IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nikolaos Theokritos Tsopanidis
November 20, 2024

Outline

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

Executive Summary

SpaceX is revolutionizing space exploration by making rockets reusable, drastically cutting costs. This project dives into what influences the success of rocket landings, aiming to improve reliability and optimize future launches.

Approach and Methods

Data Collection

Web-scraping from Wikipedia and SpaceX API for launch data

Data Analysis

Exploratory Data Analysis (EDA) with visualizations to uncover trends and patterns.

SQL queries for deeper insights into launch and payload details.

Predictive Modeling:

Machine learning algorithms trained to predict landing success.

Key Outcomes and Takeaways

Found critical factors affecting landing success, like payload mass, orbit type, and launch site.

Observed a strong correlation between more launches and improved success rates.

Achieved high accuracy with machine learning models, highlighting the potential for automating mission planning.

Introduction

Project background

The SpaceX program has deeply impacted on space exploration by achieving cost-effective rocket launches through reusable boosters. This project explores SpaceX's landing data to analyze outcomes, visualize patterns, and predict future landing successes.

Objectives and research questions

What factors influence the success or failure of SpaceX rocket landings?
How can data visualization enhance understanding of launch outcomes?
Can machine learning models accurately predict SpaceX landing success?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Tools used: BeautifulSoup, SpaceX REST API
- Perform data wrangling
  - Modified, enhanced and analyzed the dataset using Pandas & NumPy libraries
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Used Logistic Regression, Support Vector Machines (SVM), Decision Trees and K-Nearest Neighbors (KNN) and employed GridSearchCV with 10-fold cross validation to optimize each algorithm's hyperparameters

Data Collection

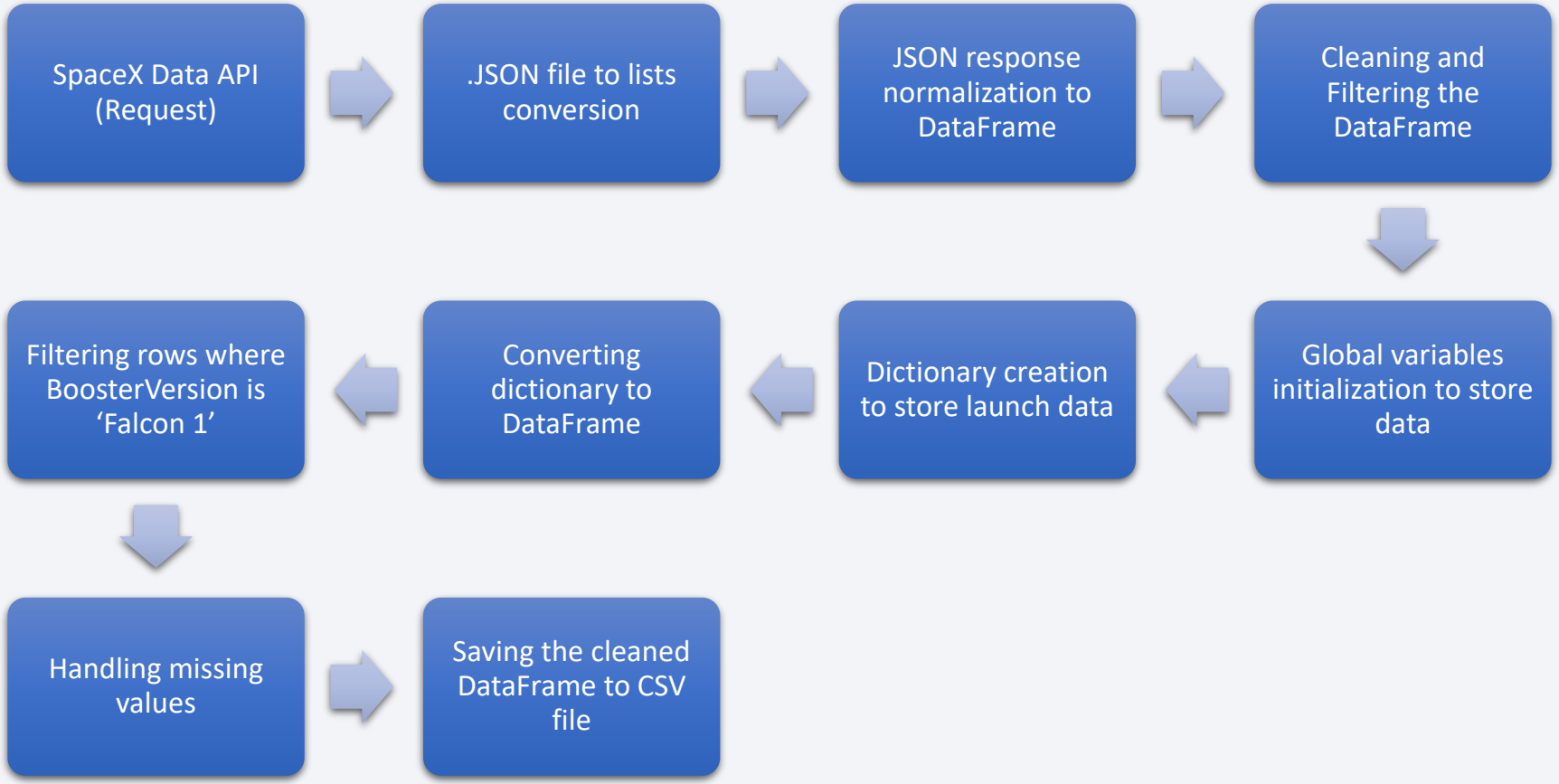
Wikipedia: "List of Falcon 9 and Falcon Heavy launches“

- Extracted data using the BeautifulSoup library to parse HTML tables.
- Defined relevant columns and exported the data as `spacex_web_scraped.csv`

SpaceX REST API:

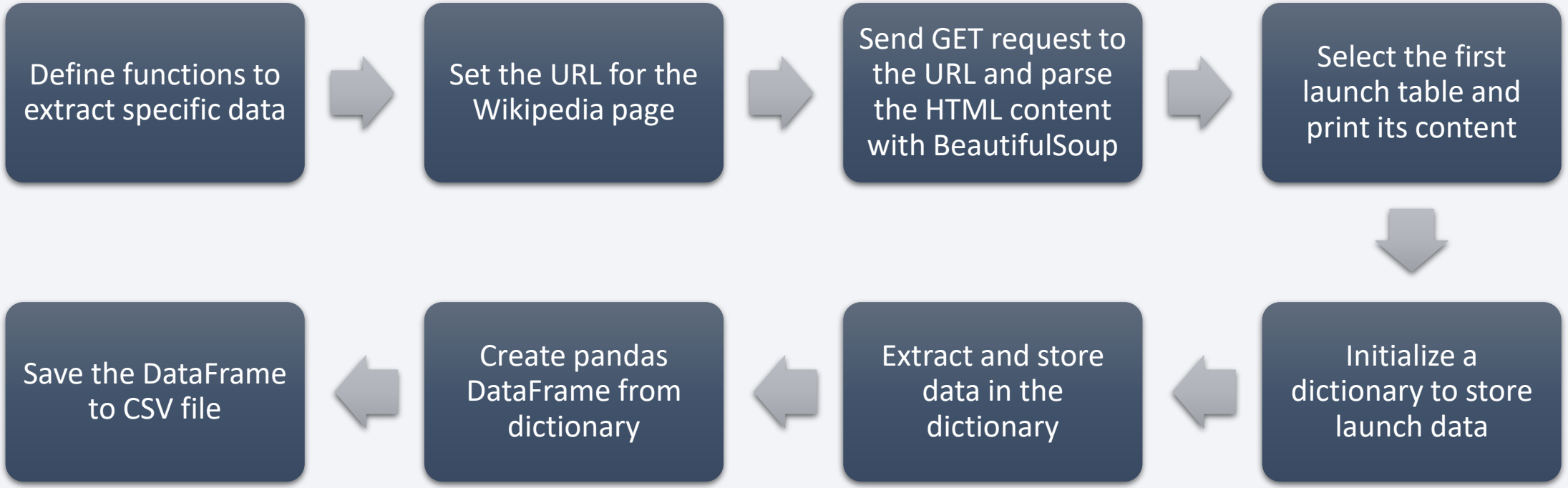
- Collected launch data via API requests.
- Appended data to lists and consolidated into a clean dataset.

Data Collection – SpaceX API



GitHub link: https://github.com/tSopermon/certificate_assignments/blob/main/applied_datascience_capstone/using_spacex_api.ipynb

Data Collection - Scraping



GitHub link: https://github.com/tSopermon/certificate_assignments/blob/main/applied_datascience_capstone/DataCollect_webscr.ipynb

Data Wrangling

Data Cleaning Process

- Removed duplicate rows to ensure data integrity.
- Converted columns like `date_utc` to **datetime** format for better readability
- Created new columns and populated them with corresponding data
- Replaced missing values in the "LandingPad" column with the **mean value**
- Final dataset saved as `dataset_part_1.csv`

Challenges encountered and solutions applied

- Handling missing values in critical columns. Replaced missing values with statistical measures like the mean.
- Ensuring consistent data formatting across multiple sources. Standardized formats and transformed columns into appropriate datatypes.

GitHub link: https://github.com/tSopermon/certificate_assignments/blob/main/applied_datascience_capstone/data-wrangling/labs-jupyter-spacex-Data%20wrangling-v2.ipynb

EDA with Data Visualization

Key Visualizations

- Outcome Trends
  - Plotted `Flightnumber` vs. `PayloadMass` with outcomes overlayed (using Matplotlib and Seaborn).
  - Highlighted trends: Increased flight numbers correlate with higher success rates; payload mass impacts landing success.
- Site and Orbit Analysis
  - Boxplots and bar charts
    - Payload vs. Launch Site
    - Orbit success rates
    - Payload vs. Orbit type
  - Scatter and line plots for trends over time.

