



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Devendrasing Deshmukh
May 12, 2025



Outline

- [Executive Summary](#)
- [Introduction](#)
- [Methodology](#)
- [Results](#)
- [Conclusion](#)
- [Appendix](#)

Executive Summary

Summary of Methodologies:

- Data ingested via SpaceX REST API calls and Wikipedia web-scraping, cleaned and merged into a unified dataset.
- Performed data wrangling: missing-value handling, feature engineering (binary “Class” target).
- Exploratory Data Analysis using scatter, bar and line plots; SQL queries to profile payloads and outcomes.
- Interactive visual analytics: Folium maps for launch-site geospatial insights; Plotly Dash dashboard with filters and dynamic plots.
- Predictive modeling: built and tuned Logistic Regression, SVM, Decision Tree, and KNN via GridSearchCV; evaluated on test set

Summary of Results:

- **Launch-Site Performance:** KSC LC-39A highest success rate (76.9%); CCAFS SLC-40 most launches but lower reliability.
- **Payload Insights:** Heavy payloads (>9 000 kg) show high success at KSC LC-39A and CCAFS SLC-40; no clear mass–success correlation overall.
- **Orbit Trends:** ES-L1, GEO, HEO, SSO orbits achieved 100% success; VLEO emerging with >80% success; LEO around 70%.
- **Temporal Trend:** Success rate steadily increased since 2013; no failures after flight #79.
- **Model Accuracy:** Logistic Regression, SVM, KNN tied at ~83% accuracy; Decision Tree at ~78%; no false negatives in best models.

Introduction

Project Background & Context:

- Analyse historic SpaceX Falcon 9 first-stage landing data to understand factors driving mission success.
- Leverage both geospatial and temporal analyses to inform future launch-site and payload strategies.

Key Questions:

- Which launch-site and payload characteristics most influence landing success?
- How do orbit types and booster versions correlate with outcomes?
- Can classification models reliably predict landing success based on mission parameters?

Section 1

Methodology

Methodology

Executive Summary

1) Data collection methodology:

- Retrieved primary launch records via SpaceX API (/v4/launches/past) and auxiliary endpoints (/rockets, /payloads, /launchpads, /cores).
- Scraped supplementary historical data from Wikipedia using BeautifulSoup

2) Data Wrangling & Processing:

- Flattened JSON payloads, merged API and scraped datasets.
- Handled missing values, removed duplicates, engineered binary target “Class” (landing success).

3) Exploratory Data Analysis (EDA) using visualization and SQL:

- Visualized relationships (e.g., Payload Mass vs. Flight Number, Orbit vs. Success) using scatter, bar, and line plots.
- Executed SQL queries in Jupyter to profile launch sites, payload statistics, and outcome counts

4) Interactive Visual Analytics:

- **Folium:** Mapped site locations with success/failure markers; computed proximities to coastlines, highways, railways, cities.
- **Plotly Dash:** Built dashboard with launch-site dropdown, payload slider, success pie chart, and scatter plot filtered by booster version

5) Predictive Analysis (Classification):

- Prepared features (one-hot encoding, scaling), split data into train/test sets.
- Trained and tuned Logistic Regression, SVM, Decision Tree, and KNN via GridSearchCV.
- Evaluated models on test set using accuracy and confusion matrix; selected best model based on performance

Data Collection

The dataset was gathered through a combination of **REST API calls** and **web scraping techniques**, ensuring both historical accuracy and rich metadata. The process involved the following steps:

1. SpaceX REST API Calls:

- Primary launch data was retrieved using the SpaceX API endpoint:
<https://api.spacexdata.com/v4/launches/past>
- Additional metadata such as rocket types, payloads, launchpads, and core details were collected using supplementary endpoints: [/rockets](#), [/payloads](#), [/launchpads](#), and [/cores](#).

2. Data Parsing & Transformation:

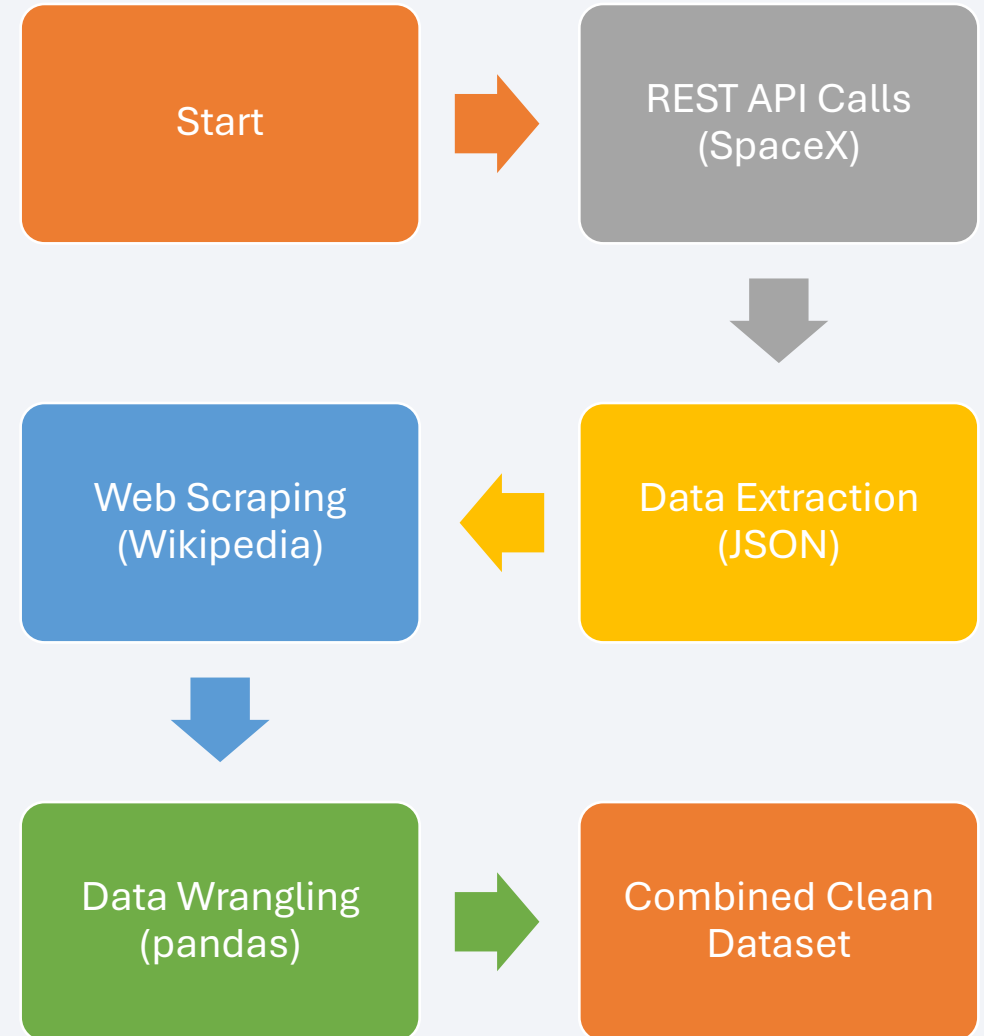
- API responses in JSON format were parsed using the Python [requests](#) and [json](#) libraries.
- Nested structures were flattened using [pandas.json_normalize\(\)](#) to build a structured dataset.

