IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Zaariq Baijnath
15/02/2024

Outline

- Executive Summary
- Introduction
- Methodology
- Results:
  - Exploratory Data Analysis (EDA) with Visualisation
  - EDA with SQL
  - Interactive Maps with Folium
  - Plotly Dash Dashboard
  - Predictive Analytics
- Discussion
- Conclusion
- Appendix

Executive Summary

- **Our research focuses on understanding the factors influencing successful rocket landings, particularly the reuse of the first stage, which significantly impacts launch costs.**
- In order to investigate this, we employed a comprehensive methodology encompassing data collection, wrangling, exploration, analysis, and predictive modelling. Utilizing SpaceX's REST API and web scraping techniques, we gathered extensive data on launch parameters, including payload, launch site, flight number, and temporal trends.
- Machine learning techniques were utilised to predict the outcome of Falcon 9 first stage landings based on various features. We performed exploratory data analysis (EDA) to understand the dataset, created a classification model pipeline, and evaluated several algorithms, including Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K Nearest Neighbors (KNN).
- Our analysis revealed that all models achieved an accuracy of approximately 83.33% on the test data. Logistic Regression and SVM outperformed the other models slightly, showcasing their potential for accurate prediction of first stage landings.
- Overall, our machine learning pipeline provides a reliable method for predicting Falcon 9 first stage landing outcomes, enabling informed decision-making in the space launch industry and facilitating cost-effective bidding strategies for space missions. Further enhancements and refinements to the model could lead to even higher prediction accuracy and broader applications in the aerospace sector.

Introduction

This capstone project focuses on predicting SpaceX's Falcon 9 first-stage reuse. Through this investigation, we'll empower Space Y, a new player in the market, to make informed pricing decisions.

Background:

- SpaceX dominates, offering launches for $62 million compared to competitors' $165 million+ prices. This is achieved by reusing the Falcon 9 first stage.
- Predicting reuse unlocks the key to price matching or undercutting SpaceX.
- Public data, exploratory data analysis and machine learning will be utilised to determine accurate predictions.
- Through comprehensive data analysis and the development of intuitive dashboards, we aim to provide actionable insights that will drive Space Y's competitiveness in the commercial space market.

Section 1

Methodology

Methodology

Methodology Summary:

1. Data collection methodology:
2. Perform data wrangling
3. Perform exploratory data analysis (EDA) using visualization and SQL
4. Perform interactive visual analytics using Folium and Plotly Dash
5. Perform predictive analysis using classification models

Data Collection

Data Preprocessing:

- Transform the scraped data into a structured dataframe using Pandas.
- Clean and preprocess the data to ensure consistency and accuracy.

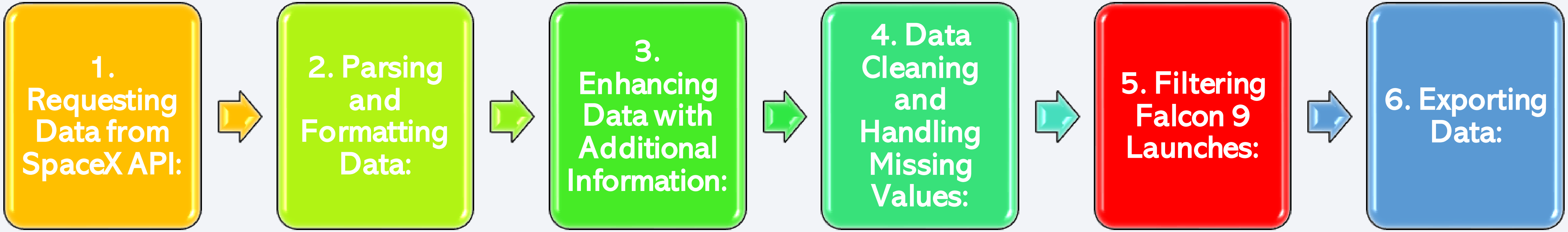

Figure 1: Flowchart representing the process steps for data collection.

Data Collection – SpaceX API

Methodology for Data Collection:

1. Requesting Data from SpaceX API:

- Utilize the requests library in Python to make HTTP GET requests to the SpaceX API endpoint (https://api.spacexdata.com/v4/launches/past).
- Retrieve JSON data containing information about past SpaceX launches, including details about rockets, payloads, launchpads, and cores.
- Ensure that the request is successful by checking the HTTP response status code (200).

2. Parsing and Formatting Data:

- Use the json_normalize() function to convert the JSON response into a structured Pandas DataFrame.
- Select relevant columns such as rocket ID, payload IDs, launchpad ID, core ID, flight number, and launch date.
- Filter out rows with multiple cores or payloads, as those are not relevant for this analysis.

3. Enhancing Data with Additional Information:

- Utilize the IDs obtained from the initial data to make additional API requests to retrieve more specific information about each launch component.
- Extract information such as booster version, payload mass, orbit, launch site details (name, longitude, and latitude), core details (landing outcome, number of flights, reuse status, etc.).

Data Collection – SpaceX API

Methodology for Data Collection:

4. Data Cleaning and Handling Missing Values:

- Address missing values in the dataset by calculating the mean payload mass and replacing NaN values with this mean.
- Ensure that the LandingPad column retains None values to represent cases where landing pads were not used.

5. Filtering Falcon 9 Launches:

- Filter the dataset to only include Falcon 9 launches by selecting rows where the BoosterVersion column equals "Falcon 9".
- Reset the FlightNumber column to ensure consistency and sequential numbering of Falcon 9 launches.

6. Exporting Data:

- Export the cleaned and filtered dataset to a CSV file for further analysis and modeling in subsequent tasks.

Data Collection - Scraping

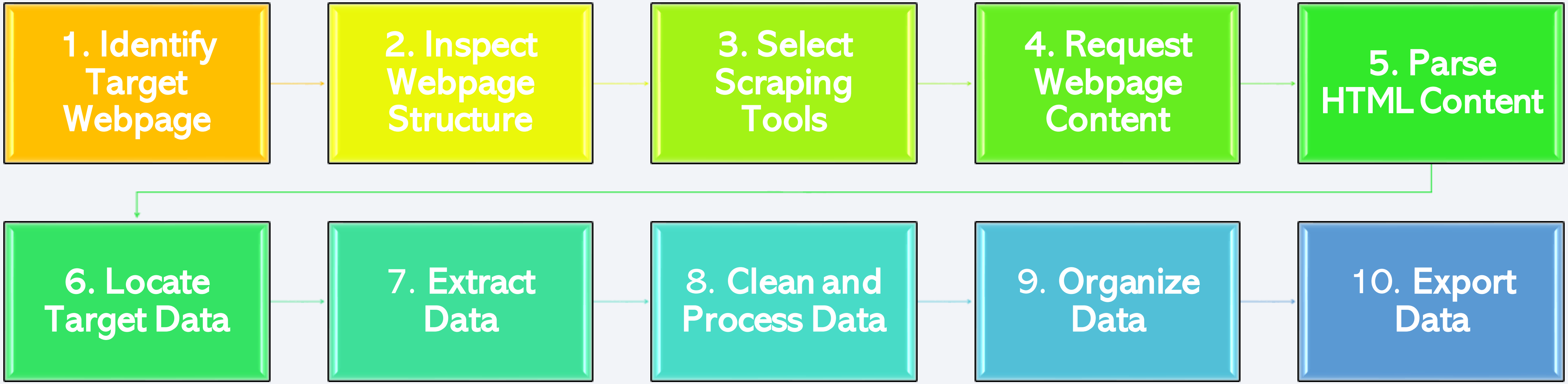

Figure 2: Flowchart representing the process steps for web scraping.

Data Collection - Scraping

Methodology for Web Scraping:

1. **Identify Target Webpage:**
   - Determine the webpage containing the desired information for scraping. In this case, the target webpage is the Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches."
2. **Inspect Webpage Structure:**
   - Analyze the structure of the target webpage to identify HTML elements containing the relevant data. This involves examining the HTML source code and understanding the organization of tables, headers, and rows.
3. **Select Scraping Tools:**
   - Choose appropriate tools for web scraping. For this task, we utilize Python libraries such as **requests** for sending HTTP requests to retrieve webpage content and **BeautifulSoup** for parsing HTML and navigating the DOM (Document Object Model) structure.
4. **Request Webpage Content:**
   - Use the **requests.get()** function to send an HTTP GET request to the URL of the target webpage. This retrieves the HTML content of the webpage, which can then be parsed.
5. **Parse HTML Content:**
   - Create a **BeautifulSoup** object by passing the HTML content obtained from the HTTP response. This allows for easy navigation and extraction of specific elements from the HTML document.

Data Collection - Scraping

Methodology for Web Scraping:

6.Locate Target Data:

- Identify the HTML elements (e.g., tables, rows, cells) that contain the desired data. This may involve inspecting the webpage source code, looking for unique identifiers, classes, or patterns that can be used to locate the relevant elements.

7.Extract Data:

- Use BeautifulSoup methods such as **find()** or **find_all()** to locate specific HTML elements containing the target data. Extract the data by accessing the text, attributes, or other properties of these elements.

8.Clean and Process Data:

- Clean the extracted data by removing any unwanted characters, formatting inconsistencies, or noise. This may involve string manipulation, regular expressions, or other data cleaning techniques.

9.Organize Data:

- Organize the extracted data into a structured format suitable for further analysis or storage. This typically involves storing the data in data structures such as lists, dictionaries, or Pandas DataFrames.

10.Export Data:

- Save the organized data to a file or database for future use. Common formats for exporting data include CSV, JSON, Excel, or SQL databases.

Data Wrangling

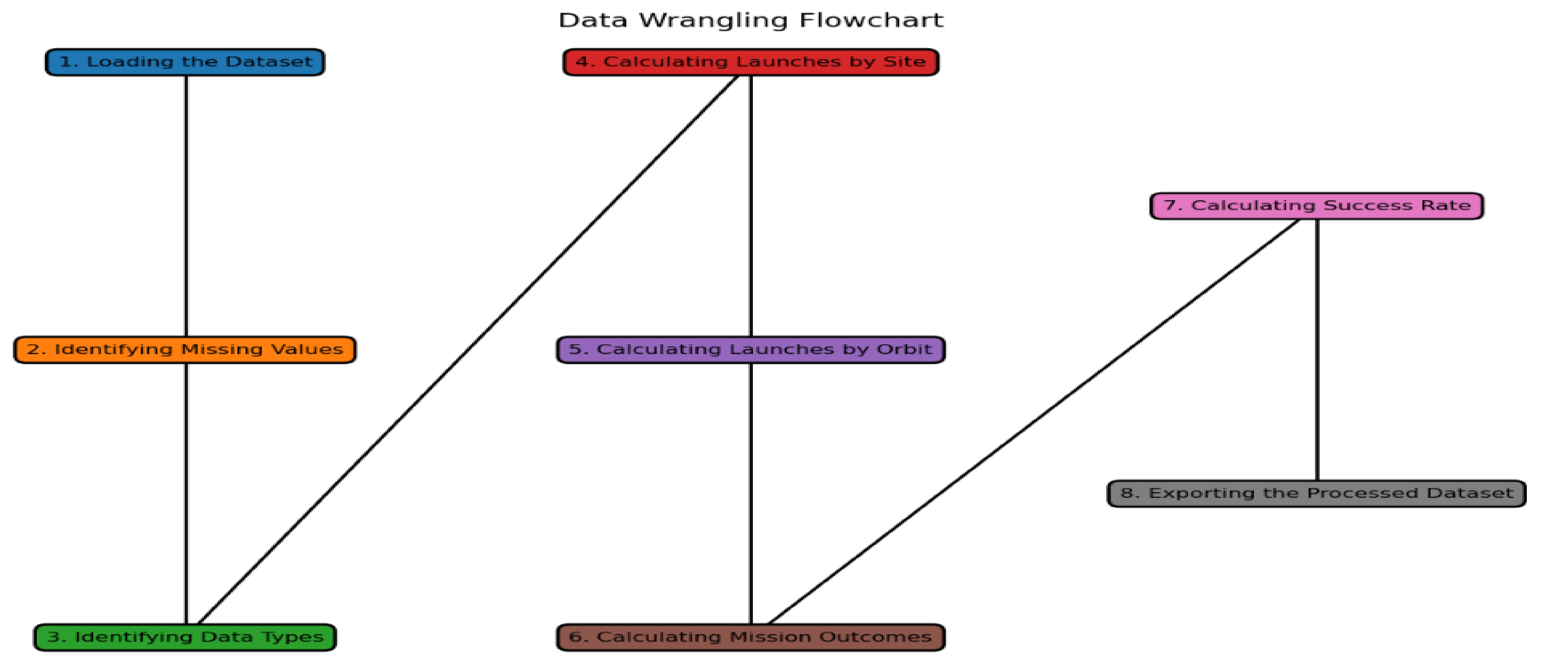

Figure 3: Flowchart representing the process steps for Data Wrangling.