Feature Engineering

- Dummy Variables Creation
  - Applied `OneHotEncoder` to categorical columns: `Orbit`, `LaunchSite`, `LandingPad`, `Serial`
  - Converted numeric columns to `float64`
- Saved enhanced dataset as `dataset_part_3.csv`

GitHub link: https://github.com/tSopermon/certificate_assignments/blob/main/applied_datascience_capstone/data-visualization-eda/jupyter-labs-eda-dataviz-v2.ipynb

EDA with SQL

SQL-Based Analytics:

- Loaded `dataset_part_1` into an SQLite database for queries.
- Key queries performed:
  - Unique launch sites and mission payloads.
  - Payload mass statistics for specific boosters and organizations (e.g., NASA).
  - Success and failure counts for landing outcomes.
  - Dates and details of specific events like the first successful ground landing.
  - Rankings of landing outcomes between specific dates.

GitHub link: https://github.com/tSopermon/certificate_assignments/blob/main/applied_datascience_capstone/eda-sql/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

Objects created and added

- `**folium.Map**`: Map initialization centered at NASA Johnson Space Center
- `**folium.Circle**`: Highlighted the NASA Johnson Space Center with a circle and a labeled marker
- `**folium.Marker**`: Marked each site on the map with circles and labeled markers
- `**folium.plugins.HeatMap**`: Visualized the density of launch sites on the map
- `**folium.plugins.MousePosition**`: Displaying the latitude and longitude of the mouse pointer on the map
- `**folium.PolyLine**`: Marked points of interest and draw lines to measure distances from launch sites to these points

Explanation

The circles were added to visually identify and label the locations of NASA Johnson space center and SpaceX launch sites. A heatmap was added to show the density and distribution of launch sites. Marker Cluster was added to manage multiple markers at the same coordinates, indicating the success or failure of launches. Mouse position plugin was added to facilitate the identification of coordinates for points of interests and lastly, distance markers and lines were added to measure and display distances between launch sites and nearby points of interests.

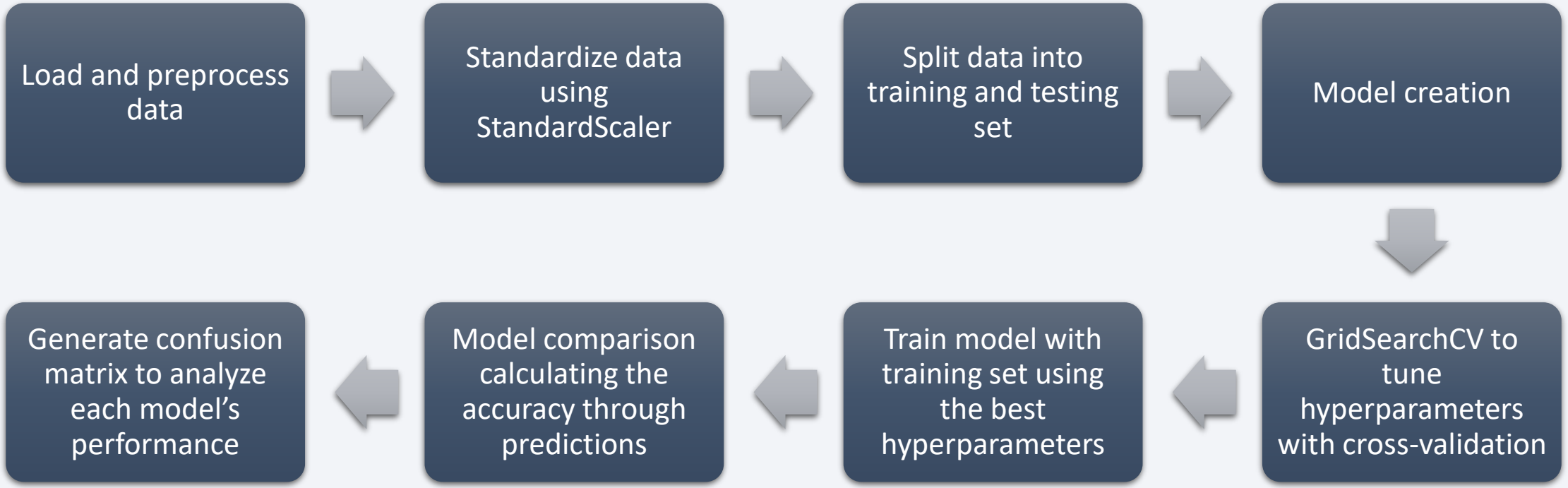
GitHub link: https://github.com/tSopermon/certificate_assignments/blob/main/applied_datascience_capstone/spacex-folium-visualization/lab-jupyter-launch-site-location-v2.ipynb

Build a Dashboard with Plotly Dash

- A **dropdown menu** for launch sites was added to provide users with the flexibility to view data for specific launch sites combined.
- **Pie chart** for launch success was added to give a visual representation of the success rate of launches at different sites.
- **Range slider** for payload mass was added to allow users to filter the data based on payload mass.
- **Scatter Plot** for payload vs. launch success was added to visualize the relationship between payload mass and launch success. This plot helps in identifying trends and correlations between the payload mass and the success of the launches,

GitHub link: https://github.com/tSopermon/certificate_assignments/blob/main/applied_datascience_capstone/spacex-dashboard/spacex_dash_app.py

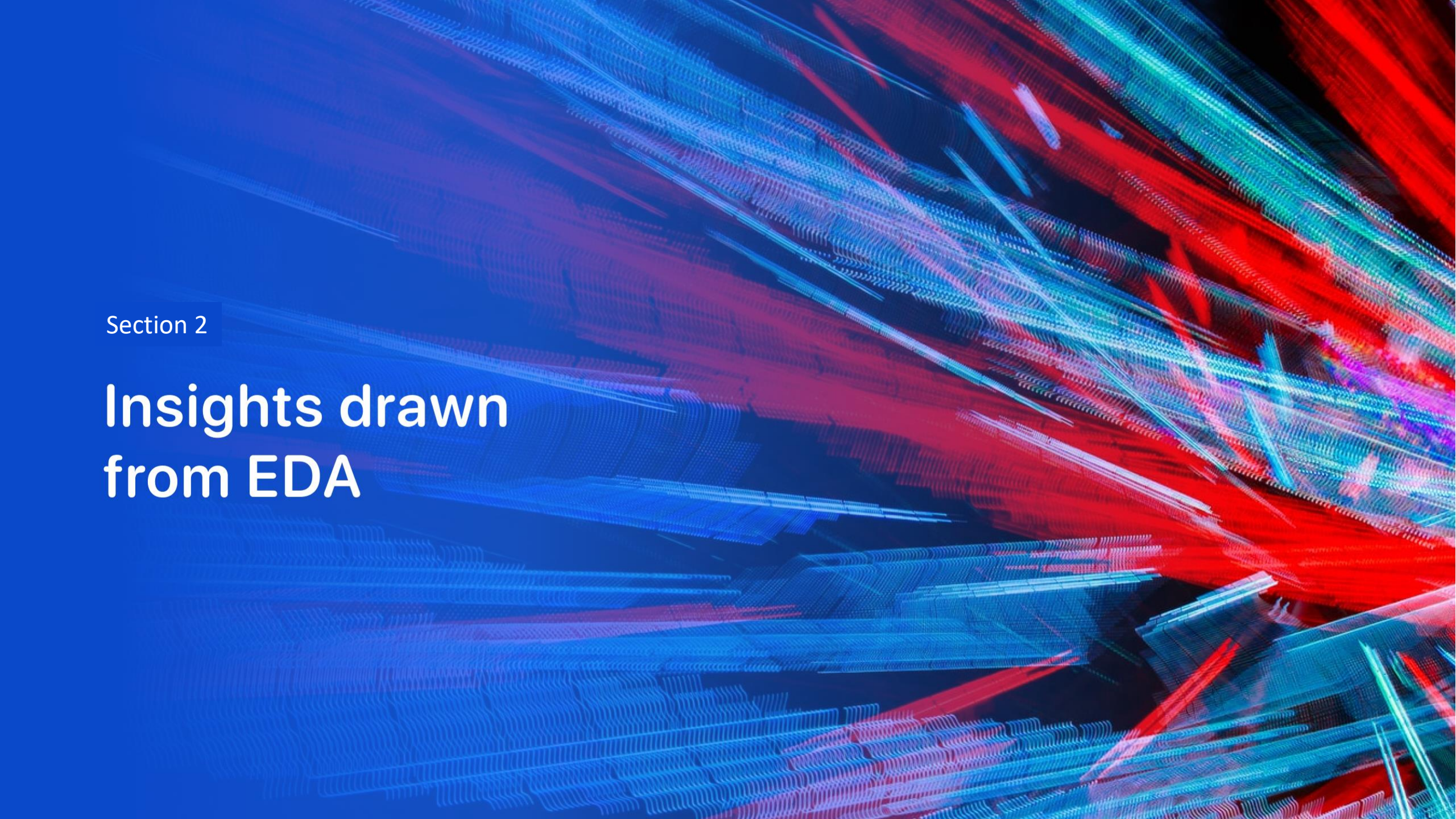
Predictive Analysis (Classification)