3. Wikipedia Web Scraping:

- Historical launch data including launch outcomes and payload mass were extracted from relevant Wikipedia pages using [BeautifulSoup](#).
- HTML tables were converted into pandas DataFrames.

4. Data Integration & Cleaning:

- Both sources (API + Wikipedia) were merged to produce a comprehensive dataset.
- Redundant and irrelevant entries were cleaned, and missing values were handled to improve data quality for analysis.



Data Collection – SpaceX API

Key Phrases:

- **Source:** SpaceX REST API
- **Purpose:** Retrieve structured data on past launches, rockets, payloads, cores, and launchpads.
- **Method:** HTTP GET requests using Python's `requests` library.
- **Format:** Data received in JSON format.
- **Tools Used:** `requests`, `pandas`, `json_normalize`

Step-by-Step Explanation:

1. Call Primary Endpoint

<https://api.spacexdata.com/v4/launches/past>

→ Fetch historical launch data (flight number, date, rocket ID, payload ID, core ID, etc.).

2. Enrich Data

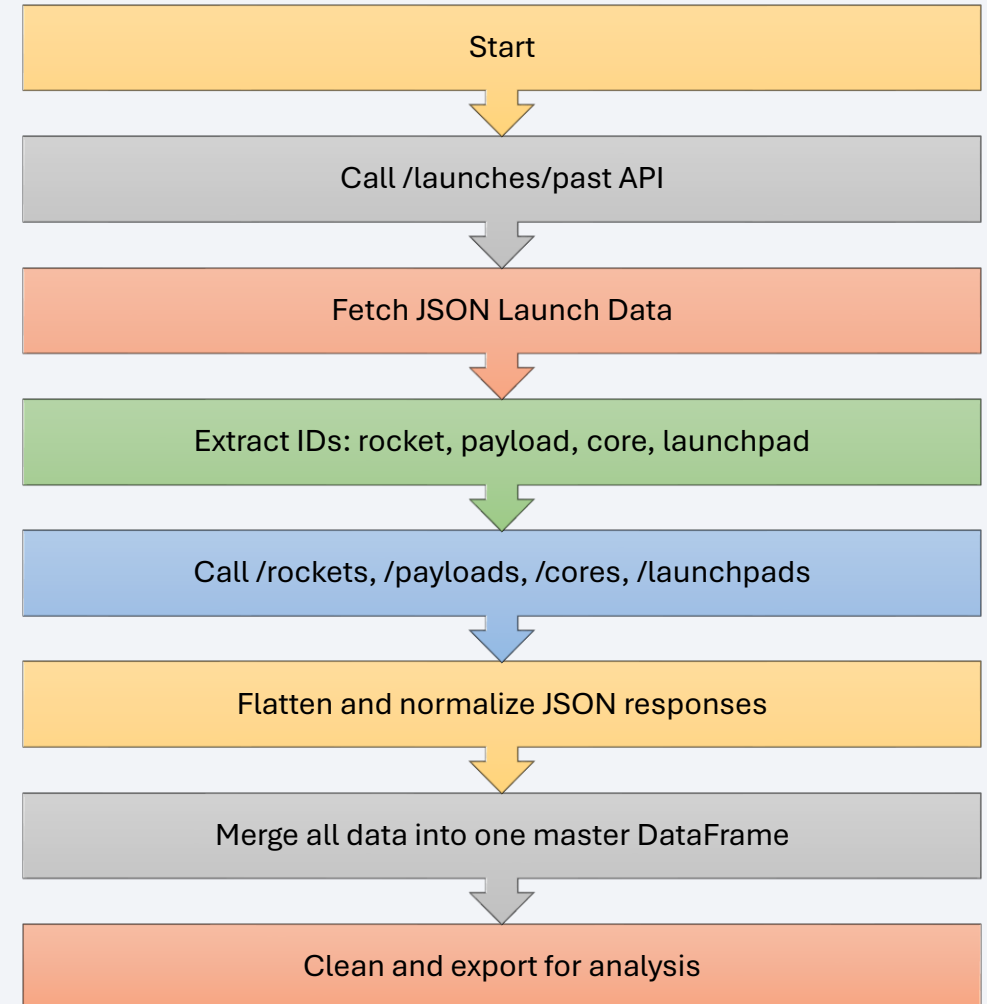
- Use `rocket`, `payload`, `launchpad`, and `core` IDs from the main dataset.
- Perform **secondary API calls** to:
 - `/rockets` → Get rocket names and configurations.
 - `/payloads` → Extract payload mass, orbit type, etc.
 - `/launchpads` → Fetch launch site names and locations.
 - `/cores` → Check reuse, landing outcomes, and booster info.

3. Normalize & Merge Data

- Flatten nested JSON structures using `json_normalize`.
- Join datasets based on IDs to create a comprehensive table.

4. Final Output

- A complete launch dataset suitable for machine learning and analysis.



