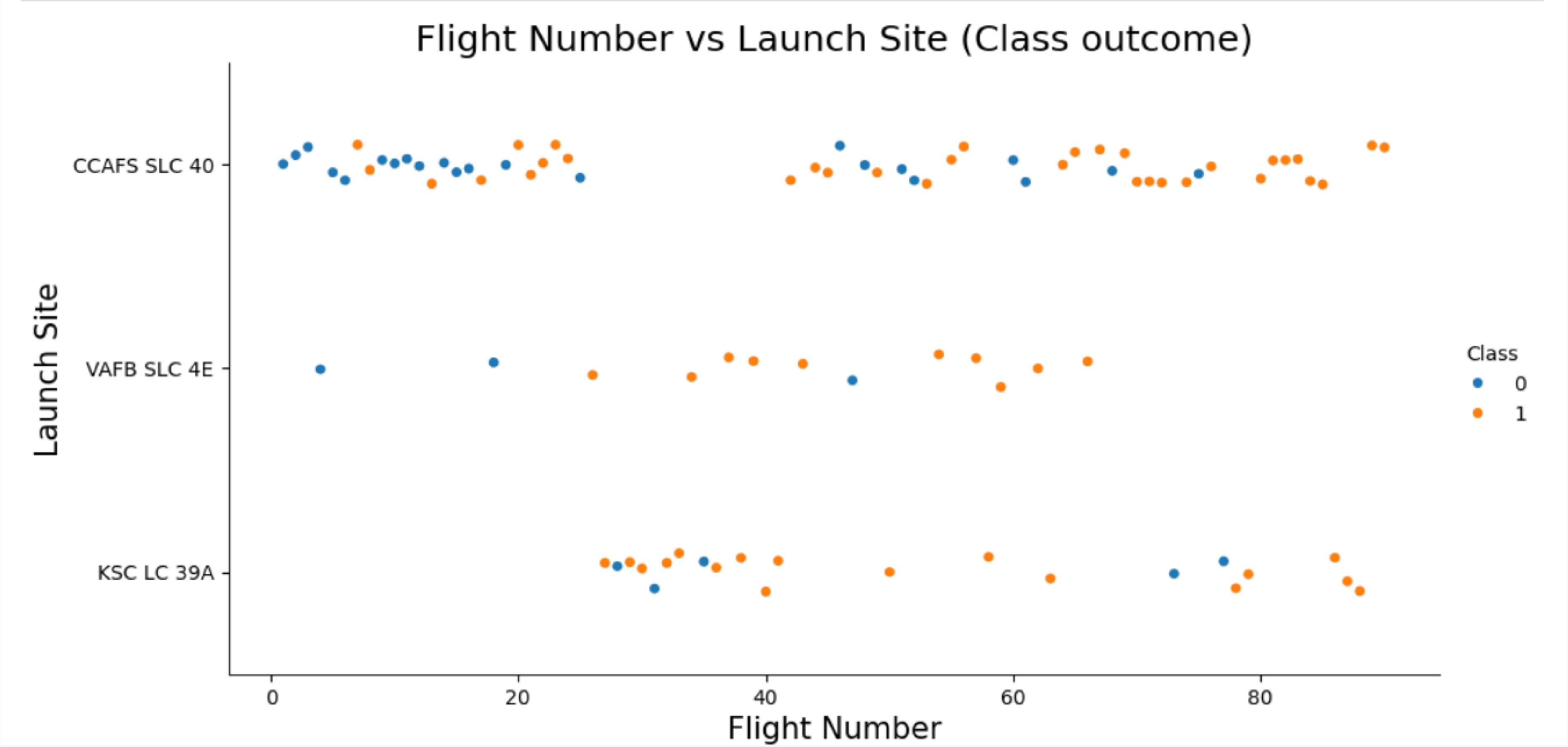
Summary of Exploratory Data Analysis (EDA) Results:

Aspect	Details
Launch Sites	- CCAFS LC-40, VAFB SLC-4E, KSC LC-39A, CCAFS SLC-40.
Total Payload Mass by NASA (CRS)	0.0 kg – No payload was carried by boosters launched by NASA (CRS).
Average Payload Mass (F9 v1.1)	0.0 kg – Booster version F9 v1.1 carried no payload.
First Successful Landing (Ground Pad)	Achieved on 2015-12-22.
Mission Outcomes (Success/Failure)	- Success: 38 (general), 14 (drone ship), 9 (ground pad).
	- Failure: 3 (general), 5 (drone ship), 2 (parachute).
	- Other: 21 (No attempt), 1 (Precluded), 5 (Controlled ocean), 2 (Uncontrolled ocean).
2015 Failure Records	- Month: January and April.
	- Landing Outcome: Failure (drone ship).
	- Booster Versions: F9 v1.1 B1012, F9 v1.1 B1015.
	- Launch Site: CCAFS LC-40.
Landing Outcomes (2010-06-04 to 2017-03-20)	- Most Frequent: "No attempt" (10).
	- Success (drone ship): 5, Failure (drone ship): 5.
	- Success (ground pad): 3, Controlled ocean: 3.
	- Other: Uncontrolled ocean (2), Failure (parachute) (2), Precluded (1).