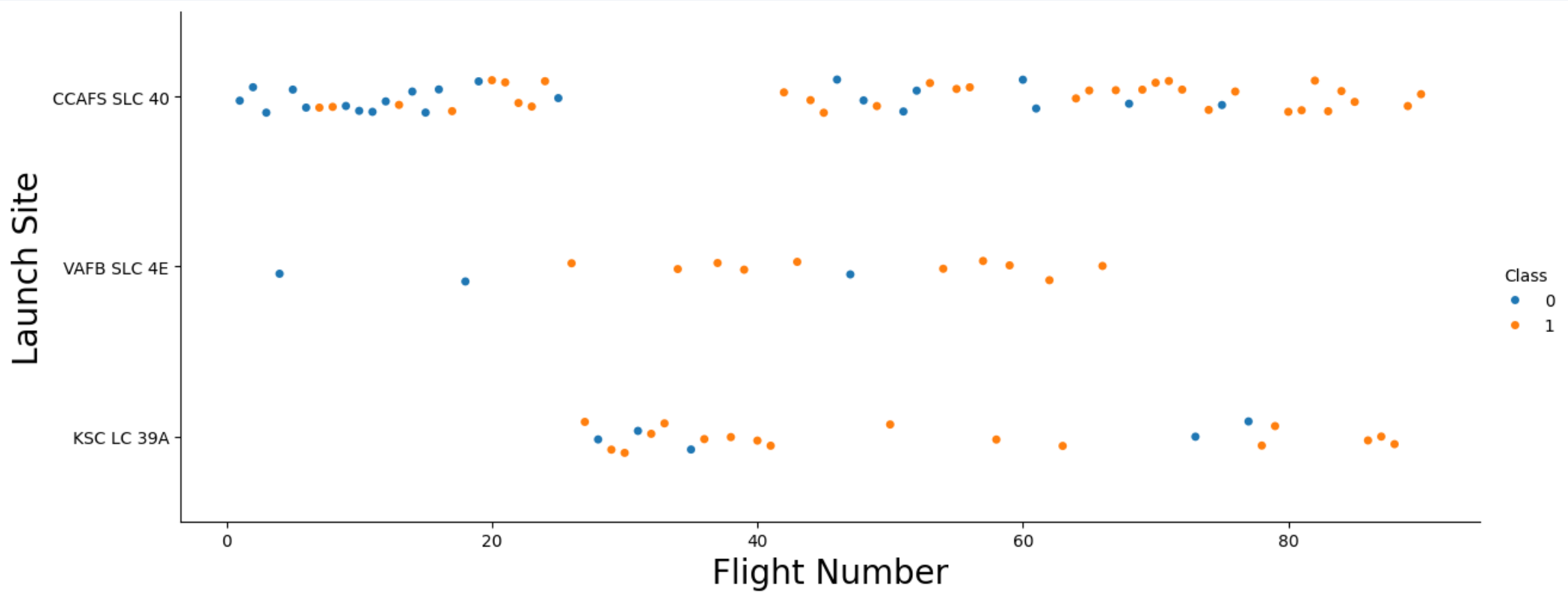
GitHub link: https://github.com/tSopermon/certificate_assignments/blob/main/applied_datascience_capstone/spacex-landing-predictions/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb

Results

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

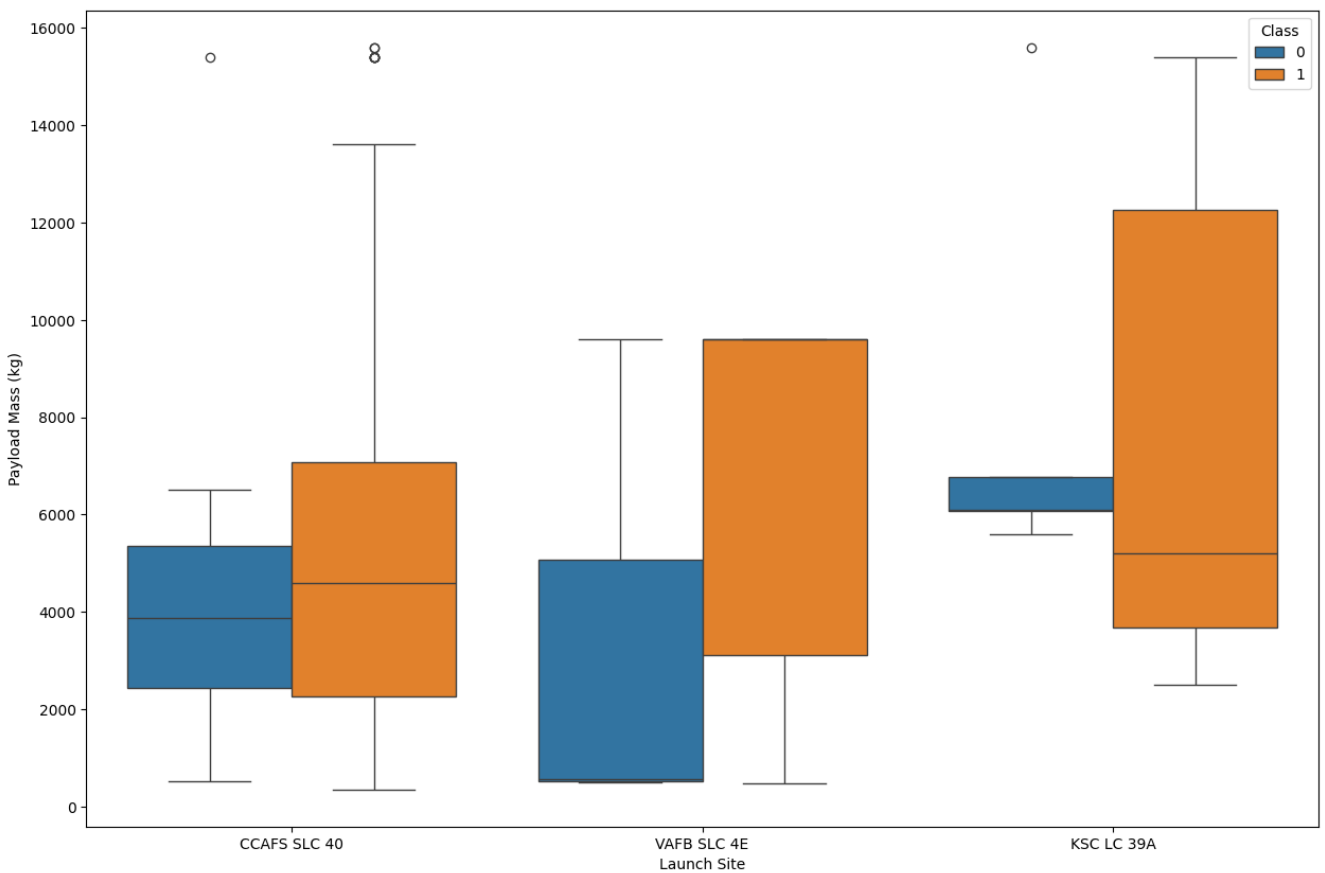

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



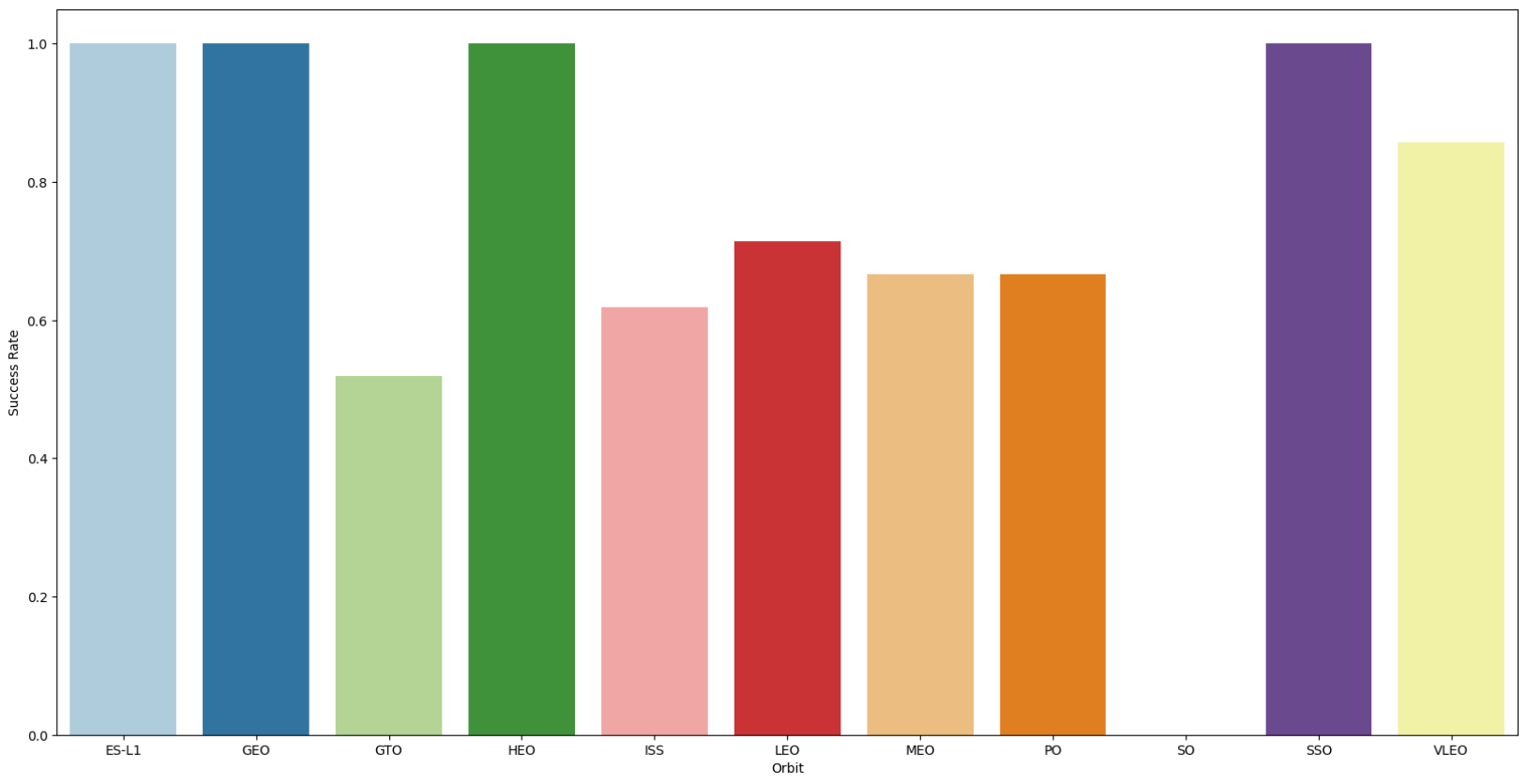
The launch site with the most rocket launches is CCAFS-SLC-40, with KSC-LC-39A and VAFB-SLC-4E following. We notice between flight numbers 25-40 that the rocket launches stopped at the CCAFS-SLC-40 and took place at KSC-LC-39A instead.

Payload vs. Launch Site



- Most launches occur at **CCAFS-SLC-40** site with distinct payload capacities.
- For **VAFB-SLC-4E** launch site, there are no rockets launched for heavy payload mass (greater than 10000kg)
- For **CCAFS-SLC-40**, most of the rockets launched had a range 2000 – 8000 kilograms of payload mass, with a few (3) heavy payload mass exceptions/outliers.
- For **KSC-LC-39A** launch site's rockets, heavier payload mass was used for their launches.