Data Collection - Scraping

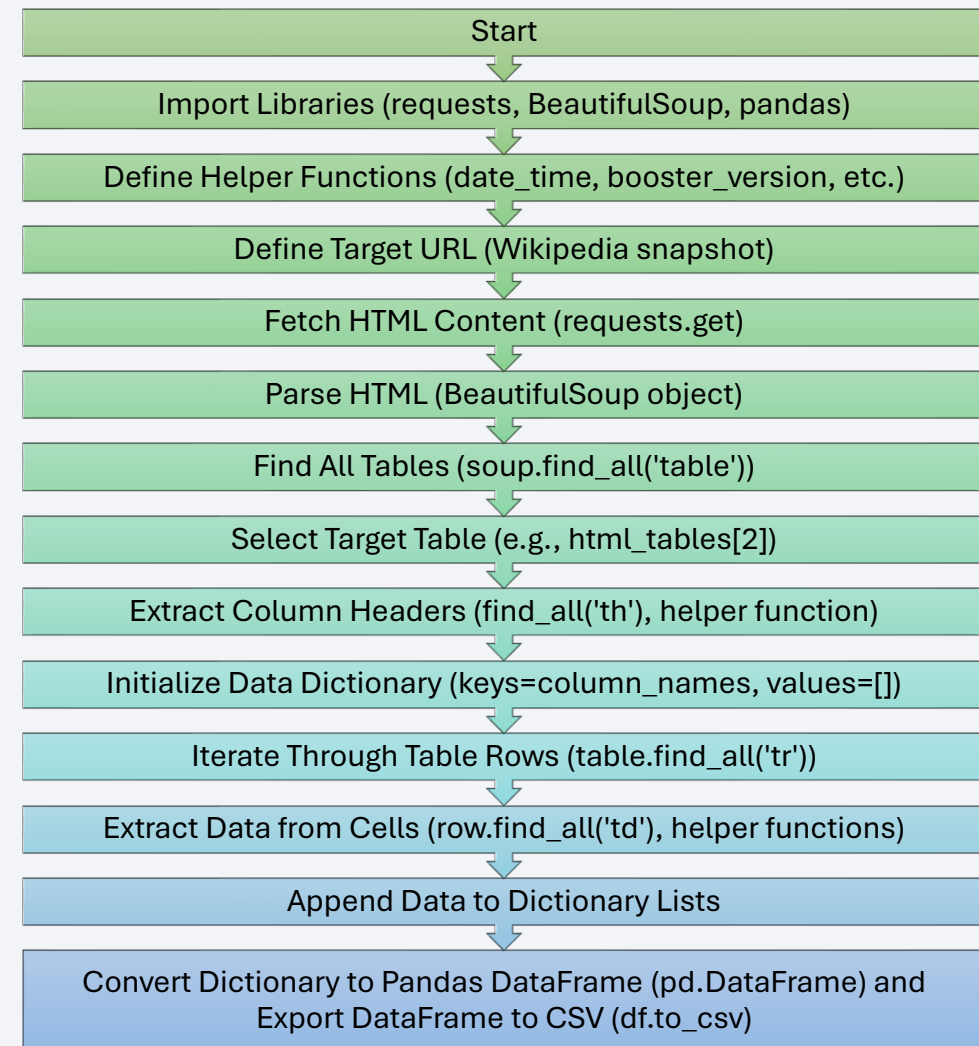
Web Scraping Process: Wikipedia Falcon 9 Launches

Goal: Extract Falcon 9 launch data from a specific Wikipedia page snapshot.

Tools: Python with `requests`, `BeautifulSoup`, `pandas`.

Steps:

1. **Fetch:** Get HTML content from the target Wikipedia URL.
2. **Parse:** Use `BeautifulSoup` to parse the fetched HTML.
3. **Identify:** Locate the specific HTML table containing launch data.
4. **Extract Headers:** Get column names from the table header (`<th>`)
5. **Extract Data:** Iterate through table rows (`<tr>`), extract data from cells (`<td>`) using helper functions.
6. **Structure:** Store extracted data in a Python dictionary.
7. **Convert:** Transform the dictionary into a Pandas DataFrame.
8. **(Optional) Export:** Save the DataFrame to a CSV file.



Data Wrangling

Key Phrases Describing the Data Wrangling Process:

1. Load Data: Import the dataset from a CSV file into a pandas DataFrame.

2. Initial Exploration:

- Calculate the percentage of missing values for each column.
- Determine the data types (numerical, categorical) for each column.

3. Analyze Categorical Features:

- Count the occurrences of each unique value in the `LaunchSite` column.
- Count the occurrences of each unique value in the `Orbit` column.
- Count the occurrences of each unique value in the `Outcome` column.

4. Define Landing Success:

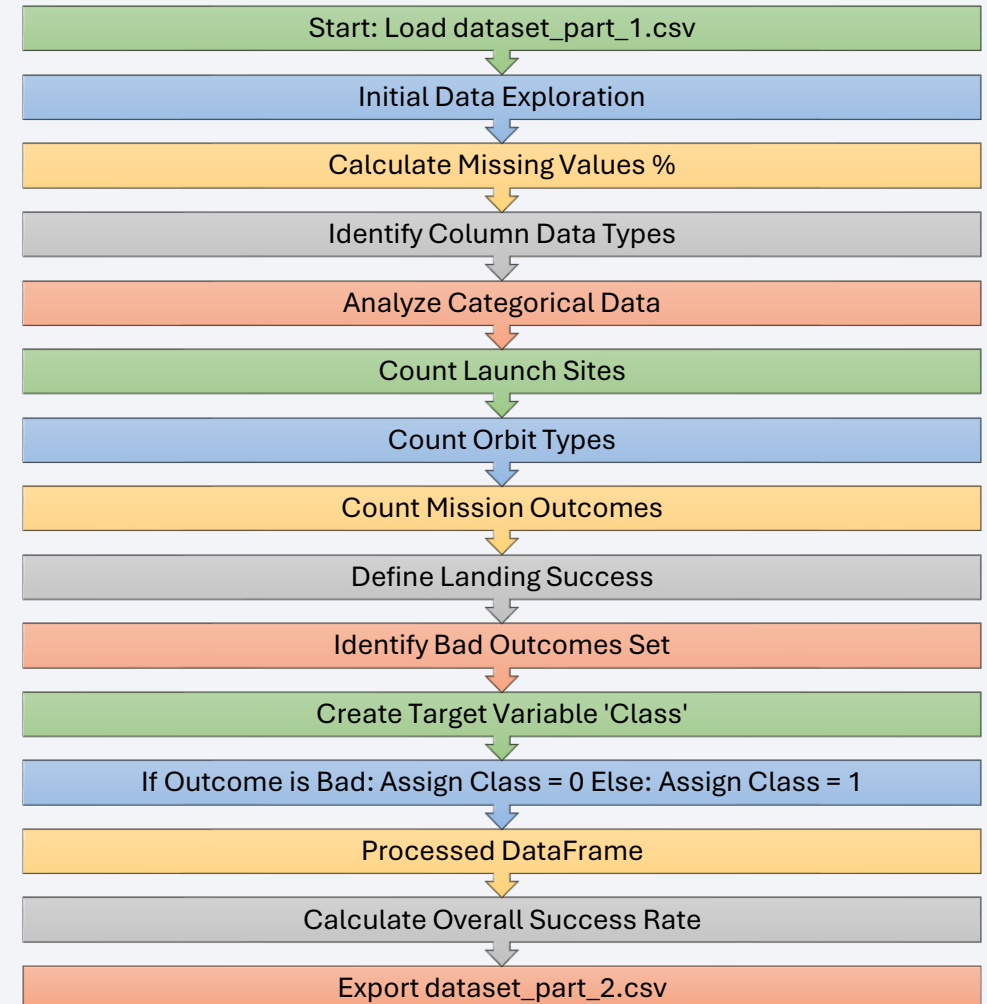
- Identify specific `Outcome` values that represent unsuccessful landings (`bad_outcomes`).