Data Wrangling

Methodology for Data Wrangling:

1. **Loading the Dataset:**
   - Load the dataset containing SpaceX launch records obtained from the previous data collection phase.
   - Use pandas **read_csv()** function to read the dataset into a DataFrame.
2. **Identifying Missing Values:**
   - Use the **isnull()** method along with **sum()** to identify the number of missing values in each column.
   - Calculate the percentage of missing values for each attribute by dividing the count of missing values by the total number of records and multiplying by 100.
3. **Identifying Data Types:**
   - Use the **dtypes** attribute to identify the data type of each column in the DataFrame.
   - This step helps distinguish between numerical and categorical variables, which is crucial for subsequent analysis.
4. **Calculating Launches by Site:**
   - Use the **value_counts()** method on the 'LaunchSite' column to determine the number of launches from each launch site.
   - This step provides insights into the distribution of launches across different SpaceX launch facilities.

Data Wrangling

Methodology for Data Wrangling:

5.Calculating Launches by Orbit:
- •Use the **value_counts()** method on the 'Orbit' column to determine the number of launches targeting each orbit.
- •This analysis reveals the distribution of launches across different orbital destinations.

6.Calculating Mission Outcomes:
- •Use the **value_counts()** method on the 'Outcome' column to determine the frequency of different mission outcomes.
- •Categorize the outcomes into successful and unsuccessful based on predefined criteria (e.g., whether the booster landed successfully).
- •Extract the classification variable that represents the outcome of each launch, where 0 indicates an unsuccessful landing and 1 indicates a successful landing.
- •Create a new column ('Class') in the DataFrame to store the landing outcome labels.

7.Calculating Success Rate:
- •Calculate the success rate by taking the mean of the 'Class' column, where a value of 1 represents a successful landing.
- •The success rate provides an overall measure of the proportion of successful landings among all recorded launches.

8.Exporting the Processed Dataset:
- •Save the modified DataFrame, including the newly created 'Class' column, to a CSV file for further analysis and modeling in subsequent tasks.
- •Use the **to_csv()** method to export the DataFrame to a CSV file, ensuring that the index is not included in the exported file.

EDA with Data Visualization

- **Methodology for Exploratory Data Analysis (EDA)**
- Exploratory Data Analysis (EDA) is a crucial step in understanding the dataset and uncovering patterns, trends, and relationships within the data. The methodology for EDA involves several steps:

1. **Calculating Launches by Site**: Determine the number of launches that have occurred at each launch site to understand the distribution of launches.
2. **Calculating Launches by Orbit**: Analyze the distribution of launches across different orbit types to identify any patterns or trends.
3. **Calculating Mission Outcomes**: Determine the success rate of missions for each orbit type to understand the overall performance of the Falcon 9 rocket.
4. **Visualizing Relationships**: Use visualizations such as scatter plots, bar charts, and line charts to explore the relationships between variables, such as flight number, payload mass, launch site, and orbit type.
5. **Feature Engineering**: Select relevant features for predictive modeling and preprocess the data accordingly. This may involve creating dummy variables for categorical features, casting numeric columns to a consistent data type, and performing any necessary transformations.

EDA with SQL

Summary of SQL Queries and Methodology

This SQL notebook is designed to analyze the SpaceX dataset using SQL queries. Below is a summary of the SQL queries performed along with the methodology:

- **Task 1: Display the names of unique launch sites in the space mission**

✓ SQL Query: SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;

✓ Methodology: This query retrieves distinct launch site names from the dataset to identify unique launch sites involved in SpaceX missions.

- **Task 2: Display 5 records where launch sites begin with the string 'CCA'**

✓ SQL Query: SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;

✓ Methodology: This query filters the dataset to display records where the launch site name starts with the string 'CCA' and limits the output to 5 records.

- **Task 3: Display the total payload mass carried by boosters launched by NASA (CRS)**

✓ SQL Query: SELECT SUM("PAYLOAD_MASS__KG_") AS total_payload_mass FROM SPACEXTBL WHERE "Customer" = 'NASA (CRS)';

✓ Methodology: This query calculates the total payload mass carried by boosters launched by NASA (CRS) by summing up the payload masses for relevant records.

EDA with SQL

- **Task 4: Display the average payload mass carried by booster version F9 v1.1**

✓ SQL Query: SELECT AVG("PAYLOAD_MASS__KG_") AS average_payload_mass FROM SPACEXTBL WHERE "Booster_Version" like '%F9 v1.1';

✓ Methodology: This query calculates the average payload mass carried by boosters of version F9 v1.1 by taking the average of payload masses for relevant records.

- **Task 5: List the date when the first successful landing outcome in ground pad was achieved**

✓ SQL Query: SELECT MIN("Date") AS first_successful_landing_date FROM SPACEXTBL WHERE "Landing_Outcome" = 'Success (ground pad)';

✓ Methodology: This query retrieves the earliest date when a successful landing outcome on the ground pad was achieved.

- **Task 6: List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000**

✓ SQL Query: SELECT "Booster_Version" FROM SPACEXTBL WHERE "Landing_Outcome" = 'Success (drone ship)' AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000;

✓ Methodology: This query filters the dataset to retrieve the booster names where the landing outcome was success on the drone ship and the payload mass was within the specified range.

EDA with SQL

- **Task 7: List the total number of successful and failure mission outcomes**

✓ SQL Query: SELECT MISSION_OUTCOME, COUNT(*) as total_number_successes_or_failures FROM SPACEXTBL GROUP BY MISSION_OUTCOME;

✓ Methodology: This query counts the occurrences of each mission outcome (success or failure) and presents the total count for each outcome.

- **Task 8: List the names of the booster versions which have carried the maximum payload mass using a subquery**

✓ SQL Query: SELECT "Booster_Version" FROM SPACEXTBL WHERE "Payload_Mass__kg_" = (SELECT MAX("Payload_Mass__kg_") FROM SPACEXTBL);

✓ Methodology: This query retrieves the names of booster versions that carried the maximum payload mass by comparing each payload mass with the maximum payload mass obtained from a subquery.

- **Task 9: List the records displaying the month names, failure landing outcomes in drone ship, booster versions, and launch sites for the months in the year 2015**

✓ SQL Query: SELECT substr(Date, 6, 2) AS month, Date, Booster_version, Launch_site, "Landing _Outcome" FROM SPACEXTBL WHERE "Landing _Outcome" = 'Failure (drone ship)' AND substr(Date, 1, 4) = '2015';

✓ Methodology: This query retrieves records for months in the year 2015 where the landing outcome was a failure on the drone ship, along with booster versions and launch sites.

EDA with SQL

- **Task 10: Rank the count of landing outcomes between specific dates in descending order**

✓ SQL Query: 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;

✓ Methodology: This query ranks the count of landing outcomes (such as failure on the drone ship or success on the ground pad) between specified dates in descending order.

These SQL queries help to extract valuable insights from the SpaceX dataset, providing information about launch sites, payload masses, landing outcomes, booster versions, and more.

Build an Interactive Map with Folium

Methodology used to create the Interactive Maps with Folium:

1. **Importing Required Libraries**: The first step involves importing the necessary libraries, including Folium and pandas, which are used for data manipulation and visualization.
2. **Loading Data**: Data related to launch sites and launch records is loaded into a pandas DataFrame. This data includes information such as launch site coordinates, launch outcomes, and other relevant details.
3. **Initializing the Map**: A Folium map object is created to serve as the canvas for visualizing geographical data. The map is initialized with a specific center location and zoom level, providing a starting point for further exploration.
4. **Adding Markers and Circles**: Markers are added to represent launch sites on the map, with each marker displaying the name of the launch site when clicked. Circles are also added around each launch site to highlight their coverage areas or proximity.
5. **Handling Marker Clusters**: To improve map readability and performance, marker clusters are utilized to group nearby markers with the same coordinates. This helps in managing large numbers of markers more efficiently, especially when dealing with multiple launch records at the same location.
6. **Drawing Polylines**: Polylines are drawn to visualize the distances between launch sites and selected points of interest, such as coastlines, cities, railways, or highways. These polylines provide a clear representation of spatial relationships and proximity.
7. **Interactivity**: Interactive features such as MousePosition are added to the map, allowing users to explore and interact with geographical data more effectively. This includes obtaining coordinates for specific points on the map and calculating distances between them.
8. **Finalizing and Displaying the Map**: Once all map elements and interactivity features are added, the final map is displayed. Users can interact with the map to explore different aspects of the geographical data and gain insights into launch site locations and their surroundings.

Build an Interactive Map with Folium

Summary of the map objects created:

1. **Markers**: Markers were added to represent the launch sites on the map. Each marker indicated the location of a launch site, and a label with the launch site name was displayed when the marker was clicked. These markers provided a visual representation of the launch sites on the map.

2. **Circles**: Circles were added around each launch site to highlight the area of influence or proximity. These circles had a radius of 1000 meters and were filled with a color to distinguish them from other map elements. The circles helped visualize the coverage area around each launch site.

3. **Marker Clusters**: Marker clusters were utilized to handle multiple markers with the same coordinates, such as multiple launch records at the same launch site. Marker clusters grouped nearby markers into clusters, improving map readability and performance, especially when dealing with a large number of markers.

4. **Polyline**: Polyline was drawn to represent the distance between a launch site and a selected point of interest, such as a coastline or city. The polyline showed the shortest path between the launch site and the selected point, providing a visual representation of the distance.

- These map objects were added to achieve various objectives in the analysis:
- Markers were used to indicate the locations of launch sites, making it easier to identify them on the map.
- Circles were added to visualize the coverage area or proximity around each launch site, helping to understand the spatial distribution of launch sites.
- Marker clusters improved map readability by grouping nearby markers, reducing clutter and improving performance.
- Polylines were drawn to represent distances between launch sites and selected points of interest, aiding in the analysis of proximity to railways, highways, coastline, and cities.
- Overall, these map objects helped in exploring and analyzing the spatial relationships and proximity of launch sites to various features, contributing to a better understanding of the launch site locations and their surroundings.

Build a Dashboard with Plotly Dash

Here's a summary of the plots/graphs and interactions included in the dashboard

1. Dropdown List for Launch Site Selection (Task 1):

1. A dropdown list allows users to select a specific launch site or view data for all sites.
2. This interaction enables users to filter launch data based on their area of interest, providing more focused insights.

2. Pie Chart Showing Total Successful Launches (Task 2):

1. The pie chart displays the total count of successful and failed launches for the selected launch site.
2. Users can visualize the success rate of launches at different sites, helping them identify sites with higher success rates.
3. The chart dynamically updates based on the selected launch site, providing site-specific success rate information.

3. Range Slider for Payload Selection (Task 3):

1. A range slider allows users to select a payload mass range, specifying the minimum and maximum payload values.
2. This interaction enables users to filter launch data based on payload mass, focusing on launches within a specific payload range.

4. Scatter Chart Showing Payload vs. Launch Success/Failure (Task 4):

1. The scatter chart illustrates the correlation between payload mass and launch success or failure.
2. Users can observe how payload mass influences launch outcomes, identifying any trends or patterns.
3. The chart dynamically updates based on the selected launch site and payload range, providing customized insights.

5. Callback Functions for Dynamic Updates:

1. Callback functions are implemented to update the plots and charts dynamically in response to user interactions.
2. These functions ensure that the dashboard remains interactive and responsive, providing real-time visualizations based on user selections.

Build a Dashboard with Plotly Dash

Overall, these plots and interactions are included in the dashboard to enable users to explore SpaceX launch records, analyze key factors affecting launch success, and gain valuable insights into launch site performance and payload characteristics.