Section 2

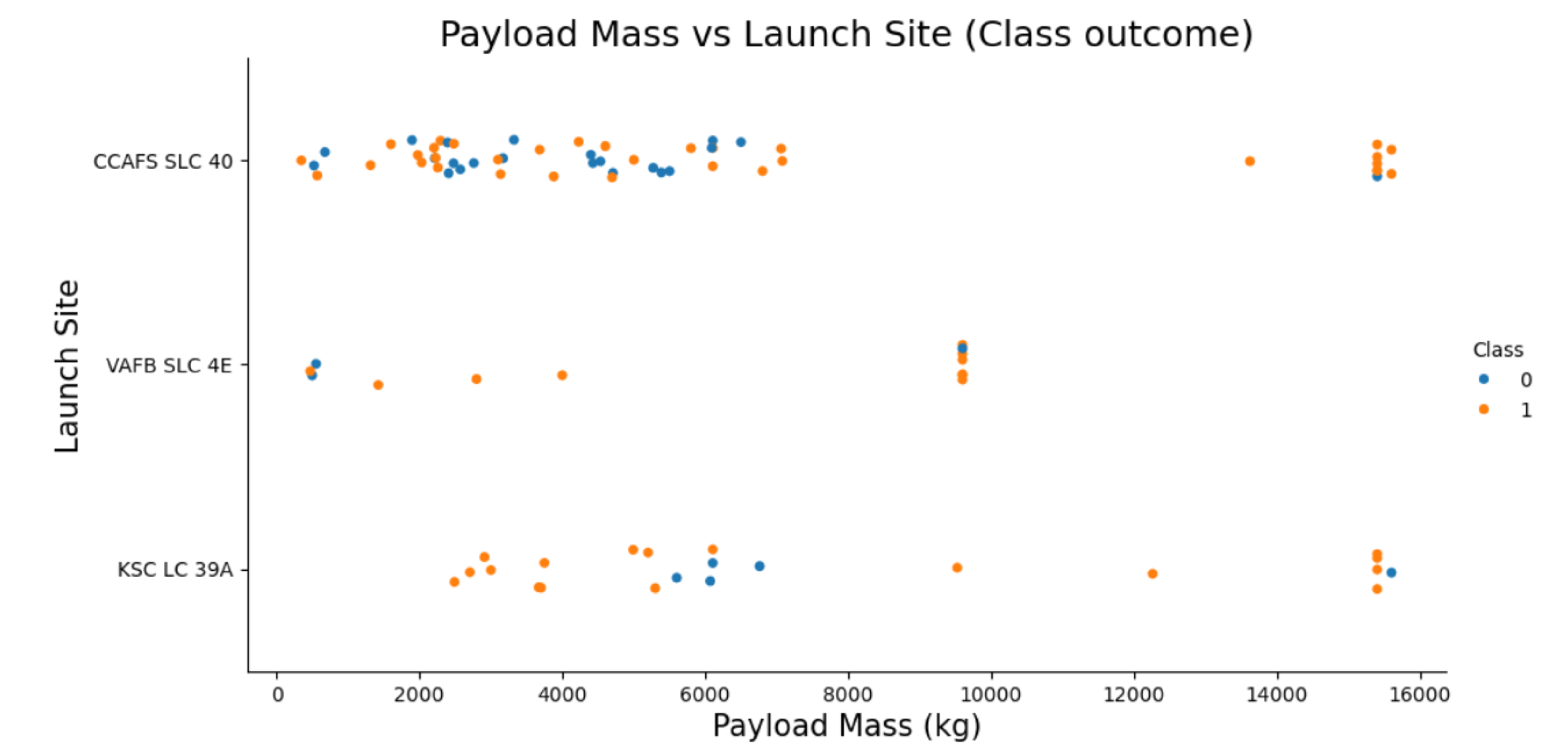
Insights drawn from EDA

Flight Number vs. Launch Site



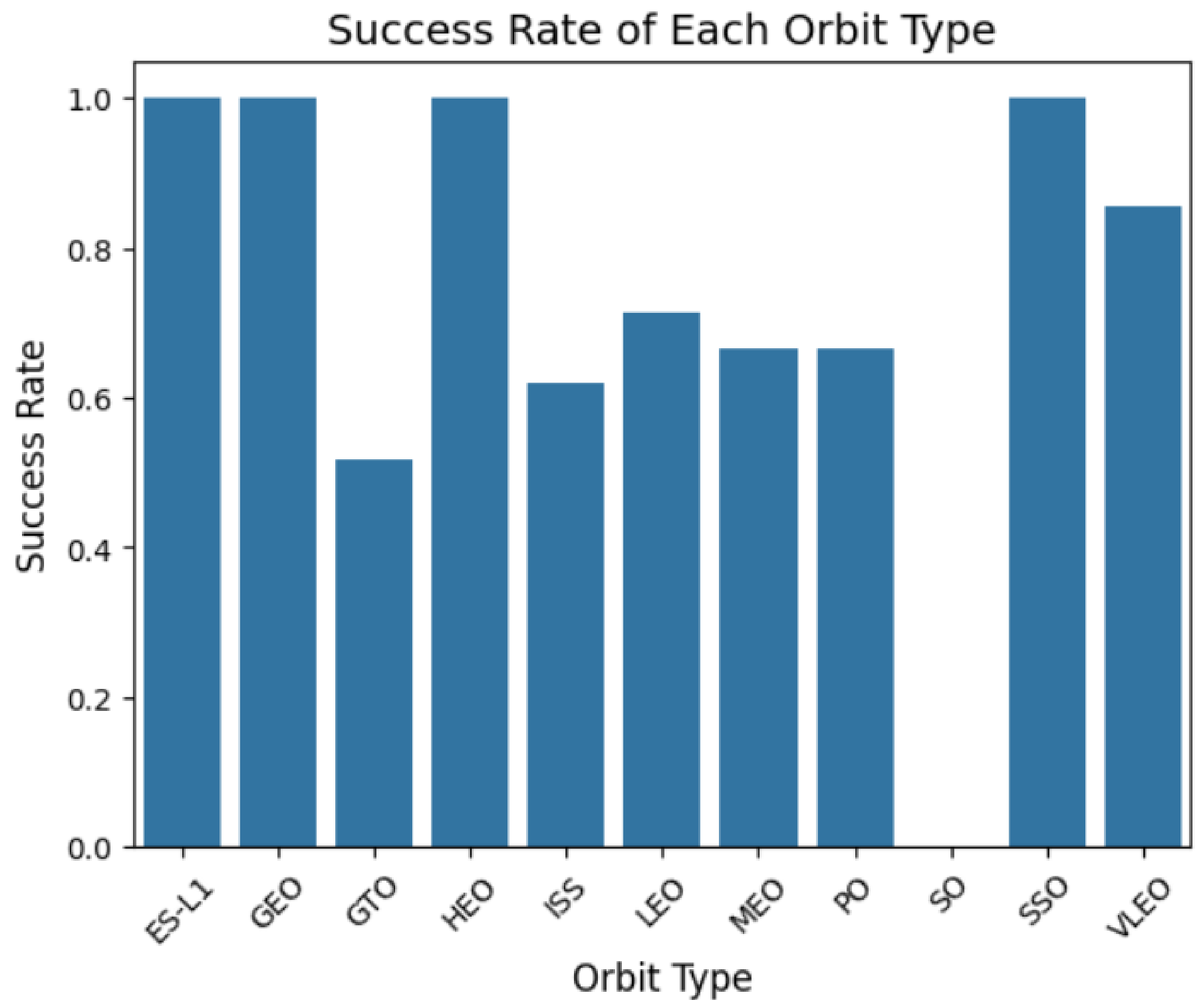
- Different launch sites (CCAFS SLC 40, VAFB SLC 4E, KSC LC 39A) show varying success (Class = 1) and failure (Class = 0) rates as the flight numbers increase.
- Success rates improve with increasing flight numbers, suggesting that operational experience and iterations contribute to better outcomes.

