Winning Space Race with Data Science

Lessio Igor

21 November 2023

Outline

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

Here the Main (GitHub)

Executive Summary

- Summary of methodologies
- Summary of all results

Here the Main (GitHub)

Introduction

Here the Main (GitHub)

Project Background and Context:

The space industry is rapidly evolving with companies like Virgin Galactic, Rocket Lab, Blue Origin, and SpaceX leading the way in making space travel more affordable and accessible.
SpaceX, in particular, has revolutionized the industry with achievements like ISS missions, the Starlink project, and manned spaceflights.

A key factor in SpaceX's success is the cost-effectiveness of its Falcon 9 rocket, partly due to the reusable first stage. Understanding and predicting the reusability of this stage is crucial for cost estimation and competitive positioning in the market.

Your project focuses on SpaceX's Falcon 9 rocket, particularly analyzing the first stage, which is crucial for the rocket's cost-effectiveness and operational success.

Problems to Address:

Determine the cost of each SpaceX launch by analyzing various factors, including the likelihood of the first stage's successful landing.

Develop predictive models to assess whether the first stage of Falcon 9 will land successfully, using machine learning techniques and publicly available data.
Create comprehensive dashboards for visualizing and interpreting SpaceX launch data, aiding decision-making processes in your hypothetical company, Space Y.
Your role as a data scientist in Space Y, a new competitor in the space industry, involves leveraging data science and machine learning to make informed predictions about launch costs and the reusability of SpaceX's rockets.

Section 1

Methodology

Methodology

Here the Main (GitHub)

Executive Summary

Data Collection Methodology:

Utilized public datasets from SpaceX to gather comprehensive information on Falcon 9 launches.
Data was sourced through various channels, including SpaceX's official records and other spaceflight-related databases.

Data Wrangling:

Processed the raw data to ensure quality and relevance.
Handled missing values, outliers, and inconsistencies to prepare a clean and reliable dataset for analysis.

Data Processing:

Transformed and standardized the data to fit analytical needs.
Engineered features to enhance model performance and insights, especially focusing on factors influencing rocket landings.

Exploratory Data Analysis (EDA):

Conducted thorough EDA using both visualization techniques and SQL queries.
Identified key patterns, correlations, and insights relevant to SpaceX's launch outcomes and first stage landings.

Interactive Visual Analytics:

Utilized Folium for geospatial analysis, visualizing launch site locations and trajectories.
Employed Plotly Dash to create interactive dashboards, enabling dynamic exploration of launch data.

Predictive Analysis Using Classification Models:

Developed classification models to predict the success of the first stage landing.
Models included Logistic Regression, Decision Trees, SVM, and K-Nearest Neighbors.

Model Building, Tuning, and Evaluation:

Built various machine learning models, tuning hyperparameters for optimal performance.Evaluated models using metrics like accuracy, precision, and recall, ensuring robustness and reliability of predictions.

Data Collection

The foundation of our analysis lies in the robust data collection process, ensuring a comprehensive dataset was curated to feed our predictive models. Our journey began with two primary sources:

- **SpaceX API**: Through automated GET requests to the SpaceX API, we retrieved detailed launch records. This step was vital to gather real-time, accurate data directly from the field.
- **Web Scraping Wikipedia**: We employed web scraping techniques to extract historical launch records from the "List of Falcon 9 and Falcon Heavy launches" Wikipedia page. This approach allowed us to compile a rich dataset that includes past mission records crucial for our analysis.

Here the Main (GitHub)

Data Collection

Each dataset underwent a rigorous cleaning and formatting phase, ensuring consistency and reliability. Key tasks in this phase included:

- **Parsing JSON**: After receiving JSON responses from the SpaceX API, we parsed and transformed the data into a structured format.
- **HTML Table Extraction**: Our scraping scripts were fine-tuned to accurately identify and parse tabular data from Wikipedia, converting it into a manipulable DataFrame.
- **Data Filtration**: We filtered the datasets to include only 'Falcon 9' launches, aligning our dataset with the project's focus.
- **Handling Missing Values**: Recognizing the impact of incomplete data on our analyses, we meticulously addressed missing values through imputation strategies, ensuring no gaps in the data.

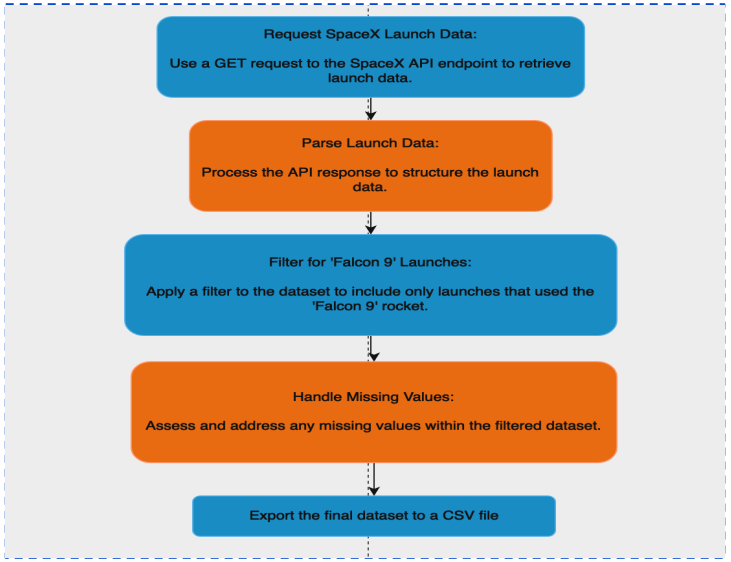
Here the Main (GitHub)

Data Collection – SpaceX API

Here the Main (GitHub)

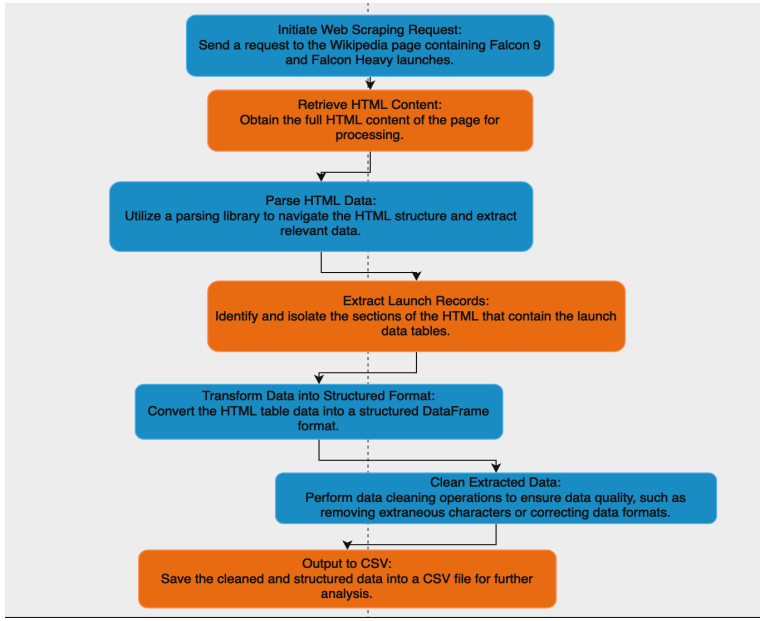
Data Collection with SpaceX API:

Leveraging SpaceX's API, I systematically retrieved launch data, ensuring meticulous record-keeping and analysis of the database also with external tools.
The GitHub repository (GitHub) documents our comprehensive process from API endpoint querying to JSON data extraction, parsing, and CSV output.



Data Collection – Scraping

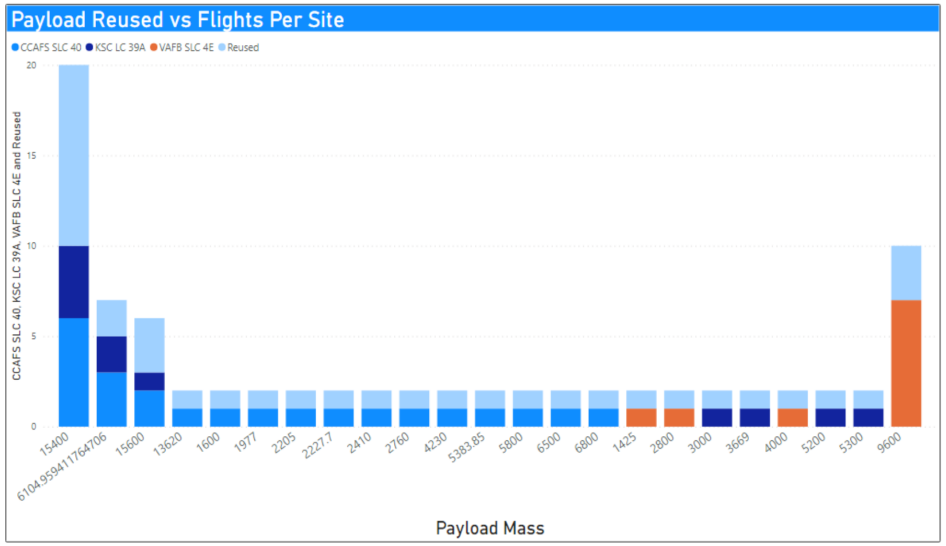
The data scraping procedure involves systematically requesting the Falcon 9 and Heavy launch records from Wikipedia, parsing through the HTML to extract launch data tables, and transforming this information into a clean, structured CSV file. The process is optimized for accuracy and efficiency, ensuring reliable data for subsequent analysis. (GitHub)



Here the Main (GitHub)

Data Collection – Payload Reused vs Flights Per Site

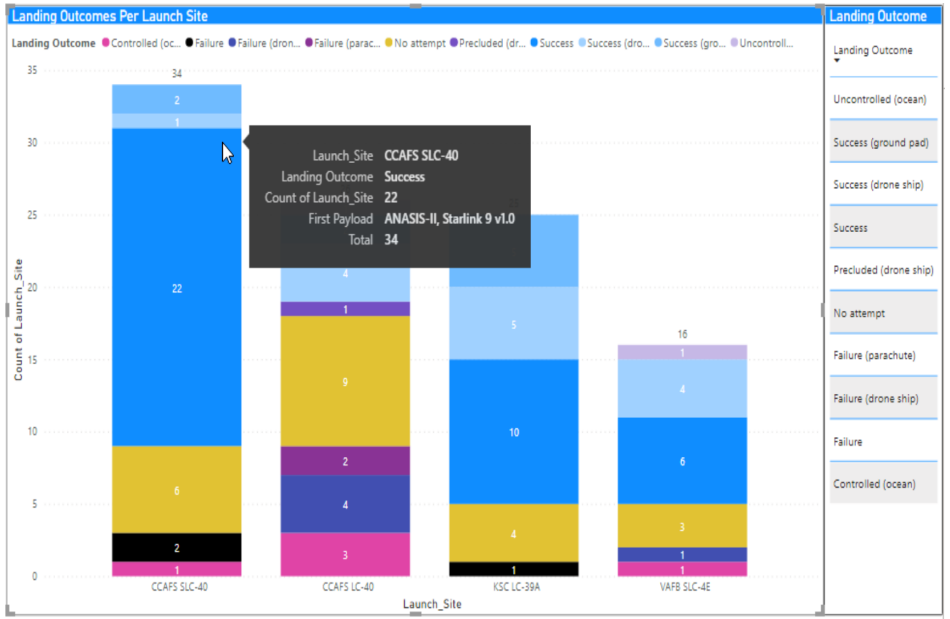
This bar chart delineates the correlation between payload mass and the number of times it was reused across different SpaceX launch sites. The dominant blue hues represent payloads that have been reused, revealing a trend where higher payload masses tend to have fewer reuses. Notably, CCAFS SLC 40 showcases the most significant reuse activity, indicative of its pivotal role in SpaceX's cost-cutting endeavors through payload recycling. Non Filtered.



Here the Main (GitHub)

Data Collection – Landing Outcomes Per Launch Site (1)

Through a series of stacked bar charts, we can scrutinize the diverse landing outcomes associated with each SpaceX launch site. Each color segment epitomizes a unique landing status, such as controlled ocean landings, successful drone ship landings, and outright failures. The data manifest a clear narrative: CCAFS SLC-40, a site with the most launches, also sports a complex mosaic of outcomes, while VAFB SLC 4E, with fewer launches, demonstrates a comparably higher proportion of successful ground pad landings.

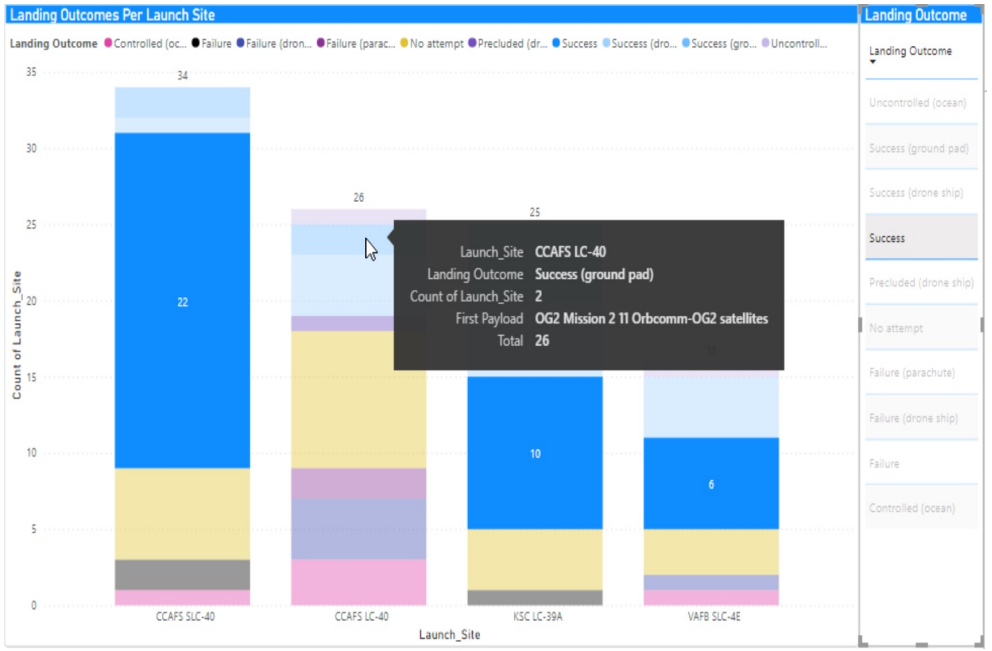


Here the Main (GitHub)

Here the Main (GitHub)

Data Collection – Landing Outcomes Per Launch Site (2)

Through a series of stacked bar charts, we can scrutinize the diverse landing outcomes associated with each SpaceX launch site. Each color segment epitomizes a unique landing status, such as controlled ocean landings, successful drone ship landings, and outright failures. The data manifest a clear narrative: CCAFS SLC-40, a site with the most launches, also sports a complex mosaic of outcomes, while VAFB SLC 4E, with fewer launches, demonstrates a comparably higher proportion of successful ground pad landings.



Here the Main (GitHub)

Here the Main (GitHub)

Data Collection – Landing Outcomes Per Launch Site (3)

Through a series of stacked bar charts, we can scrutinize the diverse landing outcomes associated with each SpaceX launch site. Each color segment epitomizes a unique landing status, such as controlled ocean landings, successful drone ship landings, and outright failures. The data manifest a clear narrative: CCAFS SLC-40, a site with the most launches, also sports a complex mosaic of outcomes, while VAFB SLC 4E, with fewer launches, demonstrates a comparably higher proportion of successful ground pad landings.

Here the Main (GitHub)

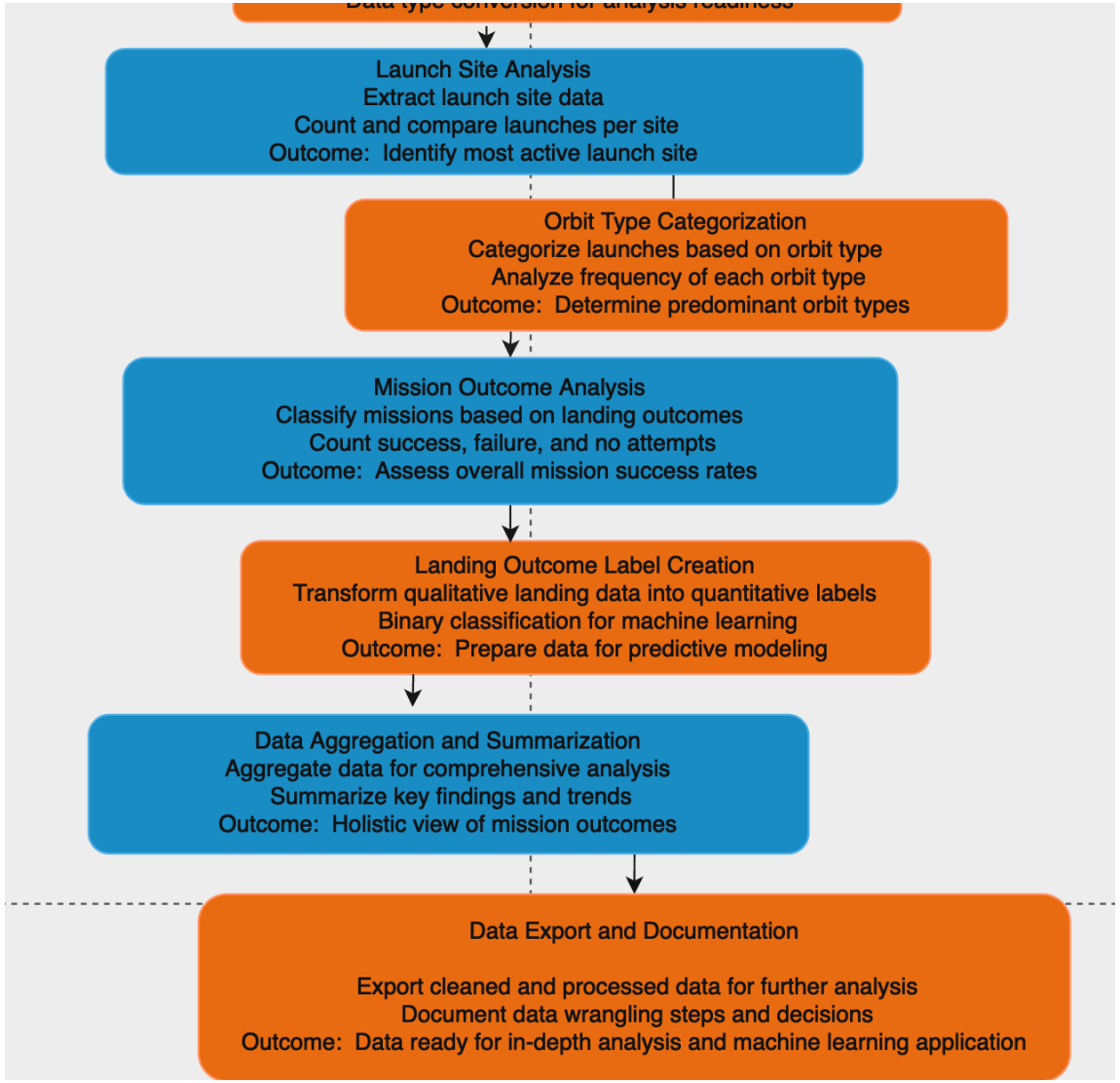
Data Collection – Landing Outcomes Per Launch Site (4)

Through a series of stacked bar charts, we can scrutinize the diverse landing outcomes associated with each SpaceX launch site. Each color segment epitomizes a unique landing status, such as controlled ocean landings, successful drone ship landings, and outright failures. The data manifest a clear narrative: CCAFS SLC-40, a site with the most launches, also sports a complex mosaic of outcomes, while VAFB SLC 4E, with fewer launches, demonstrates a comparably higher proportion of successful ground pad landings.

Here the Main (GitHub)

Data Wrangling Launch Site Analysis

- In the realm of data science, the process of data wrangling is a crucial step. My journey in this project embarked on a path of transforming raw SpaceX launch data into actionable insights.
- This phase involved meticulous data categorization, in-depth orbit analysis, precise outcome labeling, and assessing the efficiency of SpaceX's various launch sites.
- Each step was executed with care to ensure the integrity and relevance of the data, laying the groundwork for accurate predictive modeling and strategic decision-making.
- This overview highlights my methodical approach to uncovering the hidden stories within the SpaceX launch data.