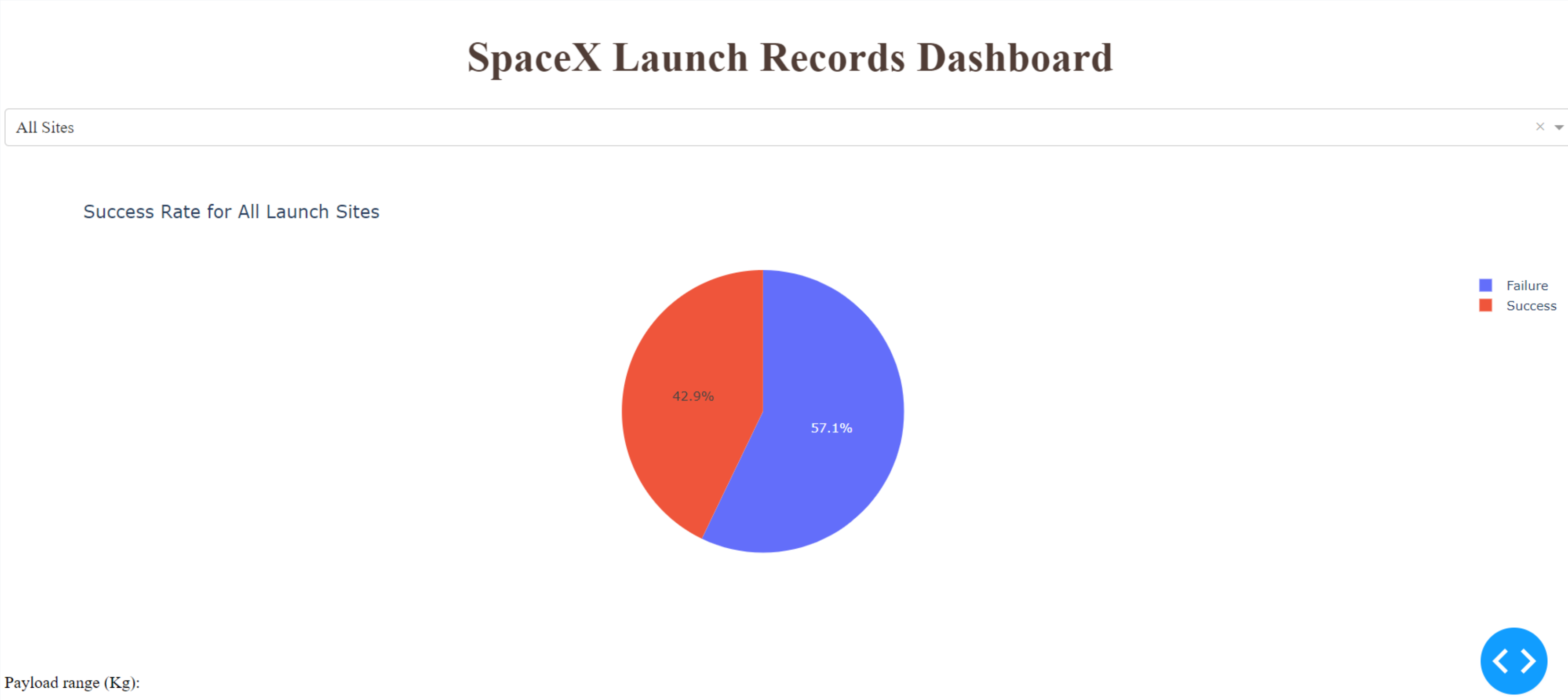

Figure 4: Plotly Dashboard depicting the pie chart success rate for all launch sites

Build a Dashboard with Plotly Dash

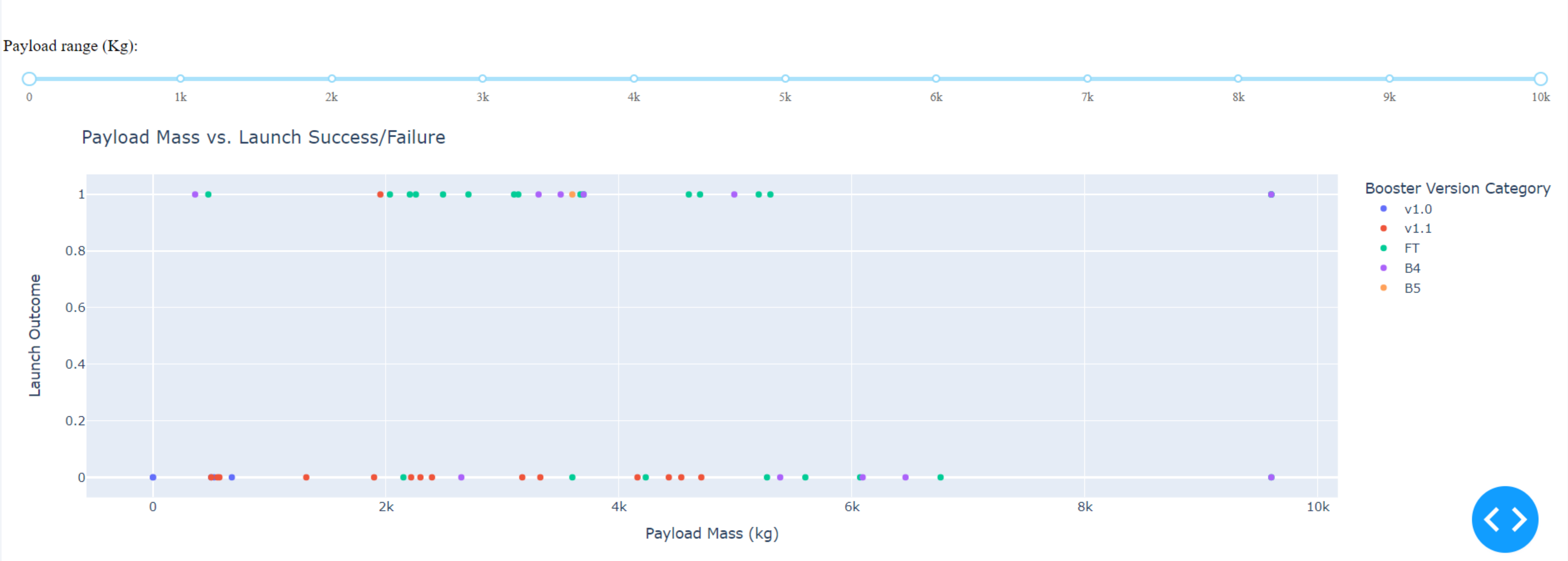

Figure 5: Plotly Dashboard depicting the scatter chart for the payload mass against launch success/failure.

Build a Dashboard with Plotly Dash

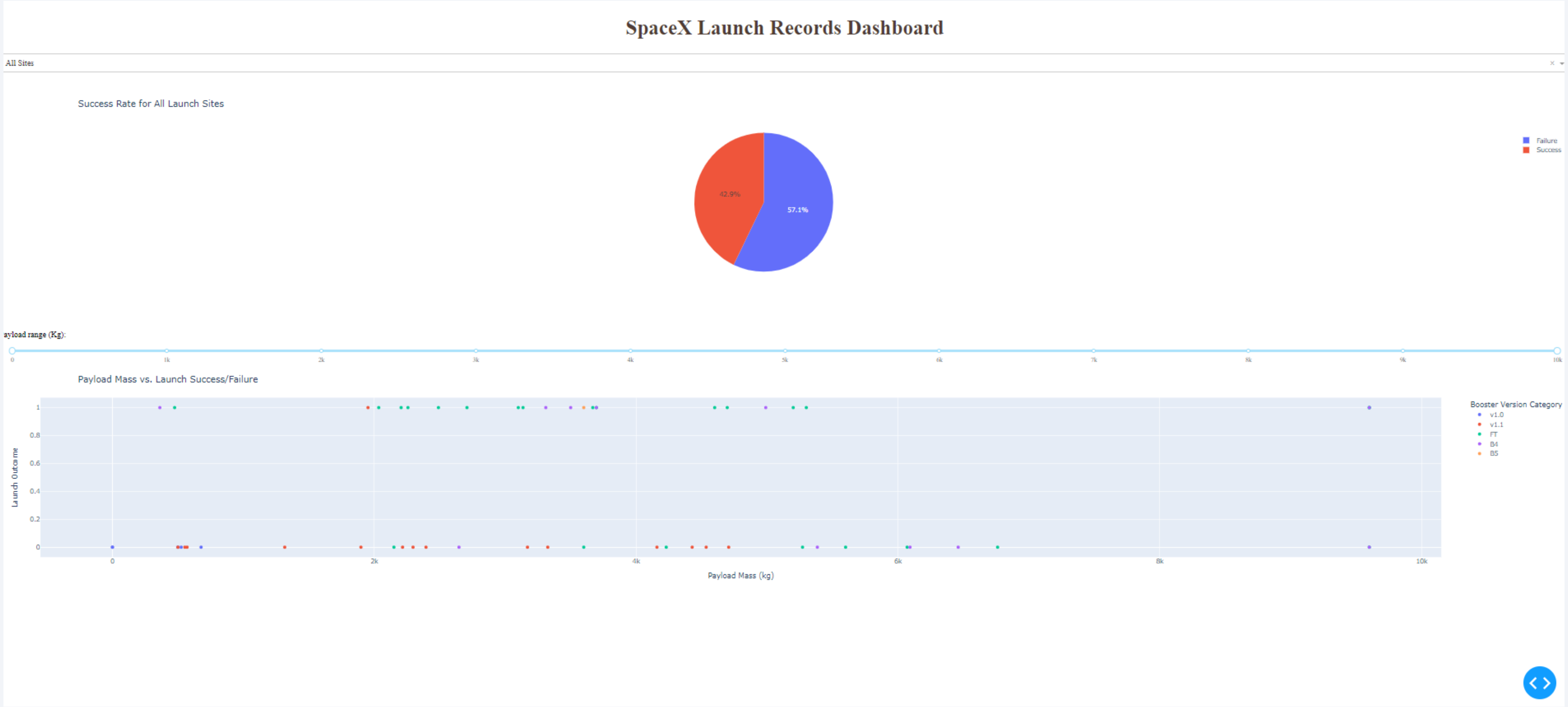

Figure 6: Plotly Dashboard depicting the pie chart success rate for all launch sites

Predictive Analysis (Classification)

Summarized overview of the model development process:

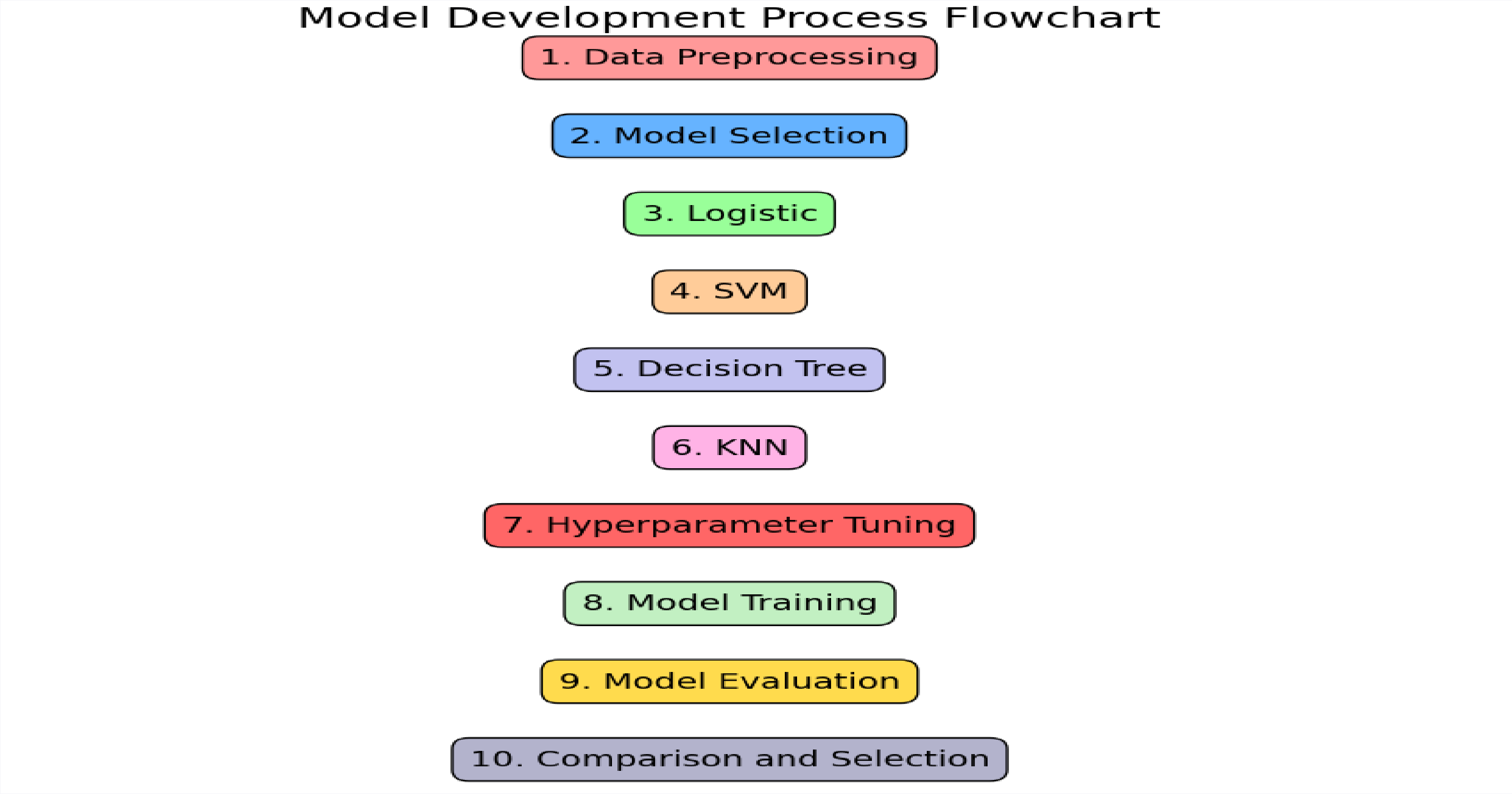

Figure 7: Flowchart of the model development process

Predictive Analysis (Classification)

Summarized overview of the model development process:

1. **Data Preprocessing**:
    1. The dataset is loaded and preprocessed, including standardization of the feature data and splitting into training and test sets.
2. **Model Selection and Hyperparameter Tuning**:
    1. Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K Nearest Neighbors (KNN) classifiers are chosen.
    2. GridSearchCV is employed to search for the best hyperparameters for each classifier using cross-validation.
3. **Model Training**:
    1. Each classifier is trained on the training data with the best hyperparameters obtained from the GridSearchCV.
4. **Model Evaluation**:
    1. The accuracy of each model is evaluated using the test data.
    2. Confusion matrices are plotted to visualize the performance of each model, particularly in terms of true positives, true negatives, false positives, and false negatives.
5. **Comparison and Selection of the Best Performing Model**:
    1. The accuracies of all models are compared.
    2. The model with the highest accuracy on the test data is selected as the best performing model.