Payload vs. Launch Site



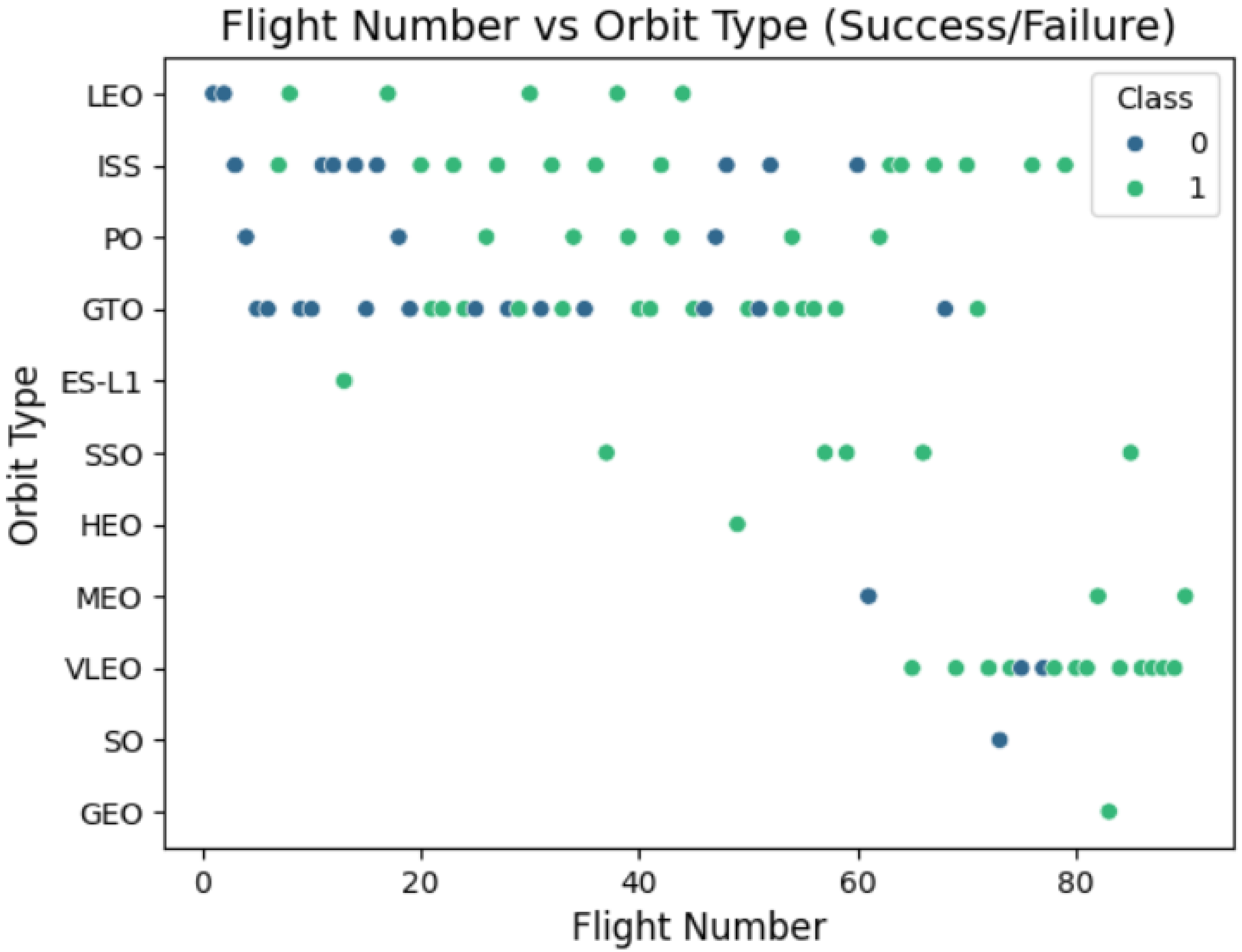
- Each launch site has handled a range of payload masses.
- Success (Class = 1) is observed across payload masses at all three sites, but specific sites may handle certain payload ranges more successfully.

Success Rate vs. Orbit Type



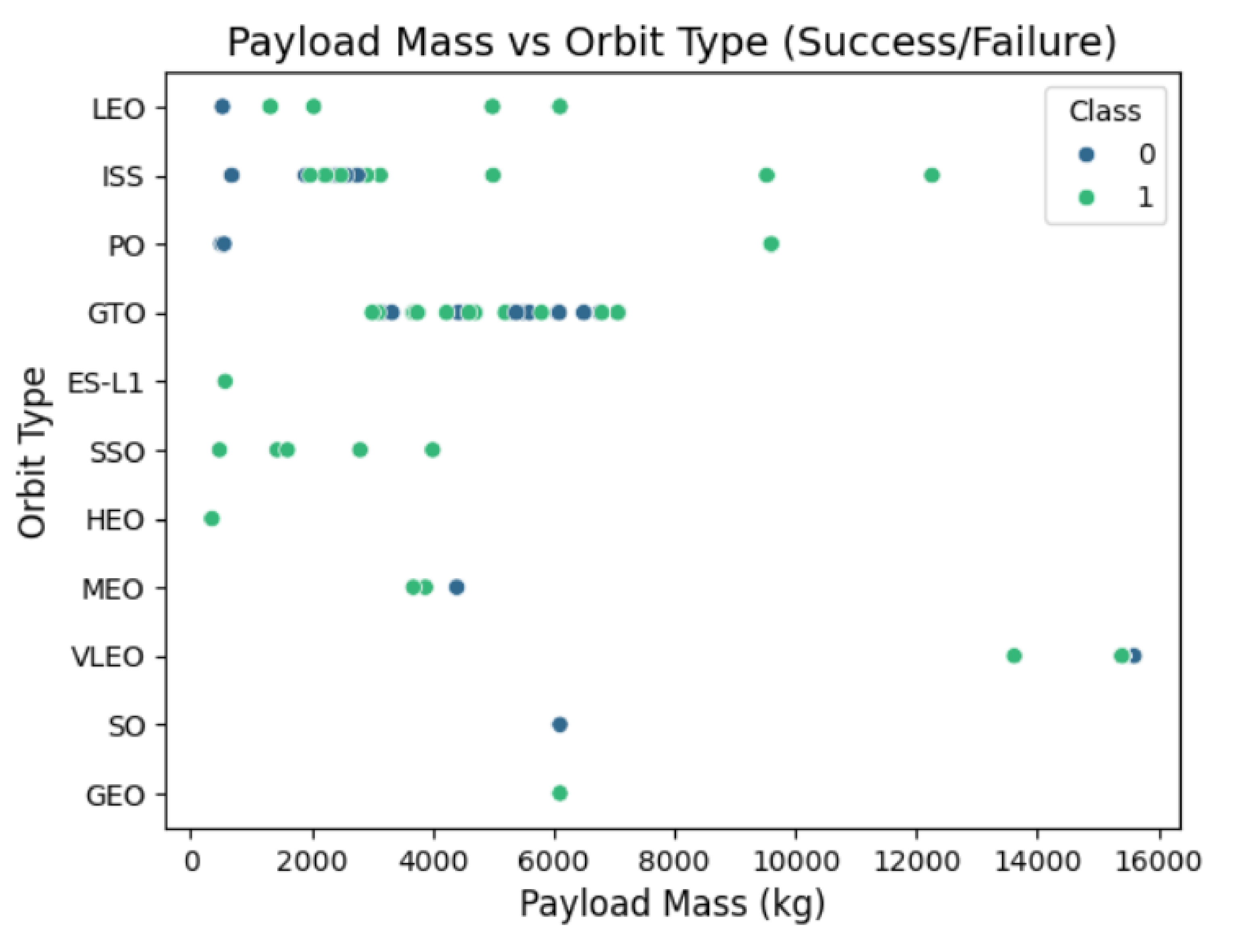
- **Highest Success Rates**: The orbit types ES-L1, GEO, and SSO have the highest success rates, each at 1.0.
- **Lowest Success Rate**: The orbit type GTO has the lowest success rate, around 0.5.
- **Varying Success Rates**: The orbit types HEO, ISS, LEO, MEO, PO, SO, and VLEO have varying success rates between 0.6 and 0.9.
- **VLEO**: The orbit type VLEO has a success rate slightly above 0.8.
- **HEO and ISS**: The orbit types HEO and ISS have success rates below 0.8 but above 0.6.
- **LEO, MEO, PO, and SO**: The orbit types LEO, MEO, PO, and SO have success rates around 0.7.