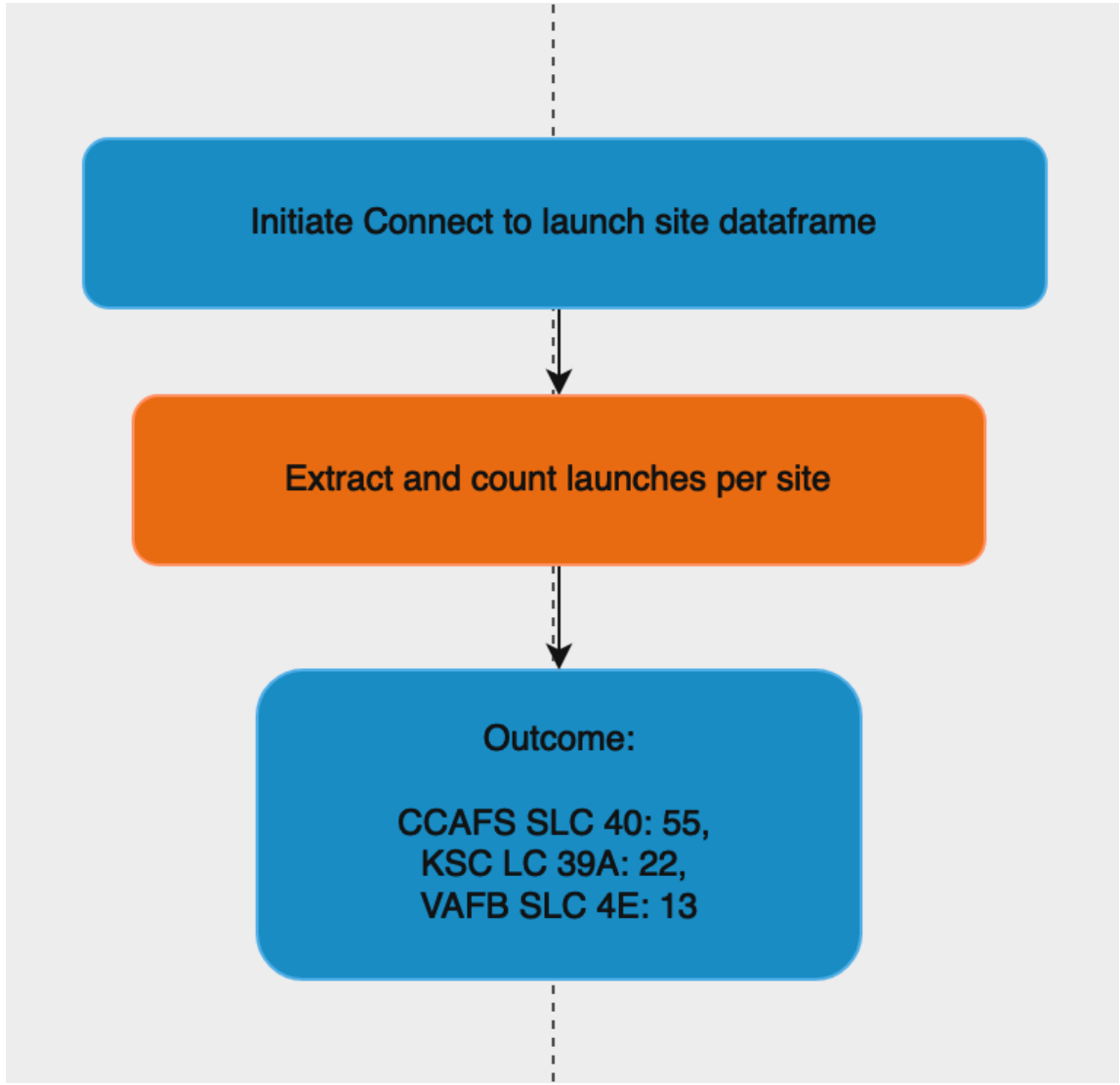
Here the Main (GitHub) - Here the (GitHub) of this Section

Data Wrangling Assessing Launch Site Utilization

- My analysis revealed distinct variations in launch activities across different sites, with **CCAFS SLC 40** emerging as the most frequently utilized site.
- This detailed examination allowed me to understand the distribution of launches, which is essential for efficient resource allocation and optimized scheduling.
- Through this analysis, I gained valuable insights into site-specific operational capacities and strategic planning for future missions.

Key Points:

CCAFS SLC 40: 55 launches
KSC LC 39A: 22 launches
VAFB SLC 4E: 13 launches



Here the Main (GitHub) - Here the (GitHub) of this Section

Data Wrangling Orbit Classification and Frequency

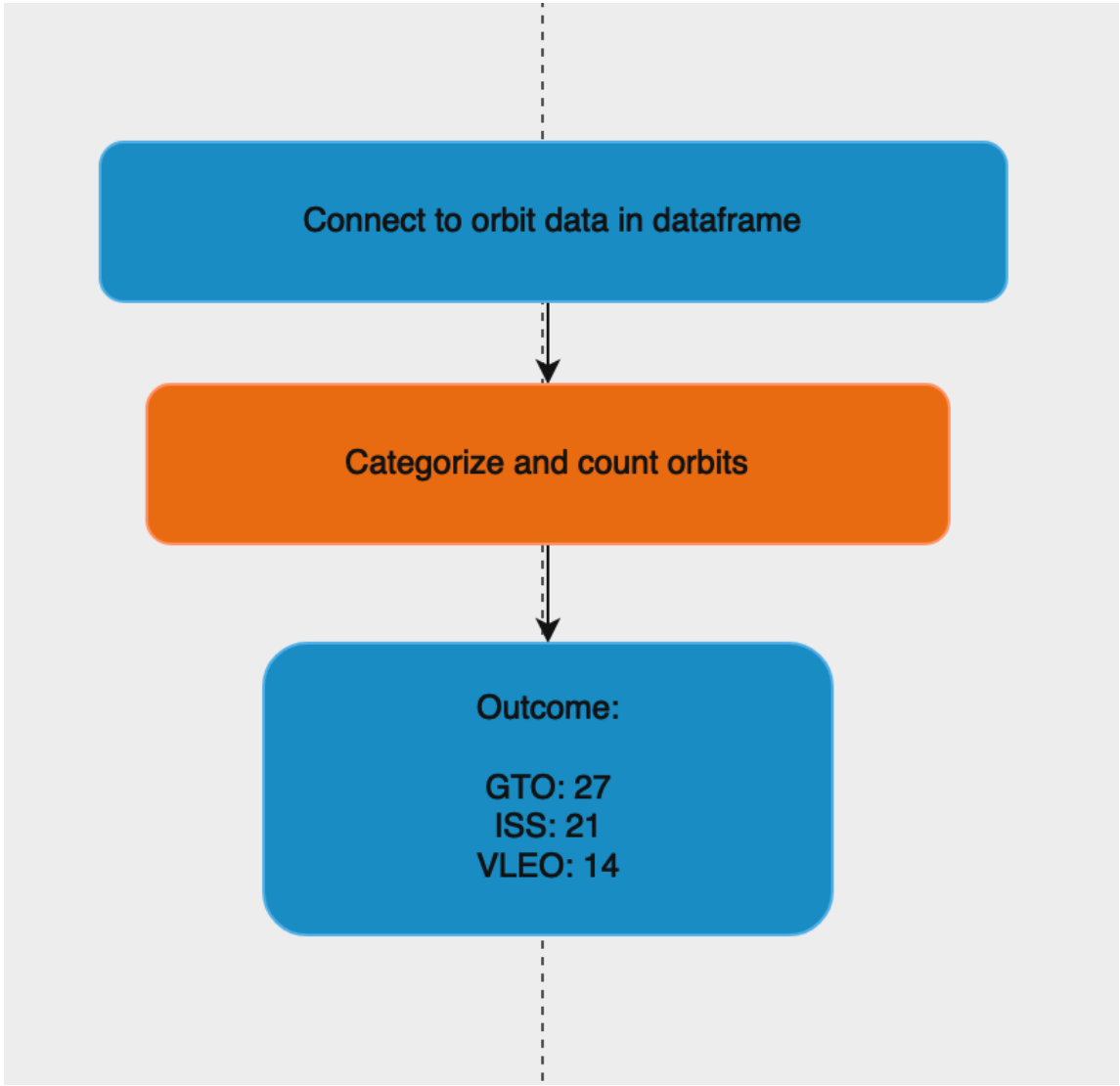
In my analysis, the orbit types used in the missions were indicative of their objectives.

The predominance of GTO orbits in the dataset highlighted a significant focus on geostationary transfer missions.

This orbit classification provided a deeper understanding of mission priorities and technological capabilities in achieving various orbital trajectories.

Key Points:

GTO: 27 occurrences
ISS: 21 occurrences
VLEO: 14 occurrences



Here the Main (GitHub) - Here the (GitHub) of this Section

Data Wrangling Orbit-Specific Landing Outcomes

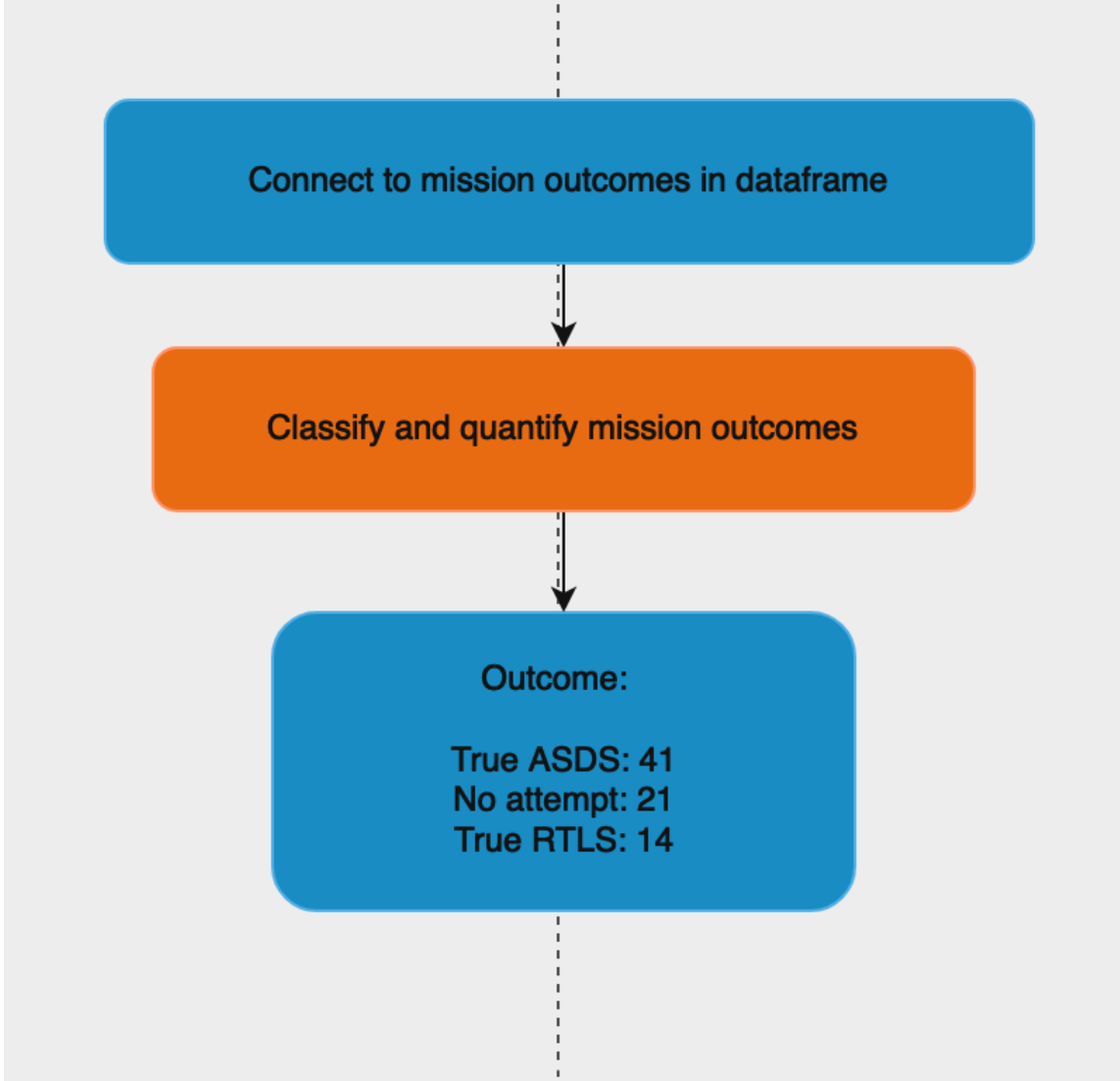
My analysis of mission outcomes by orbit type shed light on the unique challenges encountered in different orbital paths.

The high success rate in ASDS landings is a testament to the advancements in drone ship landing technologies and techniques.

This analysis was crucial in understanding the relationship between orbit types and landing success rates.

Key Points:

True ASDS: 41 successes
No attempt/Empty: 21
True RTLS: 14 successes



Here the Main (GitHub) - Here the (GitHub) of this Section

Data Wrangling Landing Outcomes and Classification

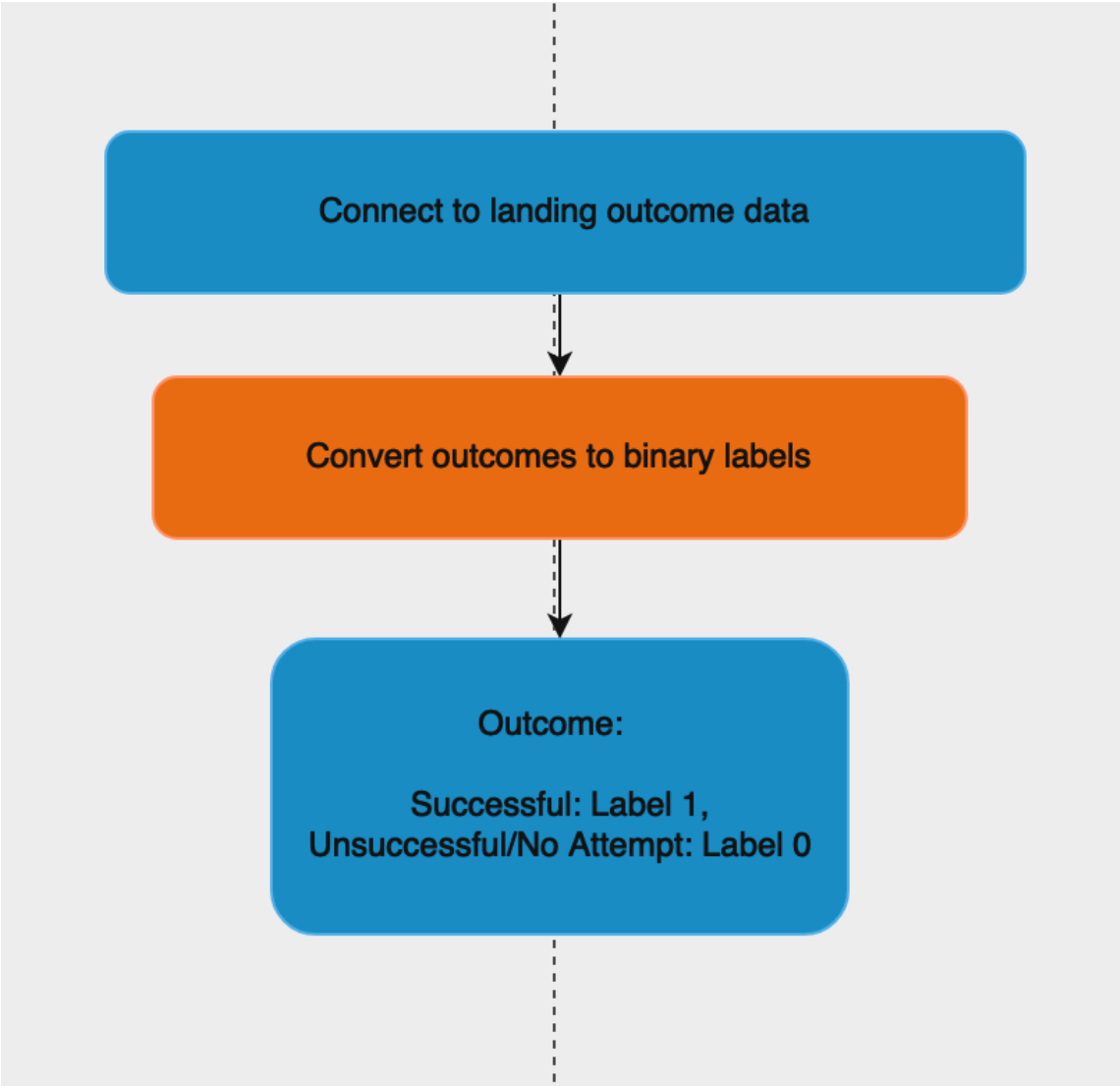
I transformed the various landing outcomes into binary classification labels to facilitate machine learning model training.

This step was instrumental in enabling accurate predictions regarding the success of future landings.

The classification process distinguished between planned and unplanned outcomes, enriching the dataset for more nuanced analysis.

Key Points:

Success (True Ocean/RTLS): = '1'
Failure (False ASDS/Ocean, None None/ASDS): = '0'



Here the Main (GitHub) - Here the (GitHub) of this Section

Data Wrangling Quantitative Summary of Mission Outcomes

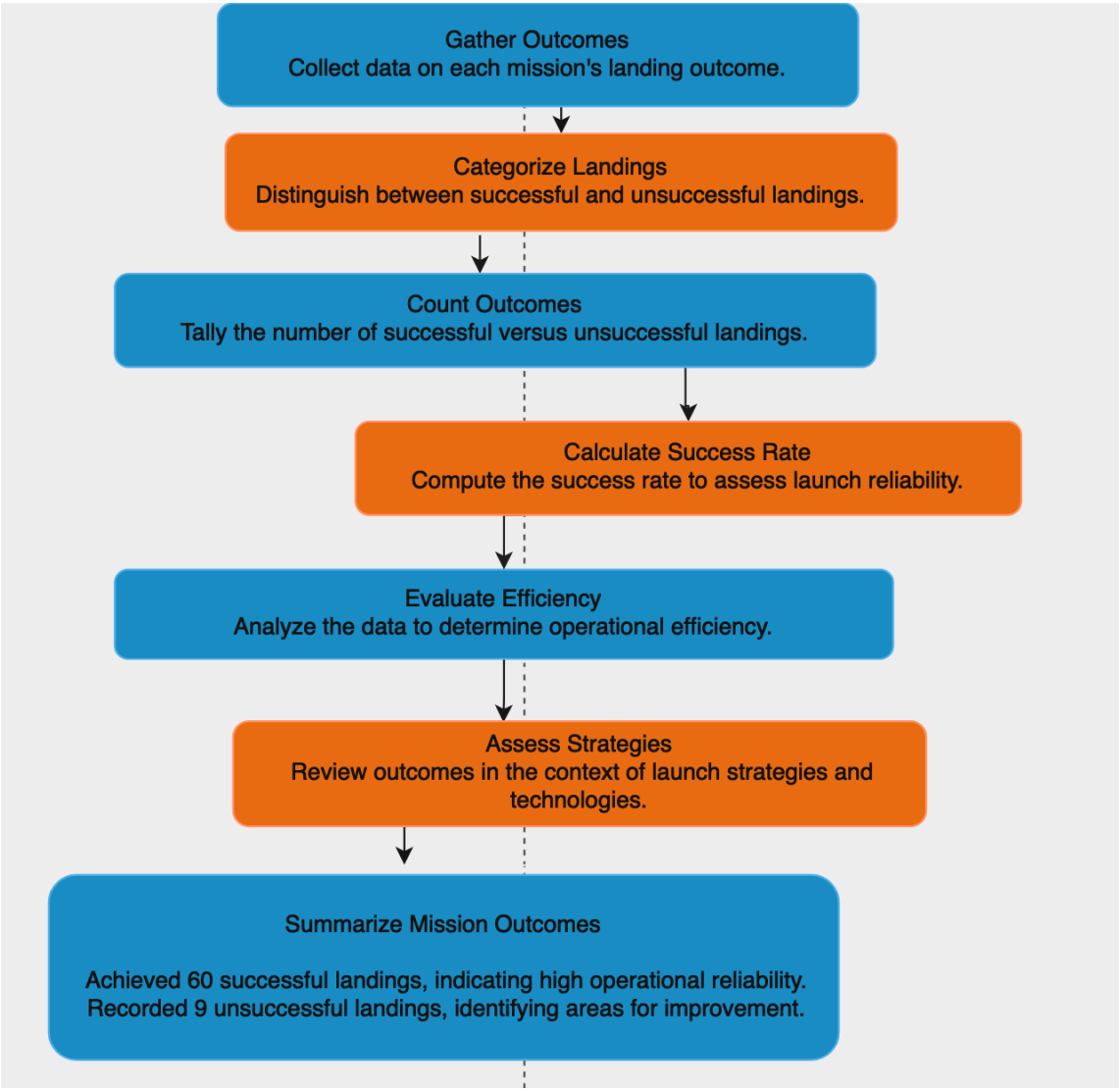
Summarizing the mission outcomes provided a clear depiction of the overall success rate.

This summary was crucial in evaluating the reliability and efficiency of launch operations.

Understanding these metrics allowed me to assess the effectiveness of different launch strategies and technologies.

Key Points:

Successful landings: 60
Unsuccessful landings: 9



Here the Main (GitHub) - Here the (GitHub) of this Section

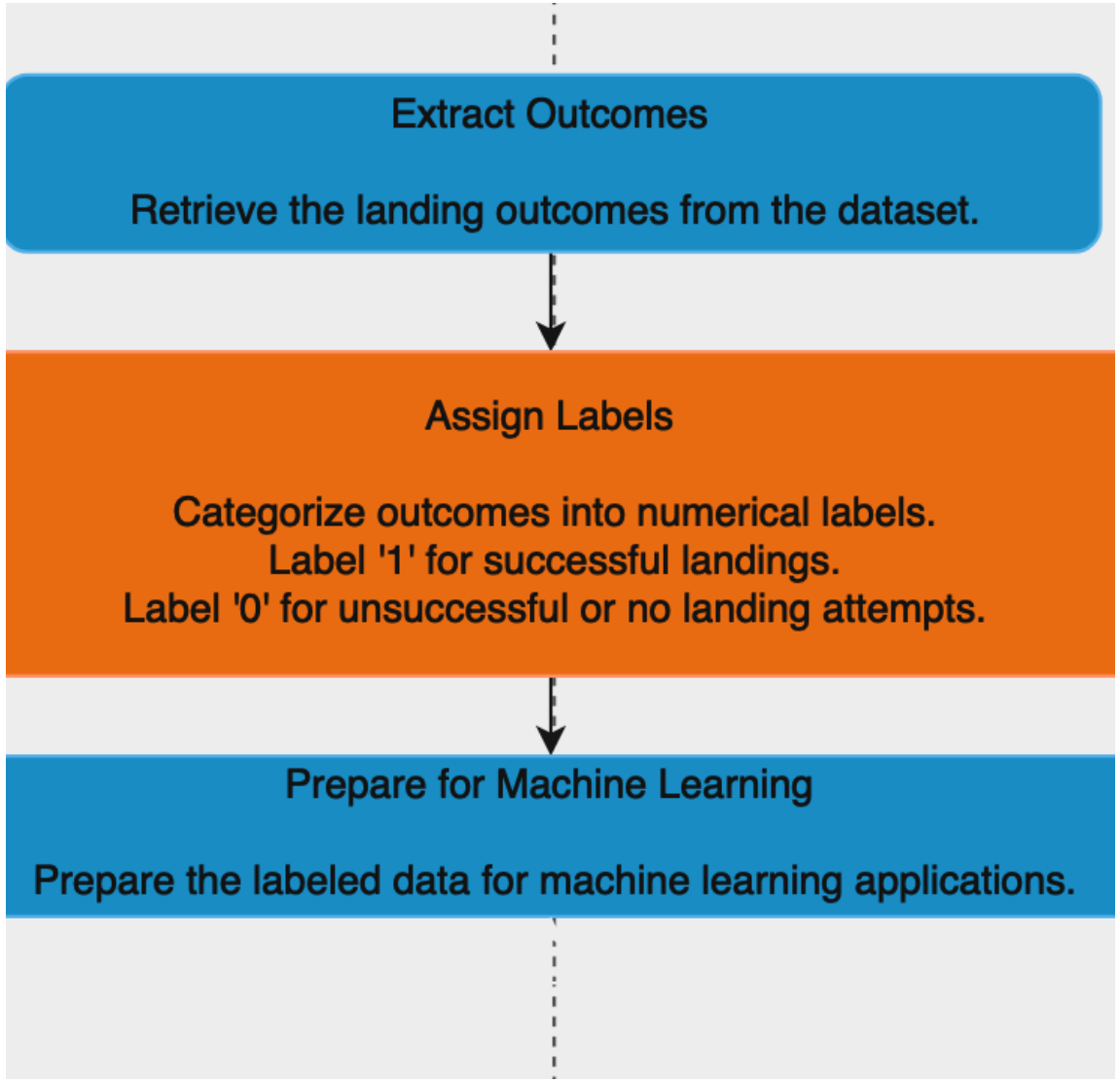
Data Wrangling Outcome Labeling for Machine Learning Readiness

Converting the outcomes into numerical labels was a critical step towards preparing the data for machine learning applications.