Success Rate vs. Orbit Type

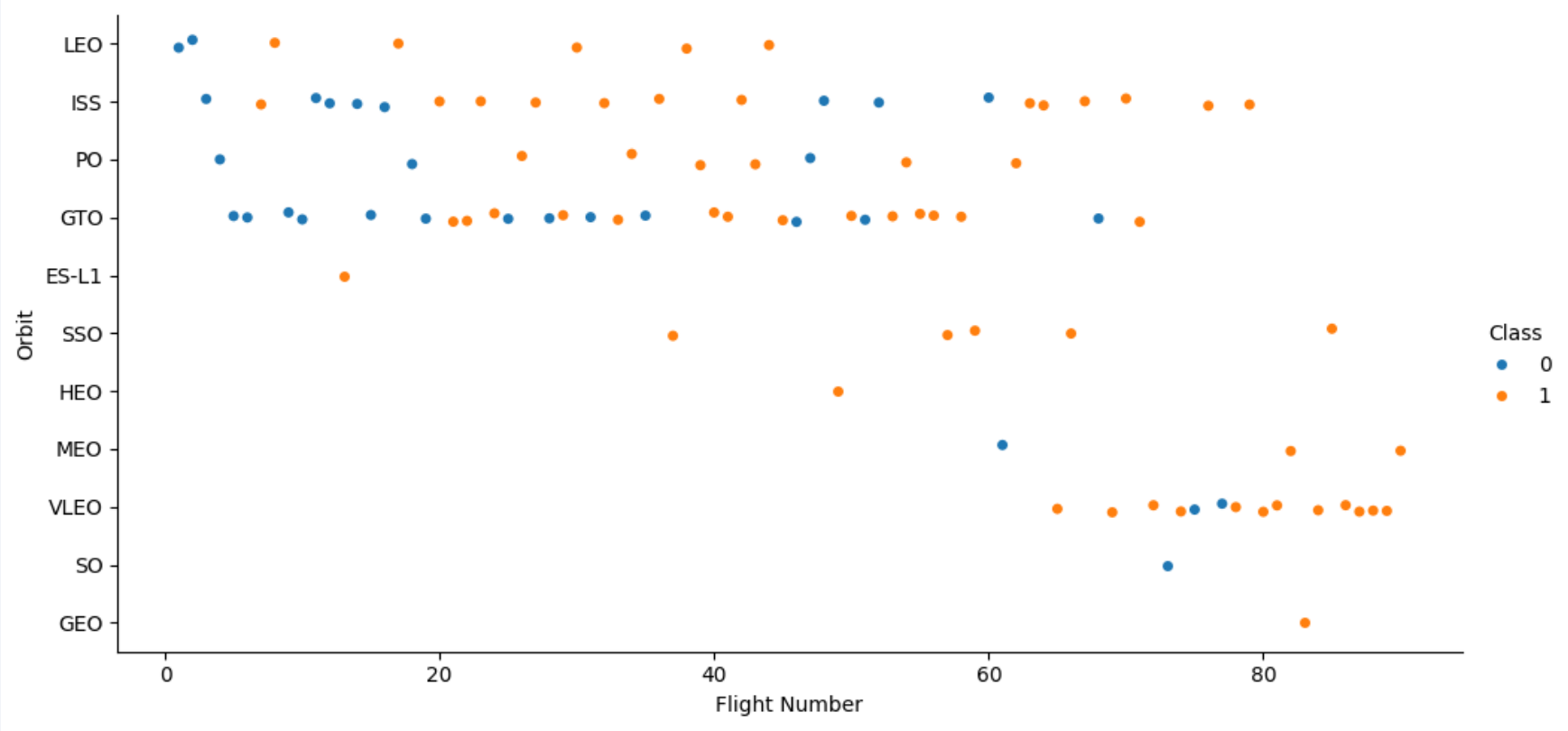


The orbits with the highest success rates are ES-L1, GEO, HEO, SSO with perfect percentage and VLEO at 85%. Orbits GTO, ISS, LEO, MEO and PO have moderate success rates around 50-70% and the orbit SO does not have any successful launches.

Analytically:

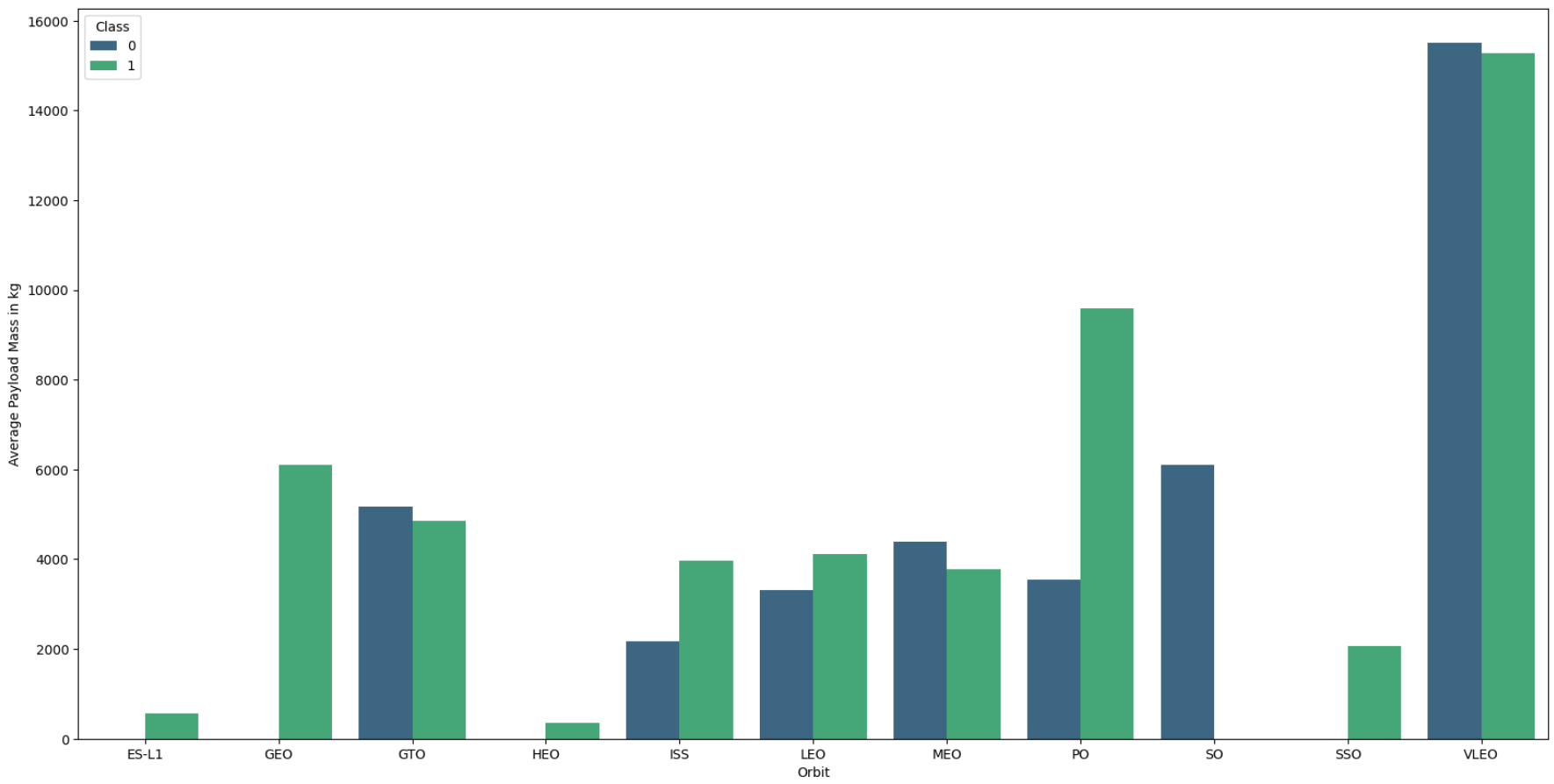
Orbit	
ES-L1	1.000000
GEO	1.000000
GTO	0.518519
HEO	1.000000
ISS	0.619048
LEO	0.714286
MEO	0.666667
PO	0.666667
SO	0.000000
SSO	1.000000
VLEO	0.857143

Flight Number vs. Orbit Type



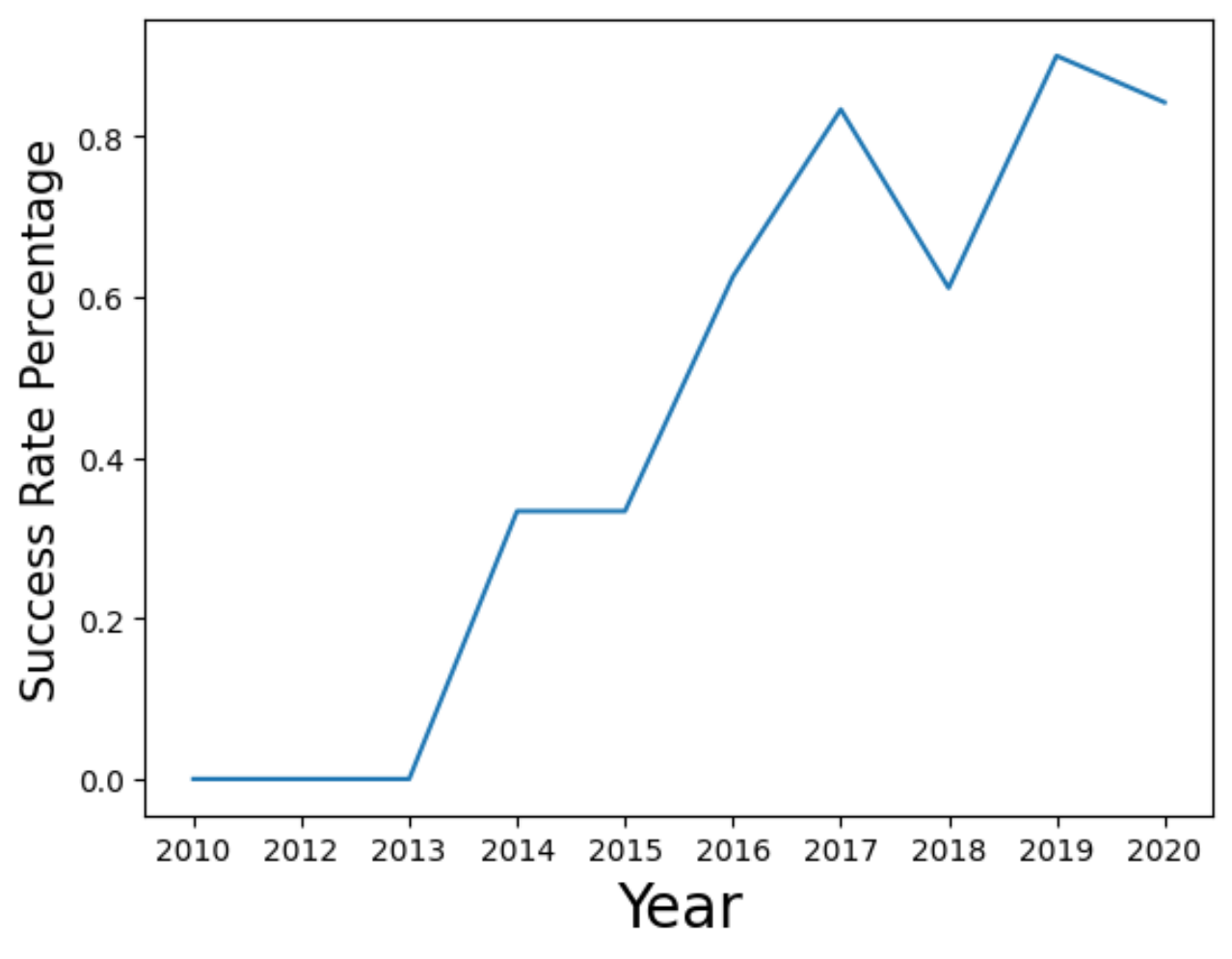
We notice at the orbits LEO, ISS, and GTO that as the flight number increases, successful launches become more frequent as well. Furthermore, some launches at specific orbits, like SO,ES-L1, HEO and GEO have one launch each, meaning that we shouldn't calculate success rates and put our trust at their percentages to generate any conclusions for these orbits.