5. Create Target Variable:

- Generate a new binary column named `Class`.
- Assign `0` to `Class` if the `Outcome` corresponds to a `bad_outcome` (unsuccessful landing).
- Assign `1` to `Class` if the `Outcome` indicates a successful landing.

6. Calculate Success Rate: Determine the overall landing success rate by calculating the mean of the `Class` column.

7. Export Processed Data: Save the modified DataFrame, including the new `Class` column, to a new CSV file (`dataset_part_2.csv`).



EDA with Data Visualization

In Exploratory Data Analysis (EDA), various charts are plotted to understand the data's characteristics, uncover patterns, identify anomalies, and test hypotheses. These charts are used to explore relationships between different variables in a dataset, likely related to space launches, focusing on factors like flight number, payload mass, launch site, orbit, and launch success (Class). Here's a summary of the charts plotted and the likely reasons for their use:

1. Scatter Plots

- LaunchSite vs. FlightNumber
- LaunchSite vs. PayloadMass
- Orbit vs. PayloadMass
- FlightNumber vs. Orbit

Why these charts were used:

To visualize relationships between two numerical variables: The basic scatter plot of `PayloadMass` vs. `FlightNumber` was used to observe if there's a trend or correlation between the mass of the payload and the sequence of flights (e.g., does payload mass increase with later flights?).

To explore relationships with a third categorical variable: The `seaborn` categorical scatter plots were used to add another dimension to the analysis. By coloring the points with the `Class` variable (likely representing launch success or failure), the analyst could investigate how the relationship between the two primary variables (e.g., `PayloadMass` and `FlightNumber`) differs for successful versus unsuccessful launches. This helps in identifying patterns such as:

- Whether payload mass and flight number combinations have different success rates.
- If success rates for different launch sites change with flight number or payload mass.
- How the success of reaching specific orbits is related to payload mass or flight number.

2. Bar Plot

```
orbit_success_rate =  
df.groupby('Orbit')['Class'].mean().sort_values(ascending=False)  
orbit_success_rate.plot(kind='bar', color='skyblue')
```

Why these charts was used:

To compare a numerical value across different categories: A bar plot is effective for comparing the average success rate (`Class`) across different discrete categories (types of `Orbit`). This allows for an easy visual comparison to identify which orbits have historically had higher or lower success rates.

3. Line Plot

```
sns.lineplot(data=df, x='Date', y='Class', marker='o', color='green')
```

Why these charts was used:

To show trends over time: A line plot is ideal for visualizing how a variable (in this case, the average success rate or `Class`) changes over a continuous period, such as `year`. This chart would clearly illustrate whether the launch success rate has been increasing, decreasing, or remaining stable over the years.



EDA with SQL

1) Display the names of the unique launch sites in the space mission

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE
```

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

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

3) Display the total payload mass carried by boosters launched by NASA (CRS)

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)'
```

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1'
```

5) List the date when the first succesful landing outcome in ground pad was acheived.

```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)'
```

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

7) List the total number of successful and failure mission outcomes

```
%sql SELECT SUM(CASE WHEN Mission_Outcome = 'Success' THEN 1 ELSE 0 END) AS Success_Count, SUM(CASE WHEN Mission_Outcome = 'Failure (in flight)' THEN 1 ELSE 0 END) AS Failure_Count FROM SPACEXTABLE;
```

8) List all the booster_versions that have carried the maximum payload mass. Use a subquery.

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

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.