Flight Number vs. Orbit Type



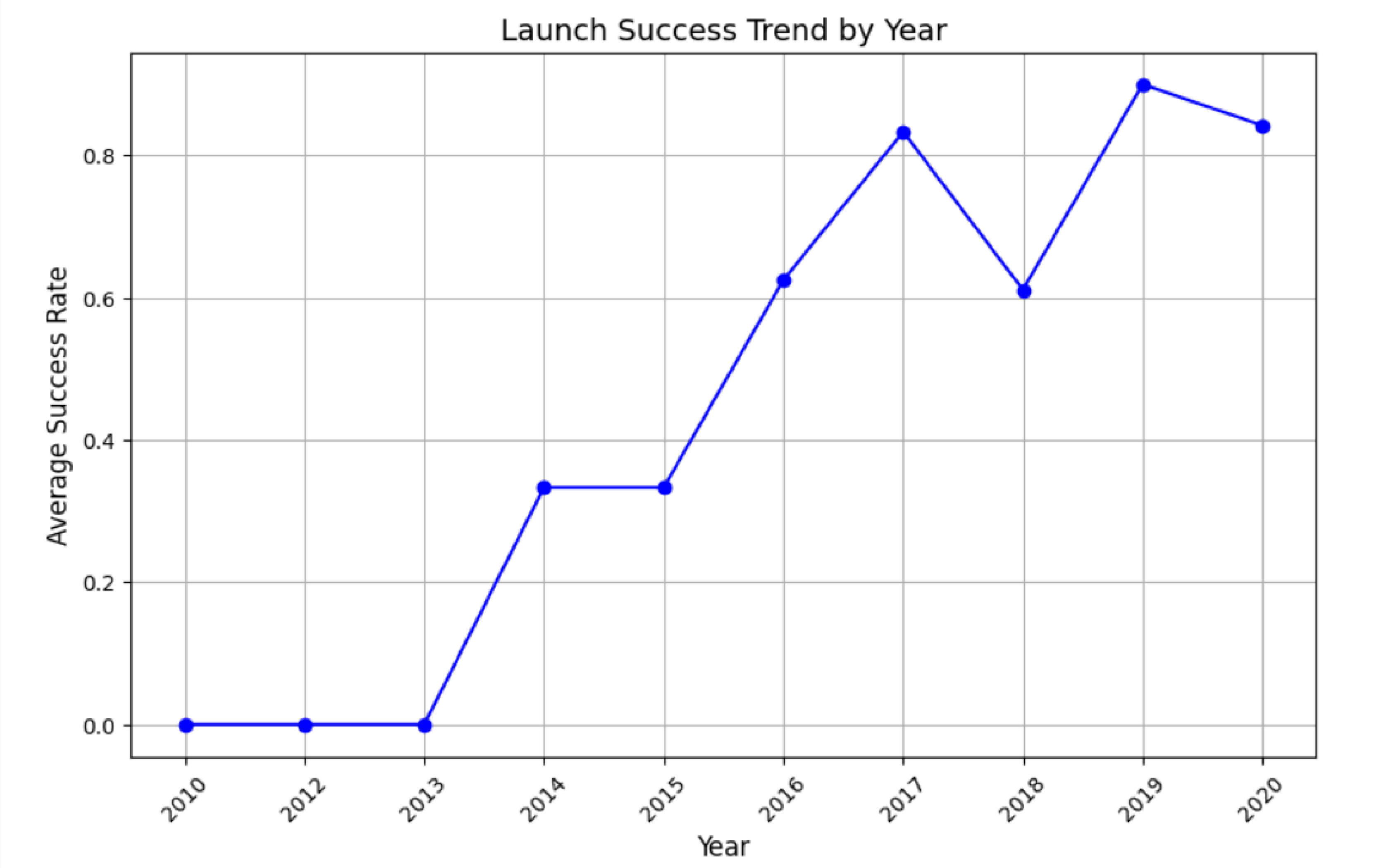
- **Success and Failure Distribution**: The plot shows the distribution of successful (green) and failed (blue) flights across different orbit types and flight numbers.
- **Orbit Types**: The orbit types include LEO, ISS, PO, GTO, ES-L1, SSO, HEO, MEO, VLEO, SO, and GEO.
- **Flight Numbers**: The x-axis represents flight numbers ranging from 0 to 100.
- **Success Rates**: The success rates vary across different orbit types, with some orbit types having more successful flights than others.

Payload vs. Orbit Type



- Payloads with varying masses have been launched into different orbit types.
- - Success (Class = 1) is generally observed across multiple orbit types, such as LEO (Low Earth Orbit), GTO (Geostationary Transfer Orbit), and others.
- Higher payload masses (e.g., >10,000 kg) seem to have more successful outcomes in certain orbit types.

Launch Success Yearly Trend



- The average success rate of launches has increased over time, particularly after 2013.
- There is a significant improvement in success rates between 2013 and 2020, indicating advancements in technology or operations.

All Launch Site Names

- We can see that unique LaunchSites are: ['CCAFS SLC 40' 'VAFB SLC 4E' 'KSC LC 39A']

```
# Get the unique launch sites
unique_launch_sites = df['LaunchSite'].unique()

# Print the unique launch sites
print("Unique LaunchSites:", unique_launch_sites)
```

```
Unique LaunchSites: ['CCAFS SLC 40' 'VAFB SLC 4E' 'KSC LC 39A']
```

Launch Site Names Begin with 'CCA'

```
# Filter records where Launch_Site begins with 'CCA'
cca_launch_sites = df[df['LaunchSite'].str.startswith('CCA')]

# Display the first 5 records
cca_launch_sites.head(5)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857	0

Total Payload Mass

- Total payload carried by boosters: 549446.3470588236 kg

```
# Calculate the total payload mass for all boosters
total_payload = df['PayloadMass'].sum()

print(f"Total payload carried by boosters: {total_payload} kg")
```

```
Total payload carried by boosters: 549446.3470588236 kg
```

Average Payload Mass by F9 v1.1

The average Payload mass by F9 v1.1 is as follows:

```
# Filter data for booster version F9 v1.1
f9_v1_1_payload = df[df['BoosterVersion'] == 'F9 v1.1']

# Calculate the average payload mass
average_payload_f9_v1_1 = f9_v1_1_payload['PayloadMass'].mean()

print(f"Average payload mass carried by F9 v1.1: {average_payload_f9_v1_1} kg")
```

```
Average payload mass carried by F9 v1.1: nan kg
```

First Successful Ground Landing Date

```python
import sqlite3

# Connect to the SQLite database (replace 'your_database.db' with your databa
conn = sqlite3.connect('df.db')
cursor = conn.cursor()

# SQL query to find the first successful landing on ground pad
query = """
SELECT MIN("Date") AS First_Successful_Landing_Date
FROM df
WHERE Landing_Outcome = 'Success (ground pad)';
"""


# Execute the query
cursor.execute(query)

# Fetch and display the result
result = cursor.fetchone()
print("First Successful Landing Date on Ground Pad:", result[0])

# Close the connection
conn.close()
```

Successful Drone Ship Landing with Payload between 4000 and 6000

```python
import sqlite3

# Connect to the SQLite database
conn = sqlite3.connect('df.db')  # Replace 'your_database.db' with your actual database file
cursor = conn.cursor()