Exploratory Data Analysis (EDA) Results:

<u>Methodology:</u>

•Conducted thorough analysis of the dataset to understand its structure, patterns, and relationships between variables.
•Explored summary statistics, distributions, and visualizations to gain insights into the data.
•Identified key features and potential challenges for predictive modelling.

<u>Key Insights after performing exploratory data analysis on the SpaceX Falcon 9 dataset:</u>

1. **Flight Number vs. Payload Mass:** There appears to be a relationship between flight number and payload mass, with higher flight numbers associated with a higher likelihood of successful landings. Additionally, heavier payloads are associated with a lower success rate for first-stage landings.
2. **Launch Site Analysis:** Different launch sites exhibit different success rates, with Cape Canaveral Space Launch Complex 40 having a success rate of 60%, while Kennedy Space Center Launch Complex 39A and Vandenberg Air Force Base Space Launch Complex 4E have success rates of 77%.
3. **Orbit Success Rate:** The success rate varies across different orbit types, with some orbits having higher success rates than others. For example, low Earth orbit (LEO) and polar orbits tend to have higher success rates compared to geostationary transfer orbits (GTO).
4. **Temporal Trends:** Over time, there appears to be an increasing trend in the success rate of Falcon 9 launches, indicating potential improvements in technology and operational procedures.
5. **Feature Importance:** Certain features, such as launch site, orbit type, and payload mass, are likely to be important predictors of mission success and should be considered in predictive modeling efforts.

By gaining insights into these key aspects of the dataset, we can better understand the factors that influence the success of Falcon 9 rocket launches and inform future decision-making processes.

Results:

Interactive Analytics (screenshots provided from slide 24-26):

•Created interactive visualizations and dashboards to showcase the data and its insights.

•Implemented user-friendly features allowing for exploration of different aspects of the data.

•Utilized screenshots to demonstrate the interactive functionalities and highlight key findings from the data.

Predictive Analysis Results:

•Developed machine learning models to predict the success or failure of Falcon 9 rocket launches.

•Evaluated the performance of various classification algorithms, including Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K Nearest Neighbors (KNN).

•Conducted hyperparameter tuning using techniques like GridSearchCV to optimize model performance.

•Assessed model accuracy, precision, recall, and F1-score to determine the predictive capability of each model.

•Identified the best-performing model based on evaluation metrics and provided insights into its predictive capabilities for future Falcon 9 launches.

Results:

All the models performed at about the same level and had the same scores and accuracy . This is likely due to the small dataset utilised for training and testing.

Due to it’s robust functionality, there are several reasons why the SVM model might perform the best:

1. **Ability to Handle Non-Linearity:** SVMs are effective in handling non-linear relationships between features and the target variable. They can use different kernel functions such as radial basis function (RBF), polynomial, and sigmoid to capture complex decision boundaries.
2. **Optimized Hyperparameters:** The GridSearchCV algorithm was used to find the best combination of hyperparameters for the SVM model, including the choice of kernel, regularization parameter (C), and kernel coefficient (gamma). This optimization process likely contributed to the model's improved performance.
3. **Robustness to Overfitting:** SVMs are less prone to overfitting, especially in high-dimensional spaces, compared to other models like decision trees. This property ensures that the model generalizes well to unseen data, leading to better performance on the test set.
4. **Effective Handling of Imbalanced Data:** If the dataset had an imbalance in the classes (e.g., more successful landings than unsuccessful ones), SVMs can handle such scenarios effectively by adjusting the class weights or using techniques like cost-sensitive learning.
5. **Versatility:** SVMs can be adapted to various types of classification problems and have been successfully applied in numerous domains, including image recognition, text classification, and bioinformatics. This versatility might have contributed to its effectiveness in predicting the landing outcome of the Falcon 9 first stage.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