```
%sql SELECT strftime('%m', Date) AS Month_Number, CASE strftime('%m', Date) WHEN '01' THEN 'January' WHEN '02' THEN 'February' WHEN '03' THEN 'March' WHEN '04' THEN 'April' WHEN '05' THEN 'May' WHEN '06' THEN 'June' WHEN '07' THEN 'July' WHEN '08' THEN 'August' WHEN '09' THEN 'September' WHEN '10' THEN 'October' WHEN '11' THEN 'November' WHEN '12' THEN 'December' END AS Month_Name, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE Landing_Outcome = 'Failure (drone ship)' AND strftime('%Y', Date) = '2015'
```

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 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

Map Object	Specific Use	Purpose
<code>folium.Map</code>	Base map canvas	Provide a foundational layer for all visual elements; initially centered on NASA Johnson Space Center.
<code>folium.Circle</code>	NASA Johnson Space Center & each launch site	Visually highlight the initial map focus and mark the geographical coordinates of launch sites.
<code>folium.Marker (DivIcon)</code>	NASA JSC & each launch site (with names)	Provide clear textual identification for key locations and launch sites directly on the map.
	Individual launch records (color-coded)	Visually represent launch outcomes (success/failure) for quick assessment of site success rates.
	Proximity points (coastline, railway, etc.)	Visually pinpoint nearby features and display their calculated distance from a launch site.
<code>folium.plugins.MarkerCluster</code>	Grouping launch outcome markers	Manage visual clutter when multiple launches originate from the same coordinates.
<code>folium.plugins.MousePosition</code>	Displaying coordinates on mouseover	Allow users to easily identify coordinates of any point of interest while exploring the map.
<code>folium.PolyLine</code>	Lines connecting launch sites to proximity points	Visually depict distances and spatial relationships between launch sites and nearby key features.



Build a Dashboard with Plotly Dash

SpaceX Dashboard: Interactive Launch Analysis

1. Launch Site Dropdown Menu: A dropdown menu () that allows users to select a specific launch site or view data for all launch sites.