# Correct SQL query
query = """
SELECT Booster_Version
FROM SPACEXTABLE
WHERE Outcome = 'Success (drone ship)'
  AND PayloadMass > 4000
  AND PayloadMass < 6000;
"""

# Execute the query
cursor.execute(query)

# Fetch results
results = cursor.fetchall()

# Display the results
for row in results:
    print("Booster Version:", row[0])

# Close the database connection
conn.close()
```

Total Number of Successful and Failure Mission Outcomes

```
SELECT Landing_Outcome, COUNT(*) AS Outcome_Count
FROM SPACEXTABLE
GROUP BY Landing_Outcome;
```

Boosters Carried Maximum Payload

```
SELECT BoosterVersion, MAX(Payload_Mass_kg) AS Max_Payload_Mass
FROM SPACEXTABLE;
```

2015 Launch Records

```
SELECT Month(Date) AS Month,
       Landing_Outcome,
       BoosterVersion,
       Launch_Site,
```

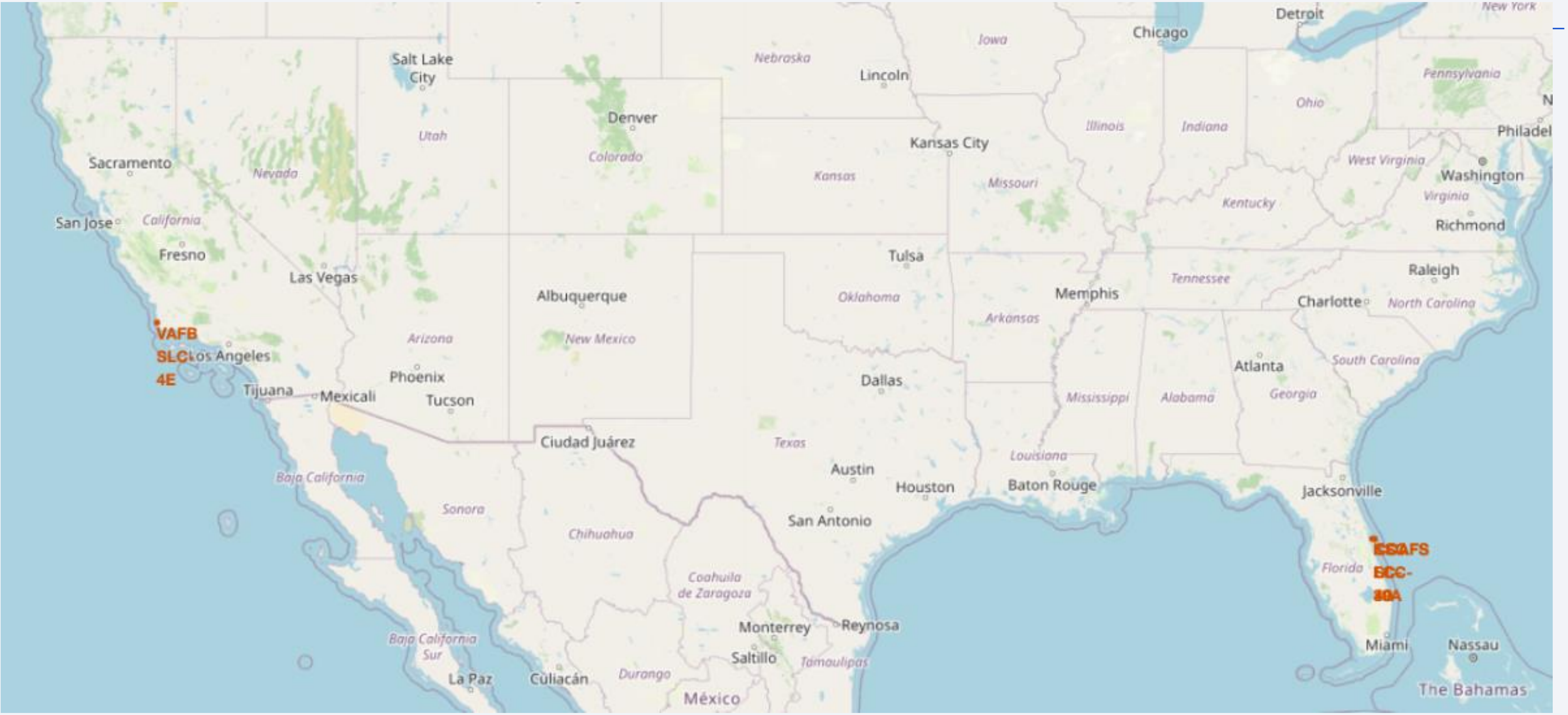
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

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