Average Payload vs. Orbit Type



Excluding SO, ES-L1, HEO and GEO for having only one launch each, we notice at the orbits ISS, LEO and PO that the launches used heavier payload mass on average on their successful launches than on their unsuccessful.

Launch Success Yearly Trend



Steady improvement in landing technologies over the years.

Annual increase in successful missions highlighted advancements in SpaceX's capabilities.

All Launch Site Names

```
cur.execute('SELECT DISTINCT "Launch_site" FROM SPACEXTABLE')
results = cur.fetchall()
for row in results:
    print(row[0])
```

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

The first line of code executes a SQL query using the `execute` method of the cursor object `cur`. The SQL query `SELECT DISTINCT “Launch_site” FROM SPACEXTABLE` is designed to select unique values from the “Launch_site” column in the table named “SPACEXTABLE”.
The second line fetches all the rows returned by the executed query and stores them in the variable `results`.
The final part of the code is a for loop that iterates over each row in the `results` list. For each row, prints the first element of the tuple.

Launch Site Names Begin with 'CCA'

```
cur.execute("SELECT * FROM SPACEXTABLE WHERE Launch_site LIKE 'CCA%' LIMIT 5" )
results = cur.fetchall()
for row in results:
    print(row)
```

The SQL query `SELECT * FROM SPACEXTABLE WHERE Launch_site LIKE 'CCA%' LIMIT 5` is designed to select all columns (*) from the table named "SPACEXTABLE" where the "Launch_site" column's value starts with the string 'CCA'.

('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

```
cur.execute("SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer LIKE 'NASA (CRS)'")
results = cur.fetchall()
for row in results:
    print(row[0])
```

45596

The SQL query `SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer LIKE 'NASA (CRS)'` is designed to calculate the total payload mass from the "PAYLOAD_MASS__KG_" column in the table named "SPACEXTABLE" for rows where the "Customer" column's value matches 'NASA (CRS)'.

Average Payload Mass by F9 v1.1

```
cur.execute("SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1'")
results = cur.fetchall()
for row in results:
    print(row[0])
```

2928.4

The SQL query `SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1'` is designed to calculate the average payload mass from the "PAYLOAD_MASS__KG_" column in the table named "SPACEXTABLE" for rows where the "Booster_Version" column's value matches 'F9 v1.1'.

First Successful Ground Landing Date

```
cur.execute("SELECT DATE FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success (ground pad)' ORDER BY DATE ASC LIMIT 1")
results = cur.fetchall()
for row in results:
    print(row[0])
```

2015-12-22

The SQL query `SELECT DATE FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success (ground pad)' ORDER BY DATE ASC LIMIT 1` is designed to select the "DATE" column from the table named "SPACEXTABLE" for rows where the "Landing_Outcome" column's value matches 'Success (ground pad)'.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
cur.execute("SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE (Landing_Outcome LIKE 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)")
results = cur.fetchall()
for row in results:
    print(row[0])
```

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

The SQL query `SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE (Landing_Outcome LIKE 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)` is designed to select unique values from the "Booster_Version" column in the table named "SPACEXTABLE".

Total Number of Successful and Failure Mission Outcomes

```
cur.execute("SELECT Mission_Outcome, COUNT(Mission_Outcome) FROM SPACEXTABLE GROUP BY Mission_Outcome")
results = cur.fetchall()
for row in results:
    print(row)
```

```
('Failure (in flight)', 1)
('Success', 98)
('Success ', 1)
('Success (payload status unclear)', 1)
```

The SQL query `SELECT Mission_Outcome, COUNT(Mission_Outcome) FROM SPACEXTABLE GROUP BY Mission_Outcome` is designed to select each unique value in the "Mission_Outcome" column from the table named "SPACEXTABLE" and count how many times each value appears.

Boosters Carried Maximum Payload

```
cur.execute("SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)")
results = cur.fetchall()
for row in results:
    print(row)
```

```
('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 `SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)` is designed to select unique values from the "Booster_Version" column in the table named "SPACEXTABLE".

2015 Launch Records

```
cur.execute("""
    SELECT
        CASE strftime('%m', DATE)
            WHEN '01' THEN 'January'
            WHEN '02' THEN 'February'
            WHEN '03' THEN 'March'
            WHEN '04' THEN 'April'
            WHEN '05' THEN 'May'
            WHEN '06' THEN 'June'
            WHEN '07' THEN 'July'
            WHEN '08' THEN 'August'
            WHEN '09' THEN 'September'
            WHEN '10' THEN 'October'
            WHEN '11' THEN 'November'
            WHEN '12' THEN 'December'
        END AS month_name,
        Landing_Outcome,
        Booster_Version,
        Launch_site
    FROM SPACEXTABLE
    WHERE
        Landing_Outcome = 'Failure (drone ship)' AND
        strftime('%Y', DATE) = '2015'
""")
results = cur.fetchall()
for row in results:
    print(row)
```

The SQL query is designed to select several columns from the table named "SPACEXTABLE".
The SELECT clause includes a CASE statement that converts the month part of the "DATE" column into its corresponding month name. The strftime('%m', DATE) function extracts the month as a two-digit string from the "DATE" column. The CASE statement then maps each month number to its respective month name (e.g., '01' to 'January', '02' to 'February', etc.). The result is aliased as month_name. Additionally, the query selects the "Landing_Outcome", "Booster_Version", and "Launch_site" columns.

('January', 'Failure (drone ship)', 'F9 v1.1 B1012', 'CCAFS LC-40')
('April', 'Failure (drone ship)', 'F9 v1.1 B1015', 'CCAFS LC-40')

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

```
cur.execute("""
    SELECT
        Landing_Outcome,
        COUNT(Landing_Outcome) AS outcome_count
    FROM SPACEXTABLE
    WHERE
        DATE BETWEEN '2010-06-04' AND '2017-03-20'
    GROUP BY
        Landing_Outcome
    ORDER BY
        outcome_count DESC;
            """)
results = cur.fetchall()
for row in results:
    print(row)
```

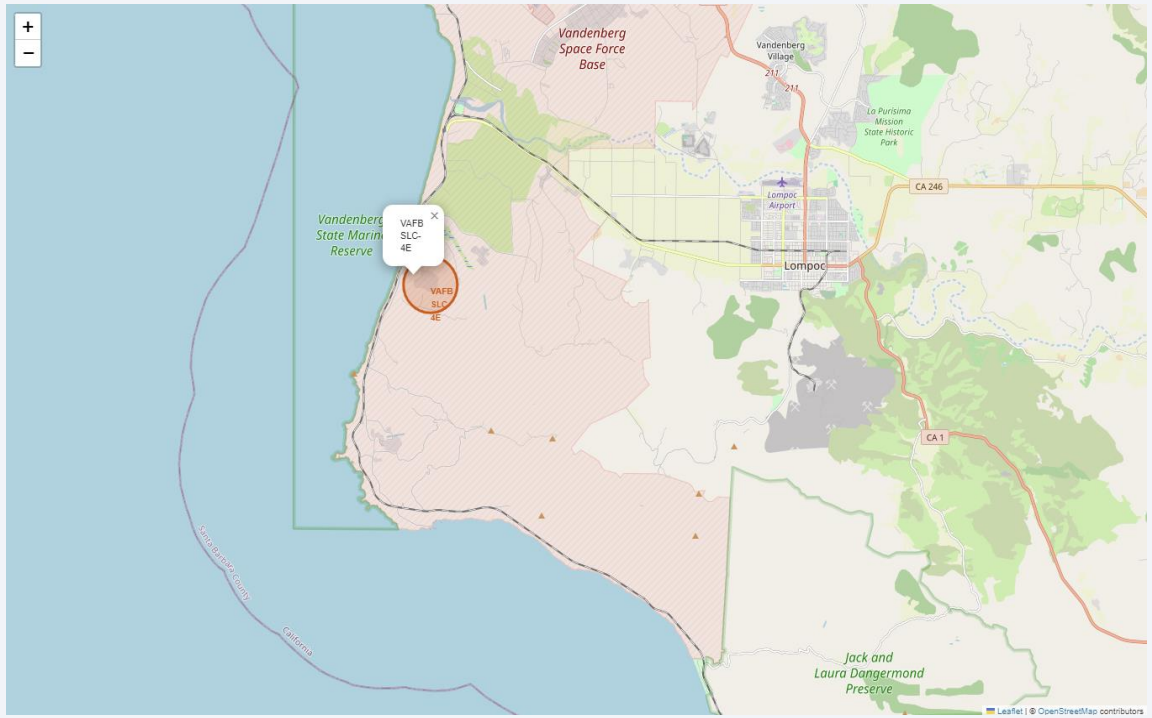
```
('No attempt', 10)
('Success (drone ship)', 5)
('Failure (drone ship)', 5)
('Success (ground pad)', 3)
('Controlled (ocean)', 3)
('Uncontrolled (ocean)', 2)
('Failure (parachute)', 2)
('Precluded (drone ship)', 1)
```