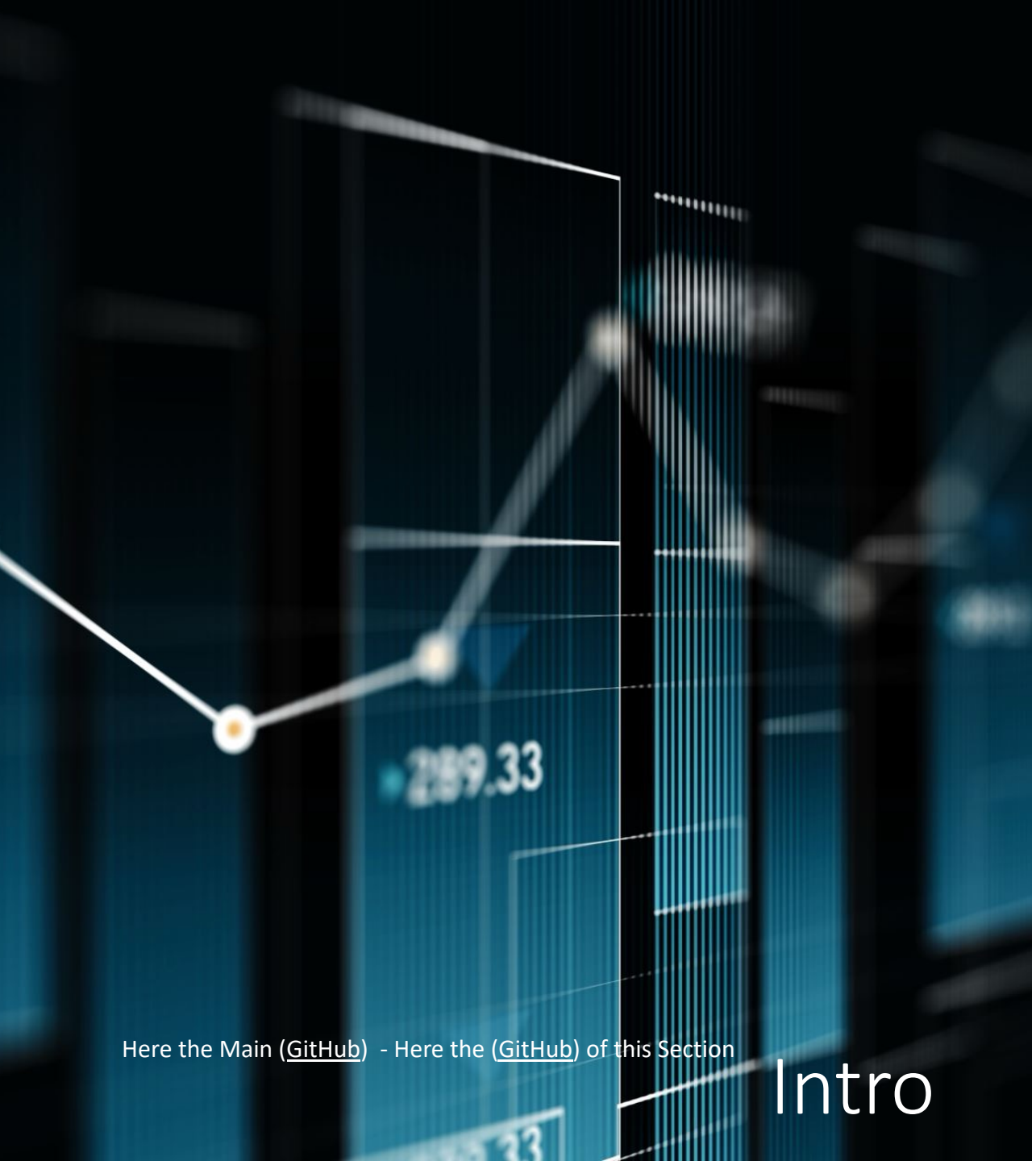
This process of labeling facilitated the use of supervised learning algorithms to predict the likelihood of successful landings, underscoring the potential of machine learning in enhancing space mission strategies.

Key Points:

Label '1': Indicative of successful landings
Label '0': Representative of unsuccessful or no landing attempts



Here the Main (GitHub) - Here the (GitHub) of this Section



Here the Main (GitHub) - Here the (GitHub) of this Section

Intro

EDA with Data Visualization

In this analysis, we delve into the world of SpaceX Falcon 9 rocket launches. SpaceX has revolutionized the aerospace industry by significantly reducing launch costs through reusability. Our objective is to predict the success of Falcon 9 first stage landings. By understanding the data and engineering relevant features, we aim to uncover insights that can inform decision-making and improve success rates.

Let's embark on this journey of exploration and preparation.

EDA with Data Visualization

Here the Main (GitHub) - Here the (GitHub) of this Section

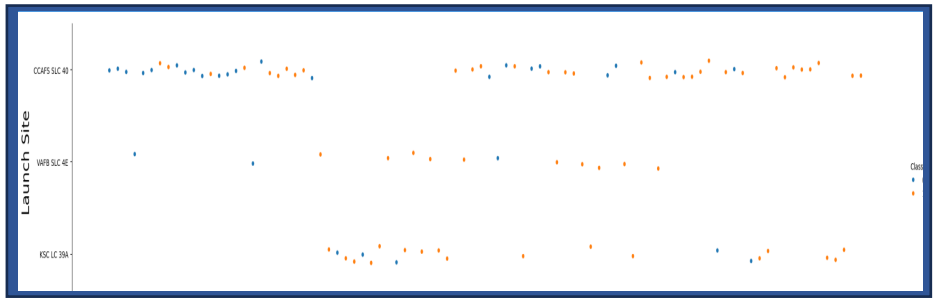
Task 1

Visualize the relationship between Flight Number and Launch Site

A scatter plot was created with Flight Number on the x-axis and Launch Site on the y-axis.

The hue represents the Class (0 for unsuccessful, 1 for successful) of landing.

This plot helps visualize the relationship between the flight number and launch site in terms of landing success.



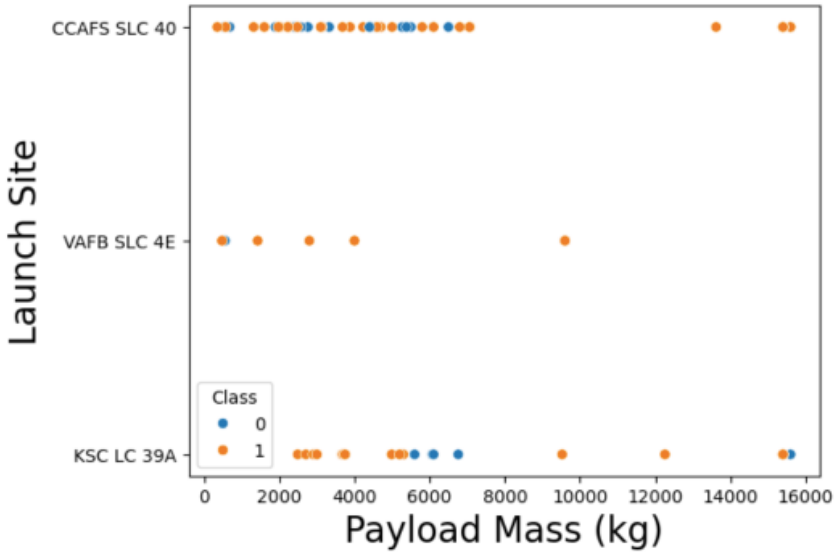
EDA with Data Visualization

Here the Main (GitHub) - Here the (GitHub) of this Section

Task 2

Visualize the relationship between Payload and Launch Site

Another scatter plot was generated with Payload Mass (kg) on the x-axis and Launch Site on the y-axis.
The hue indicates the landing success class.
This chart aims to show the relationship between payload mass and launch site concerning landing outcomes.



EDA with Data Visualization

Here the Main (GitHub) - Here the (GitHub) of this Section

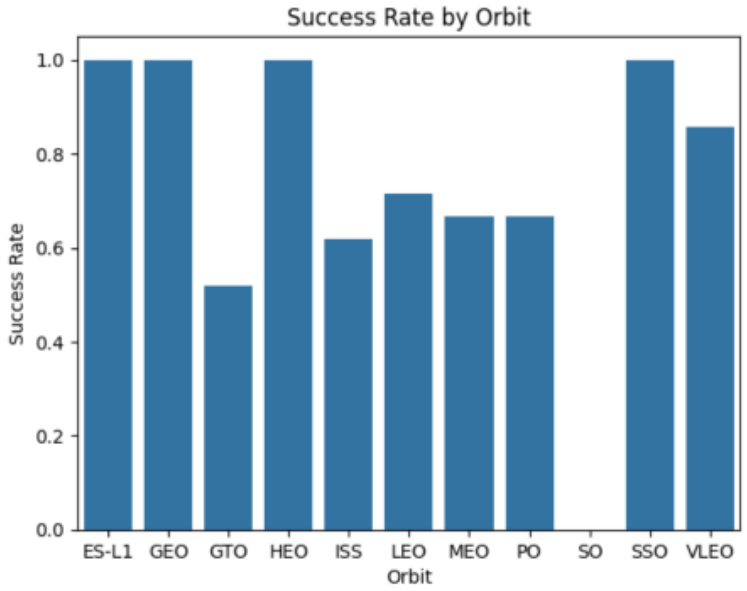
Task 3

Visualize the relationship between success rate of each orbit type

A bar chart displaying the success rate of each orbit type was plotted.

The success rate was calculated by grouping data by the Orbit column and calculating the mean of the Class column.

This chart provides insights into the success rates for different orbit types.



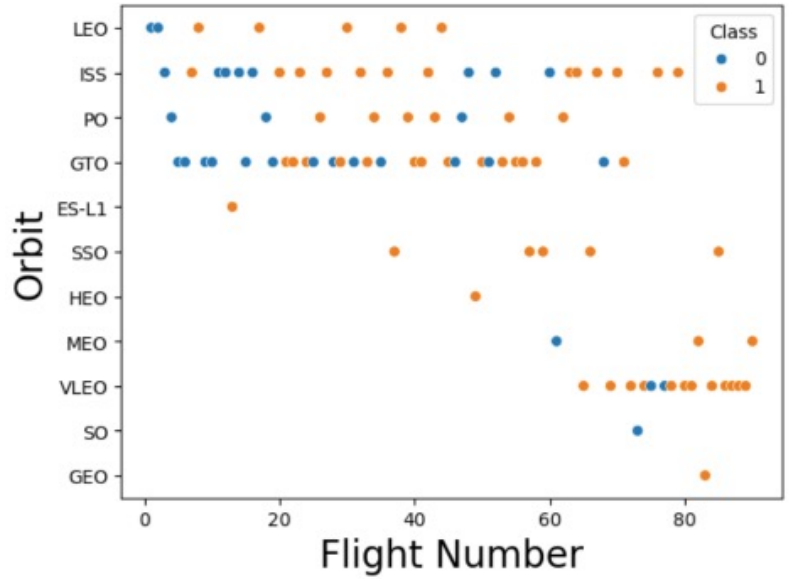
EDA with Data Visualization

Here the Main (GitHub) - Here the (GitHub) of this Section

Task 4

Visualize the relationship between Flight Number and Orbit type

Another scatter plot was created with Flight Number on the x-axis and Orbit on the y-axis. The hue represents the landing success class. This chart illustrates the relationship between flight numbers and orbit types in terms of landing success.



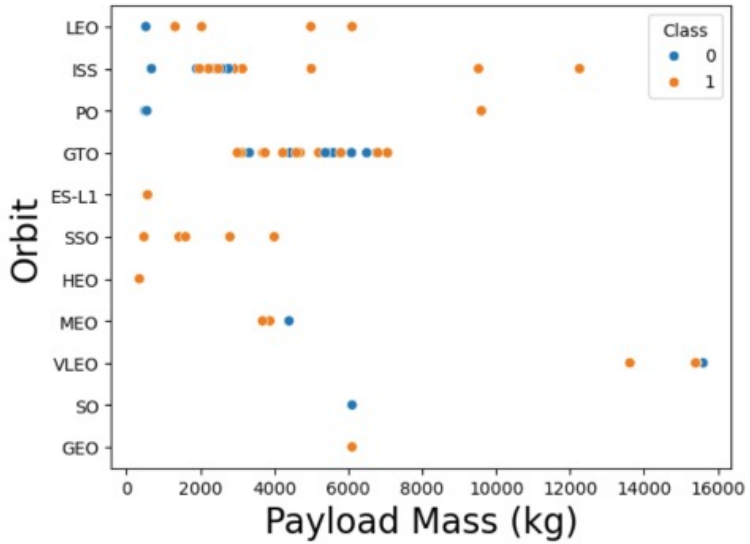
EDA with Data Visualization

Here the Main (GitHub) - Here the (GitHub) of this Section

Task 5

Visualize the relationship between Payload and Orbit type

A scatter plot was generated with Payload Mass (kg) on the x-axis and Orbit on the y-axis.
The hue indicates the landing success class.
This chart helps explore how payload mass relates to orbit types in terms of landing outcomes.



EDA with Data Visualization

Here the Main (GitHub) - Here the (GitHub) of this Section

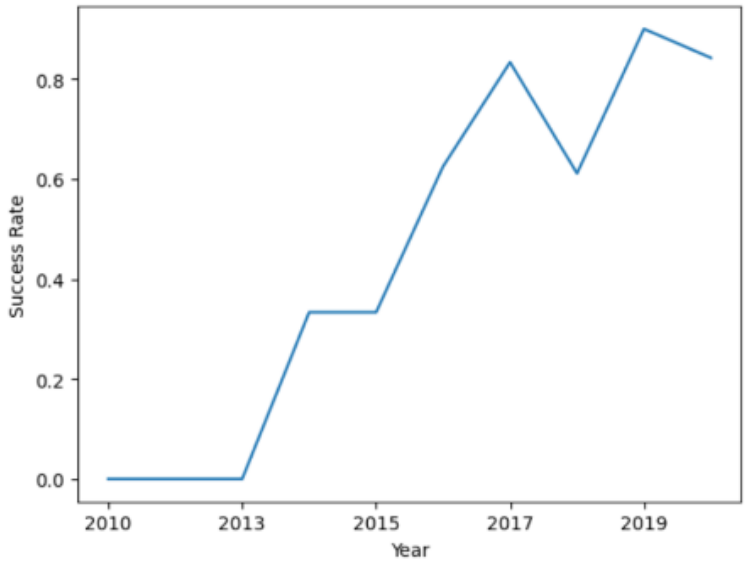
Task 6

Visualize the launch success yearly trend

A line chart was created to visualize the launch success trend over the years.

The 'Year' column was extracted from the 'Date' column, and the success rate was calculated per year.

This chart shows the annual trend in launch success.



EDA with Data Visualization

Here the Main (GitHub) - Here the (GitHub) of this Section

Task 7

Create dummy variables for categorical columns

In this step, dummy variables were created for categorical columns like Orbit, LaunchSite, LandingPad, and Serial.

This transformation converts categorical data into numerical format for machine learning models.

```
### TASK  7: Create dummy variables to categorical columns

features_one_hot = pd.get_dummies(features, columns=['Orbit', 'LaunchSite', 'LandingPad', 'Serial'])
features_one_hot.head()
```

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048
0	1	6104.959412	1	False	False	False	1.0	0	0	0	...	0
1	2	525.000000	1	False	False	False	1.0	0	0	0	...	0
2	3	677.000000	1	False	False	False	1.0	0	0	0	...	0
3	4	500.000000	1	False	False	False	1.0	0	0	0	...	0
4	5	3170.000000	1	False	False	False	1.0	0	0	0	...	0

EDA with Data Visualization

Here the Main (GitHub) - Here the (GitHub) of this Section

Task 8

Cast all numeric columns to float64

Finally, all numeric columns in the dataset were cast to the float64 data type to ensure uniform data types for modeling.

```
### TASK  8: Cast all numeric columns to `float64`

features_one_hot = features_one_hot.astype('float64')
features_one_hot.head()
```

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048
0	1.0	6104.959412	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0
1	2.0	525.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0
2	3.0	677.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0
3	4.0	500.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0
4	5.0	3170.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0

EDA with SQL

Here the Main (GitHub) - Here the (GitHub) of this Section

EDA with SQL 1

Display the names of the unique launch sites in the space mission
This query retrieves the unique launch site names from the dataset.

- SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;

Display 5 records where launch sites begin with the string 'CCA'
This query retrieves 5 records where the launch sites begin with the string 'CCA'.

- SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;

Here the (GitHub)

EDA with SQL

Here the Main (GitHub) - Here the (GitHub) of this Section

EDA with SQL 2

Display the total payload mass carried by boosters launched by NASA (CRS)
This query calculates the total payload mass carried by boosters launched by NASA (CRS).

- SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer LIKE 'NASA (CRS)';

Display average payload mass carried by booster version F9 v1.1
This query calculates the average payload mass carried by booster version F9 v1.1.

- SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1';

Here the (GitHub)

EDA with SQL

Here the Main (GitHub) - Here the (GitHub) of this Section

EDA with SQL 3

List the date when the first successful landing outcome in ground pad was achieved.
This query finds the earliest date when the first successful landing on a ground pad was achieved.

- SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success (ground pad)';

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
This query retrieves the names of boosters that had a successful landing on a drone ship with a payload mass between 4000 and 6000 kg.

- SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;

Here the (GitHub)

EDA with SQL

Here the Main (GitHub) - Here the (GitHub) of this Section

EDA with SQL 4

List the total number of successful and failure mission outcomes
This query counts the total number of successful and failed mission outcomes.