Section 3

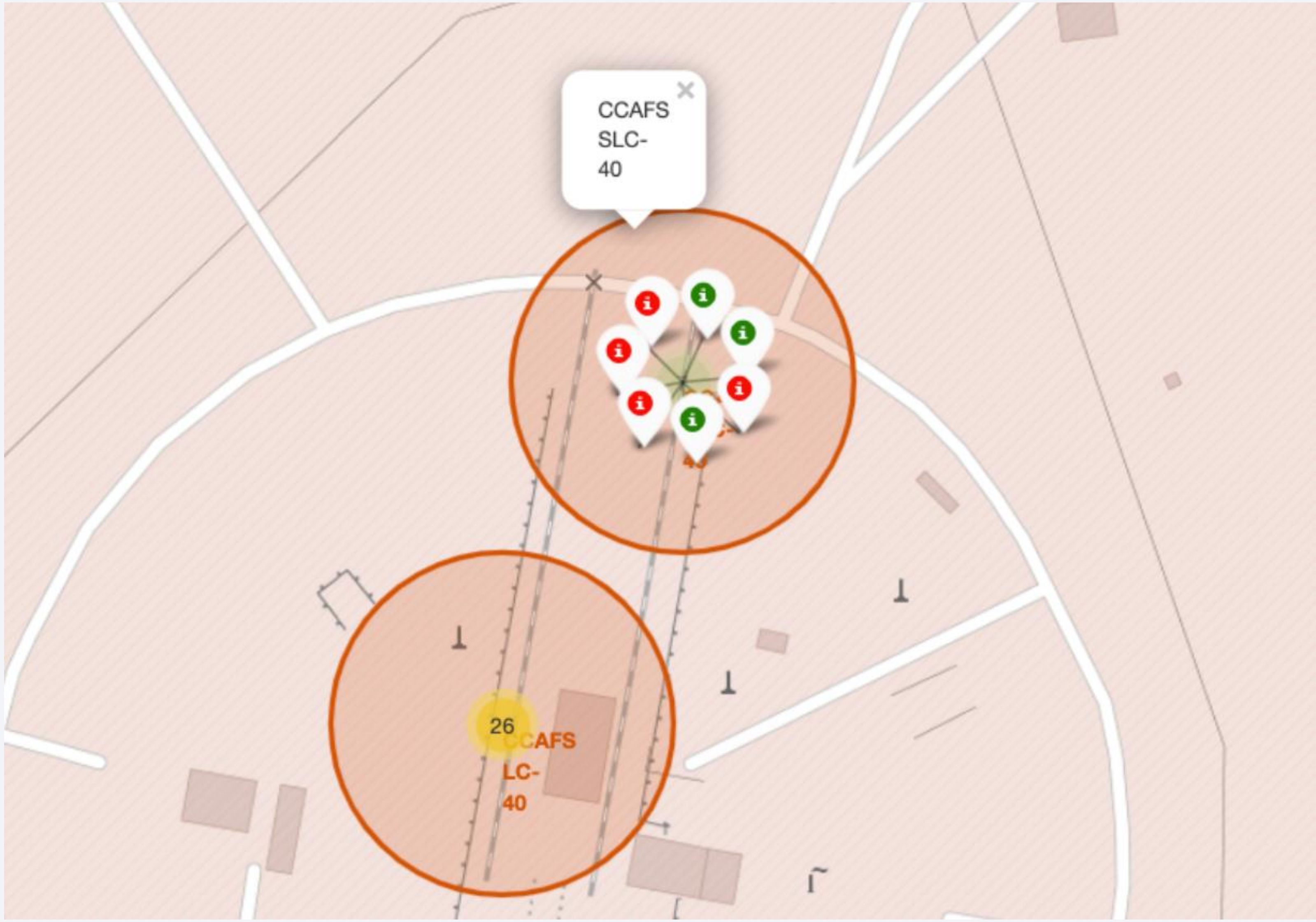
Launch Sites Proximities Analysis

Folium Map with all Launch sites



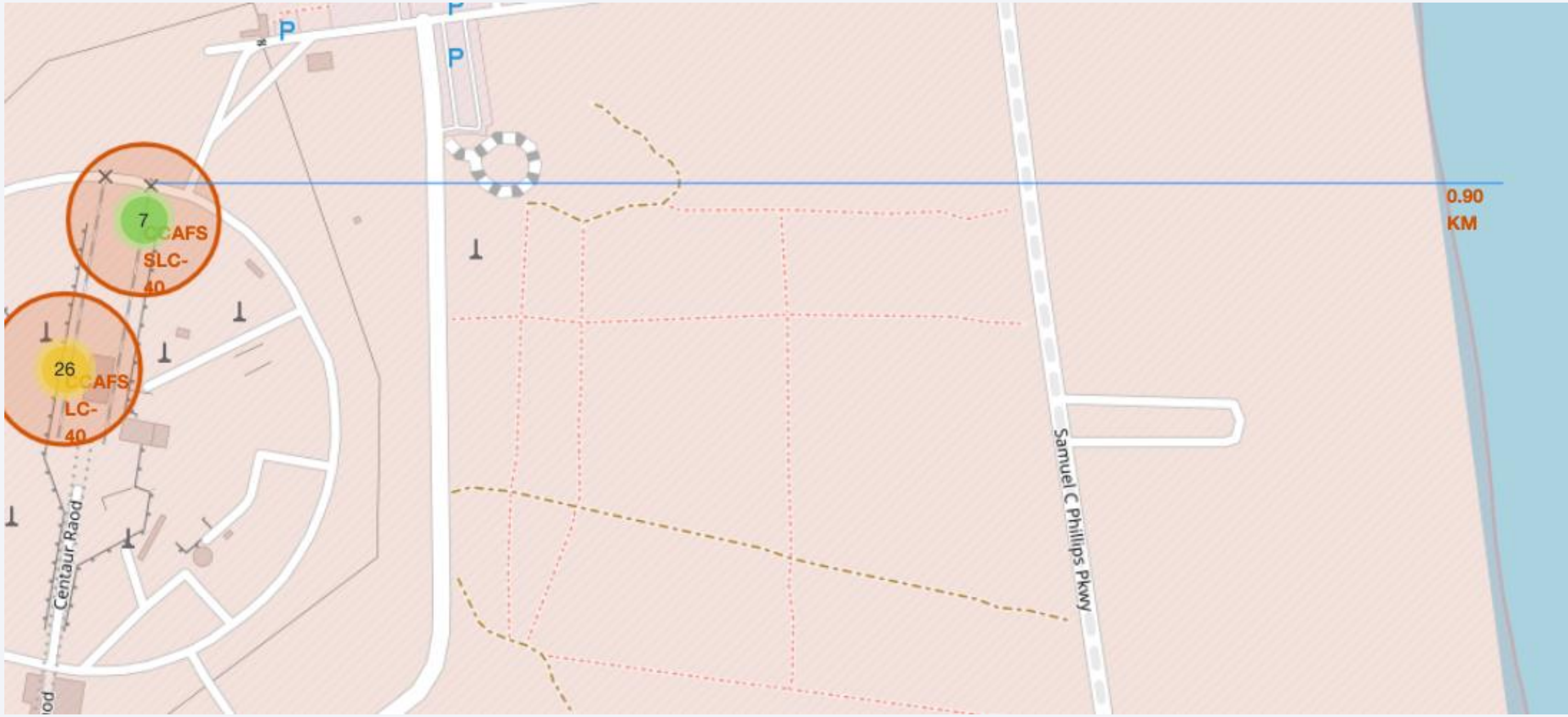
- All launch sites are in proximity to the Equator line
- All launch sites are in very close proximity to the coast

Folium Map - Launch Outcomes



- The green color represents successful launches whereas the red ones depict failed launches

Folium Map – Launch site and its Proximities

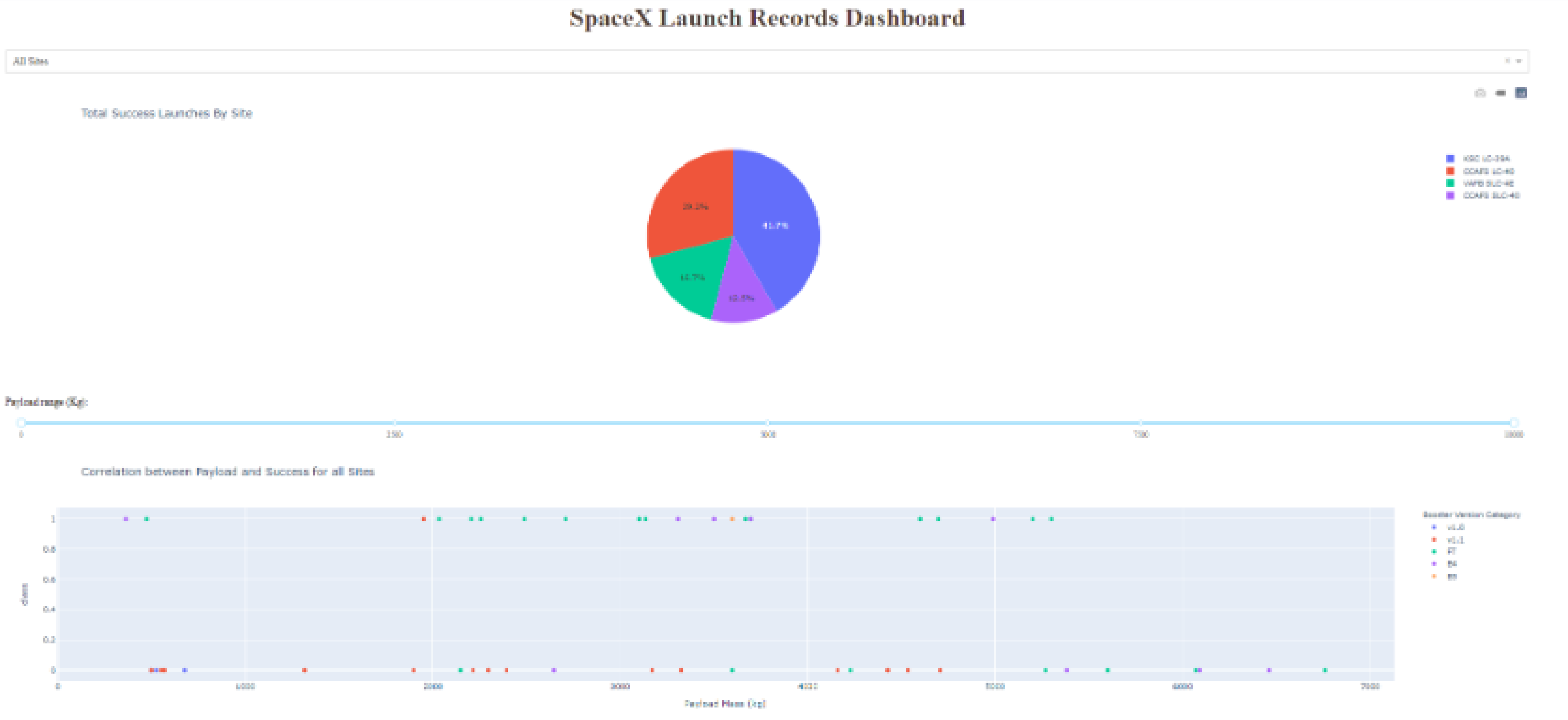


Using this ww can see that the launch site is in proximity to highway and railway but not to the city.

Section 4

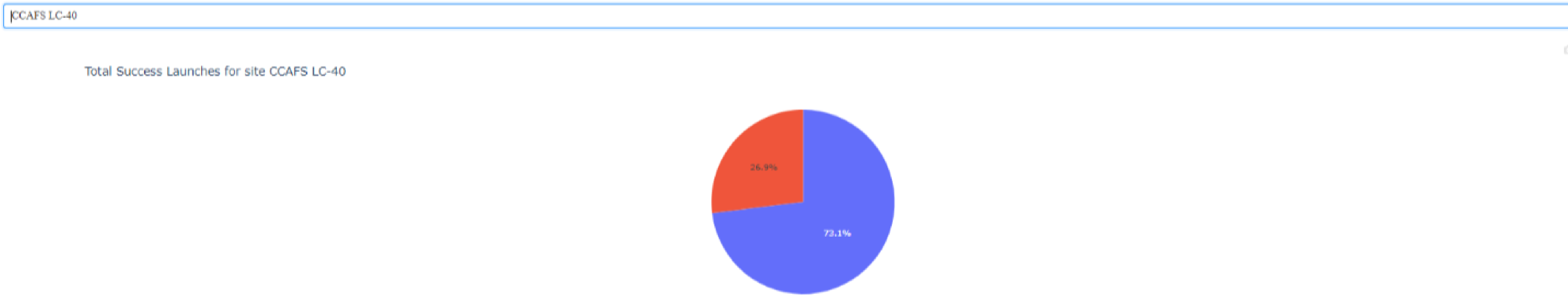
Build a Dashboard with Plotly Dash

SpaceX Launch Records Dashboard



Highest Launches

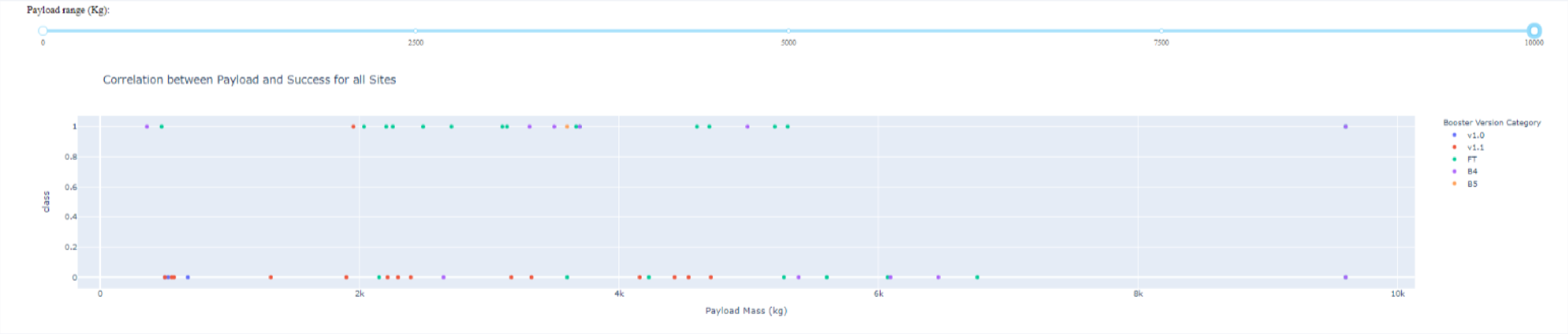
- Pie chart for is selected



The pie chart provides the following conclusions:

- **Success Rate**: The success rate for launches at the CCAFS LC-40 site is 73.1%.