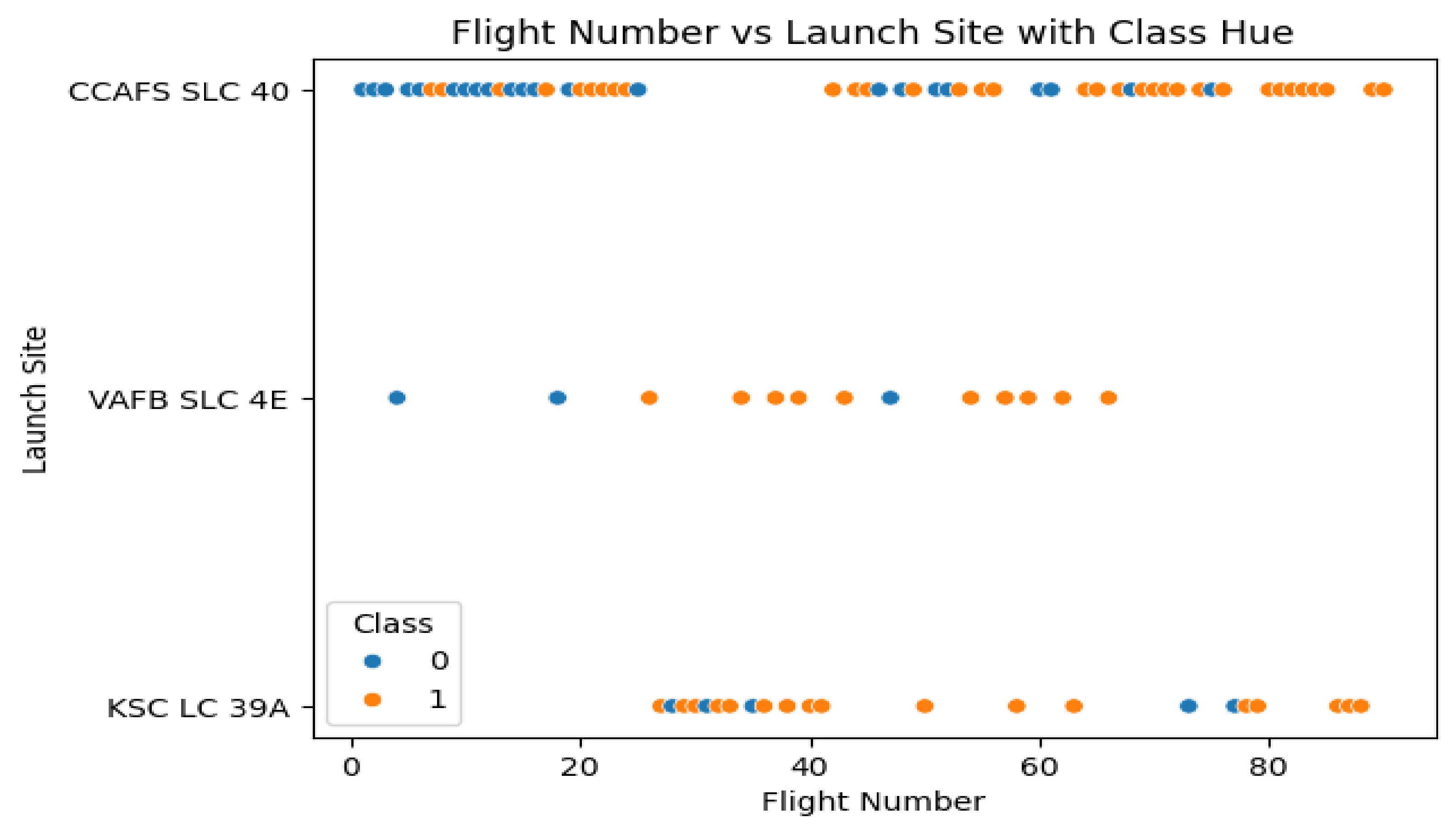

Figure 8: Flight number vs launch site

Payload vs. Launch Site

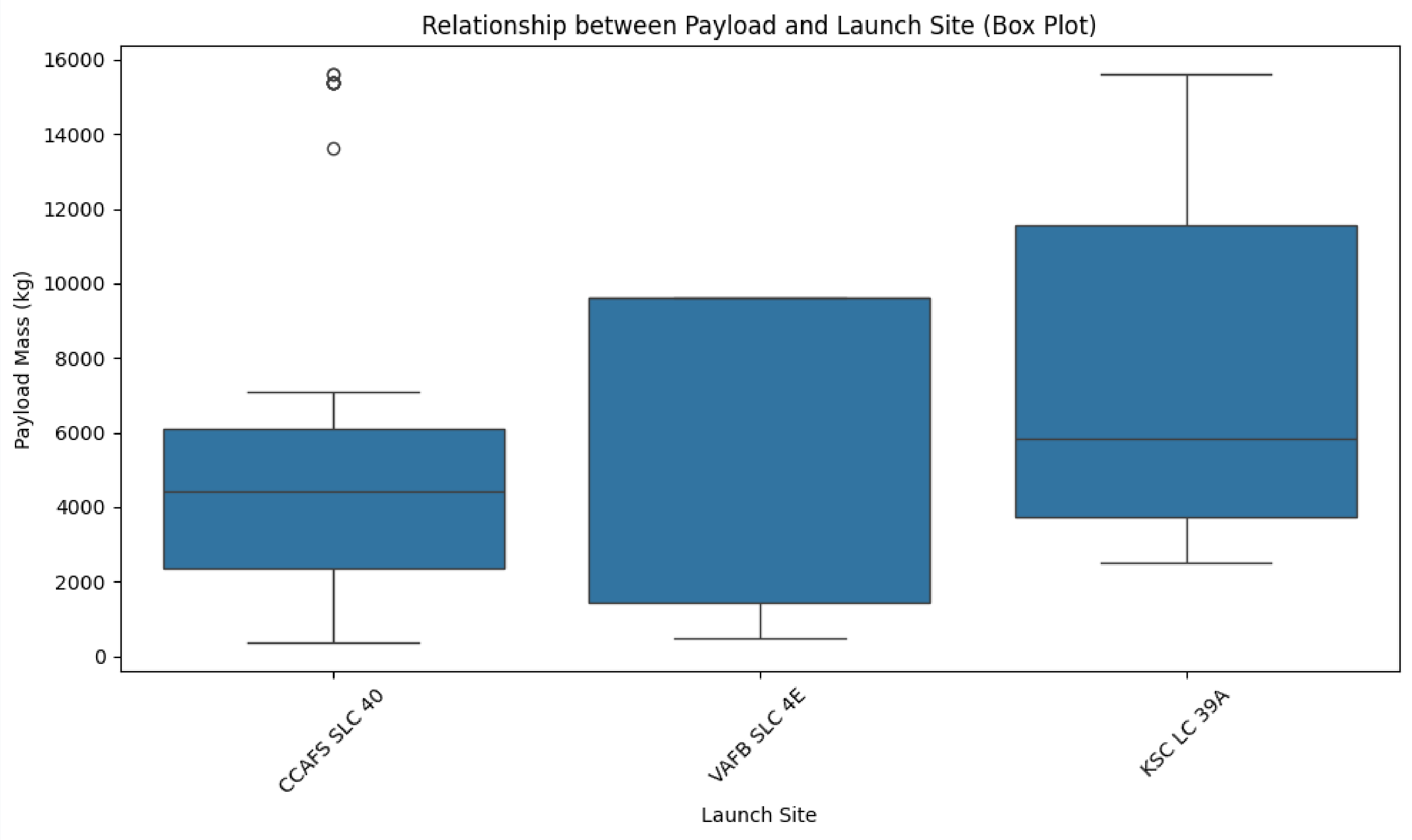

Figure 9: Payload vs launch site

Success Rate vs. Orbit Type

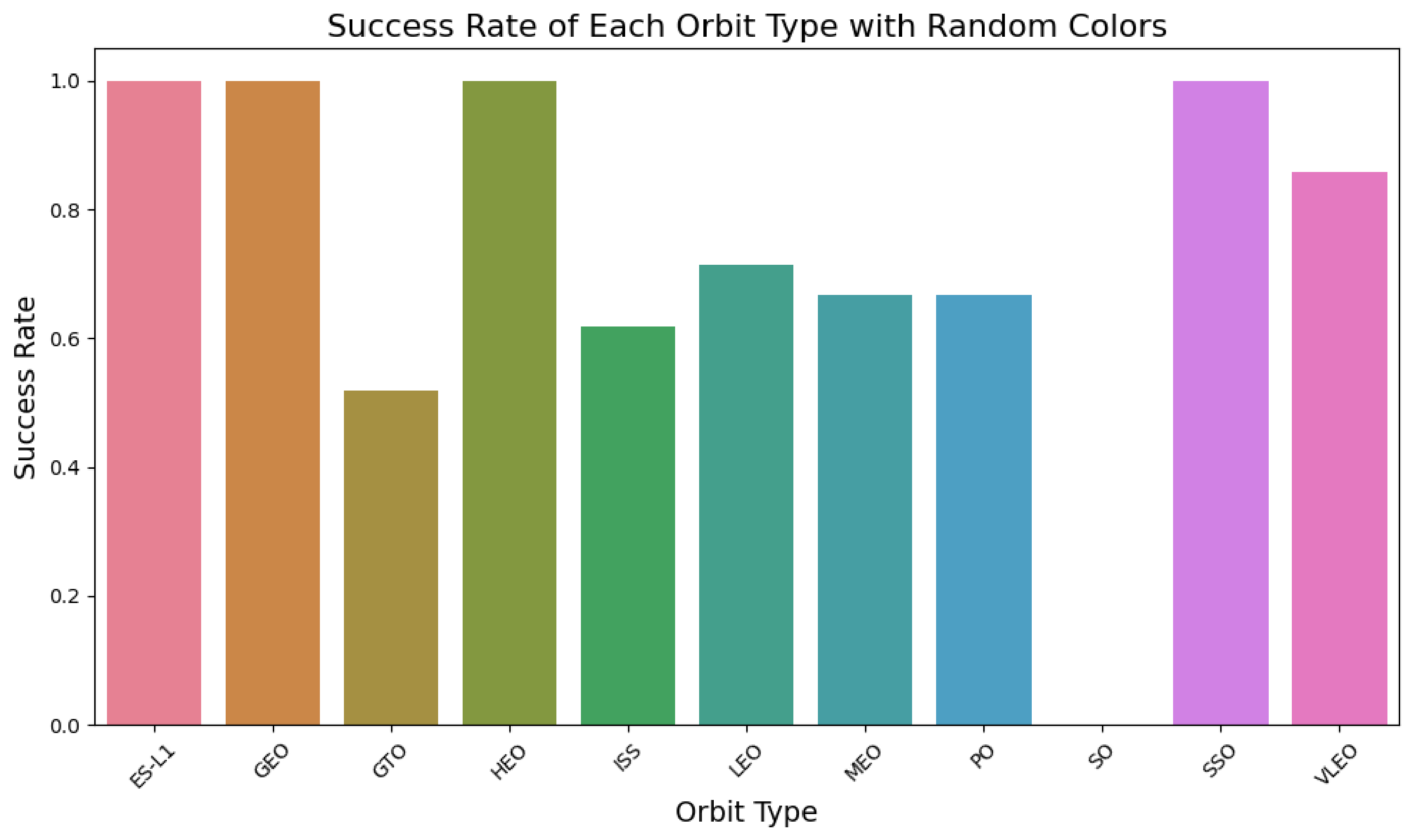

Figure 10: Success rate vs orbit type

Flight Number vs. Orbit Type

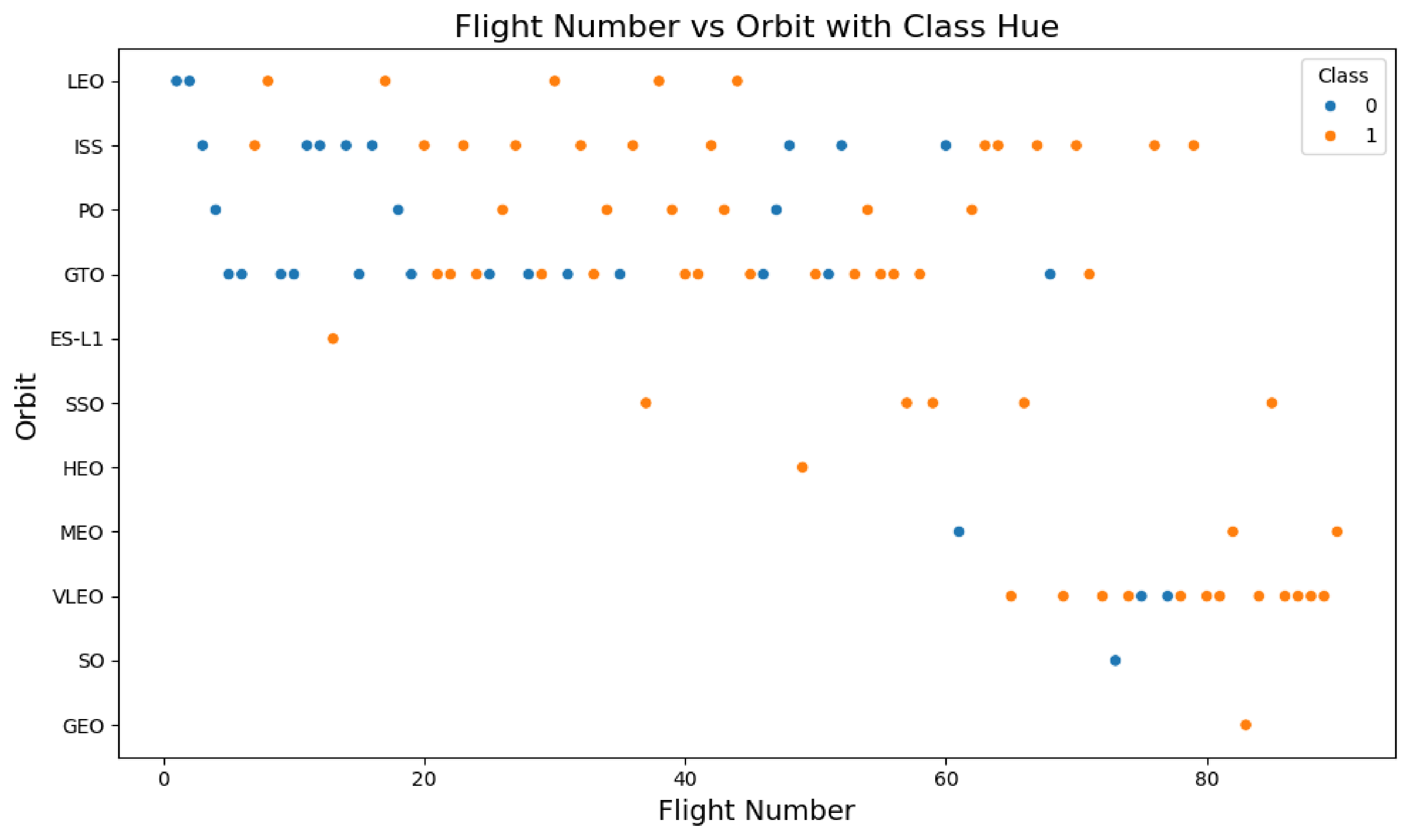

Figure 11: Flight number vs orbit type

Payload vs. Orbit Type

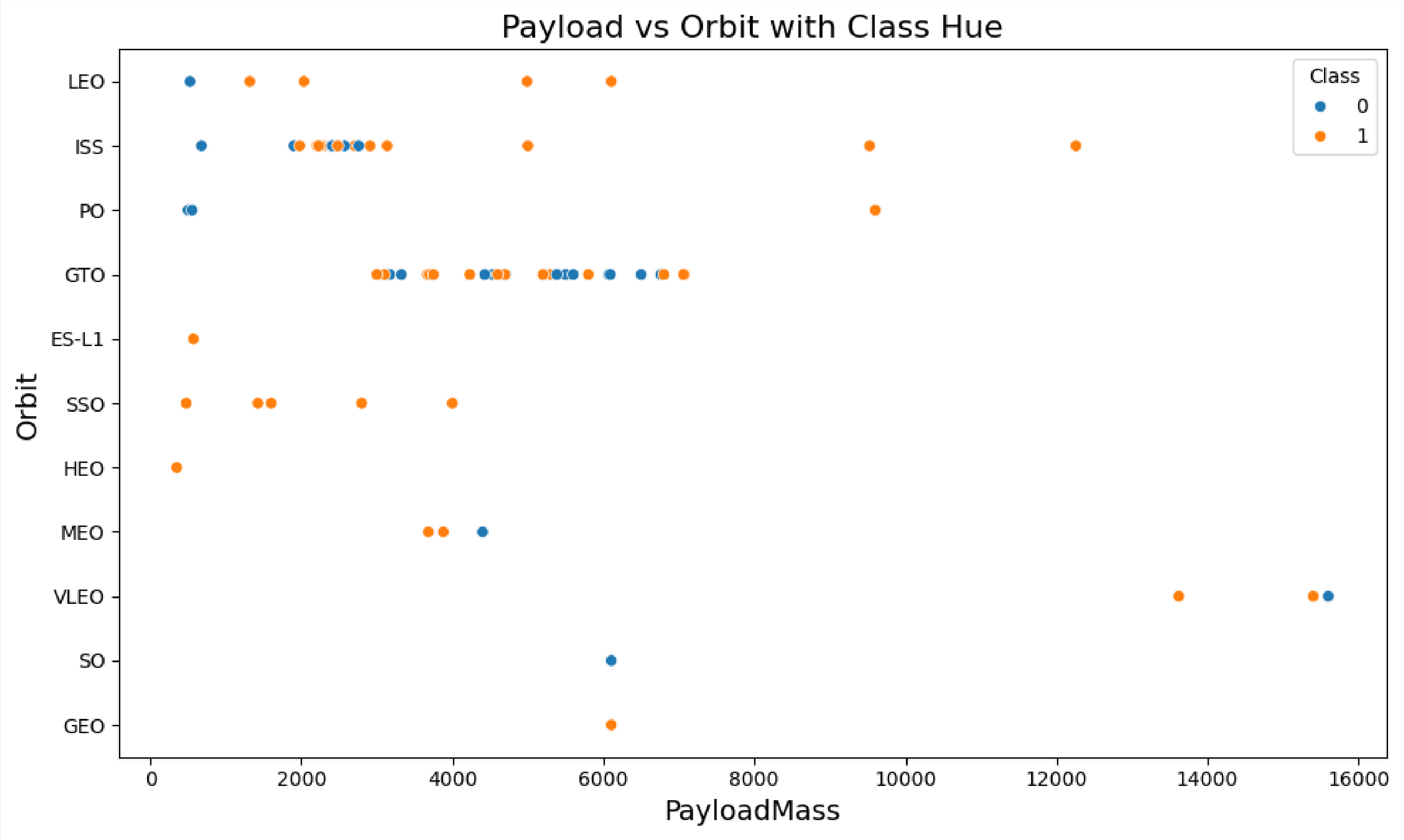

Figure 12: Payload vs orbit type

Launch Success Yearly Trend

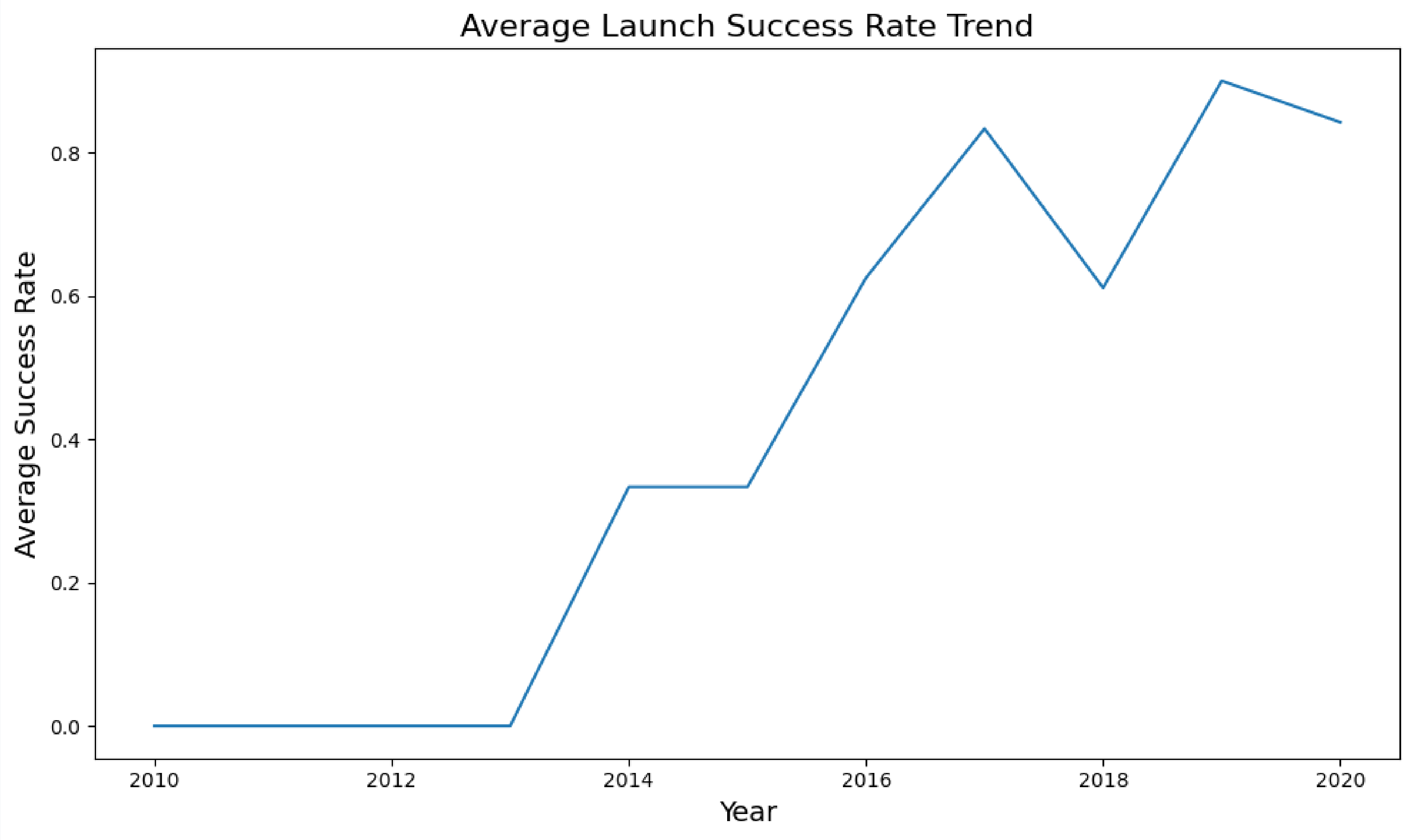

Figure 13: Launch Success Yearly Trend

All Launch Site Names (Task completed in Juypter but saved offline and screenshots are shown from Visual Studio Code due the notebook being saved on the desktop).

Task 1

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

Figure 14: Screenshot depicting the SQL code for All Launch Site Names

Launch Site Names Begin with 'CCA'

Task 2

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

```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE'CCA%' LIMIT 5;
```

Python

* 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

Figure 15: Screenshot depicting the SQL code for Launch Site Names Begin with 'CCA'

Total Payload Mass

Task 3

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

Figure 16: Screenshot depicting the SQL code for Total Payload Mass

Average Payload Mass by F9 v1.1

Task 4

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

```
[28]
%sql SELECT AVG("PAYLOAD_MASS__KG_") AS average_payload_mass FROM SPACEXTBL WHERE "Booster_Version" like '%F9 v1.1'
```

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

average_payload_mass
2928.4

Figure 17: Screenshot depicting the SQL code for Average Payload Mass by F9 v1.1

First Successful Ground Landing Date

Task 5

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 (ground pad)';
```

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

first_successful_landing_date
2015-12-22

Figure 18: Screenshot depicting the SQL code for First Successful Ground Landing Date

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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 SPACEXTBL WHERE "Landing_Outcome" = 'Success (drone ship)' AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000;
```

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

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

Figure 19: Screenshot depicting the SQL code for Successful Drone Ship Landing with Payload between 4000 and 6000

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number_successes_or_failures \
FROM SPACEXTBL \
GROUP BY MISSION_OUTCOME;
```

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

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

Figure 20: Screenshot depicting the SQL code for Total Number of Successful and Failure Mission Outcomes

Boosters Carried Maximum Payload

Task 8

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

```
[44]: %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

Figure 21: Screenshot depicting the SQL code for Boosters Carried Maximum Payload