Why it was added: This interactive element () provides users with the flexibility to filter the dashboard's data based on the launch site of interest. This helps in comparing the performance of different launch sites or focusing on a particular one.

2. Success Pie Chart: A pie chart () that visualizes launch success rates.

How it interacts: The chart dynamically updates based on the selection made in the "Launch Site Dropdown Menu" ().

- If "All Sites" is selected, the pie chart shows the total successful launches for each site ().
- If a specific launch site is selected, the pie chart displays the proportion of successful versus failed launches for that particular site ().

Why it was added: This visual representation () makes it easy to quickly understand the overall success of launches across different sites or the success/failure ratio for a specific site. Pie charts are effective for showing proportions of a whole.

3. Payload Range Slider: A range slider () that allows users to select a minimum and maximum payload mass in kilograms.

Why it was added: This interaction () enables users to filter the data displayed in the scatter chart based on a specific payload mass range. This is useful for investigating how payload mass affects launch outcomes.

4. Payload vs. Success Scatter Chart: A scatter chart () that plots payload mass against launch outcome (class), with points colored by the "Booster Version Category".

How it interacts: This chart is updated based on selections from both the "Launch Site Dropdown Menu" and the "Payload Range Slider" ().

- The selected launch site filters the data to show only launches from that site (or all sites).
- The selected payload range filters the data to include only launches within that mass range.

Why it was added: This chart () helps to visualize the relationship and correlation between payload mass and launch success. Coloring by "Booster Version Category" adds another dimension to the analysis, potentially revealing if certain booster versions are more successful with specific payload masses.



Predictive Analysis (Classification)

Predictive Analysis (Classification) Model Development Summary

The provided notebook details the process of building, evaluating, and improving classification models to predict an outcome, likely related to SpaceX launch success. The primary goal was to identify the best performing classification model from a set of candidates.