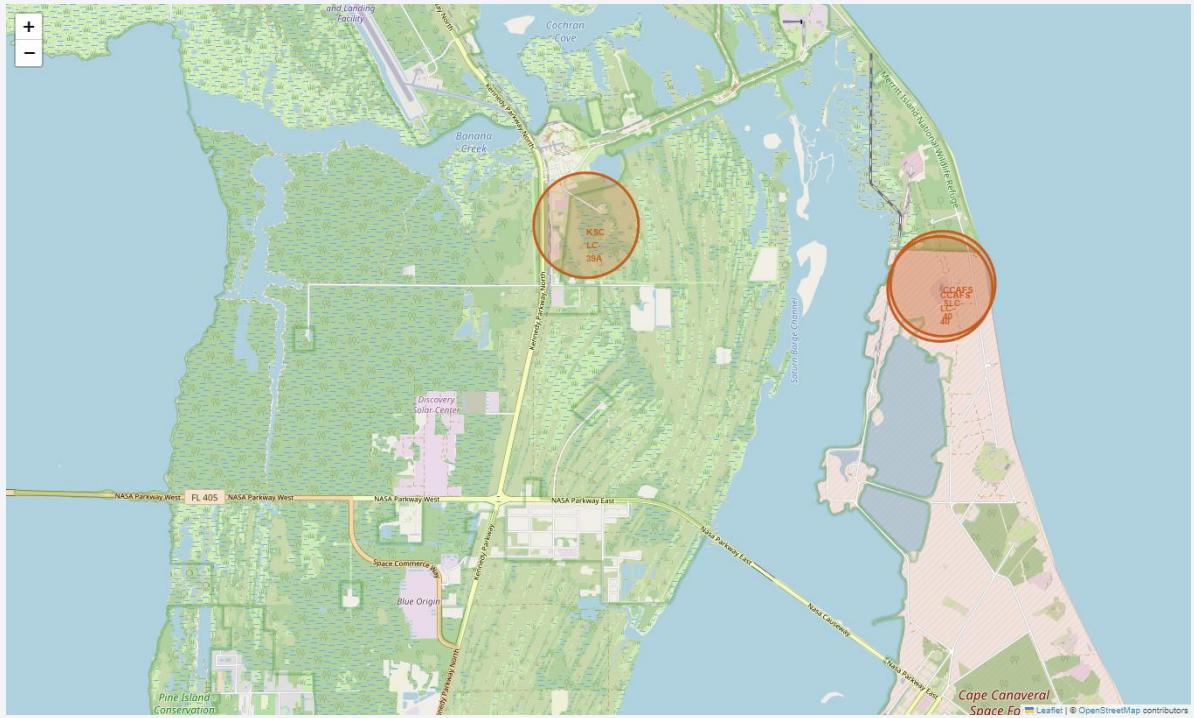
The SQL query is designed to select the "Landing_Outcome" column and count the number of times each unique landing outcome appears within a specified date range.
The SELECT clause includes the "Landing_Outcome" column and uses the `COUNT(Landing_Outcome)` function to count the occurrences of each landing outcome. The result of this count is aliased as outcome_count.
The `FROM SPACEXTABLE` clause specifies the table from which to retrieve the data. The WHERE clause filters the rows to include only those where the "DATE" column falls between '2010-06-04' and '2017-03-20'. The `GROUP BY Landing_Outcome` clause groups the results by the unique values in the "Landing_Outcome" column, ensuring that the count is calculated for each distinct landing outcome. The `ORDER BY outcome_count DESC` clause sorts the results in descending order based on the count of each landing outcome, so the most frequent outcomes appear first.

Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>



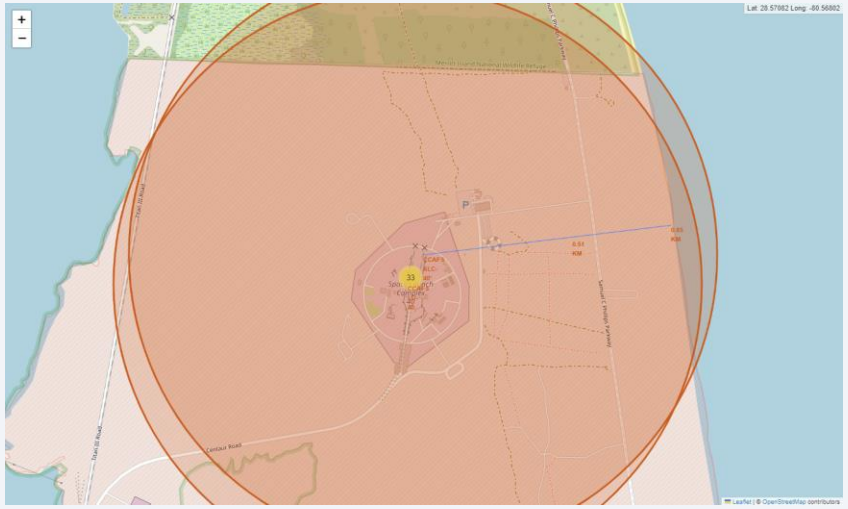


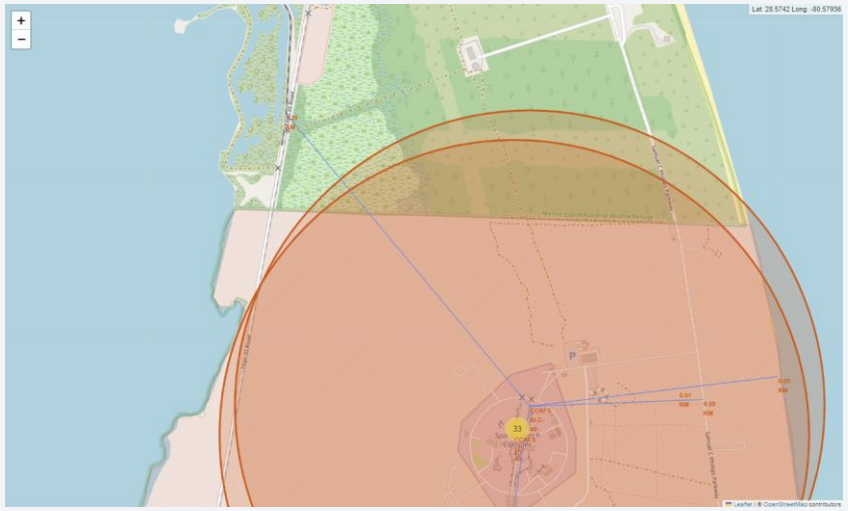
The left image shows the folium map, the circle indicating the region of the VAFB SLC-4E launch site and its label.
The right image shows the folium map, the circles indicating the regions of the KSC LC-39A, CCAFS SLC-40 and CCAFS LC-40 launch sites and their corresponding labels.

<Folium Map Screenshot 2>

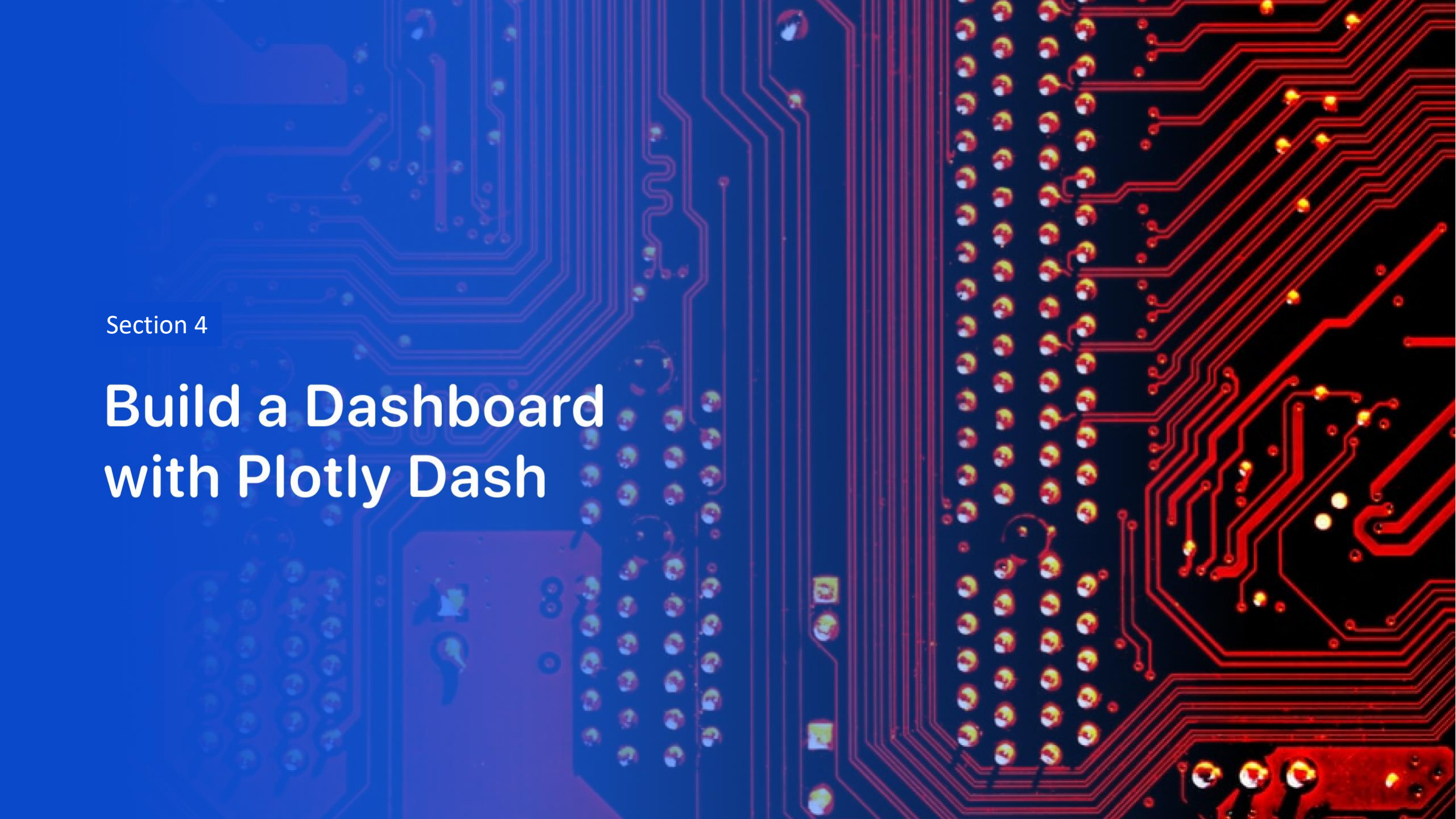
The code then iterates over each row in the `spacex_df` DataFrame, which contains data about SpaceX launches, including their coordinates and a color indicator for the launch outcome. For each row, a `Marker` object is created using the latitude (Lat) and longitude (Long) values from the DataFrame. The marker's icon is customized using the `folium.Icon` class, where the `color` is set to 'white' and the `icon_color` is dynamically set based on the `marker_color` value from the DataFrame. This customization visually indicates whether the launch was successful or failed.