- SELECT
  SUM(CASE WHEN Mission_Outcome LIKE 'Success%' THEN 1 ELSE 0 END) AS Successful_Missions,
  SUM(CASE WHEN Mission_Outcome LIKE 'Failure%' THEN 1 ELSE 0 END) AS Failed_Missions
  FROM SPACEXTABLE;

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

- SELECT Booster_Version
  FROM SPACEXTABLE
  WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTABLE
  );

Here the (GitHub)

EDA with SQL

Here the Main (GitHub) - Here the (GitHub) of this Section

EDA with SQL 5

List the records which will display the month names, failure landing_outcomes in drone ship, booster versions, and launch_site for the months in year 2015.This query retrieves records that display the month names, failure landing outcomes in a drone ship, booster versions, and launch sites for the months in the year 2015.

```
WITH MonthData AS (
    SELECT
        substr(Date, 6,2) AS Month,
        Landing_Outcome,
        Booster_Version,
        Launch_Site
    FROM
        SPACEXTABLE
    WHERE
        substr(Date,0,5)='2015'
        AND Landing_Outcome LIKE 'Failure (drone ship)'
)
SELECT
    CASE
        WHEN Month='01' THEN 'January'
         WHEN Month='02' THEN 'February'
        WHEN Month='03' THEN 'March'
        WHEN Month='04' THEN 'April'
        WHEN Month='05' THEN 'May'
        WHEN Month='06' THEN 'June'
        WHEN Month='07' THEN 'July'
        WHEN Month='08' THEN 'August'
        WHEN Month='09' THEN 'September'
        WHEN Month='10' THEN 'October'
        WHEN Month='11' THEN 'November'
        WHEN Month='12' THEN 'December'
    END AS MonthName,
    Landing_Outcome,
    Booster_Version,
    Launch_Site
FROM
    MonthData;
```

EDA with SQL

Here the Main (GitHub) - Here the (GitHub) of this Section

EDA with SQL 6

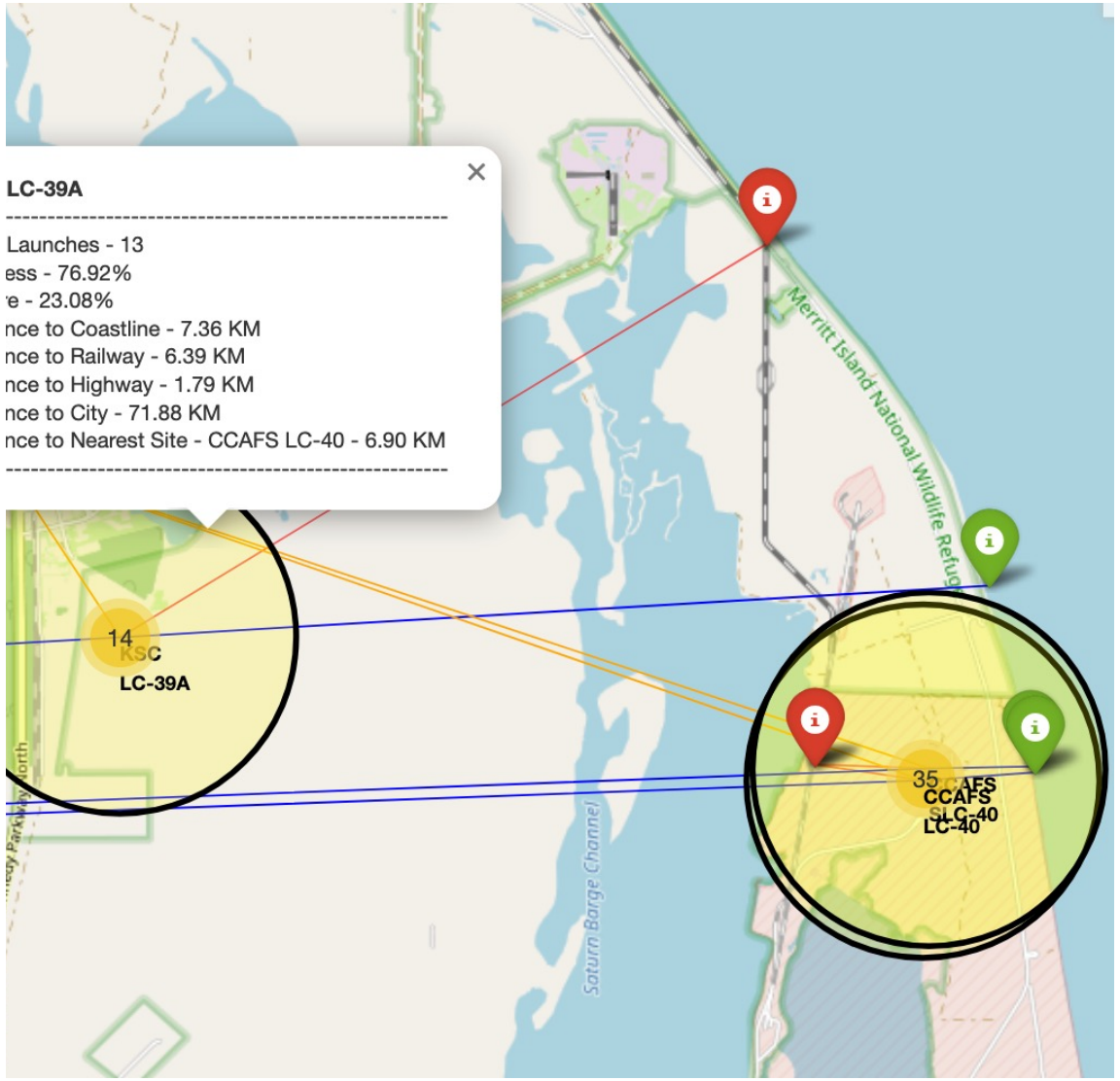
Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order. This query ranks the count of landing outcomes such as "Failure (drone ship)" or "Success (ground pad)" between the specified date range in descending order.

- ```
  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;
  ```
  ```

Build an Interactive Map with Folium

In this assignment, we explore the location of SpaceX launch sites and perform interactive visual analytics using Folium, a Python library for creating interactive maps. The goal is to visualize and analyze launch site locations, their proximity to key infrastructure (such as coastlines, railways, highways, and cities), and success rates.

Key Insights:
By visualizing the launch site location and its proximity to infrastructure, we can gain insights into SpaceX's strategic selection of launch sites.
This analysis helps us answer questions like whether launch sites are close to railways, highways, coastlines, and whether they maintain a safe distance from urban areas.
The use of Folium allows for interactive exploration of launch site data and geographic relationships.



Here the Main (GitHub) - Here the (GitHub) of this Section

Build an Interactive Map with Folium Map Markers

Green Markers for Coastline Coordinates:

These green markers represent the fixed coastline coordinates for specified launch sites.They help visualize the proximity of launch sites to coastlines, which can be important for oceanic launches, safety, and risk reduction.

Red Markers for Railway Coordinates:

These red markers represent the fixed railway coordinates for specified launch sites.

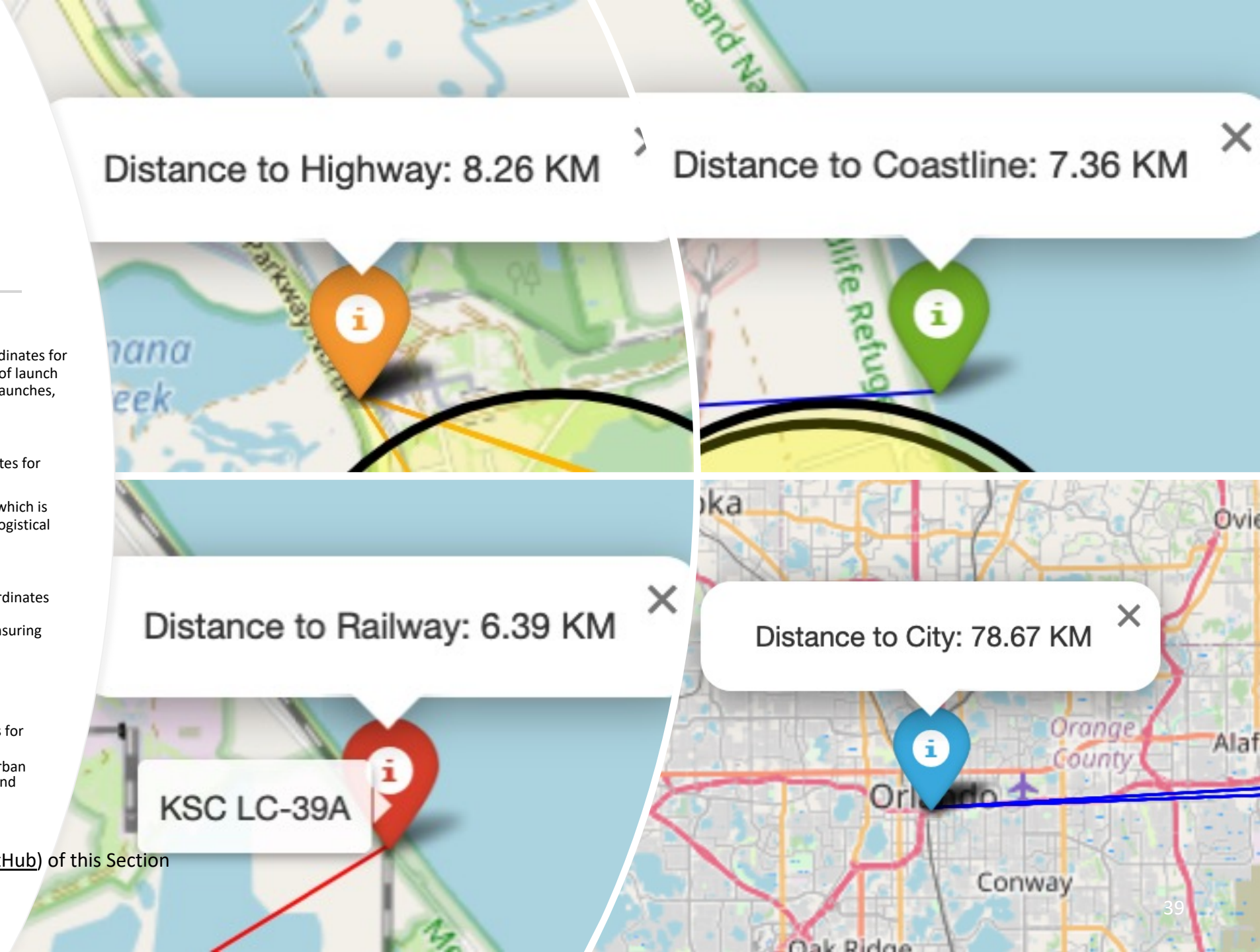
They indicate the proximity of launch sites to railways, which is crucial for transporting heavy payloads and enhancing logistical efficiency.

Orange Markers for Highway Coordinates:

These orange markers represent the fixed highway coordinates for specified launch sites.
They show the proximity of launch sites to highways, ensuring smooth transit of equipment and personnel.

Blue Markers for City Coordinates:

These blue markers represent the fixed city coordinates for specified launch sites.
They illustrate the distance between launch sites and urban areas, maintaining a safe distance for safety protocols and environmental considerations.

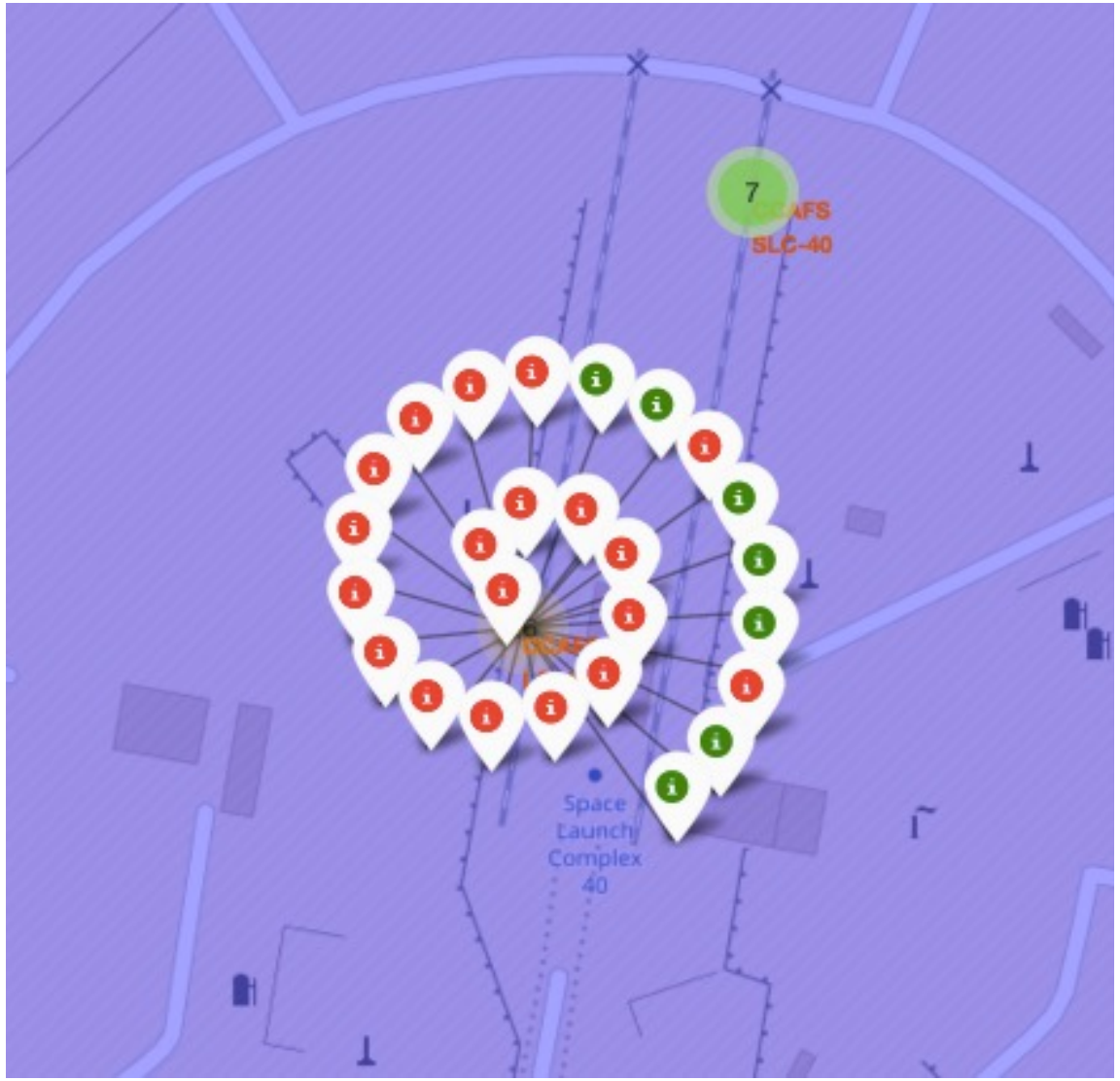


Here the Main (GitHub) - Here the (GitHub) of this Section

Build an Interactive Map with Folium Site Markers

In this assignment, we added 2 kind of markers for the success or not success of the landings using Marker Cluster.

A MarkerCluster is used to group markers for better map organization.It allows users to interact with clustered markers, making it more manageable when there are many markers on the map.



Here the Main (GitHub) - Here the (GitHub) of this Section

Build an Interactive Map with Folium – Polylines – Circles - Popups

PolyLines are used to connect launch sites to coastline, railway, highway, and city coordinates.
They visually display the distances between launch sites and these key points.

Circles with yellow fill color and black borders are created at the launch site coordinates.

Popup labels for these circles provide information about the launch site, including its name, success/failure rates, distances to coastline, railway, highway, city, and the nearest launch site.

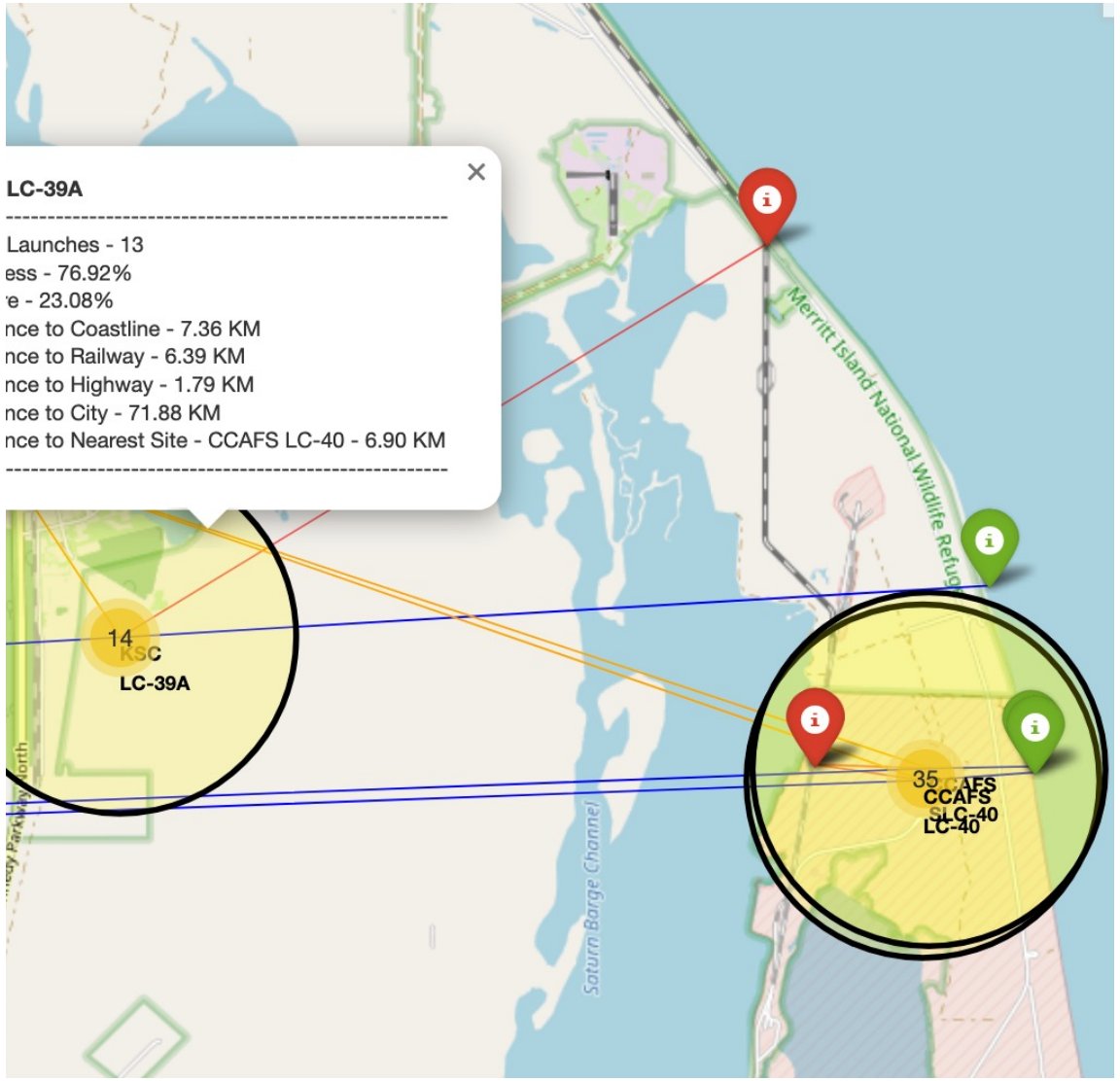
Here the Main (GitHub) - Here the (GitHub) of this Section

Build an Interactive Map with Folium

In this assignment, we explore the location of SpaceX launch sites and perform interactive visual analytics using Folium, a Python library for creating interactive maps. The goal is to visualize and analyze launch site locations, their proximity to key infrastructure (such as coastlines, railways, highways, and cities), and success rates.

Key Insights:
By visualizing the launch site location and its proximity to infrastructure, we can gain insights into SpaceX's strategic selection of launch sites.

This analysis helps us answer questions like whether launch sites are close to railways, highways, coastlines, and whether they maintain a safe distance from urban areas. The use of Folium allows for interactive exploration of launch site data and geographic relationships.



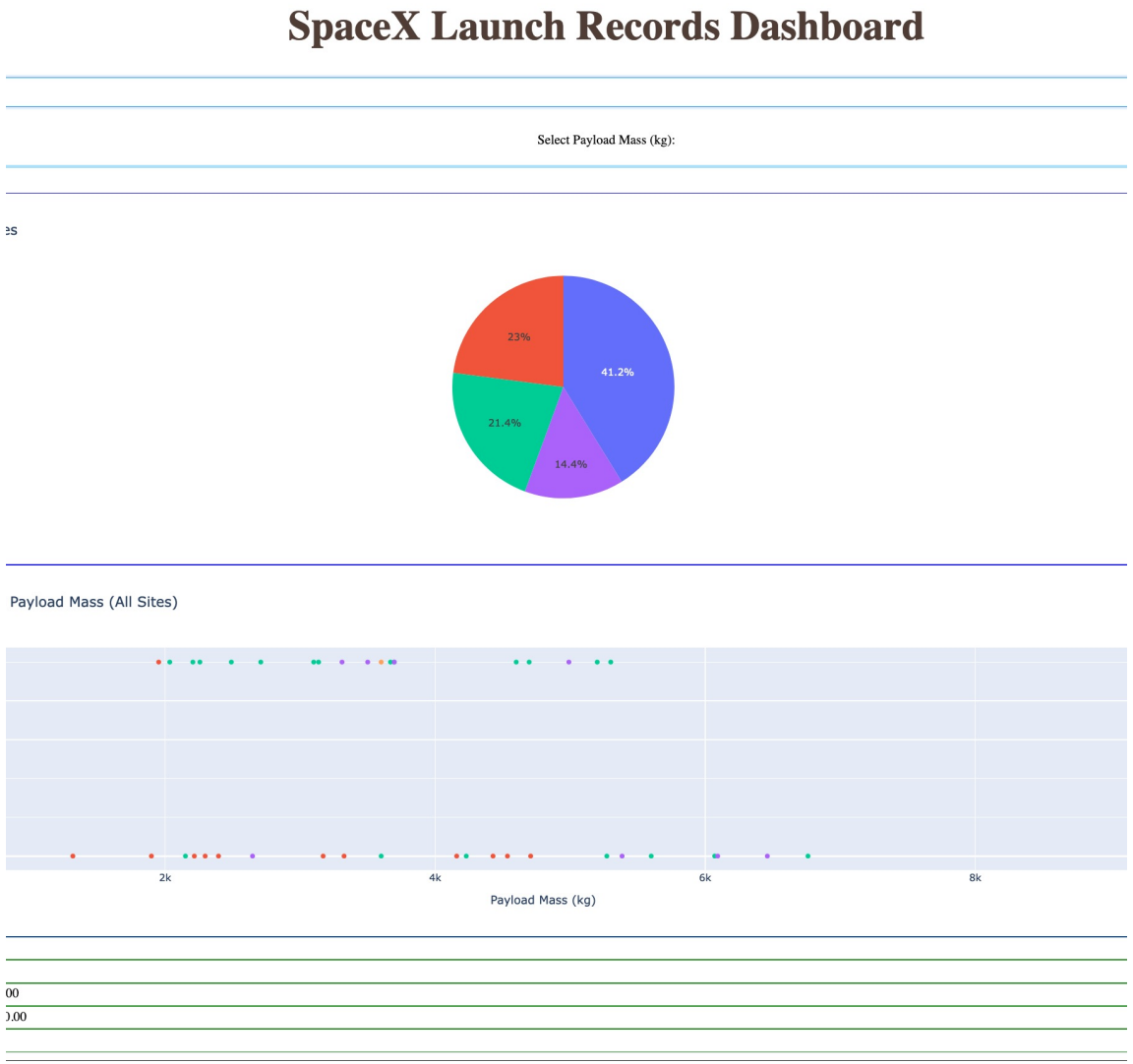
Here the Main (GitHub) - Here the (GitHub) of this Section

Build a Dashboard with Plotly Dash

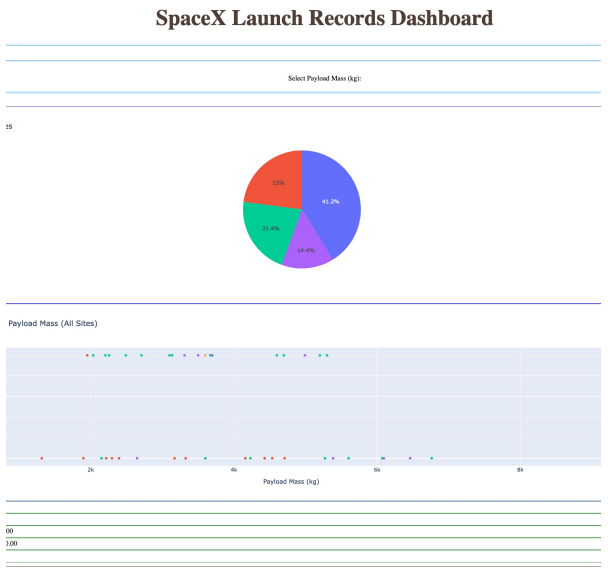
In the Plotly dashboard, the following plots/graphs and interactions have been added:

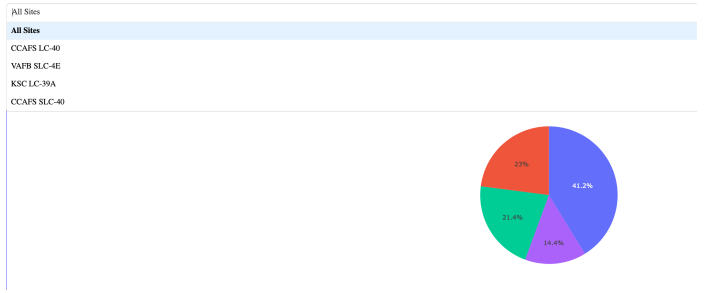
- **Dropdown List for Launch Site Selection**: A dropdown list has been added to enable the selection of launch sites. Users can choose from different launch sites, including "All Sites," "CCAFS LC-40," "VAFB SLC-4E," "KSC LC-39A," and "CCAFS SLC-40." This dropdown allows users to filter data based on launch site.
- **Payload Mass Slider**: A range slider is included to select the payload mass (in kilograms). Users can specify a range for payload mass between 0 and 10,000 kg.
- **Pie Chart - Success Rate**: A pie chart named "Success Rate for All Sites" displays the success rate of launches for all launch sites. The chart shows the percentage of successful launches relative to the total number of launches. This chart is interactive and updates based on the selected launch site.
- **Scatter Chart - Payload Mass vs. Success**: A scatter chart titled "Successful Launches by Payload Mass (All Sites)" shows the relationship between payload mass and launch success. It visualizes successful launches as points on the chart, with payload mass on the x-axis and success (1 for success, 0 for failure) on the y-axis. Users can filter data by launch site and payload mass range.

The interactions and visualizations are added to provide users with insights into SpaceX launch records and enable them to explore the success rates and payload masses associated with different launch sites. Users can interactively select launch sites and payload mass ranges to analyze the data more effectively. These plots and interactions are designed to facilitate data exploration and decision-making related to SpaceX launches.



Here the Main (GitHub) - Here the (GitHub) of this Section



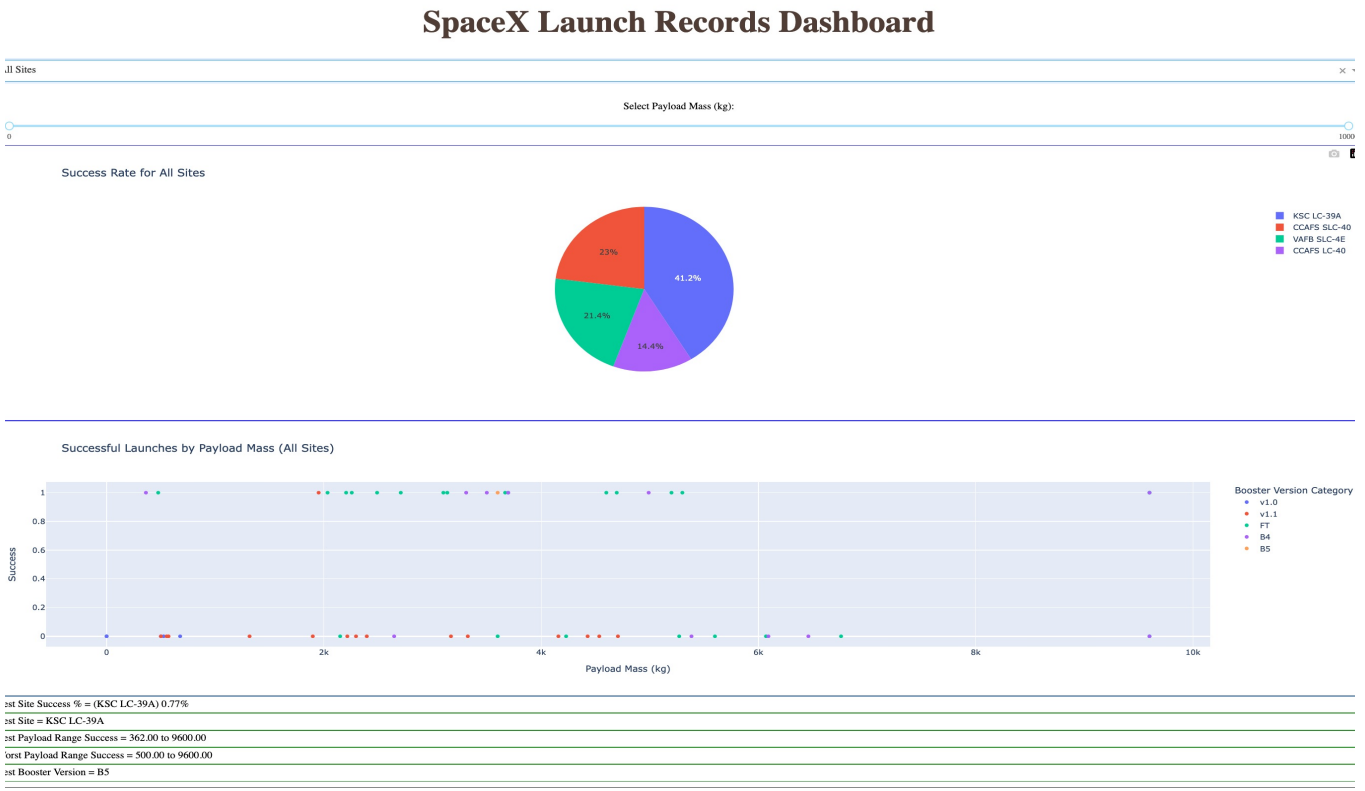


Build a Dashboard with Plotly Dash

Here the Main (GitHub) - Here the (GitHub) of this Section

Build a Dashboard with Plotly Dash

Full Dashboard Layout



Here the Main (GitHub) - Here the (GitHub) of this Section

Predictive Analysis (Classification)

Each of these methods was evaluated using cross-validation (cv=10) to estimate their performance on unseen data. The best hyperparameters for each method were determined through this evaluation, and their respective accuracies on the test data were calculated and presented. Additionally, confusion matrices were generated to visualize the performance of each model in terms of true positives, true negatives, false positives, and false negatives.
The best-performing method among these models is the Decision Tree with an accuracy of 0.8607 on the validation data. However, it should be noted that the accuracy on the test data is lower, indicating some overfitting.

1. **Logistic Regression:**
   - Logistic Regression is a binary classification algorithm that models the probability of an input belonging to a particular class.
   - In this case, it's used to predict whether the Falcon 9 first stage will land successfully (a binary outcome: Yes or No).
   - Hyperparameter tuning was performed using GridSearchCV to find the best parameters ('C', 'penalty', and 'solver') that maximize accuracy.
2. **Support Vector Machine (SVM):**
   - SVM is a powerful classification algorithm that finds a hyperplane that best separates data points of different classes.
   - Various kernels ('linear', 'rbf', 'poly', 'sigmoid') were tested along with different values of 'C' and 'gamma' for hyperparameter tuning.
   - The goal is to find the hyperparameters that maximize accuracy.
3. **Decision Tree:**
   - Decision Tree is a tree-like structure that makes decisions by splitting data based on feature values.
   - Hyperparameters include 'criterion', 'splitter', 'max_depth', 'max_features', 'min_samples_leaf', and 'min_samples_split'.
   - The objective is to find the best combination of hyperparameters that results in the highest accuracy.
4. **K-Nearest Neighbors (KNN):**
   - KNN is a simple yet effective classification algorithm that classifies data points based on the majority class among their nearest neighbors.
   - Parameters like 'n_neighbors', 'algorithm', and 'p' were tuned to optimize accuracy.
   - The algorithm tries different values of 'n_neighbors' and distance metrics ('p') to find the best configuration.

Here the(GitHub)

Here the Main (GitHub) - Here the (GitHub) of this Section

Predictive Analysis (Confusion Matrix Task 5)