The process involved these key stages:

1. Data Preparation:

- Loading the dataset.
- Feature engineering and selection, where relevant features for prediction were chosen.
- Preprocessing data, which likely included encoding categorical variables (e.g., using one-hot encoding) and potentially feature scaling.
- Splitting the data into training (`X_train`, `Y_train`) and testing (`X_test`, `Y_test`) sets to evaluate model performance on unseen data.

2. Model Training and Hyperparameter Tuning:

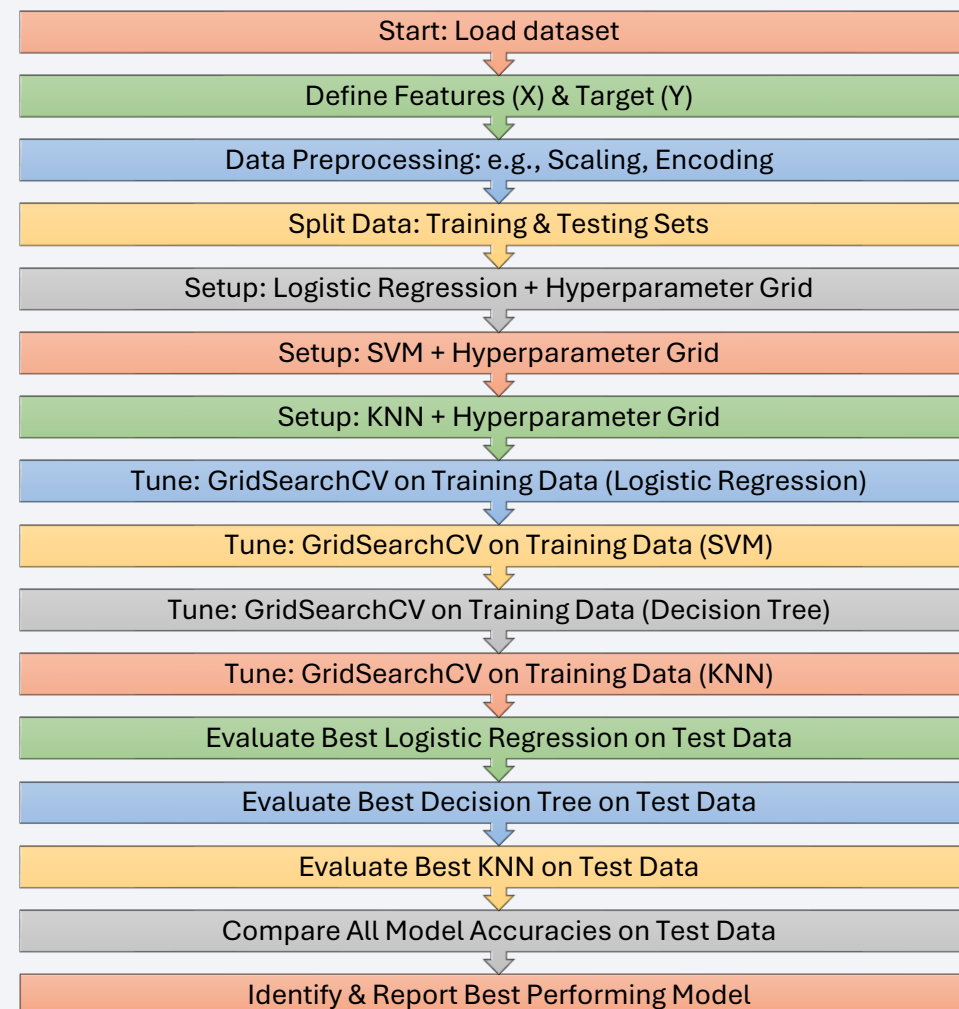
- Four different classification algorithms were selected for training:
 - Logistic Regression
 - Support Vector Machine (SVM)
 - Decision Tree
 - K-Nearest Neighbors (KNN)
- For each algorithm, hyperparameter tuning was performed using `GridSearchCV`. This method systematically works through multiple hyperparameter combinations and, using cross-validation on the training data, identifies the set of hyperparameters that yields the best performance for each model type.

3. Model Evaluation:

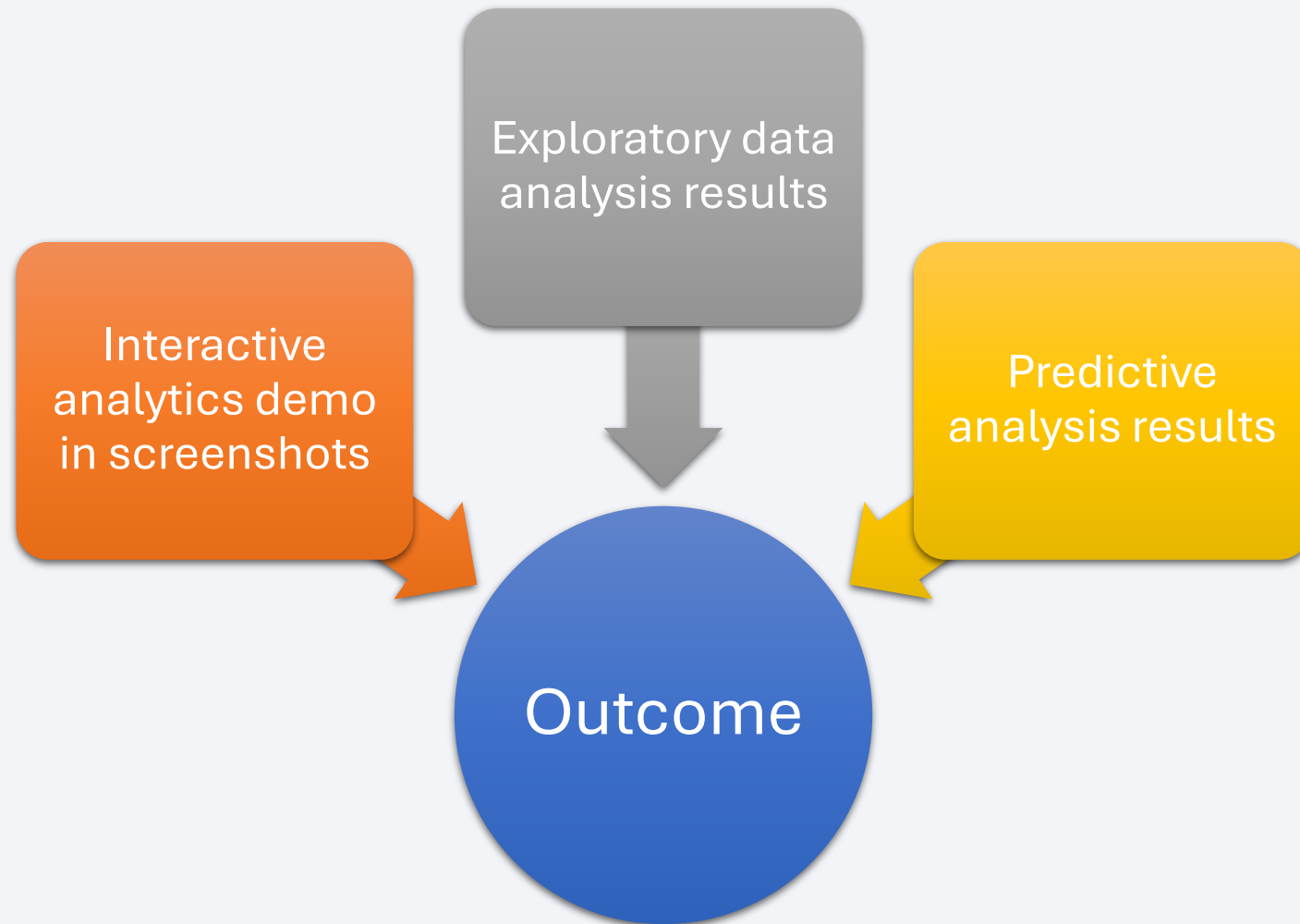
- After identifying the best hyperparameters for each model using `GridSearchCV` on the training data, each tuned model was then evaluated on the separate test set (`X_test`, `Y_test`).
- The primary metric used for evaluation was accuracy, which measures the proportion of correct predictions made by the model on the test data

4. Best Model Selection:

- The accuracies of all the tuned models (Logistic Regression, SVM, Decision Tree, KNN) on the test data were compared.
- The model with the highest accuracy on the test set was declared the best performing classification model. In this specific case, **Logistic Regression** was found to be the best model with an accuracy of approximately **83.33%**.



Results



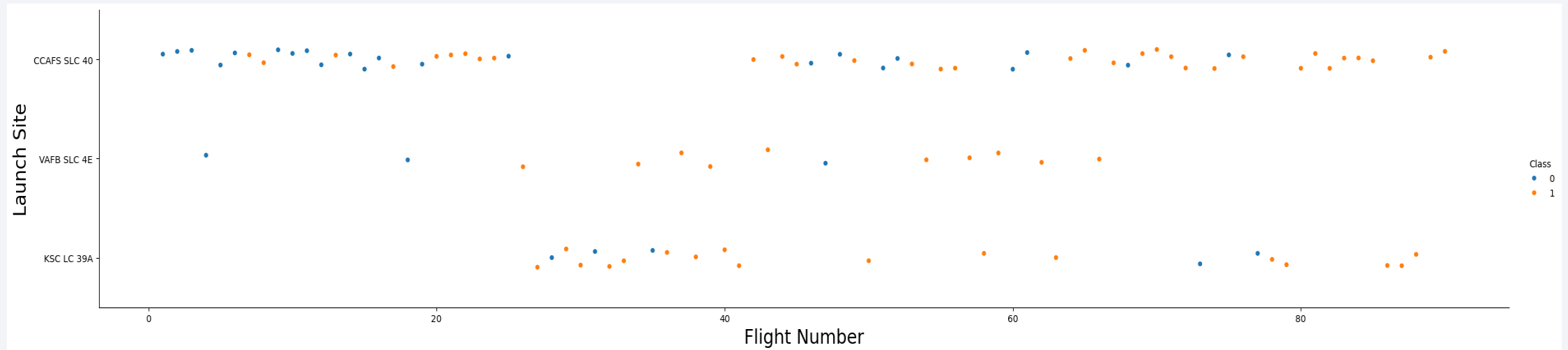


Section 2

Insights drawn from EDA

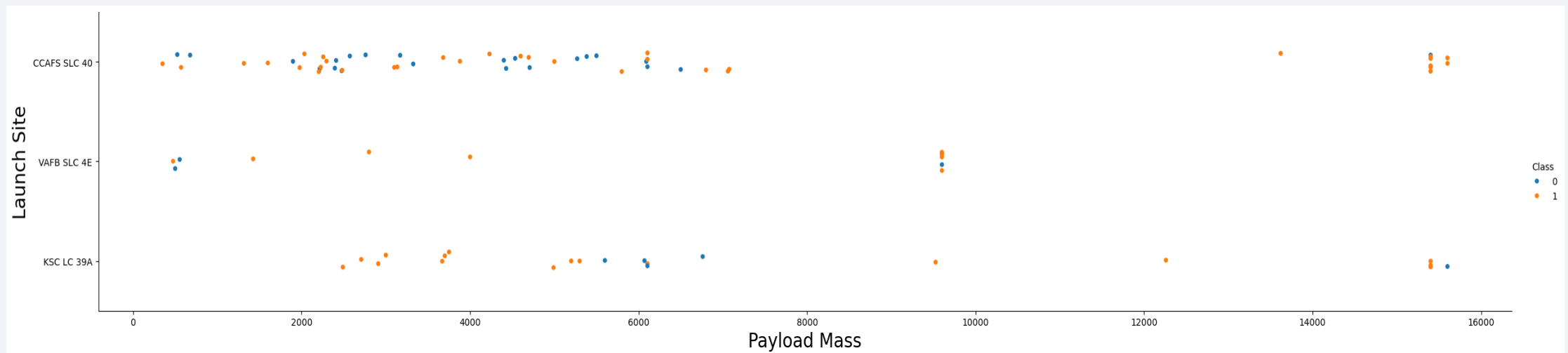
Flight Number vs. Launch Site