<Folium Map Screenshot 3>

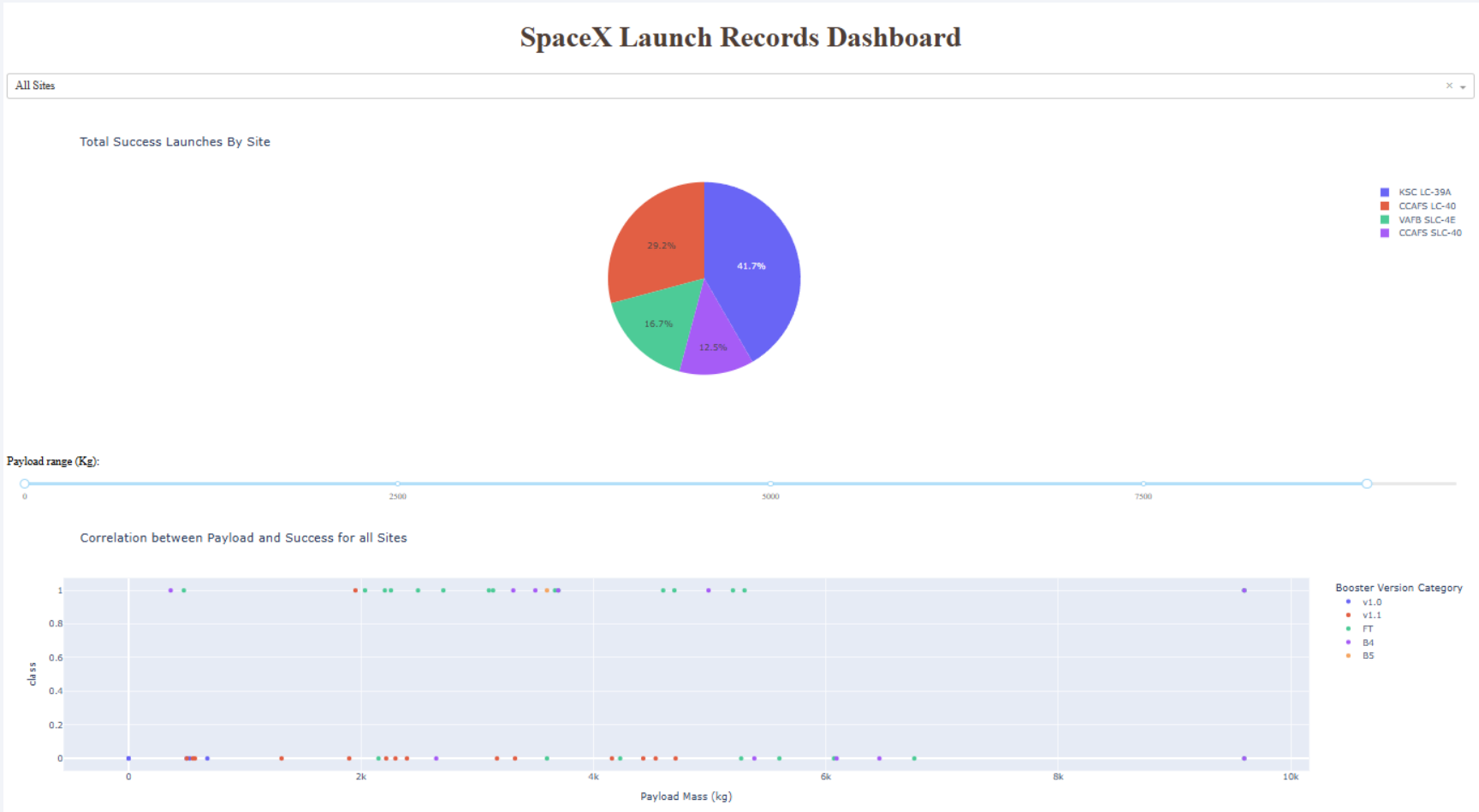




The folium map displays the distance between the launch site and the closest point of the coastline to it.
A folium marker was used to display the calculated distance.
To visually connect the launch site and the coastline, folium.Polyline is drawn between the two points. This line is added to the map, providing a clear visual representation of the distance.

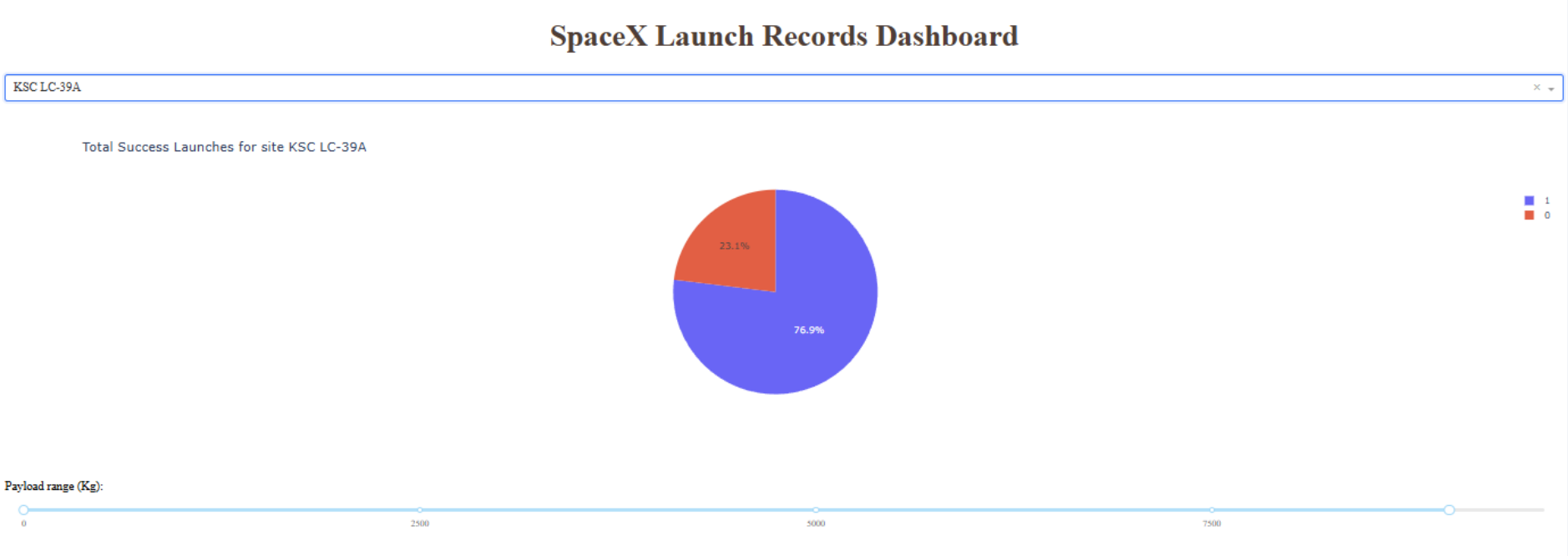

Section 4
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>



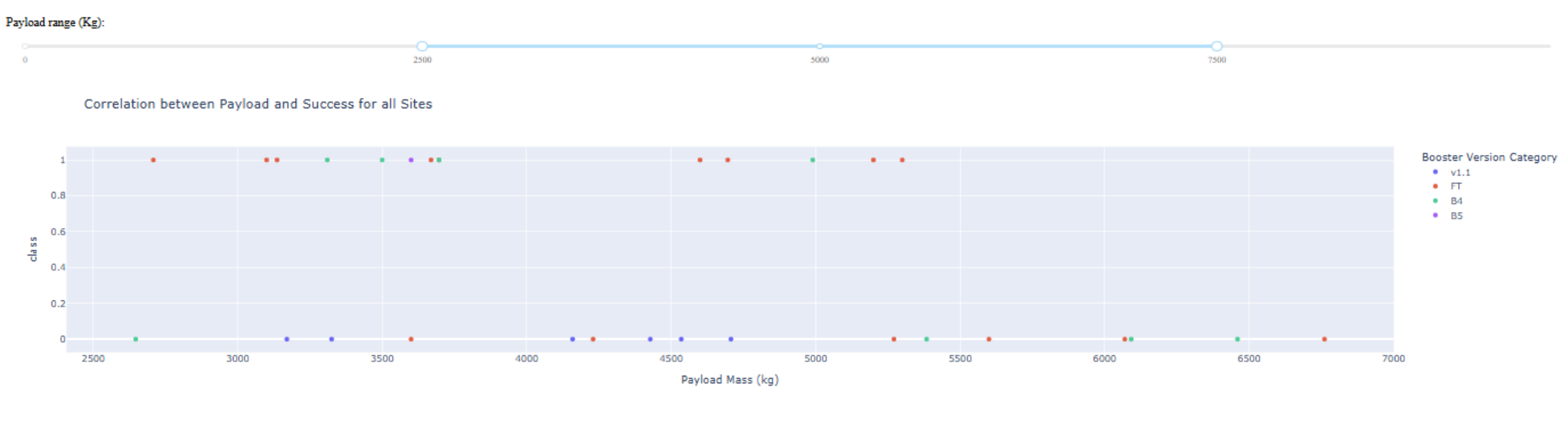
Here's the SpaceX launch records dashboard, created for the completion of the course's interactive visualization lab.
In this screenshot, the dashboard shows in a pie chart the success rates of launches in each launch site and the corresponding scatter plot displaying the payload mass used for every launch, separated by the mission's success or failure.