```
#=================================================
#
# Calculate the accuracy on the test data using the method score
#
#=================================================

accuracy_on_test = logreg_cv.score(X_test, Y_test)
# The `score` method calculates the accuracy of the model on the test data.
# It compares the predicted values (`yhat`)
# to the true values (`Y_test`) and computes the accuracy.

# Print the accuracy on the test data
print("Accuracy on test data:", accuracy_on_test)
# This line prints the accuracy achieved by the
# logistic regression model on the test data.

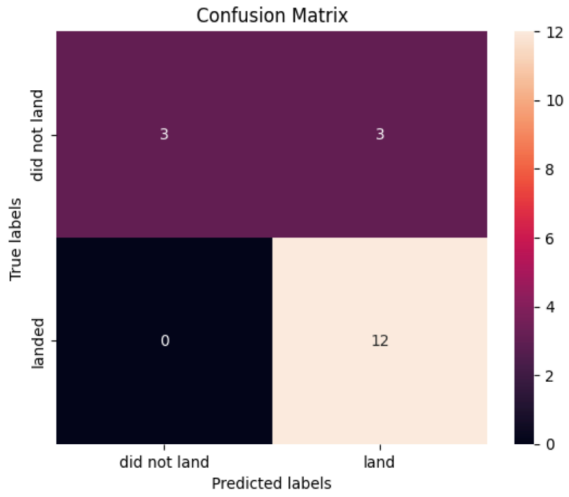
# Now, let's create a confusion matrix to visualize the model's performance.
from sklearn.metrics import confusion_matrix  # Import confusion_matrix function

# Predict the class labels for the test data
yhat = logreg_cv.predict(X_test)
# We use the trained model (logreg_cv)
# to predict class labels for the test data.

# Plot the confusion matrix
plot_confusion_matrix(Y_test, yhat)
# The `plot_confusion_matrix` function is
# a custom function defined earlier that
# displays the confusion matrix.
# It takes the true labels (`Y_test`) and predicted labels (`yhat`) as input.

#=================================================
#
# Calculate the accuracy on the test data using the method score
#
#=================================================
```

```
Accuracy on test data: 0.8333333333333334
```



Here the Main (GitHub) - Here the (GitHub) of this Section

Predictive Analysis (Confusion Matrix Task 7)

```
#====================================================
# Calculate the accuracy on the test data using the method score
#====================================================

svm_accuracy = svm_cv.score(X_test, Y_test)
# The score method calculates the accuracy of the SVM model on the test data.

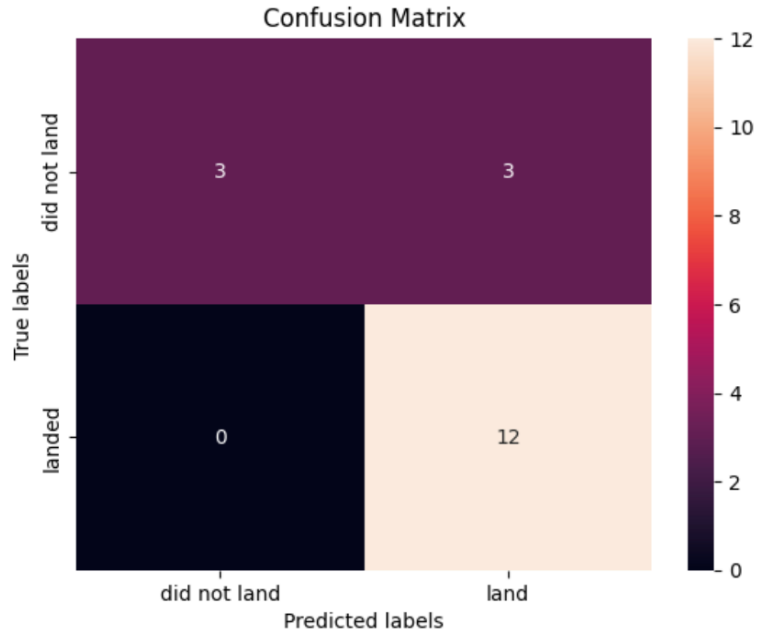
print("Accuracy on test data:", svm_accuracy)
# This line prints the accuracy achieved on the test data.

# Plot the confusion matrix
yhat_svm = svm_cv.predict(X_test)
# Make predictions on the test data using the trained SVM model.

plot_confusion_matrix(Y_test, yhat_svm)
# This line plots the confusion matrix to
# visualize the model's performance on the test data.

#====================================================
# Calculate the accuracy on the test data using the method score
#====================================================
```

```
Accuracy on test data: 0.8333333333333334
```



Here the Main (GitHub) - Here the (GitHub) of this Section

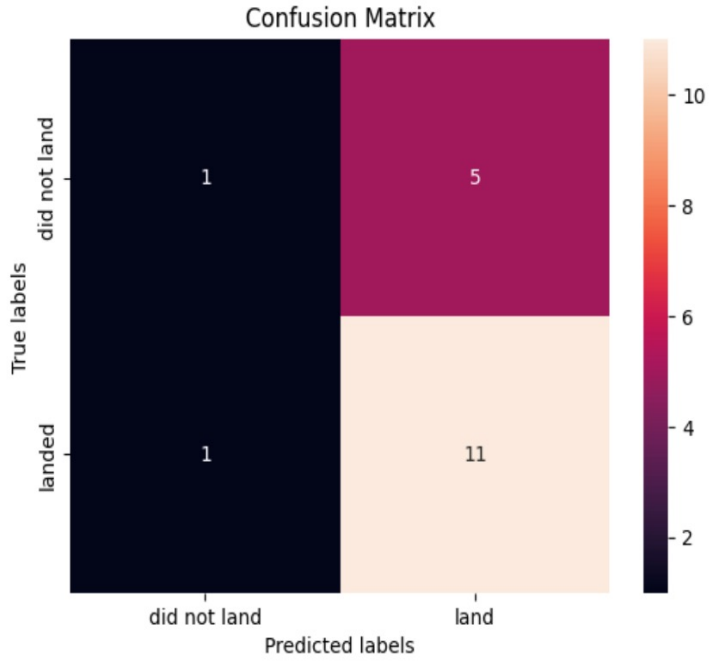
Predictive Analysis (Confusion Matrix Task 9)

```
#=====================================================
# Calculate the accuracy of the decision tree classifier on the test data
#=====================================================
accuracy_tree = tree_cv.score(X_test, Y_test)
print("Accuracy on Test Data (Decision Tree):", accuracy_tree)

# Plot the confusion matrix
yhat_tree = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test, yhat_tree)

#=====================================================
# Calculate the accuracy of the decision tree classifier on the test data
#=====================================================
```

```
Accuracy on Test Data (Decision Tree): 0.6666666666666666
```



Here the Main (GitHub) - Here the (GitHub) of this Section

Predictive Analysis (Confusion Matrix Task 11)

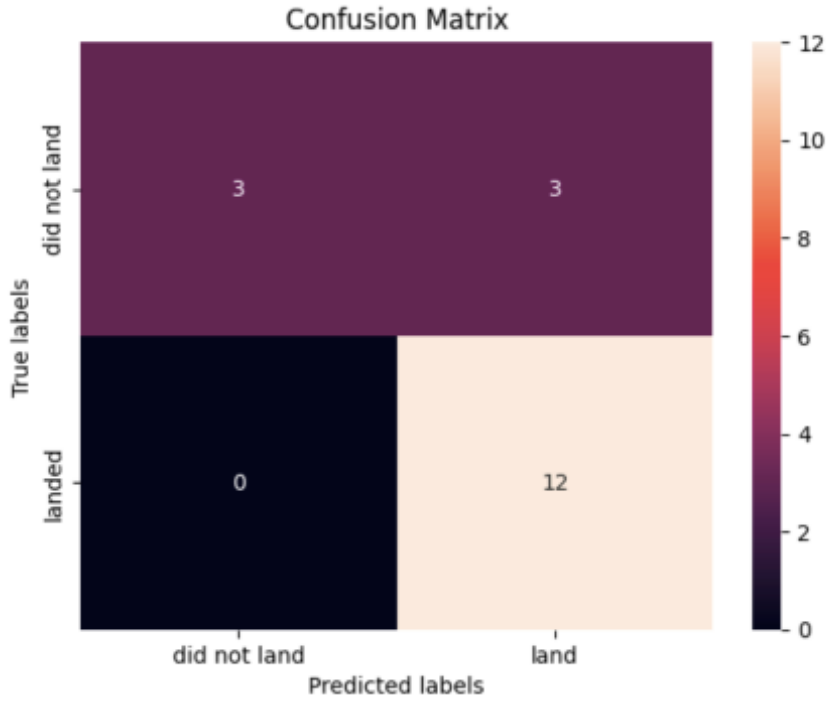
```
#=====================================================
# Calculate the accuracy of knn_cv on the test data using the method score
#=====================================================

accuracy = knn_cv.score(X_test, Y_test)
print("Accuracy on Test Data:", accuracy)

# We can plot the confusion matrix
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test, yhat)

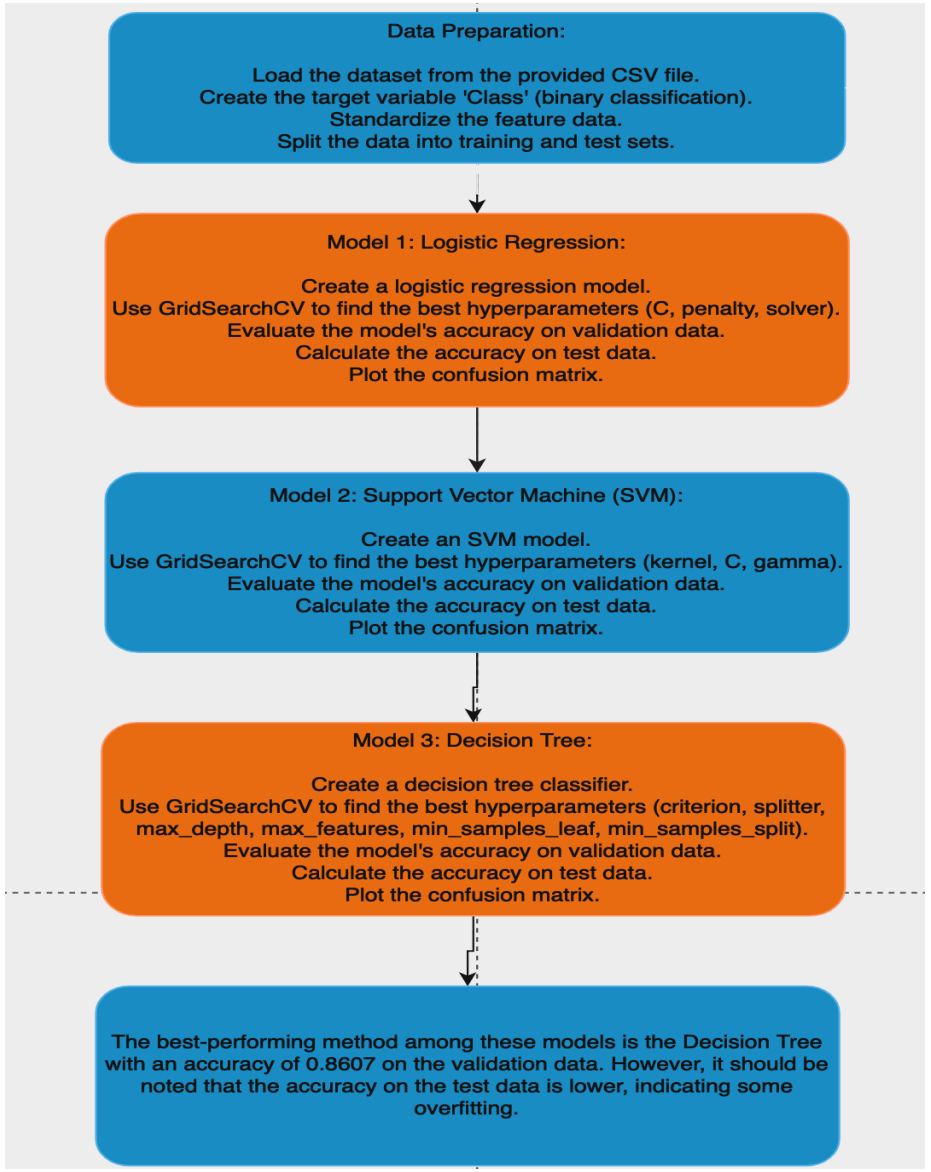
#=====================================================
# Calculate the accuracy of knn_cv on the test data using the method score
#=====================================================
```