- **Failure Rate**: The failure rate for launches at the CCAFS LC-40 site is 26.9%.

Payload vs. Launch Outcome scatter plot for all sites



- **Payload Mass and Success**: The plot shows the correlation between payload mass (in kilograms) and success (class) for all sites. The success class is binary (0 or 1).
- **Booster Version Categories**: The points on the scatter plot are color-coded based on the booster version category, including v1.0, v1.1, FT, B4, and B5.
- **Payload Range**: The payload mass ranges from 0 to 10,000 kg.
- **Performance and Reliability**: The plot helps in understanding the performance and reliability of different booster versions with varying payloads.

Section 5

Predictive Analysis (Classification)

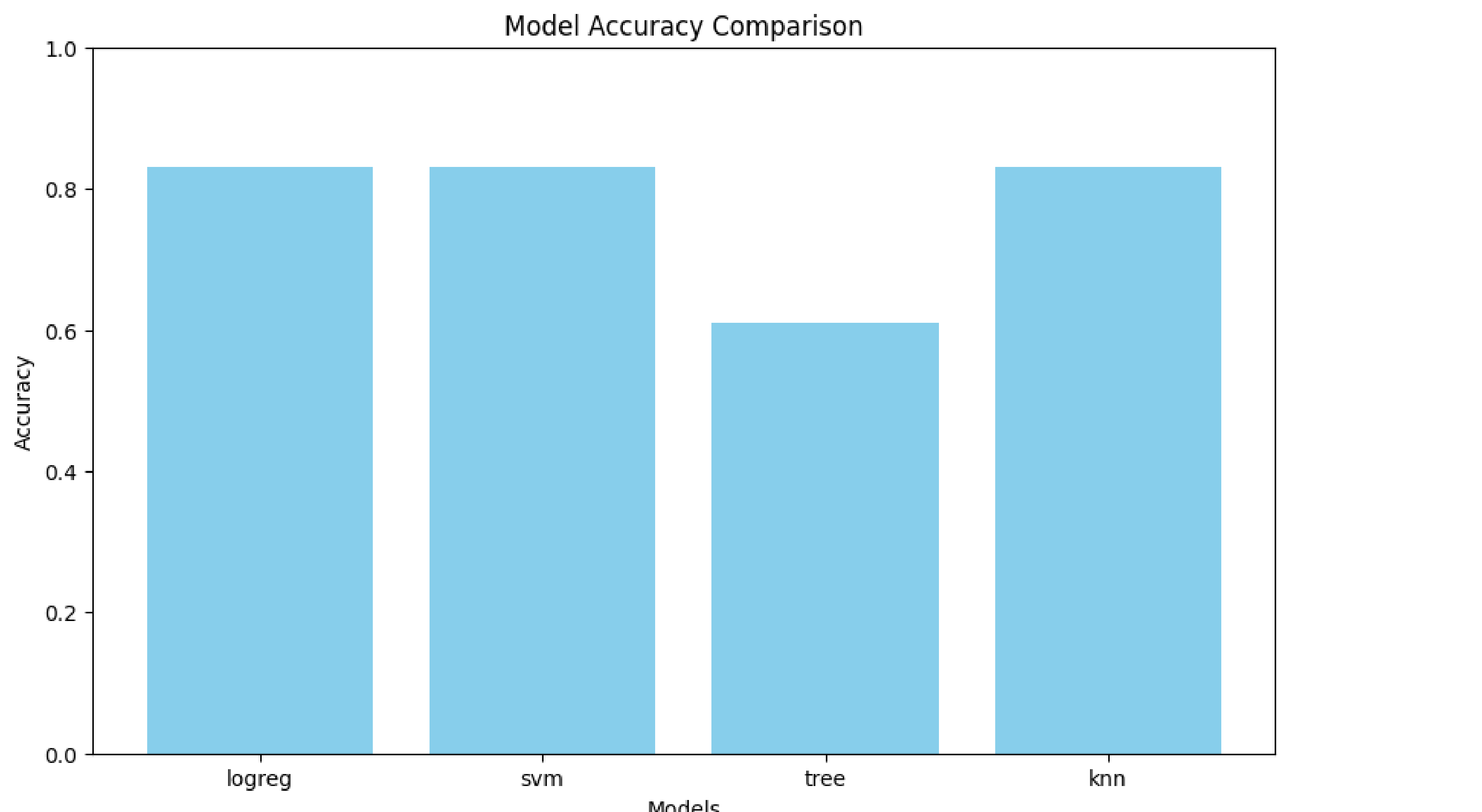
Classification Accuracy

- As we can see logistic regression is the best performing model

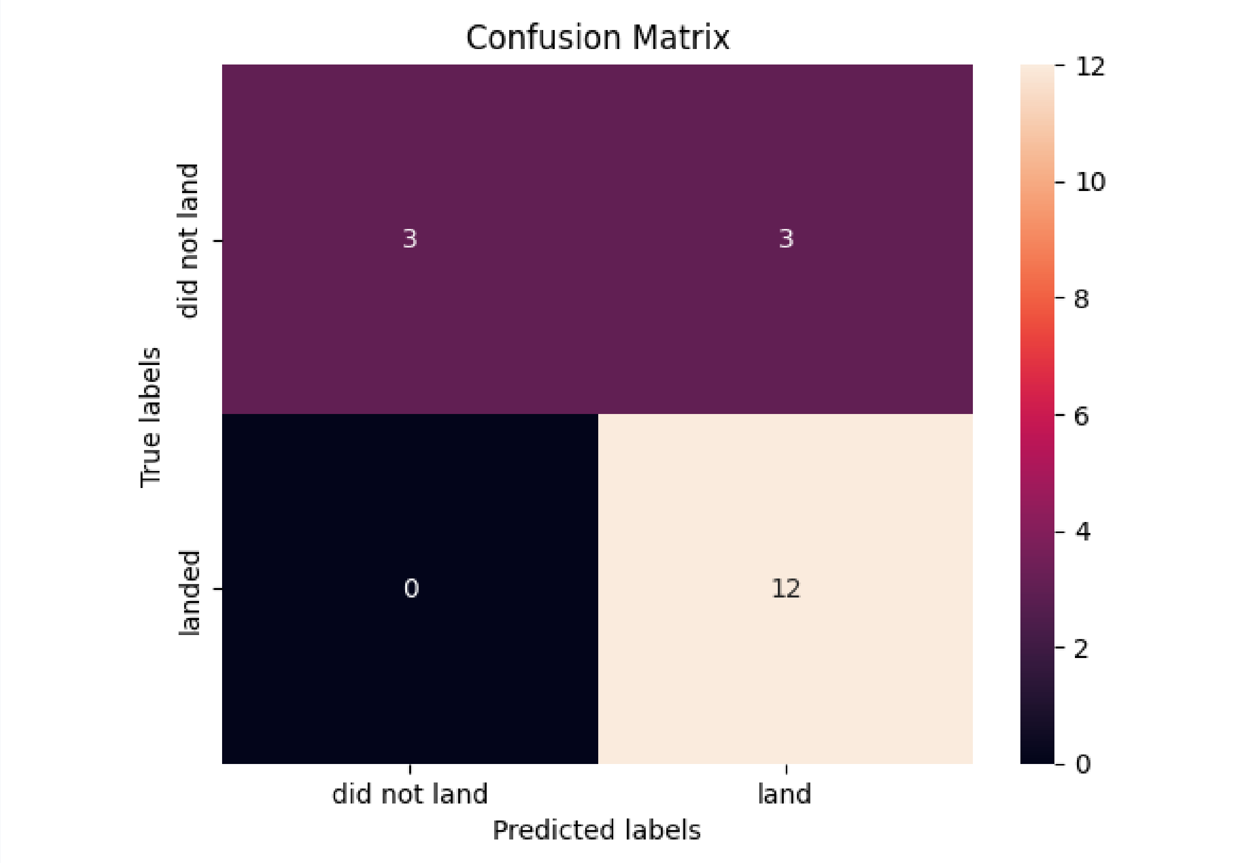
```python
import matplotlib.pyplot as plt