<Dashboard Screenshot 2>



In this screenshot, the displayed pie chart shows the launch site with the highest success ratio in its rocket launches (**KSC LC-39A**), comparatively with the other launch sites that recorded lower success ratio.

<Dashboard Screenshot 3>

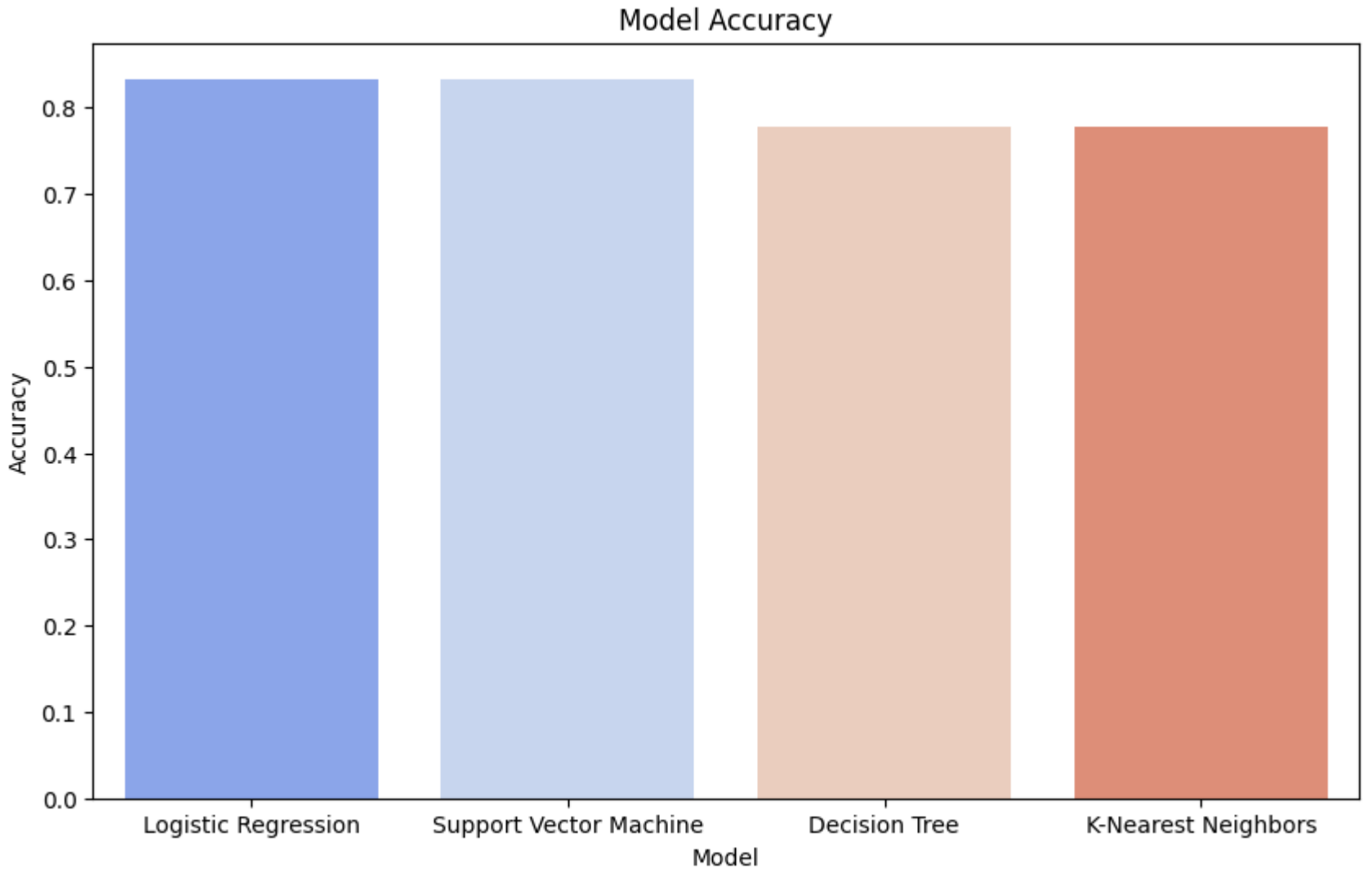


In this screenshot, the displayed scatter plot shows the payload mass (in kg) for every launch site that took place on each launch site, modified by setting the payload range from 2500kg to 7500kg.

Section 5

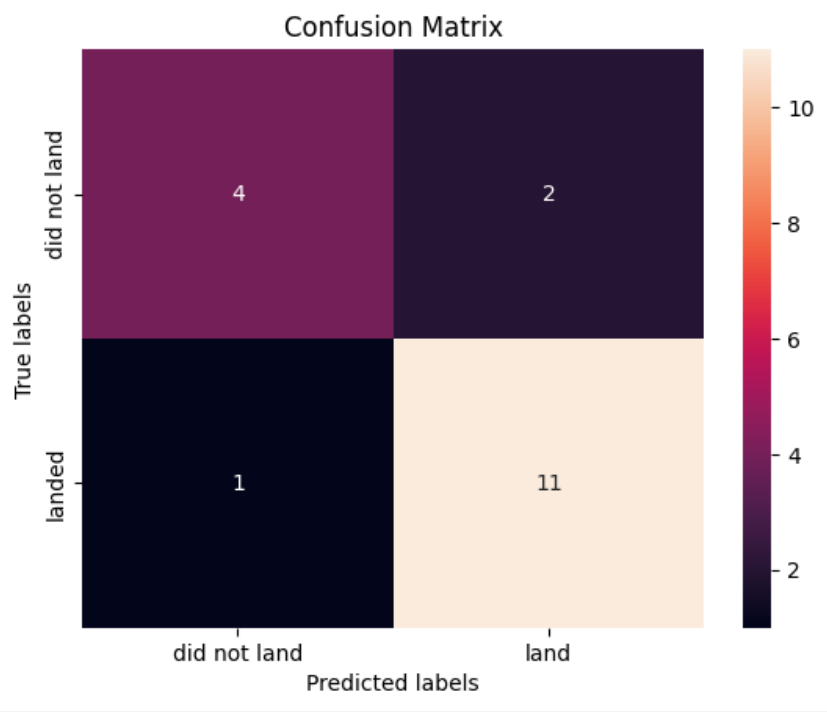
Predictive Analysis (Classification)

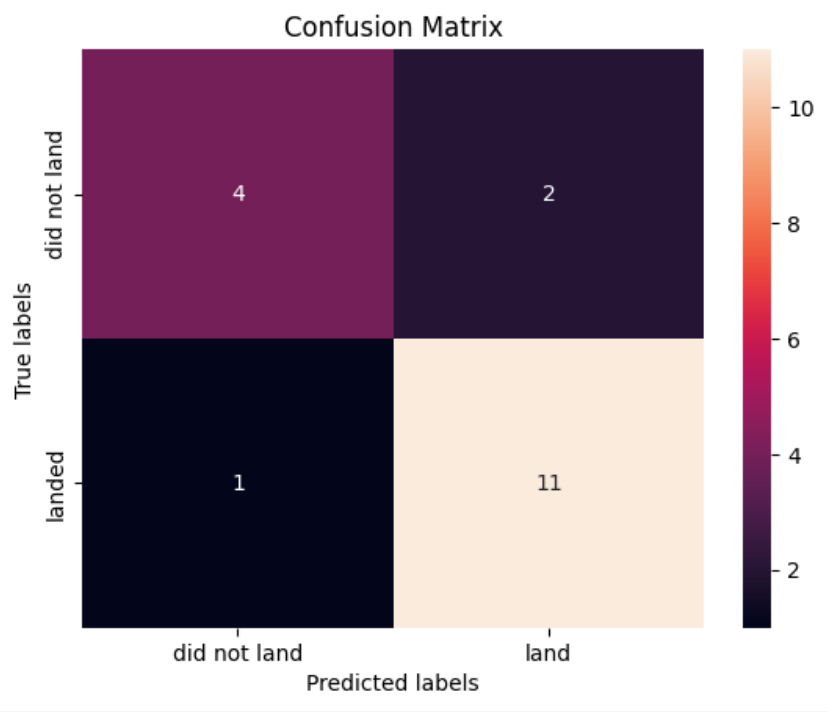
Classification Accuracy



This bar chart is designed to show each model's predictive efficiency calculated by the accuracy. It is shown that the Logistic Regression and the SVM models performed better than Decision Tree and KNN models, reaching the accuracy of 83.3%.

Confusion Matrix





The left picture is the confusion matrix of the Logistic Regression model and the right the SVM's.

Both models performed well on their predictions and considering their confusion matrixes, both classified 11 true landings and 4 true unsuccessful landings, with only 3 of the predictions being falsely classified, 2 on the unsuccessful and only one on the successful. That shows the model's greater efficiency on larger classes like successful landings.

Conclusions

- As shown in the displayed visualizations, the payload mass played a critical role on the mission's success ratio showing an ideal range of 4000-10000kg of mass carried inside the rockets.
- Furthermore, great progress was made as more launches took place realizing technological development through out the years.
- Most of the launches took place to the CCAFS-SLC-40 launch site, noticing a preference because of measuring smaller distances to reach points of interests like the coastline and nearby roads.
- On the other hand, the launch site with the most total successful launches is KSC LC-39A, offering a great insight for the development of the other sites.
- Machine learning models, like Logistic Regression and Support Vector Machine, achieved high accuracy in forecasting future launches and cost estimations, demonstrating the potential in the aerospace industry.

Appendix

Credits & Acknowledgements

- Applied Data Science Capstone: Instructors
  - Yan Luo
    Ph.D., Data Scientist and Developer
  - Joseph Santarcangelo
    Ph.D., Data Scientist at IBM

GitHub Profile: https://github.com/tSopermon

Thank you!