```
Accuracy on Test Data: 0.8333333333333334
```



Here the Main (GitHub) - Here the (GitHub) of this Section

Predictive Analysis (Flow Chart)



Here the Main (GitHub) - Here the (GitHub) of this Section

Predictive Analysis (Results Wrangling)

Number of Launches on Each Site:

CCAFS SLC 40: 55 launches
KSC LC 39A: 22 launches
VAFB SLC 4E: 13 launches

Number and Occurrence of Each Orbit:
GTO: 27 occurrences
ISS: 21 occurrences
VLEO: 14 occurrences
PO: 9 occurrences
LEO: 7 occurrences
SSO: 5 occurrences
MEO: 3 occurrences
ES-L1: 1 occurrence
HEO: 1 occurrence
SO: 1 occurrence
GEO: 1 occurrence

Number and Occurrence of Mission Outcomes of the Orbits:
True ASDS: 41 occurrences
None None: 19 occurrences
True RTLS: 14 occurrences
False ASDS: 6 occurrences
True Ocean: 5 occurrences
False Ocean: 2 occurrences
None ASDS: 2 occurrences
False RTLS: 1 occurrence
Landing Outcome Label Creation:

Conclusion - Data Wrangling
In the process of data wrangling, several key insights and transformations have been performed on the dataset:

- We have gained an understanding of the number of launches from each launch site, with CCAFS SLC 40 being the most frequently used launch site.
- The occurrence of different orbit types has been quantified, with GTO and ISS being the most common orbits.
- We have analyzed the mission outcomes and found that most of the missions have successful outcomes, with a variety of landing scenarios.
- A new 'landing_class' label has been created to categorize mission outcomes as either successful, failures, or empty.

These data wrangling steps have prepared the dataset for further exploration and analysis, providing a solid foundation for extracting meaningful insights and patterns related to Falcon 9 rocket launches and their outcomes.
The clean and structured data is now ready for in-depth exploratory data analysis and predictive analysis.

Here the Main (GitHub) - Here the (GitHub) of this Section

Predictive Analysis (Results DataViz)

The data visualizations unveiled essential insights. Payload mass doesn't significantly affect landing success,
as most launches result in success. Certain orbits, such as ISS and GTO, exhibit higher success rates, with a consistency of success over the years. These findings guided feature engineering and model selection, ultimately leading to the decision tree algorithm's choice as the best-performing model with 86.07% accuracy on validation data. However, it's worth noting that the test data showed slightly lower accuracy, suggesting potential overfitting.

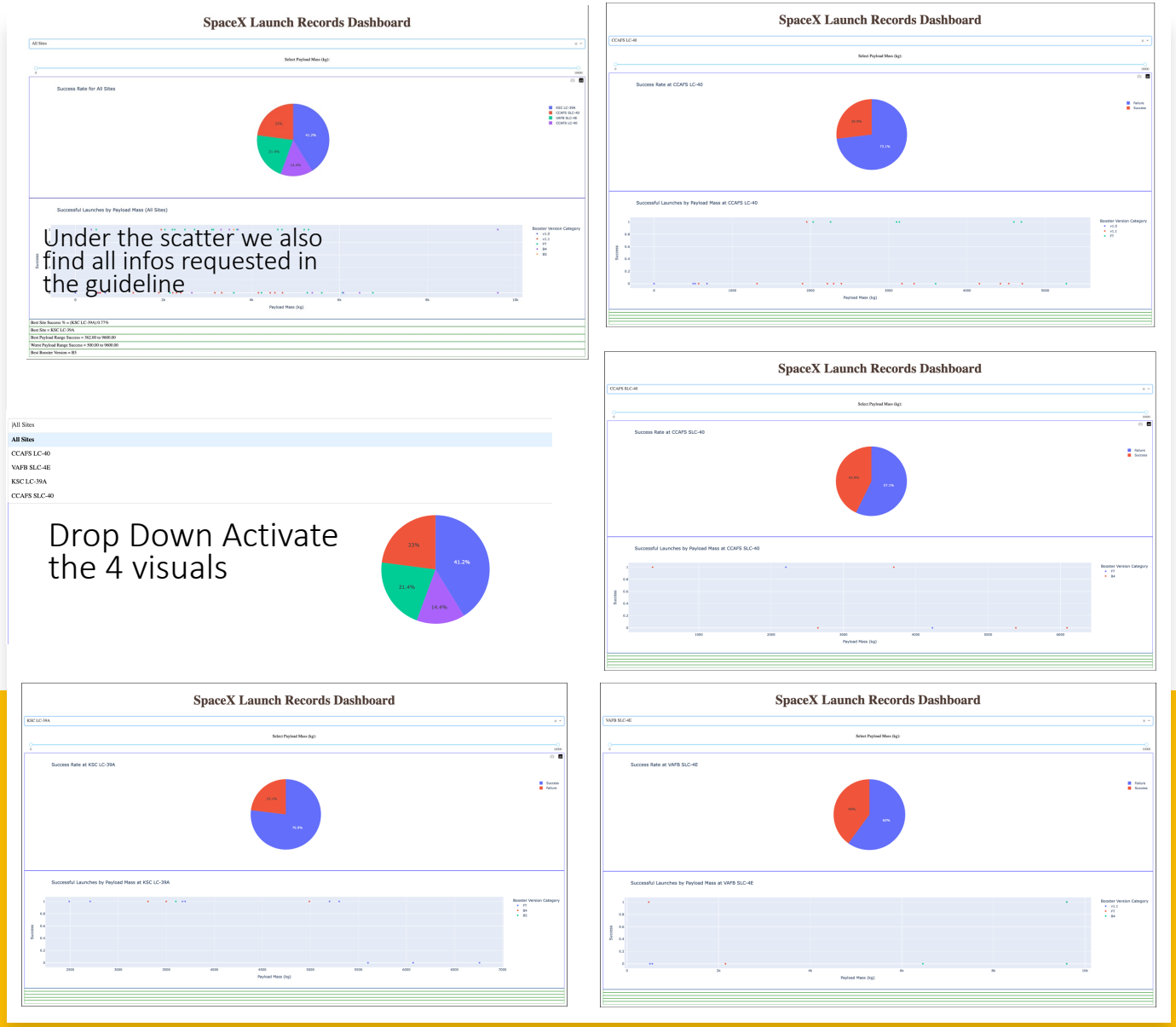
The data visualizations provided valuable insights:

- Flight Number vs. Launch Site: No clear pattern indicating payload mass affects landing success. Most launches result in success.
- Payload Mass vs. Launch Site: Payload mass varies widely, but success is consistent regardless of mass.
- Success Rate by Orbit: Certain orbits have higher success rates, with ISS and GTO having more data points.
- Flight Number vs. Orbit: No clear trend of success rate across orbits based on flight number.
- Payload Mass vs. Orbit: Heavier payloads common in GTO, but a mix of outcomes regardless of mass.
- Yearly Success Rate: Success rates consistent over the years.
- Features Engineering: Data prepared for modeling, including one-hot encoding categorical variables.

These visualizations provided critical insights for predictive analysis.

Here the Main (GitHub) - Here the (GitHub) of this Section

Predictive Analysis (DashBoard)



Here the Main (GitHub) - Here the (GitHub) of this Section

Predictive Analysis (Results ML)

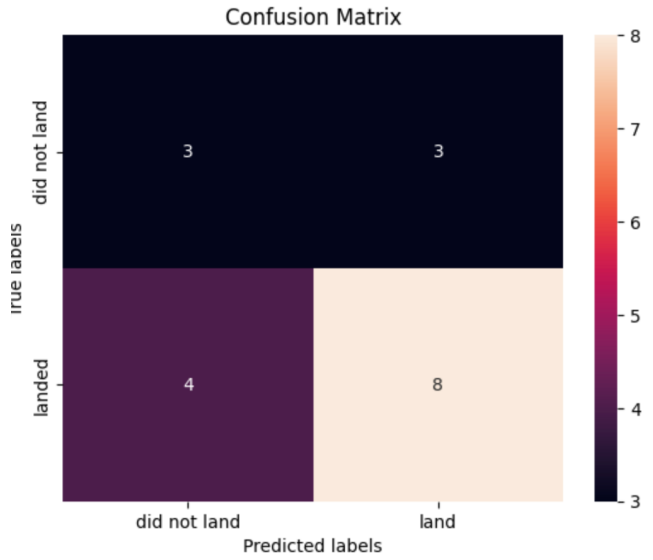
My analysis reveals that the Decision Tree model excelled in predicting Falcon 9 first stage landings.

Key points:

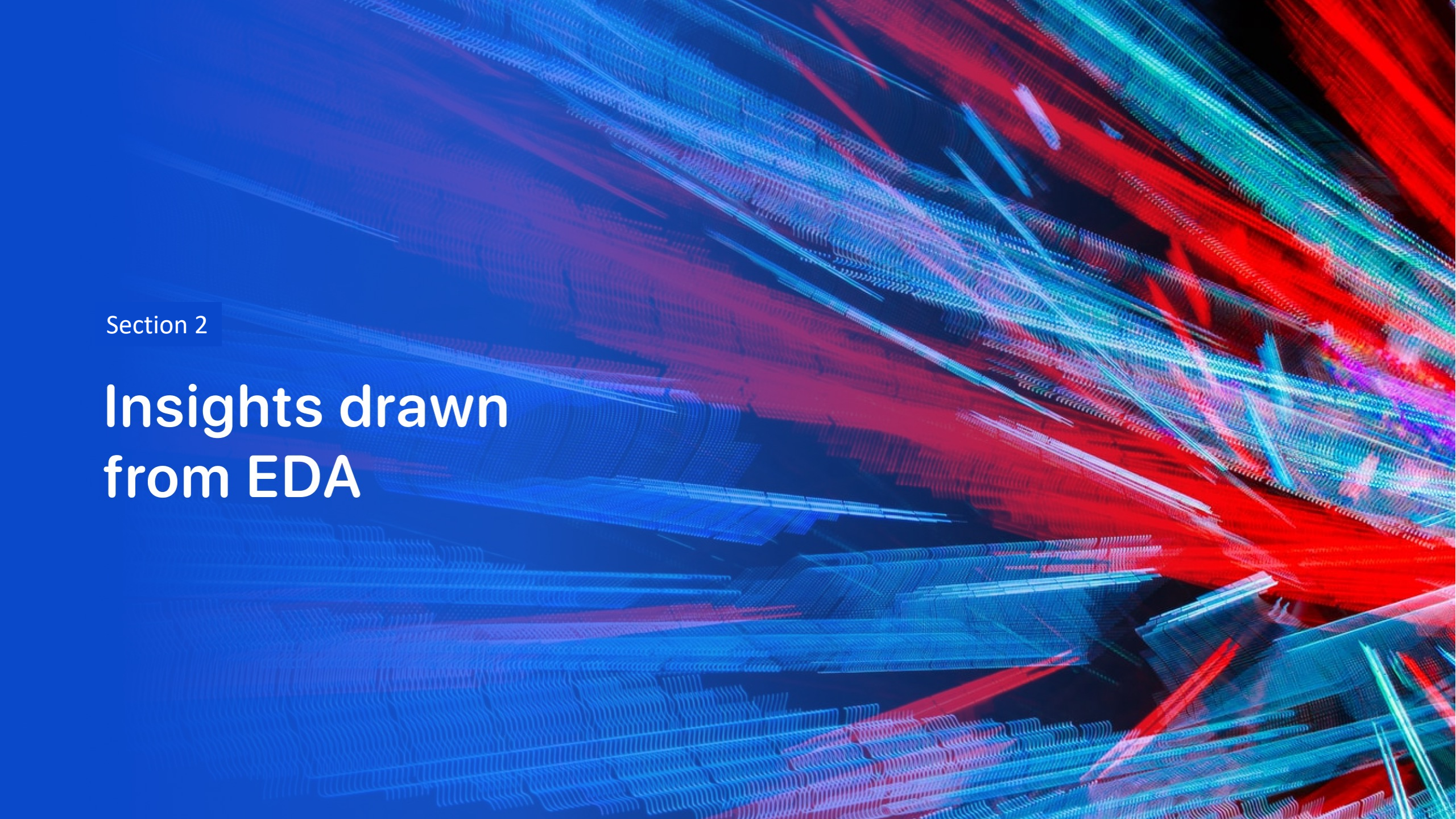
- **Decision Trees mimic human decision-making.**
- **Optimized with hyperparameters: criterion, splitter, max depth, max features, min samples leaf, and min samples split.**
- **Decision Tree's interpretability is valuable.**
- **Achieved an impressive accuracy of 0.8607 on validation data.**
- **Slight overfitting observed on test data.**
- **Visualize performance using confusion matrix.**
- **Decision Tree: Making complex decisions with high accuracy.**

In conclusion, the Decision Tree model's capacity to make data-driven decisions, coupled with careful hyperparameter tuning, led to its outstanding performance in predicting Falcon 9 first stage landings. While it exhibited some overfitting, its interpretability and accuracy make it the preferred choice for this critical task.

The confusion matrix will offer a detailed visual representation of its performance and further validate its suitability.



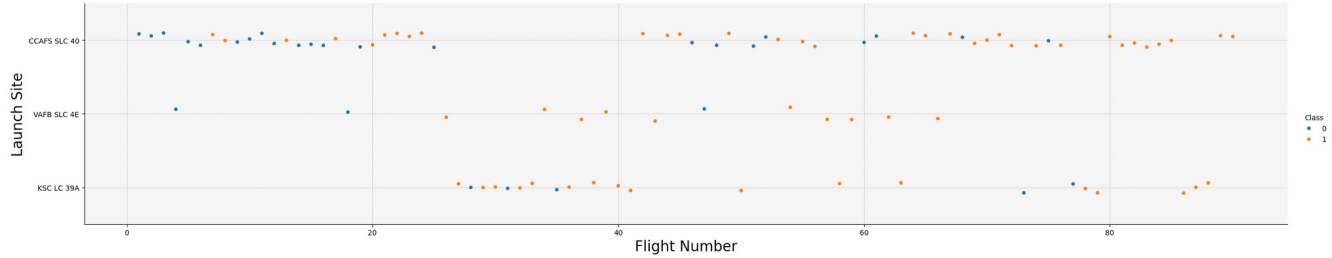
Here the Main (GitHub) - Here the (GitHub) of this Section


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

Here the Main (GitHub) - Here the (GitHub) of this Section

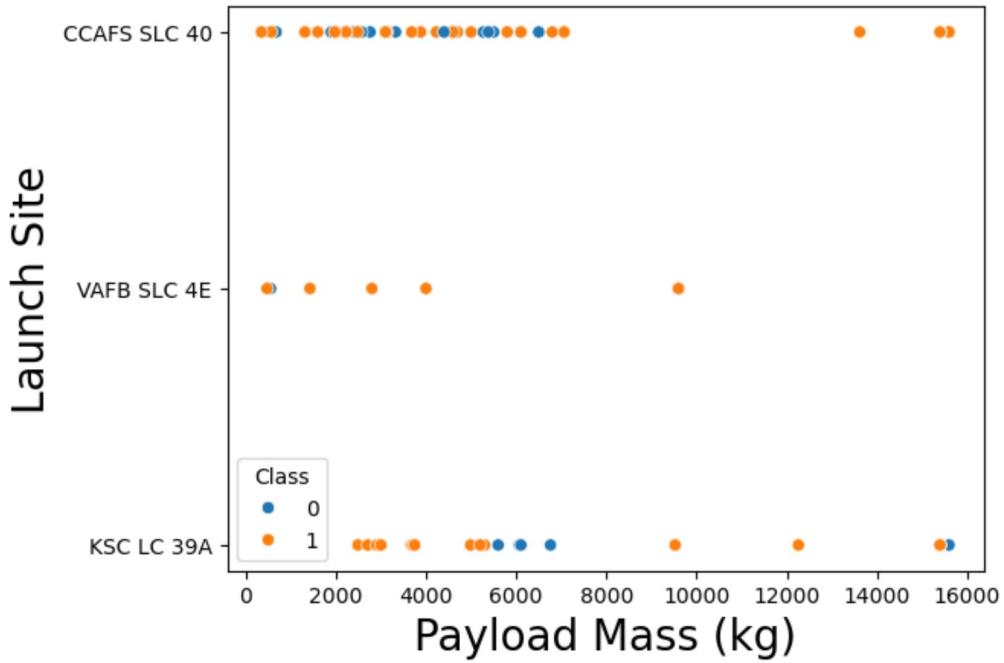
The scatter plot depicts the relationship between each flight's number and the launch site from which it was launched. Data points are differentiated by color based on a binary 'Class', where blue represents class 0 and orange represents class 1. The x-axis represents the flight number, which is an indicator of the sequence of launches, and the y-axis represents three different launch sites. There is no clear trend that indicates any correlation between the flight number and the launch site with respect to the outcome class. The plot suggests that launches occur at each site with varying outcomes, without a discernible pattern related to the flight number.



Payload vs. Launch Site

The displayed scatter plot illustrates the relationship between the payload mass of rockets and the launch sites they were launched from. On the x-axis, we have the payload mass in kilograms, showcasing a range from light to heavy payloads. The y-axis enumerates three distinct launch sites: CCAFS SLC 40, VAFB SLC 4E, and KSC LC 39A.

The plot suggests that the success of landings is not solely dependent on the payload mass, as evidenced by successful landings across the entire range of payload masses. The data points at the lower end of the payload mass spectrum also indicate successful landings, suggesting that factors other than payload mass may play a significant role in landing outcomes.



Here the Main (GitHub) - Here the (GitHub) of this Section

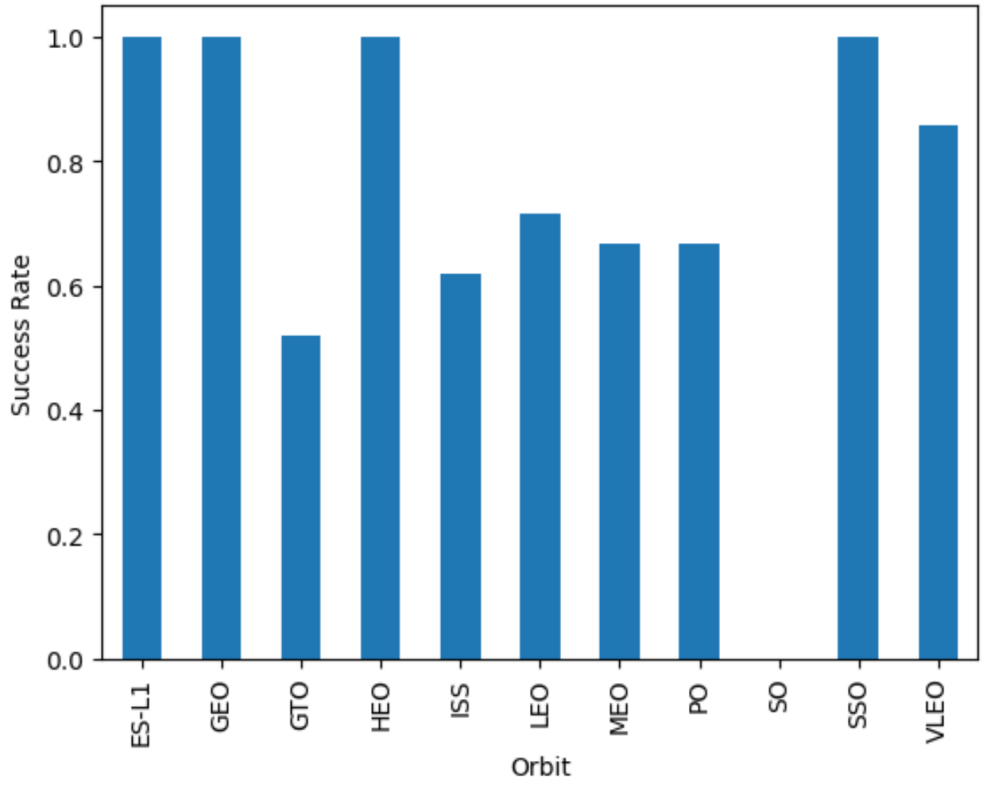
Success Rate vs. Orbit Type

Here the Main (GitHub) - Here the (GitHub) of this Section

The bar chart provides a comparative analysis of the success rates for different orbit types that SpaceX's Falcon 9 missions have targeted. Each bar represents the average success rate for a given orbit type, where the success of a landing is denoted by a 1 and a failure by a 0.

From left to right, the orbits are ES-L1, GEO, GTO, HEO, ISS, LEO, MEO, PO, SSO, and VLEO. The success rate is presented on the vertical axis, ranging from 0 to 1, where 1 signifies a 100% success rate.

Key observations from the chart are:
ES-L1 and SSO orbits have the highest average success rates, suggesting that missions to these orbits have been consistently successful.
The GEO orbit type shows a significantly lower success rate compared to others, which may indicate challenges associated with this orbit.
Other orbits, including GTO, ISS, and VLEO, also display high success rates, though not as uniformly successful as ES-L1 or SSO.



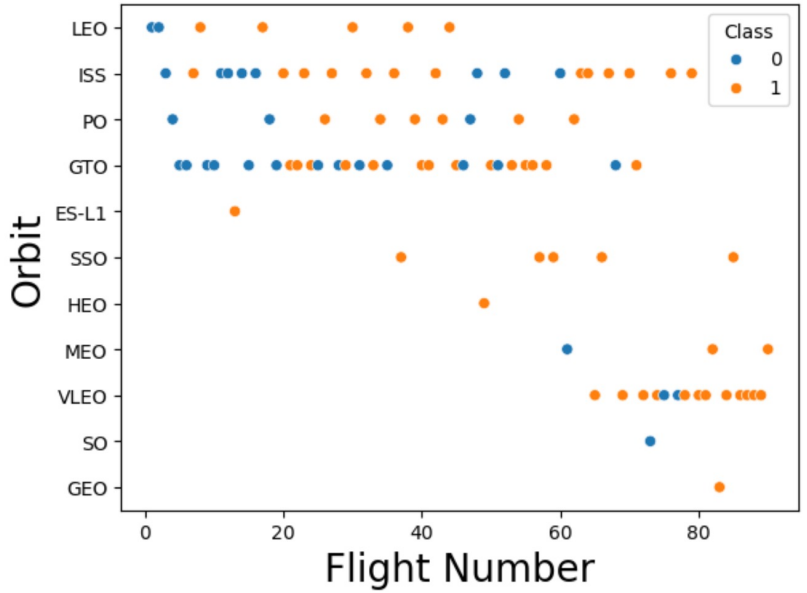
Flight Number vs. Orbit Type

Here the Main (GitHub) - Here the (GitHub) of this Section

The scatter plot visualizes Falcon 9 rocket launches, categorized by orbit type on the y-axis and sequenced by flight number on the x-axis. Each dot represents a specific launch, with the color indicating the class outcome: blue for class 0 (unsuccessful landing) and orange for class 1 (successful landing).

The distribution of dots across various orbit types—LEO, ISS, PO, GTO, ES-L1, SSO, HEO, MEO, VLEO, and GEO—demonstrates SpaceX's range of mission targets. A higher density of blue dots (unsuccessful landings) is observed in the initial flights across all orbits, which then gives way to orange dots as the flight number increases, indicating a trend of improving success rates over time.

Some orbits, like LEO and ISS, have a more consistent distribution of successful landings across flight numbers, while others like GEO and SSO show fewer flights but with a high success rate. This scatter plot underscores the evolution of SpaceX's landing success and provides insights into the relative challenges or efficiencies associated with each orbit type.



Payload vs. Orbit Type

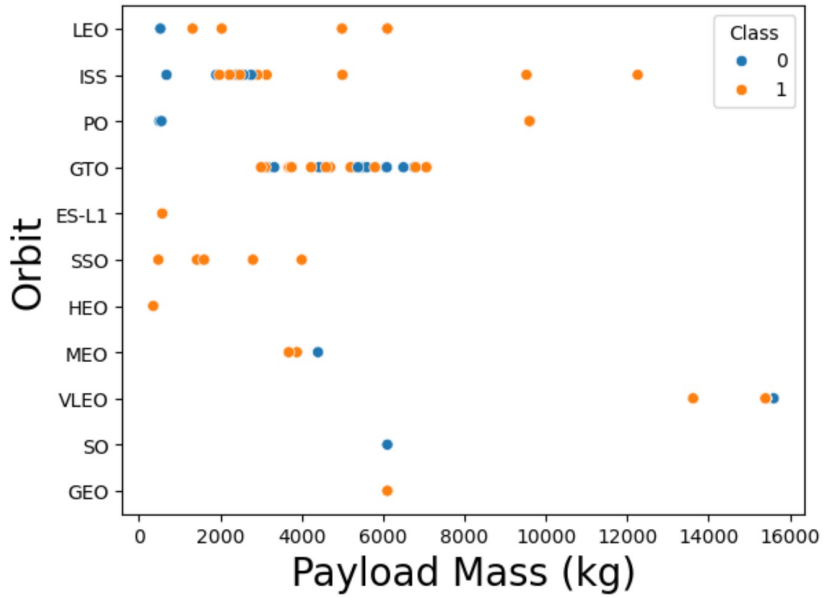
Here the Main (GitHub) - Here the (GitHub) of this Section

This scatter plot provides insights into the relationship between the mass of payloads that SpaceX Falcon 9 rockets have carried to various orbit types and the success of the first stage landing. The x-axis indicates the payload mass in kilograms, while the y-axis categorizes the orbit types the missions were destined for.

The color-coding differentiates between unsuccessful landings (class 0, blue) and successful landings (class 1, orange). The plot reveals that successful landings have occurred across a wide range of payload masses and orbit types. Notably, heavier payloads have been launched to orbits like GTO and LEO, where there's a mix of both successful and unsuccessful landings.

LEO, in particular, has seen a high volume of launches with various payload masses, mostly resulting in successful landings. In contrast, GEO orbits have had fewer launches, with a tendency towards unsuccessful landings regardless of payload mass.

Overall, the plot suggests that while payload mass is an important factor in mission design, the success of landings is influenced by a combination of payload mass and the specific requirements of the orbit type.

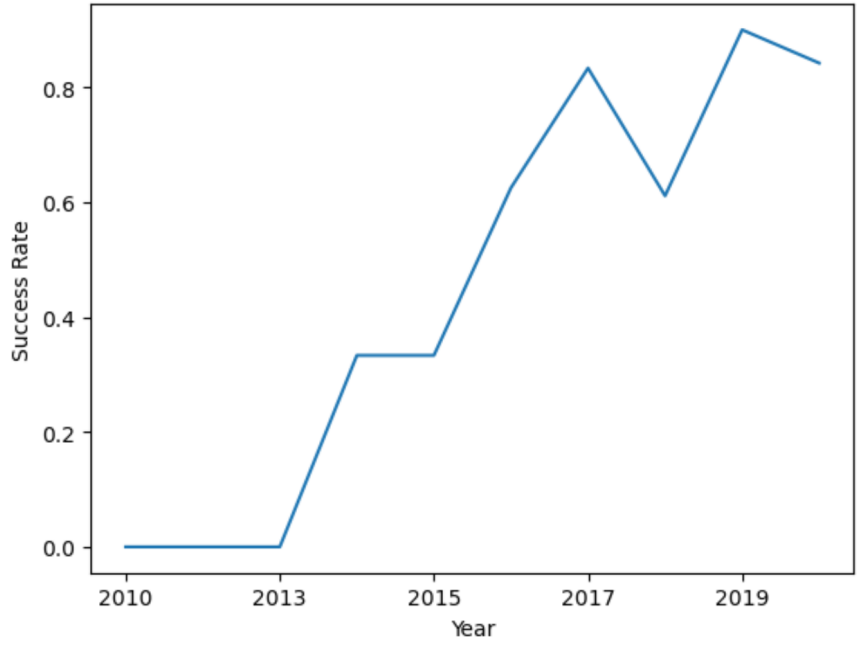


Launch Success Yearly Trend

The line chart illustrates the annual trend of SpaceX Falcon 9 rocket launch success rates from 2010 to 2019. The y-axis represents the success rate, calculated as the mean of successful (class 1) and unsuccessful (class 0) landings for each year, while the x-axis represents the year.

The chart shows an initial success rate of 0 at the start of the decade, reflecting a period of unsuccessful attempts. There is a marked uptick in the success rate beginning around 2013, indicating a turning point in the Falcon 9 program's success with landings. From 2013 onwards, there is a noticeable increase in the success rate, with some fluctuations. The year 2017 displays a peak, suggesting a period of high success for landings.

However, after 2017, there is a slight dip followed by a recovery, illustrating the challenges and subsequent improvements in SpaceX's landing technology or procedures. The overall upward trend indicates significant advancements in the company's capabilities to successfully land Falcon 9 rockets over the years.



Here the Main (GitHub) - Here the (GitHub) of this Section

All Launch Site Names

The SQL query provided is used to retrieve all unique launch site names from a table named SPACEXTABLE. The keyword DISTINCT ensures that each launch site is listed only once in the result set, even if it appears multiple times in the table.

- %sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;

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

Here the Main (GitHub) - Here the (GitHub) of this Section

Launch Site Names Begin with 'CCA'

The given SQL query selects the first five records from the SPACEXTABLE where the Launch_Site begins with 'CCA'. The LIKE 'CCA%' operator is used to match any launch site that starts with 'CCA', with the percent sign (%) functioning as a wildcard for any sequence of characters that may follow. The LIMIT 5 clause restricts the output to only the first five records that satisfy this condition. This query is useful for examining a sample of launches from launch sites with names starting with 'CCA'.