2015 Launch Records

Task 9

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
    substr(Date, 6, 2) AS month,
    Date,
    Booster_version,
    Launch_site,
    "Landing _Outcome"
FROM
    SPACEXTBL
WHERE
    "Landing _Outcome" = 'Failure (drone ship)'
    AND substr(Date, 1, 4) = '2015';
```

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

month	Date	Booster_Version	Launch_Site	"Landing _Outcome"

Figure 22: Screenshot depicting the SQL code for 2015 Launch Records

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

Task 10

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

Figure 23: Screenshot depicting the SQL code for Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

Section 3

Launch Sites Proximities Analysis

Folium: Launch site locations

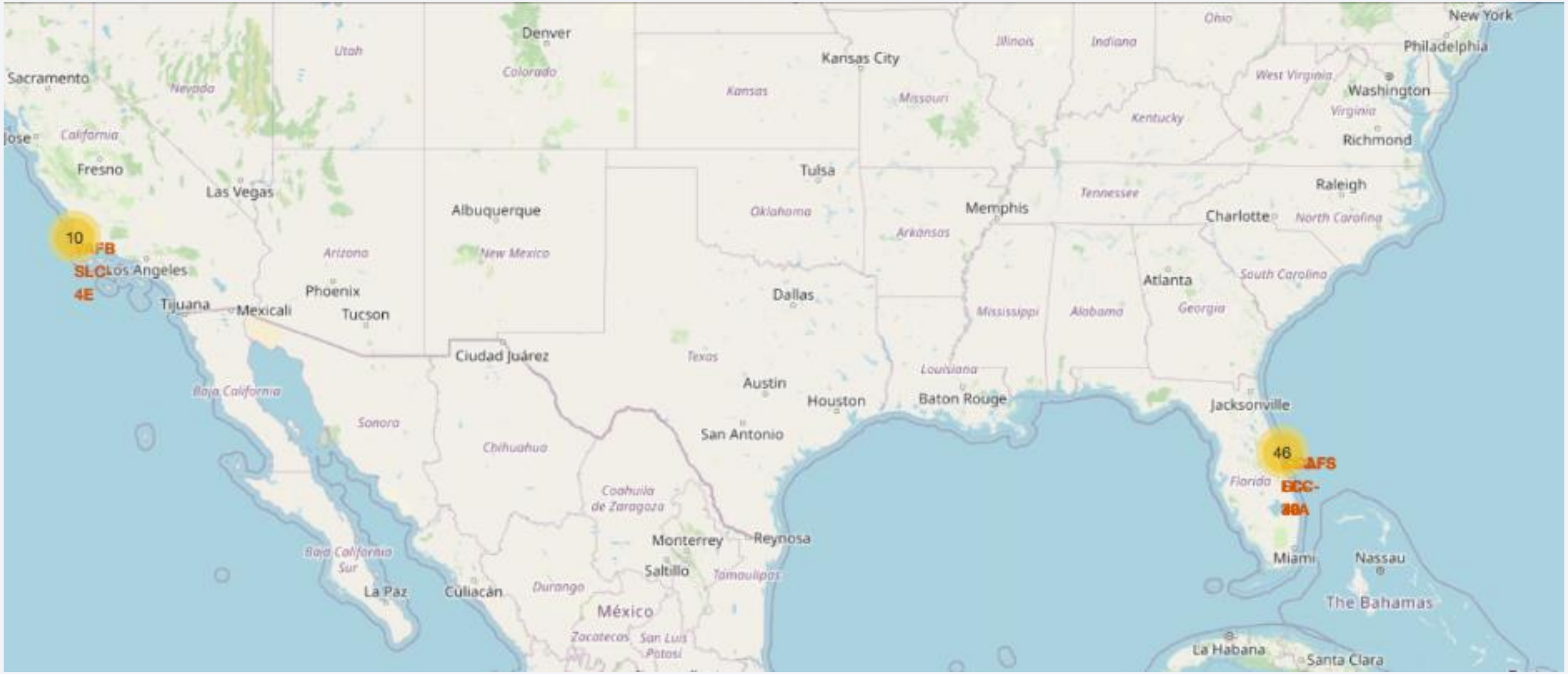

Figure 24: Screenshot depicting the folium map with launch site locations

Folium: Launch site locations

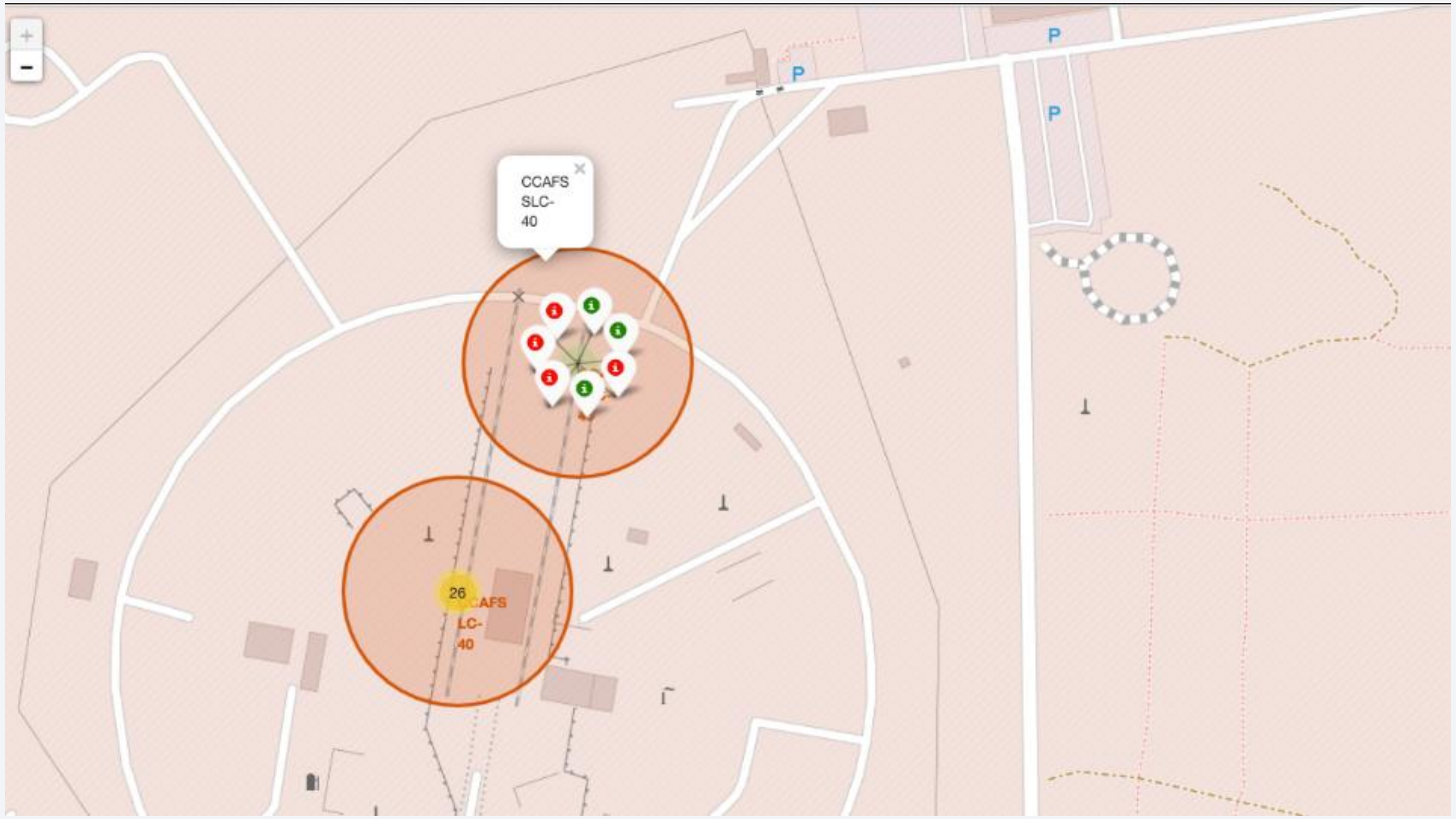

Figure 25: Screenshot depicting the folium map with specific launch sites

Folium: Launch site locations

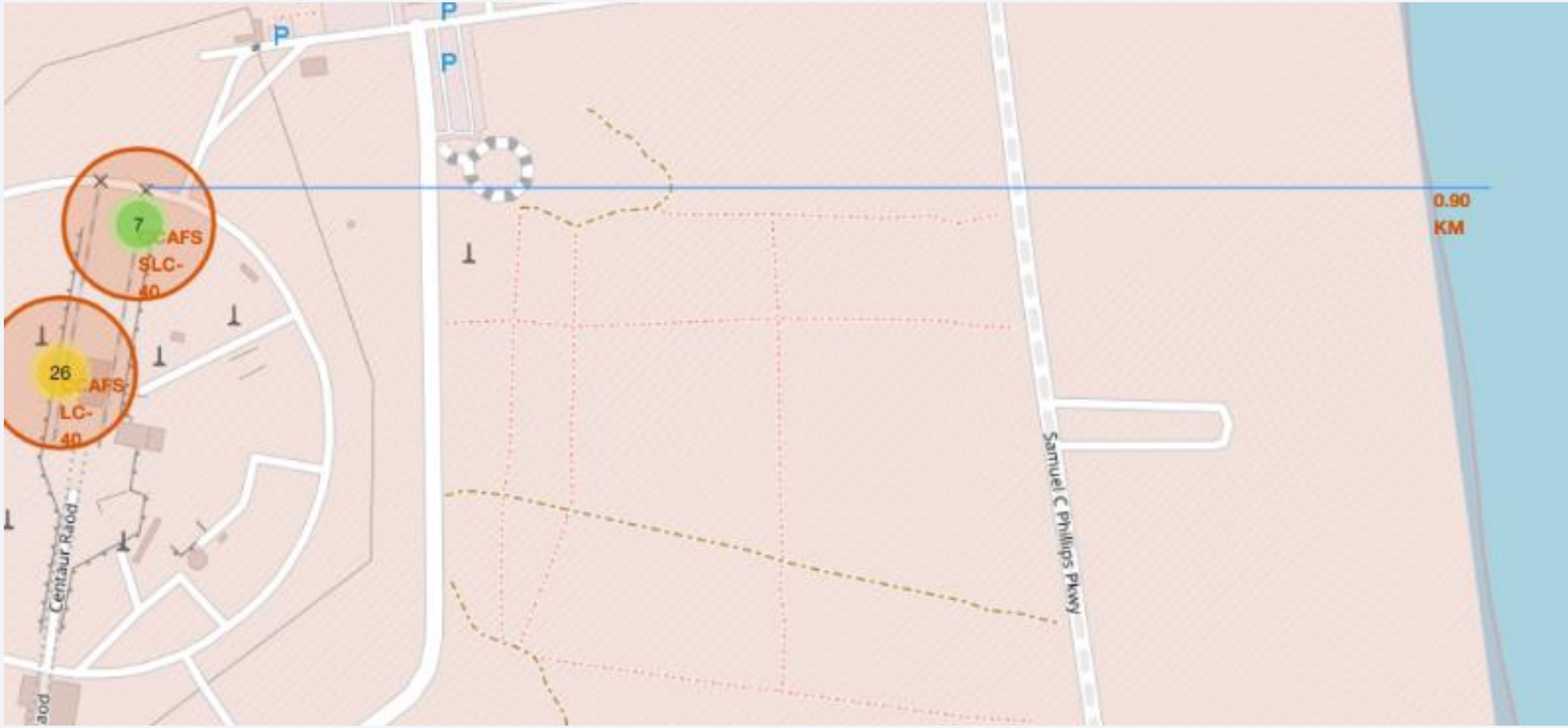

Figure 26: Screenshot depicting the folium map with the specified distance from the coastline

Section 4

Build a Dashboard with Plotly Dash

SpaceX Launch Records Dashboard

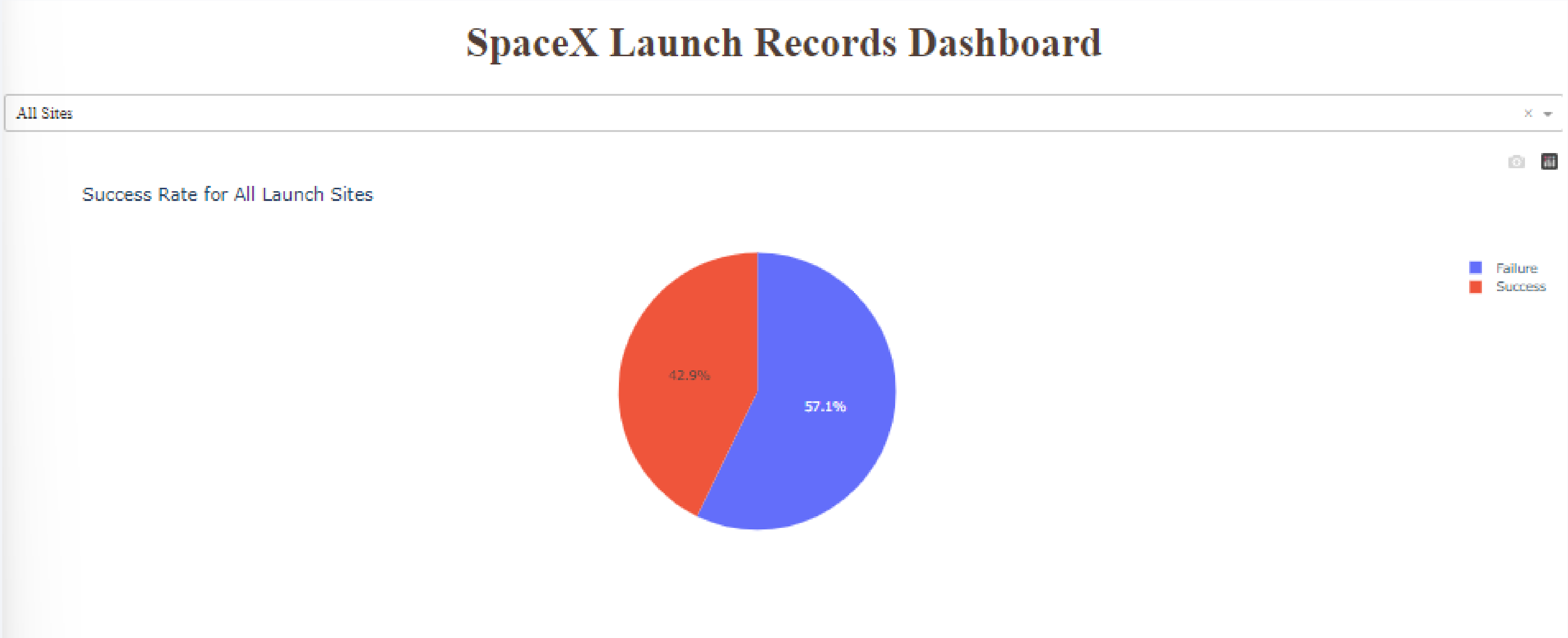

Figure 27: Screenshot depicting the launch success count for all sites, in a piechart

SpaceX Launch Records Dashboard: Launch site with the Highest Success Rate

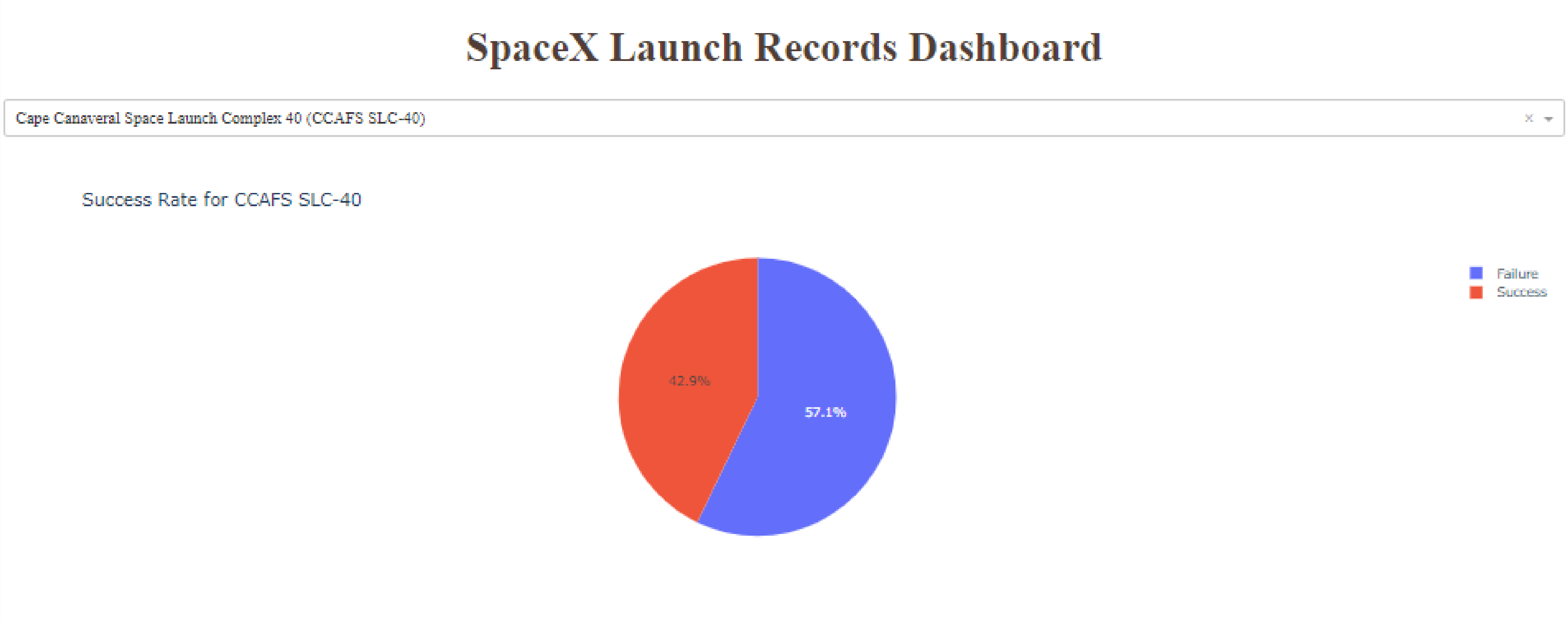

Figure 28: Screenshot depicting the launch success percentage for Cape Canaveral, in a piechart

SpaceX Launch Records Dashboard: Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

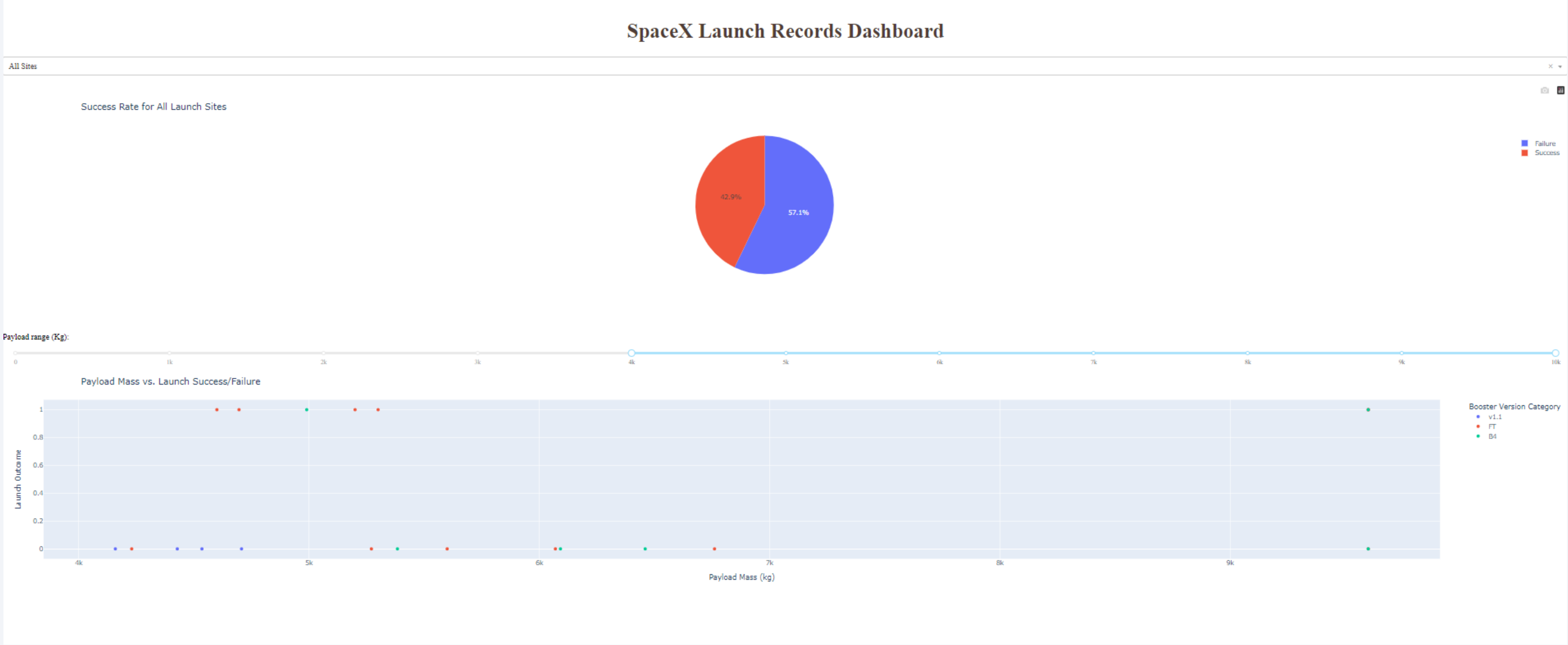

Figure 29: Screenshot depicting the launch success percentage for Cape Canaveral, in a piechart

Section 5

Predictive Analysis (Classification)

Classification Accuracy

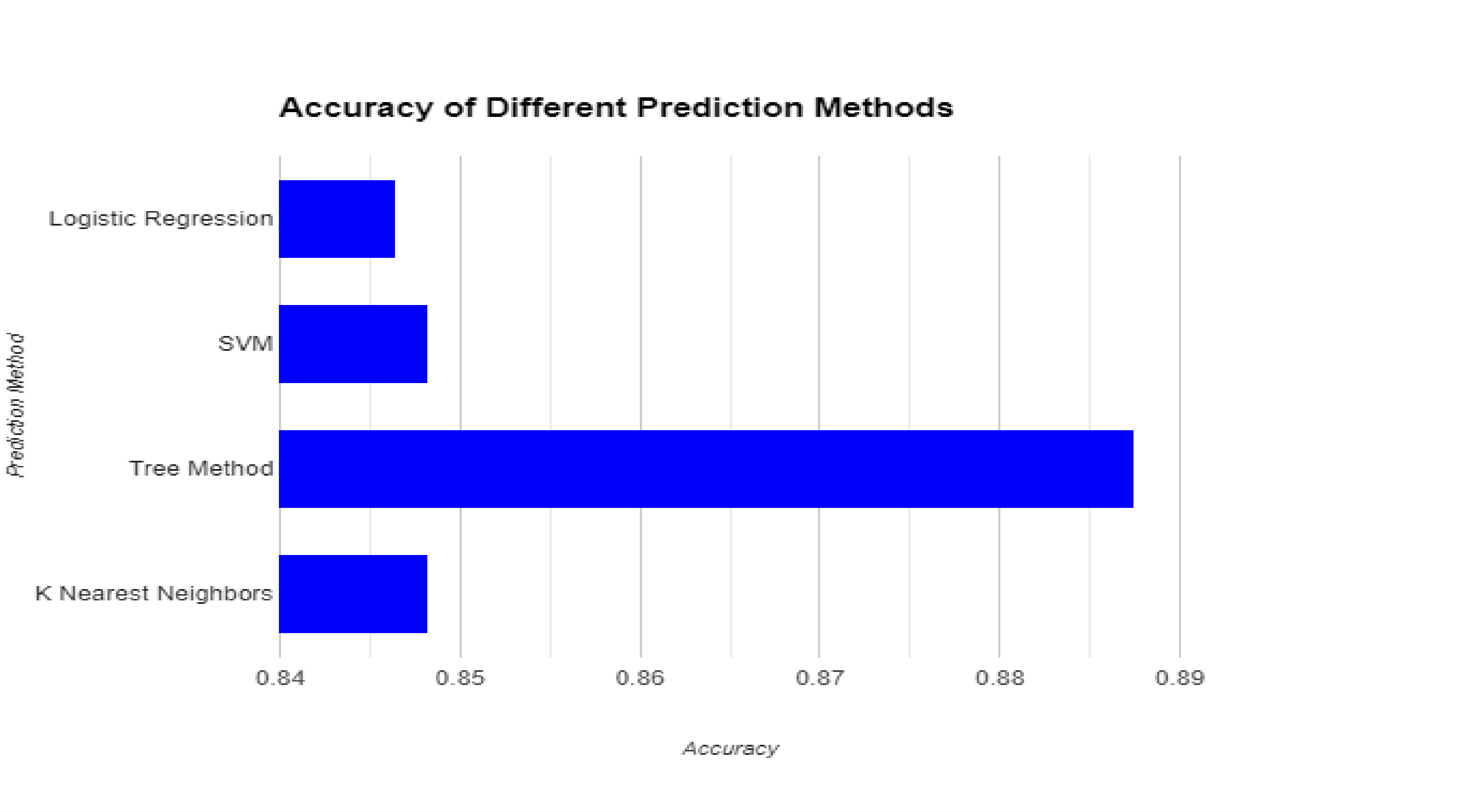

Figure 30: Bar graph depicting the accuracy of different prediction methods

Confusion Matrix

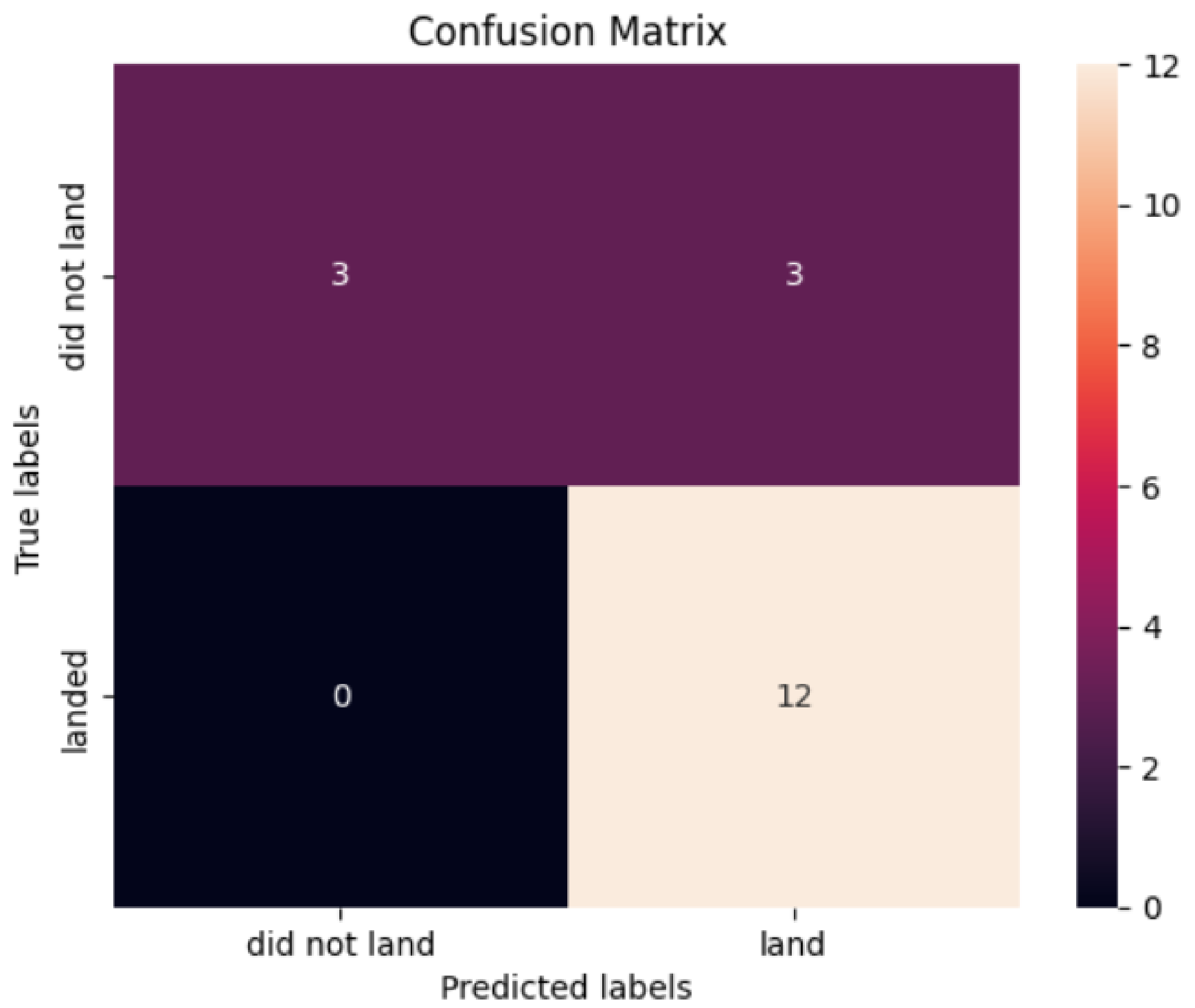