# Example model names and their accuracies
models = ['logreg', 'svm', 'tree', 'knn']
accuracies = [0.83, 0.83, 0.61, 0.83]

plt.figure(figsize=(10, 6))
plt.bar(models, accuracies, color='skyblue')
plt.title('Model Accuracy Comparison')
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.ylim(0, 1)  # Accuracy is between 0 and 1
plt.show()
```



Confusion Matrix



- Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the problem is false positives.
- Overview:
- True Postive - 12 (True label is landed, Predicted label is also landed)
- False Postive - 3 (True label is not landed, Predicted label is landed)

Conclusions

1. Successful Landing Predictions*: Logistic regression emerged as the best-performing model for predicting Falcon 9 first-stage landings, achieving high accuracy and low false positives. This can guide operational decisions for SpaceX and its competitors.

2. Key Insights from EDA:

- Launch success rates vary by site and orbit type, with CCAFS SLC-40 showing the highest success rates.

- Payload and orbit type significantly influence landing outcomes, with mid-range payloads having higher success probabilities.

- Yearly success rates highlight consistent improvements over time.

3. Cost Implications: Accurate landing predictions can enhance cost-efficiency by enabling better planning and competitive pricing for rocket launches.

4. Future Improvements:

- Fine-tuning hyperparameters and exploring ensemble models could further improve predictive performance.

- Incorporating additional features like weather conditions may refine landing outcome predictions.

Thank you!