```
SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Here the Main (GitHub) - Here the (GitHub) of this Section

Total Payload Mass

This SQL query calculates the total payload mass in kilograms for all SpaceX launches where the customer is 'NASA (CRS)'. The SUM function adds up the PAYLOAD_MASS__KG_ column values, providing a total payload mass for this specific customer. The LIKE 'NASA (CRS)' clause specifies that only records with 'NASA (CRS)' in the Customer column should be included in the sum. The output will be a single value representing the cumulative mass that SpaceX has launched for NASA's Commercial Resupply Services missions.

- %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer LIKE 'NASA (CRS)';

SUM(PAYLOAD_MASS__KG_)
45596

Here the Main (GitHub) - Here the (GitHub) of this Section

Average Payload Mass by F9 v1.1

This SQL query is designed to determine the average payload mass that the Falcon 9 version 1.1 rocket has launched. It uses the AVG() function to compute the mean value of all entries in the PAYLOAD_MASS__KG_ column from the SPACEXTABLE where the Booster_Version is like 'F9 v1.1'. This implies that it will only consider launches where the booster version specified in the record matches 'F9 v1.1'. The %sql magic command, followed by the variable $query, indicates that the query string stored in the variable query will be executed in a notebook environment that supports SQL magic commands.

```
# This SQL query calculates the average payload mass for booster version F9 v1.1
query = '''
    SELECT
      AVG(PAYLOAD_MASS__KG_)  -- This is the average function
   FROM
      SPACEXTABLE  -- This is the table name
   WHERE
      Booster_Version LIKE 'F9 v1.1';  -- This is the condition to filter the data
'''

%sql $query
```

AVG(PAYLOAD_MASS__KG_) -- This is the average function
2928.4

Here the Main (GitHub) - Here the (GitHub) of this Section

First Successful Ground Landing Date

The provided SQL query is set to find the earliest date on which SpaceX achieved a successful landing on a ground pad. It utilizes the MIN() function to locate the smallest date value, which corresponds to the earliest date in the Date column of the SPACEXTABLE. The WHERE clause restricts the search to only those records where the Landing_Outcome indicates a 'Success (ground pad)', which means that only successful ground pad landings are considered in determining the earliest successful attempt. The result of this query will be the date of the first successful ground pad landing according to the available data in the table.
This SQL query identifies the earliest date of a successful landing on a ground pad

- query = ‘’’
  SELECT
  MIN(Date) -- This is the minimum function to find the earliest date
  FROM
  SPACEXTABLE -- This is the table name
  WHERE
  Landing_Outcome LIKE 'Success (ground pad)'; -- This is the condition to filter the data for successful landings on a ground pad
- ’’
  %sql $query

MIN(Date) -- This is the minimum function to find the earliest date
2015-12-22

Here the Main (GitHub) - Here the (GitHub) of this Section

Successful Drone Ship Landing with Payload between 4000 and 6000

Here the Main (GitHub) - Here the (GitHub) of this Section

The SQL query is constructed to retrieve a list of unique Falcon 9 booster versions that have successfully landed on a drone ship and carried a payload mass between 4000 and 6000 kilograms. The DISTINCT keyword ensures that each booster version is listed only once, even if it has been used for multiple successful landings within the specified payload mass range.

The WHERE clause applies three filters:

Landing_Outcome LIKE 'Success (drone ship)' selects only the entries that indicate a successful landing on a drone ship.
PAYLOAD_MASS__KG_ > 4000 includes only those launches where the payload mass was more than 4000 kilograms.
AND PAYLOAD_MASS__KG_ < 6000 further restricts the selection to those where the payload mass was less than 6000 kilograms.

```
query = '''
SELECT DISTINCT Booster_Version FROM SPACEXTABLE
WHERE Landing_Outcome LIKE 'Success (drone ship)'
AND PAYLOAD_MASS__KG_ > 4000
AND PAYLOAD_MASS__KG_ < 6000;
%sql $query
```

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

Total Number of Successful and Failure Missio n Outcomes

The SQL query calculates the total number of successful and failed missions in the SPACEXTABLE, using conditional SUM statements to count each type based on the Mission_Outcome field.

```
SELECT
  SUM(CASE WHEN Mission_Outcome LIKE 'Success%' THEN 1 ELSE 0 END) AS
Successful_Missions, -- Condition to count successful missions
  SUM(CASE WHEN Mission_Outcome LIKE 'Failure%' THEN 1 ELSE 0 END) AS
Failed_Missions -- Condition to count failed missions
FROM
  SPACEXTABLE;
```

Successful_Missions	Failed_Missions
100	1

Here the Main (GitHub) - Here the (GitHub) of this Section

Boosters Carried Maximum Payload

Here the Main (GitHub) - Here the (GitHub) of this Section

This SQL query is designed to retrieve the Booster_Version associated with the heaviest payload mass that SpaceX has launched. The query operates in two parts:

The subquery (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE) calculates the maximum payload mass in kilograms from all records in the SPACEXTABLE.

The outer query SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = uses the result of the subquery to find the corresponding Booster_Version that has launched this heaviest payload.

The result will be the booster versions that have carried the maximum payload mass according to the SpaceX table data. If multiple booster versions have lifted this same maximum payload mass, the query will return all of them.

```
SELECT Booster_Version
FROM SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTABLE
);
```

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

2015 Launch Records

Here the Main (GitHub) - Here the (GitHub) of this Section

This SQL query employs a Common Table Expression (CTE) named MonthData to extract and transform a subset of data from the SPACEXTABLE for further processing.
The CTE MonthData selects four columns: the month part of the Date (assuming the Date is in a YYYY-MM-DD format), the Landing_Outcome, the Booster_Version, and the Launch_Site. It filters records to include only those from the year 2015 (substr(Date,0,5)='2015') and where the Landing_Outcome was a 'Failure (drone ship)'.
The main SELECT statement then transforms the month number into its corresponding name using a CASE expression. It maps each month's numeric value ('01' to '12') to its proper name ('January' to 'December').
Finally, the query selects the month names, along with the Landing_Outcome, Booster_Version, and Launch_Site from the CTE MonthData.
The result of this query will be a list of drone ship landing failures in 2015, with the month of each failure presented as a full month name instead of a number, along with details about the booster version and launch site for each of those failed landings.

```
WITH MonthData AS (
    SELECT
        sbstr(Date, 6,2) AS Month,
        Landing_Outcome,
        Booster_Version,
        Launch_Site
    FROM
        SPACEXTABLE
    WHERE
        substr(Date,0,5)='2015'
        AND Landing_Outcome LIKE 'Failure (drone ship)'
)
SELECT
    CASE
        WHEN Month='01' THEN 'January'
         WHEN Month='02' THEN 'February'
        WHEN Month='03' THEN 'March'
        WHEN Month='04' THEN 'April'
        WHEN Month='05' THEN 'May'
        WHEN Month='06' THEN 'June'
        WHEN Month='07' THEN 'July'
        WHEN Month='08' THEN 'August'
        WHEN Month='09' THEN 'September'
        WHEN Month='10' THEN 'October'
        WHEN Month='11' THEN 'November'
        WHEN Month='12' THEN 'December'
    END AS MonthName,
    Landing_Outcome,
    Booster_Version,
    Launch_Site
FROM
    MonthData;
```

MonthName	Landing_Outcome	Booster_Version	Launch_Site
October	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

The SQL query counts and sorts the different landing outcomes in the SPACEXTABLE from June 4, 2010, to March 20, 2017, grouping them by outcome and ordering by frequency in descending order.

```
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;
```

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

Here the Main (GitHub) - Here the (GitHub) of this Section

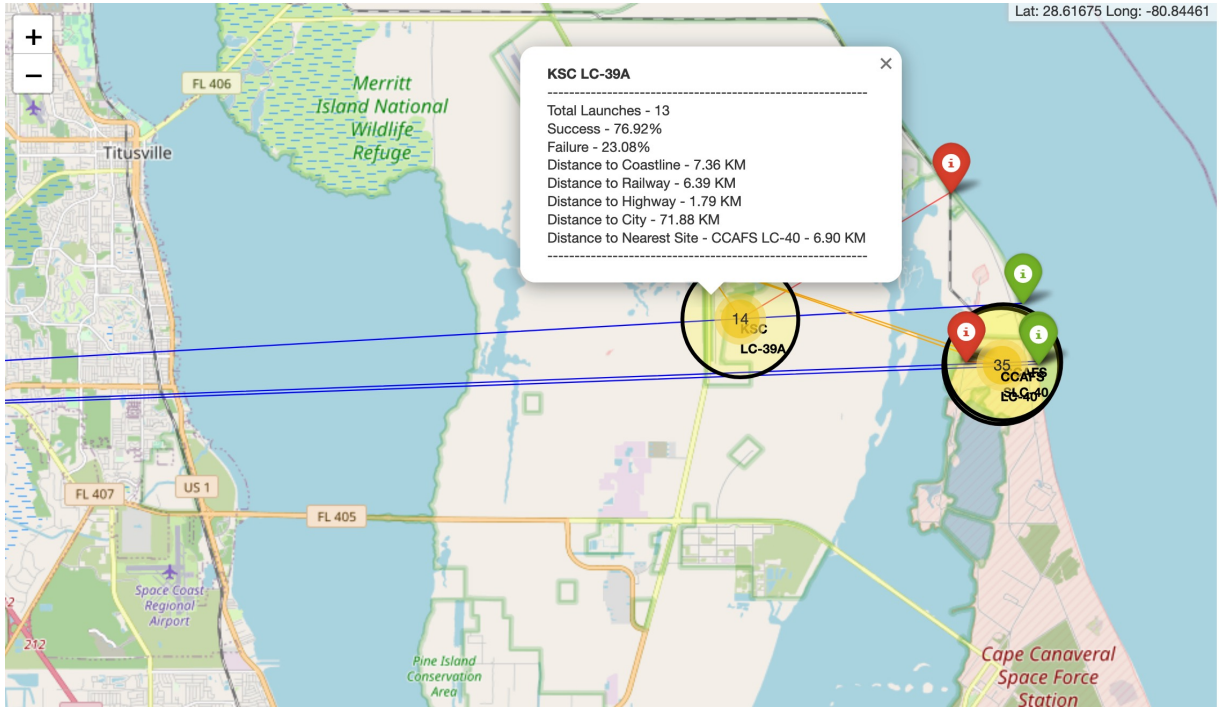

Section 3
Launch Sites
Proximities Analysis

Folium Map

PolyLines are used to connect launch sites to coastline, railway, highway, and city coordinates.
They visually display the distances between launch sites and these key points.

Circles with yellow fill color and black borders are created at the launch site coordinates.

Popup labels for these circles provide information about the launch site, including its name, success/failure rates, distances to coastline, railway, highway, city, and the nearest launch site.

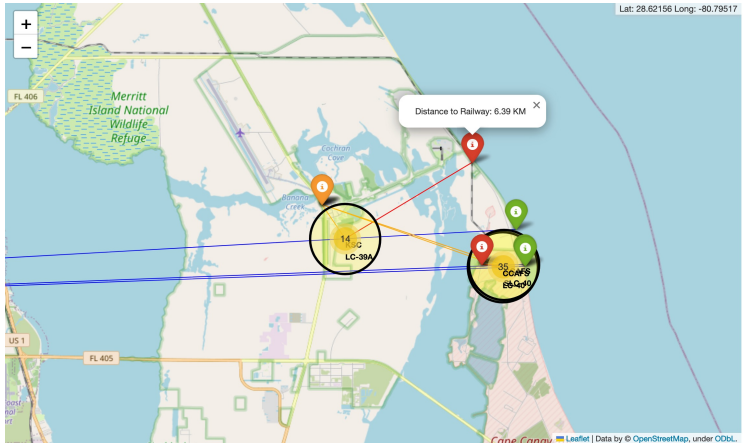


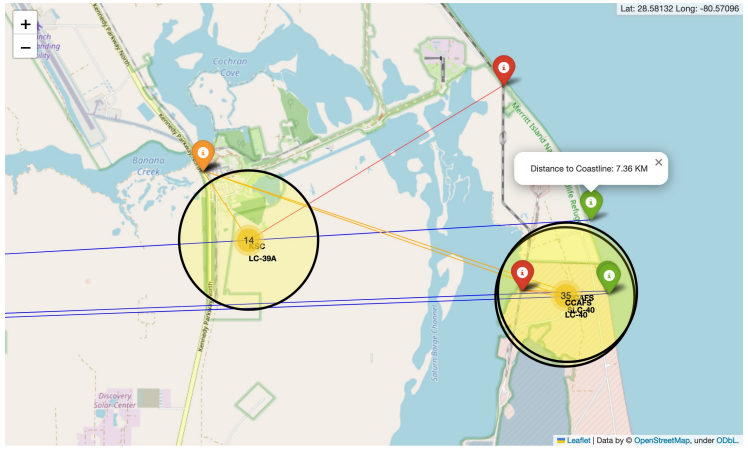
Here the Main (GitHub) - Here the (GitHub) of this Section

Folium Map CoastLine Railroad Highway

Green Markers for Coastline Coordinates:
These green markers represent the fixed coastline coordinates for specified launch sites.They help visualize the proximity of launch sites to coastlines, which can be important for oceanic launches, safety, and risk reduction.

Red Markers for Railway Coordinates:
These red markers represent the fixed railway coordinates for specified launch sites.They indicate the proximity of launch sites to railways, which is crucial for transporting heavy payloads and enhancing logistical efficiency.





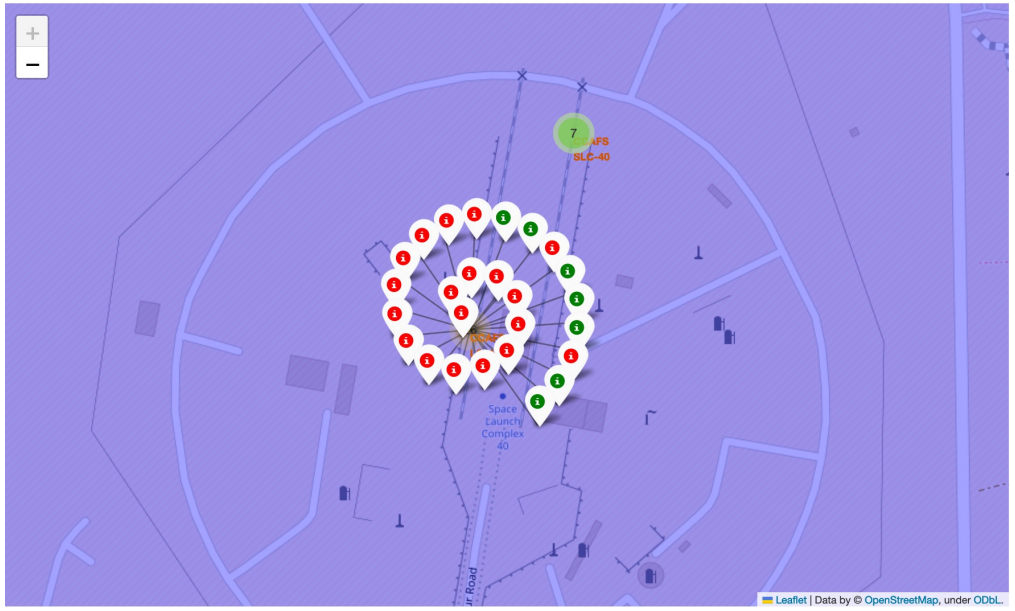
Orange Markers for Highway Coordinates:

These orange markers represent the fixed highway coordinates for specified launch sites.
They show the proximity of launch sites to highways, ensuring smooth transit of equipment and personnel.