Figure 30: Confusion matrix depicting the accuracy of the best performing model

Conclusions

1. **Exploratory Data Analysis (EDA)**:
    1. During the exploratory data analysis phase, various insights were gained from the dataset.
    2. Features such as launch success, booster version, and mission outcome were likely explored.
    3. Statistical summaries, data visualizations, and correlation analyses may have been performed to understand the relationships between different variables.
    4. The EDA phase likely helped in identifying patterns, trends, outliers, and potential areas for further investigation.
2. **Predictive Analysis**:
    1. In the predictive analysis phase, machine learning models were trained to predict the landing outcome of the Falcon 9 first stage.
    2. Models such as Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K Nearest Neighbors (KNN) were evaluated.
    3. Each model was optimized using techniques like hyperparameter tuning with cross-validation.
    4. The performance of each model was assessed using metrics such as accuracy, precision, recall, and F1-score.
    5. Based on the evaluation results, the best-performing model was identified, which in this case, appears to be the **Support Vector Machine (SVM) model (although the accuracy of each model was the same at 0.833)**.
3. **Other Sections**:
    1. Other sections of the project included data preprocessing steps such as data cleaning, feature engineering, and feature scaling.
    2. Techniques like standardization or normalization were applied to ensure that all features have the same scale.
    3. The project also included the visualization of results, such as confusion matrices, to evaluate the performance of the predictive models.
    4. Conclusions drawn from the project likely emphasized the importance of machine learning in predicting the landing outcome of the Falcon 9 first stage and its potential applications in cost estimation for space launches

Appendix – Results of Predictive Analysis:

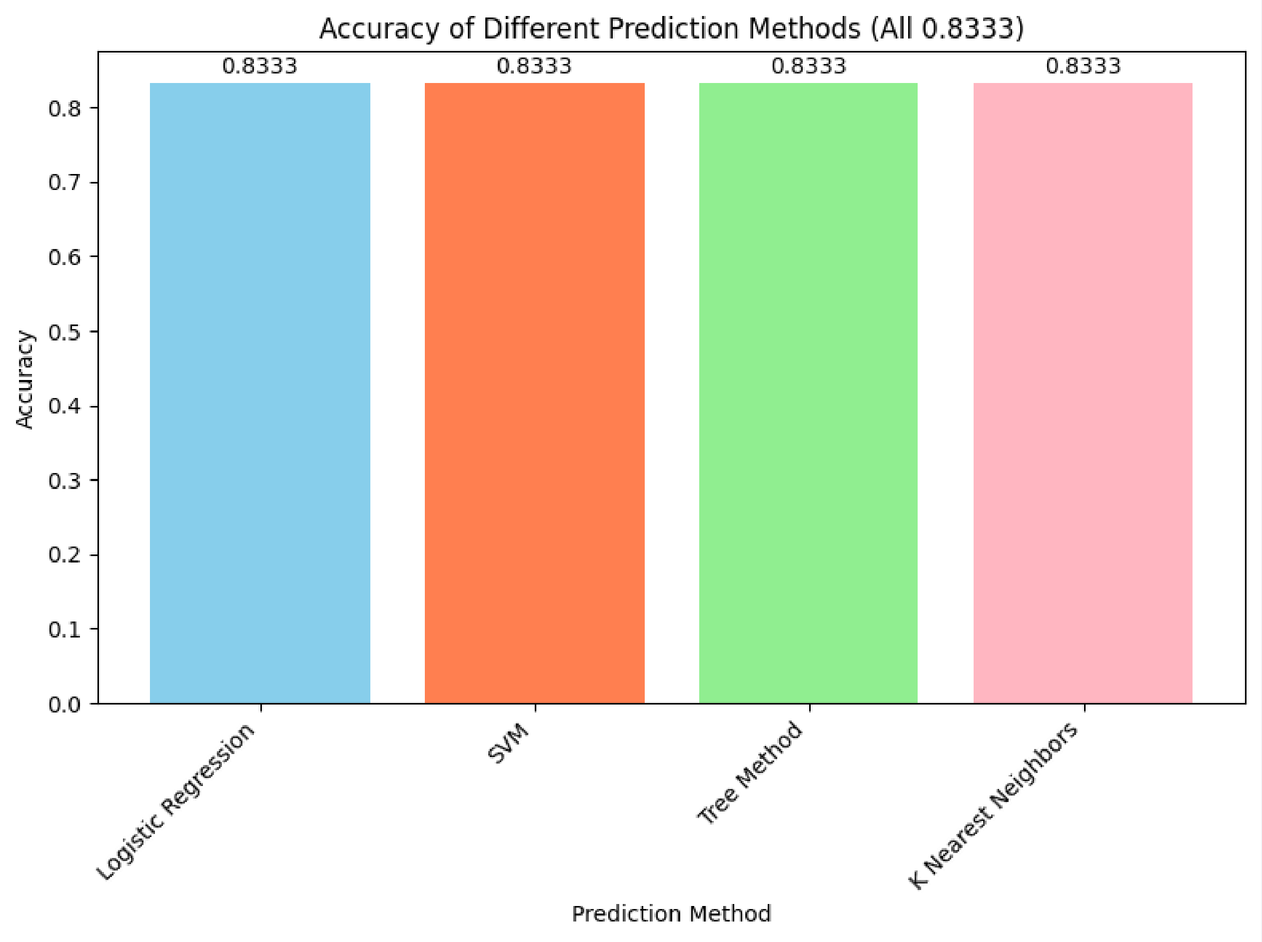

Figure 31: Bar graph depicting the accuracy various methods of predictive analysis:

Thank you!