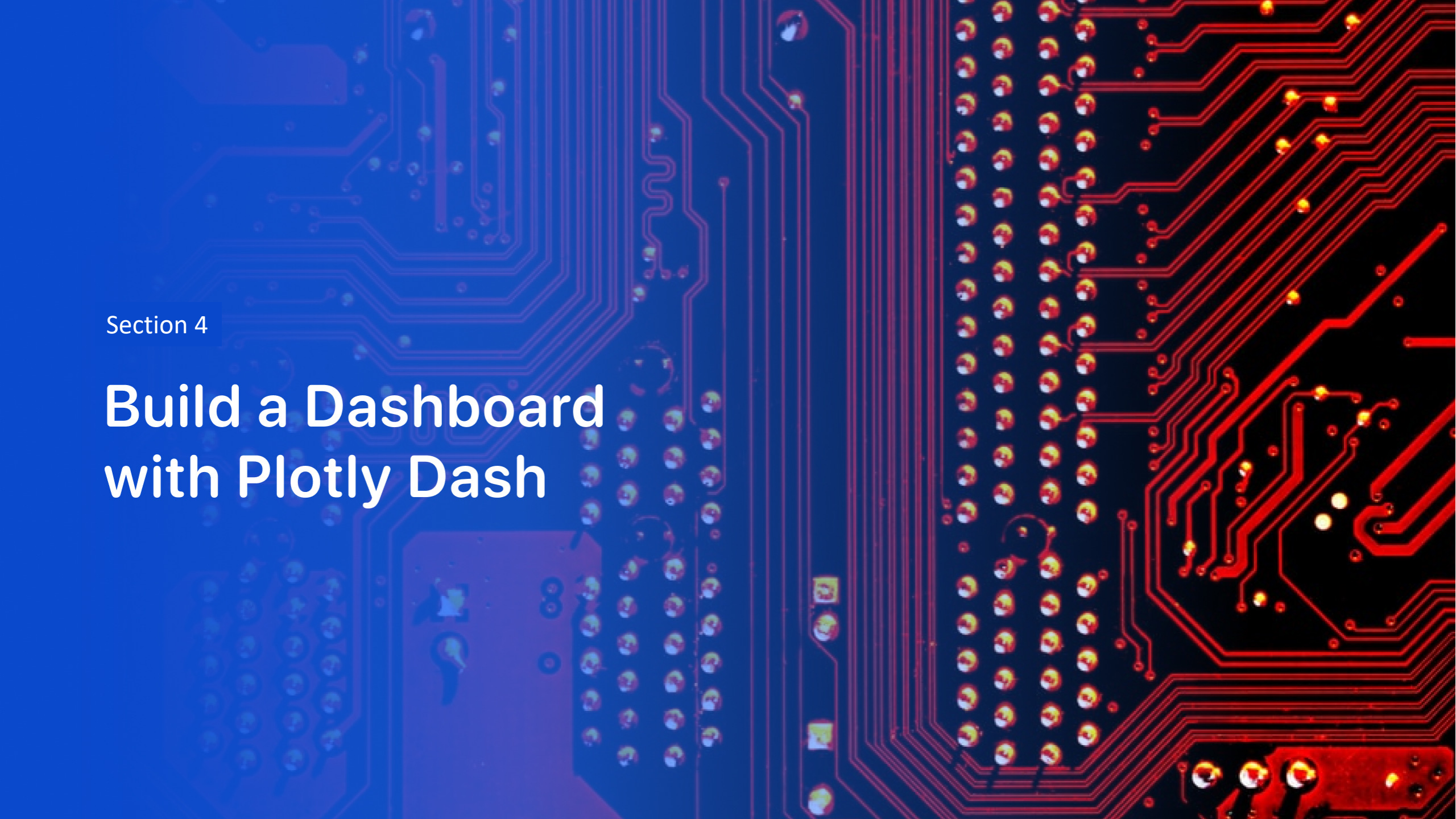
Here the Main (GitHub) - Here the (GitHub) of this Section

Folium Map Screenshot 3

A MarkerCluster is used to group markers for better map organization.It allows users to interact with clustered markers, making it more manageable when there are many markers on the map.

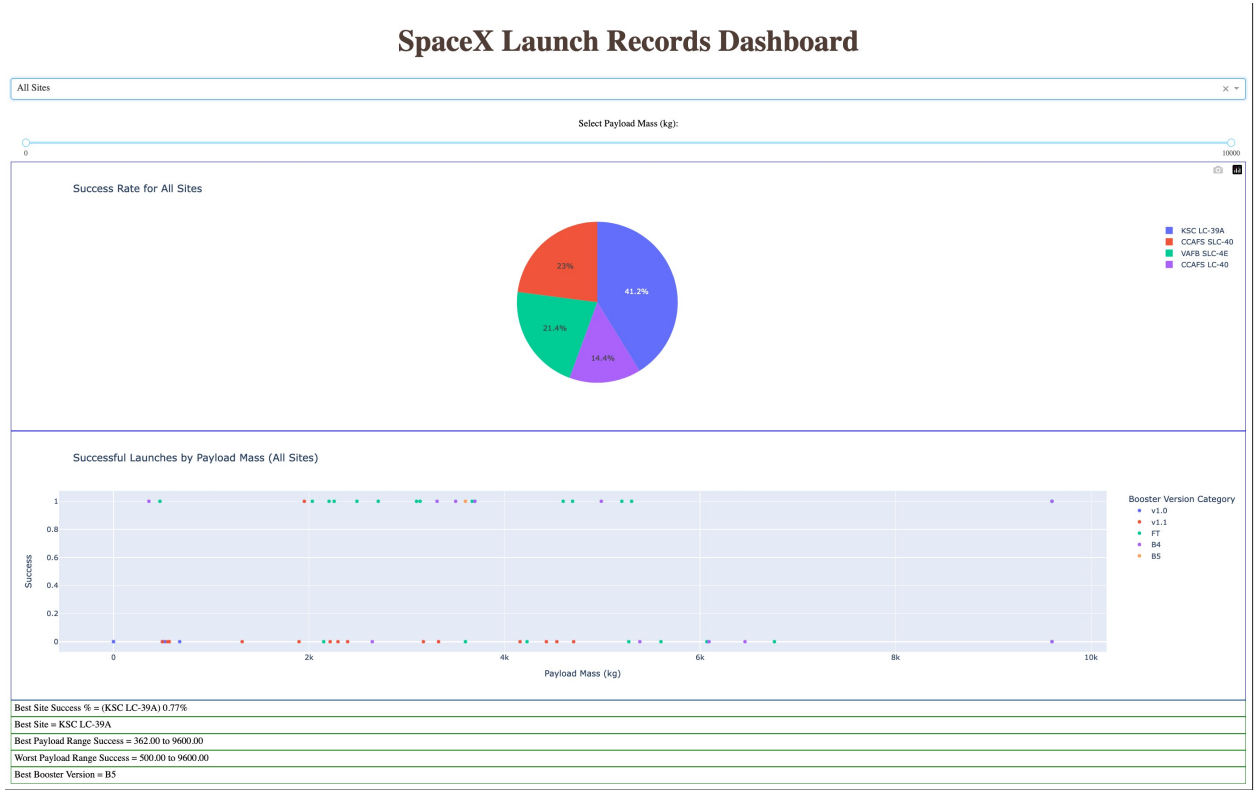


Here the Main (GitHub) - Here the (GitHub) of this Section


Section 4
Build a Dashboard with Plotly Dash

Dashboard Main

The dashboard provides users with an option to select "All Sites" from the dropdown menu.
When this option is chosen, the dashboard displays information about the best launch site, including the success percentage, the optimal payload range, and identifies the best booster overall.

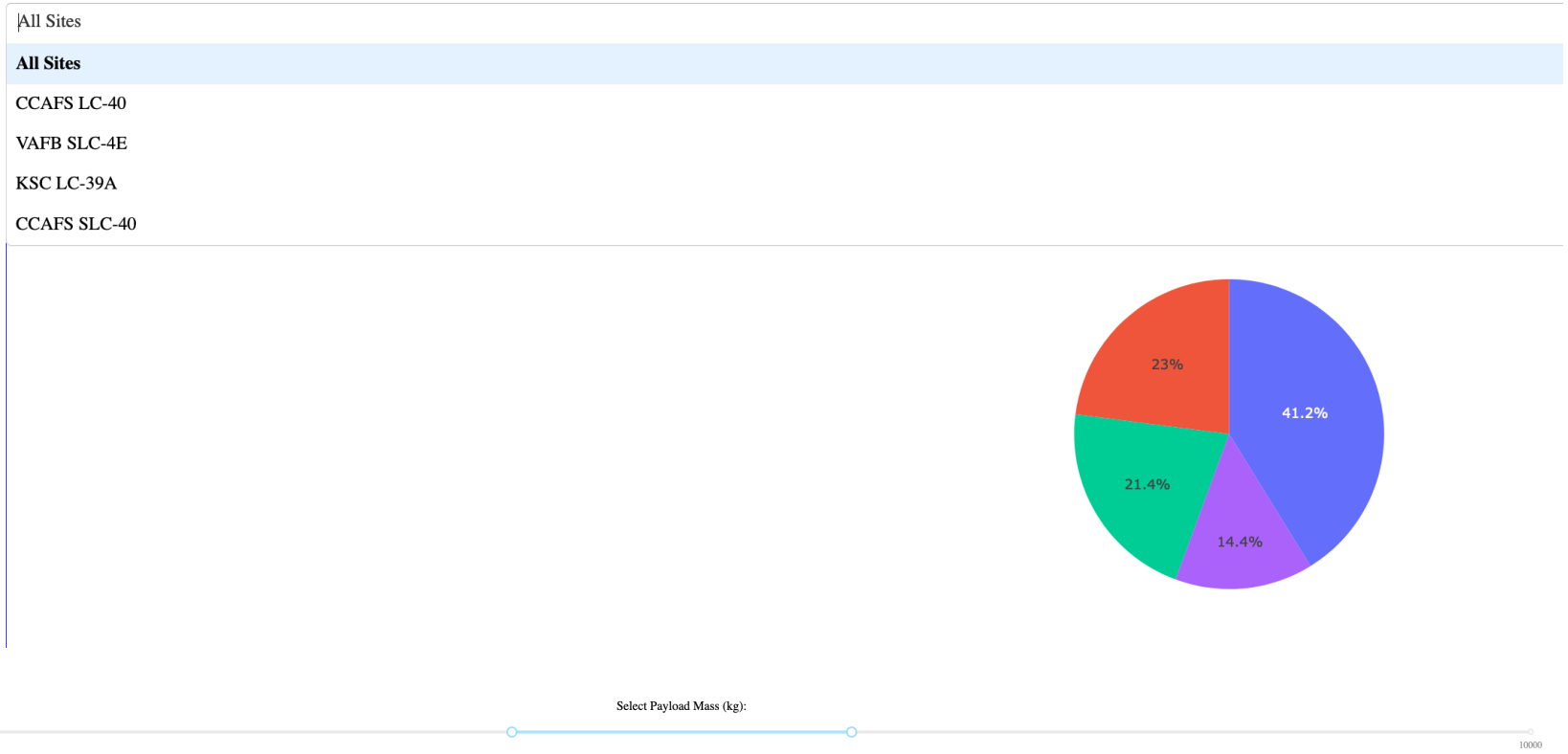


Best Site Success % = (KSC LC-39A) 0.77%
Best Site = KSC LC-39A
Best Payload Range Success = 362.00 to 9600.00
Worst Payload Range Success = 500.00 to 9600.00
Best Booster Version = B5

Here the Main (GitHub) - Here the (GitHub) of this Section

Dashboard DropDown – S.

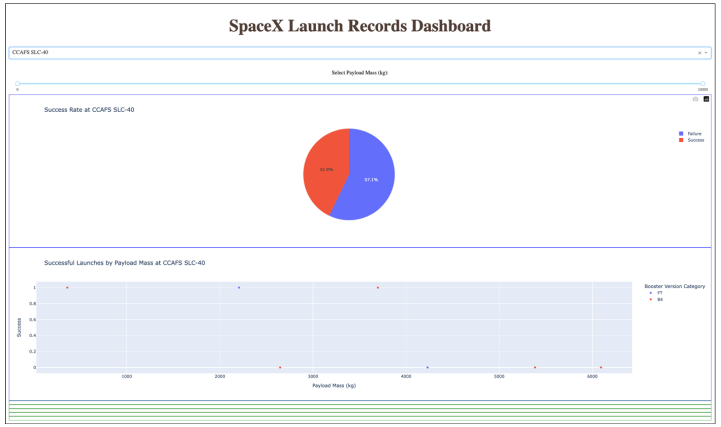
The dashboard enables user interaction by offering a dropdown menu that allows users to easily navigate through all available launch sites. Additionally, users have the flexibility to select their desired payload using this interface.

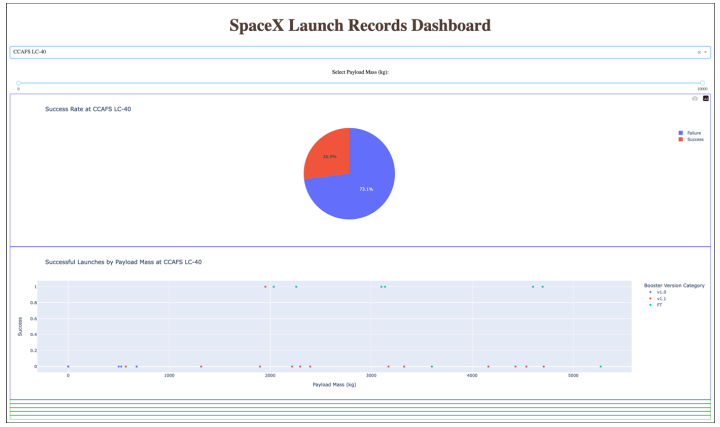


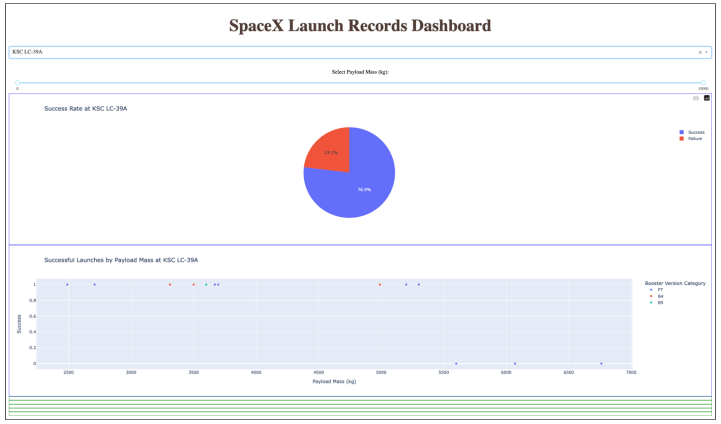
Here the Main (GitHub) - Here the (GitHub) of this Section

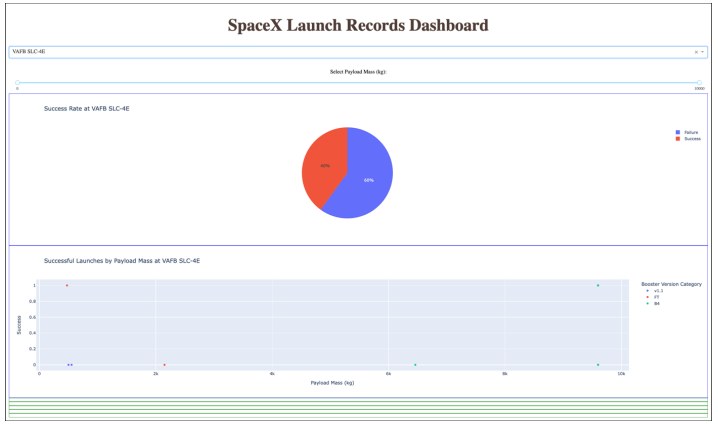
Dashboard Other Sites

Upon selecting a specific launch site, the dashboard dynamically updates all the information presented in the graphs in real-time, providing users with the most current data for their chosen site.









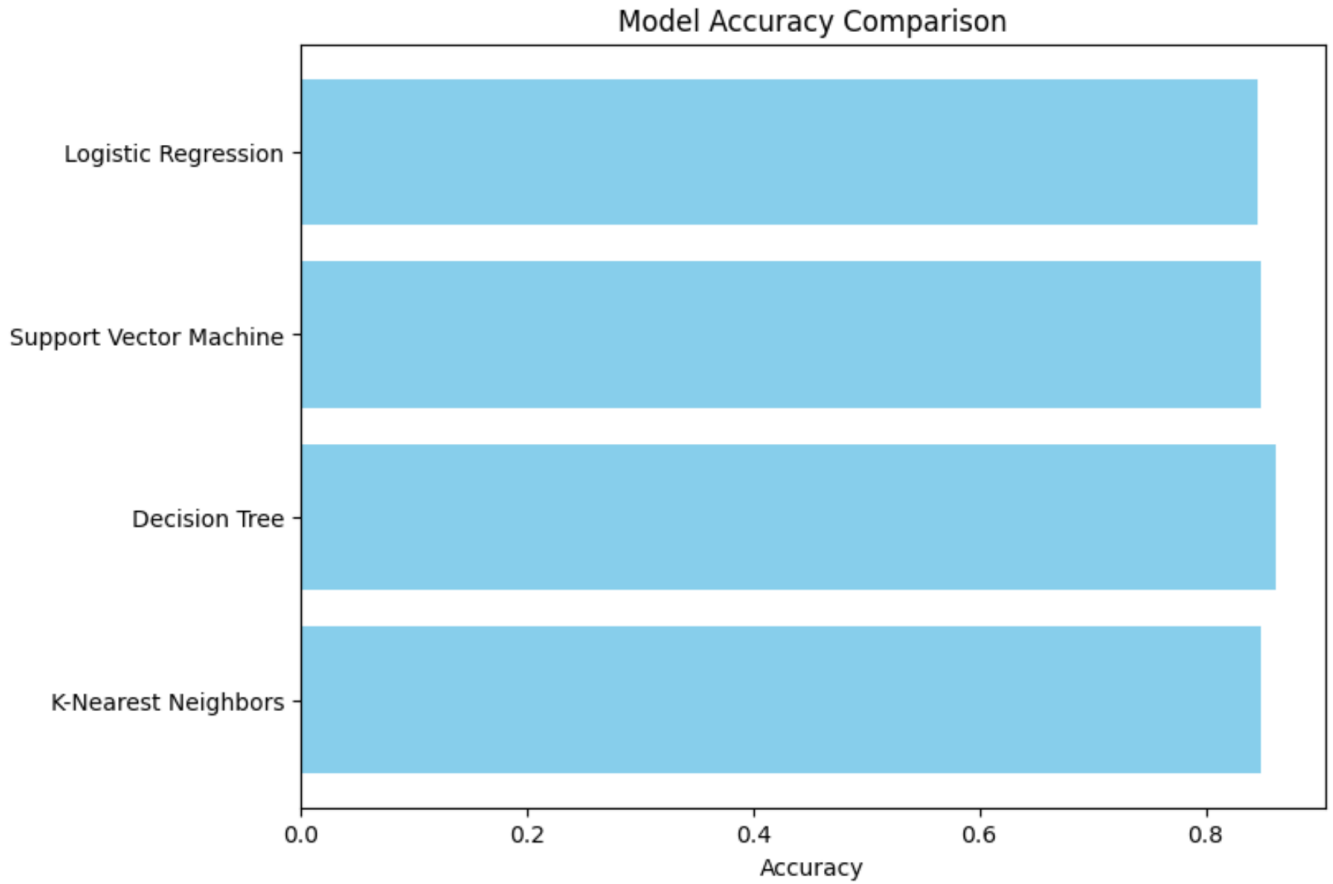
Here the Main (GitHub) - Here the (GitHub) of this Section

Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
Best-Performing Method: Decision Tree
Accuracy Score: 0.8625
```



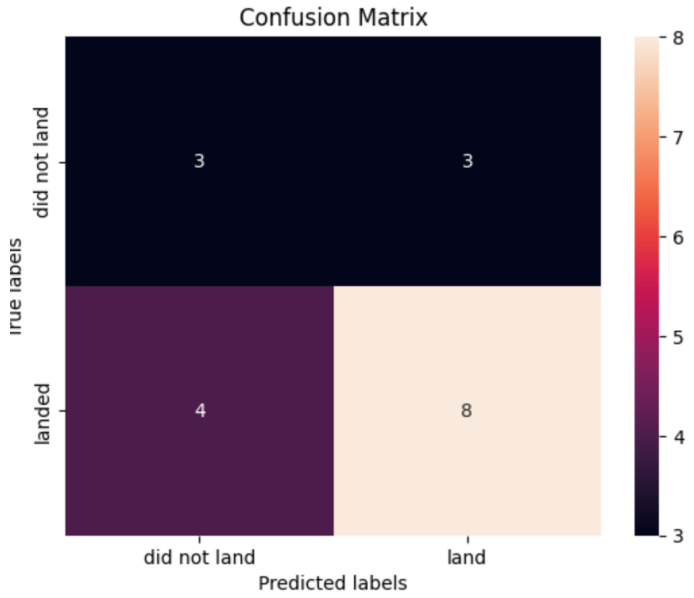
Here the Main (GitHub) - Here the (GitHub) of this Section

Confusion Matrix – Decision Tree

The confusion matrix for the Decision Tree model provides a visual representation of its performance in binary classification. It consists of four key metrics:

- True Positives (TP): The number of correctly predicted positive cases (successful landings).
- True Negatives (TN): The number of correctly predicted negative cases (unsuccessful landings).
- False Positives (FP): The number of actual negative cases incorrectly predicted as positive.
- False Negatives (FN): The number of actual positive cases incorrectly predicted as negative.

These metrics allow us to assess how well the Decision Tree model distinguishes between successful and unsuccessful Falcon 9 first stage landings. It helps in evaluating the model's accuracy and its ability to minimize false predictions.



Here the Main (GitHub) - Here the (GitHub) of this Section

Conclusions

Conclusions from SpaceX Falcon 9 First Stage Landing Prediction:

Model Selection: Among Logistic Regression, SVM, K-Nearest Neighbors, and Decision Tree, the Decision Tree model showed superior performance with an accuracy of 0.8607.

Performance Metrics: Accuracy across models varied, indicating the diverse nature of the dataset and the complexity of predicting landing outcomes.

Optimization and Tuning: Hyperparameter tuning played a crucial role in enhancing model performances, notably in Logistic Regression and SVM.

Insights for SpaceX: The analysis provides valuable predictions for landing success, directly impacting cost-efficiency and operational decisions for SpaceX.

Strategic Implications: Accurate predictions can aid SpaceX in competitive bidding, offering cost-effective solutions while maintaining high success rates.

Future of Space Exploration: The study highlights ML's potential in advancing space missions, particularly in optimizing launch strategies and reducing expenditures.

Data-Driven Decisions: The project underscores the importance of data-driven approaches in the aerospace sector, paving the way for more informed and strategic planning.

Here the Main (GitHub) - Here the (GitHub) of this Section

Appendix

I used Power BI for some plots and mainly basic functions on python. All the code is in the repository.

Here the Main (GitHub)

Thank you!