The graph comparing flight number and launch site shows that CCAFS SLC 40 has the highest number of launches and the best overall success rate. VAFB SLC 4E ranks second in success, followed by KSC LC 39A. The data also indicates that the general success rate has improved over time, with no recorded failures after flight number 79. Overall, launch sites with a higher number of flights tend to show greater success rates.



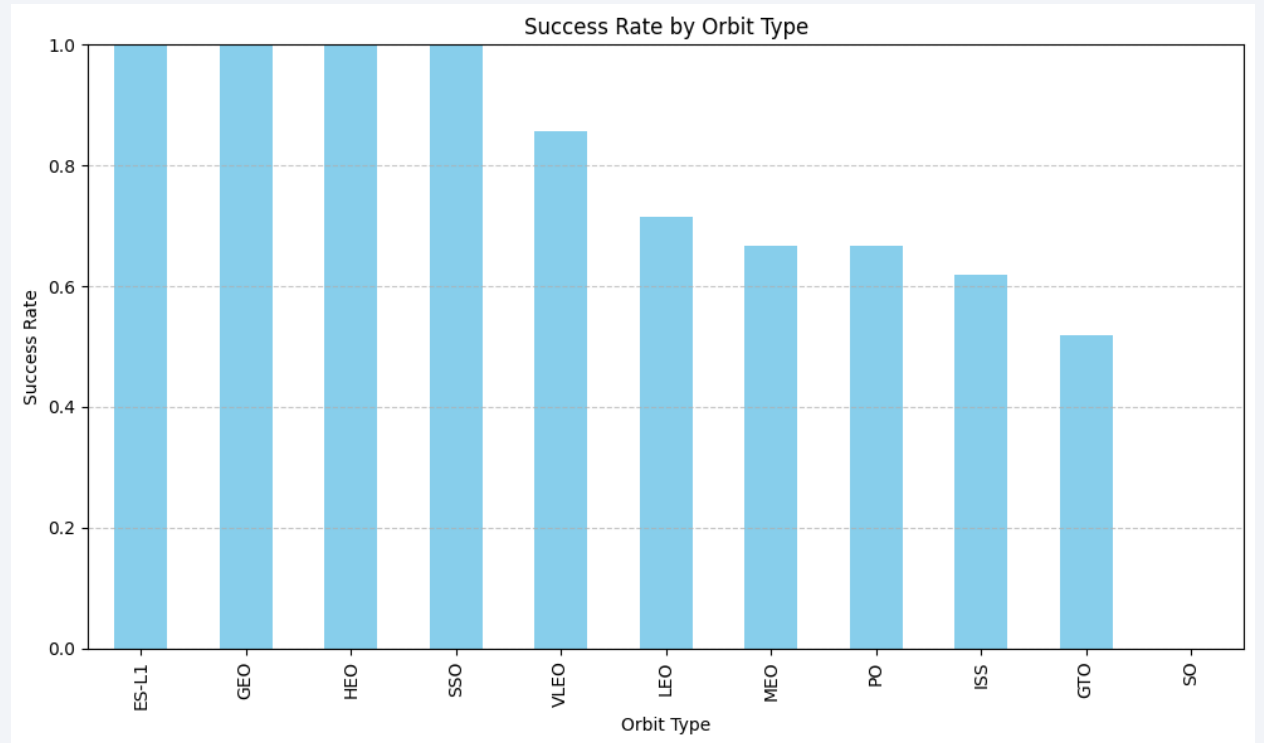
Payload vs. Launch Site

The Payload vs. Launch Site graph shows that VAFB SLC-4E has no launches with heavy payloads (over 10,000 kg). Most flights from CCAFS SLC 40 carry payloads under 8,000 kg, while the minimum payload from KSC LC 39A is around 2,000 kg. Payloads over 9,000 kg—roughly the weight of a school bus—tend to have high success rates. Only CCAFS SLC 40 and KSC LC 39A appear capable of handling payloads over 12,000 kg. While heavier payloads (over 7,000 kg) generally show higher success rates, the graph does not reveal a clear pattern indicating that launch success is directly dependent on payload mass or launch site.



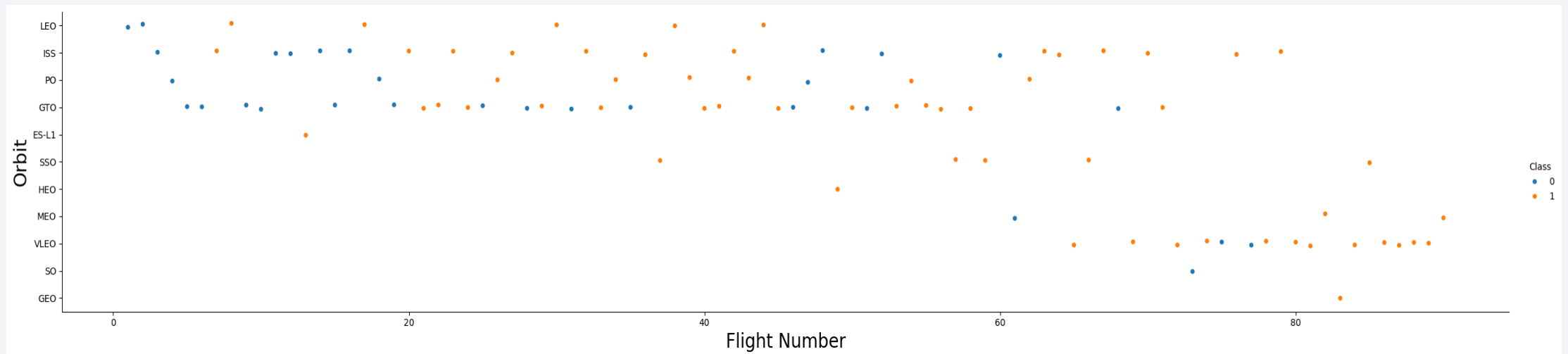
Success Rate vs. Orbit Type

The Success Rate vs. Orbit bar chart shows that ES-L1, GEO, HEO, and SSO have the 100% and highest success rates. These are followed by VLEO with over 80% success and LEO with over 70%.



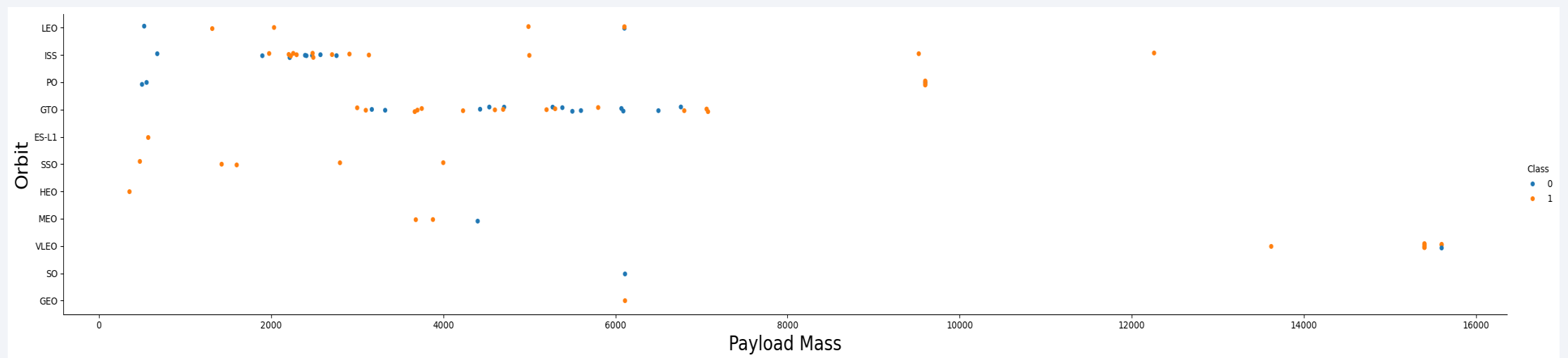
Flight Number vs. Orbit Type

The Flight Number vs. Orbit Type graph shows that only GTO, PO, ISS, and LEO orbits have flight numbers ranging from 0 to 100, while other orbits start around 40–60 and go up to 100. Overall, success rates have improved over time across all orbits. VLEO appears to be an emerging opportunity due to its recent rise in launch frequency. In LEO, success seems to correlate with the number of flights, whereas no such relationship is evident for GTO.



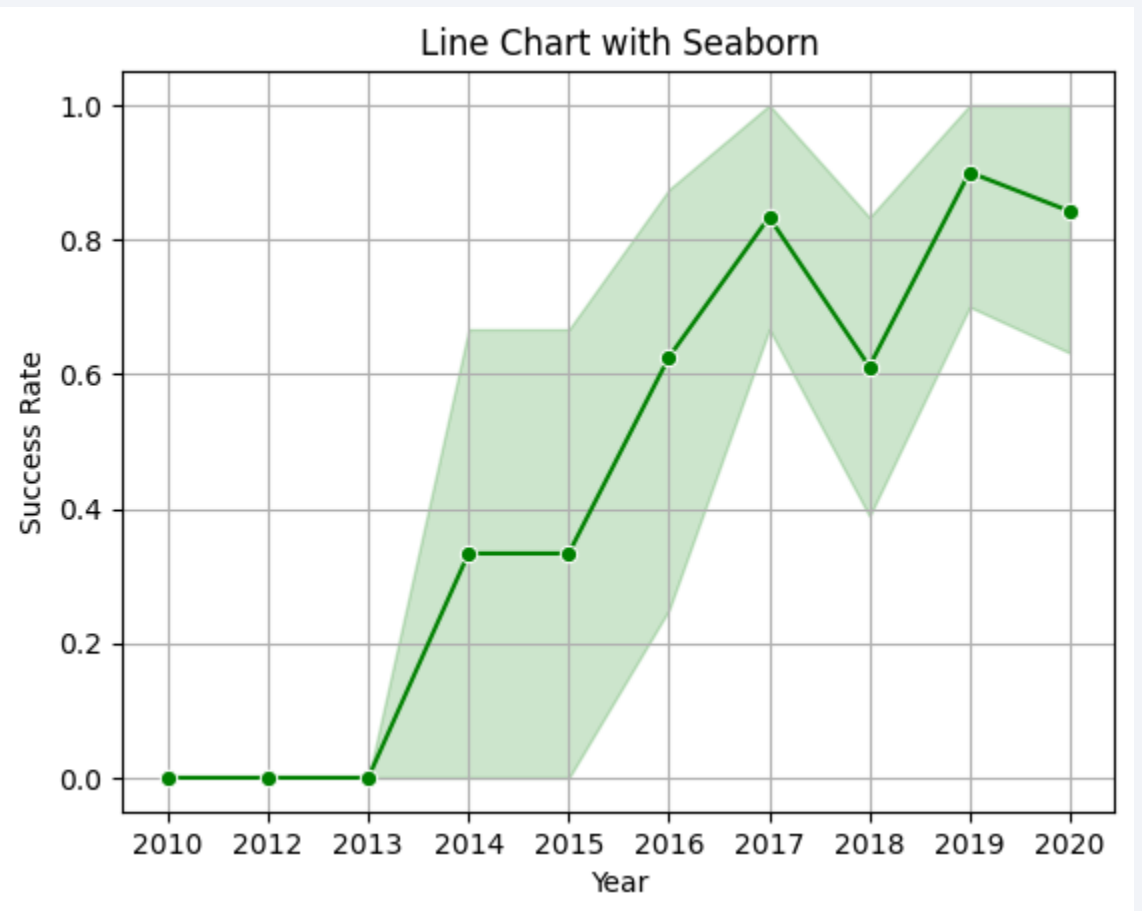
Payload Mass vs. Orbit Type

The Payload Mass vs. Orbit Type plot shows that launches with payloads over 8,000 kg are rare and mostly successful in VLEO, ISS, and PO orbits. Other orbits typically involve payloads under 8,000 kg. The ISS orbit supports the widest range of payload masses and maintains a high success rate. There are only a few launches to SO and GEO orbits. While heavy payloads show positive outcomes in Polar, LEO, and ISS orbits, there is no clear relationship between payload mass and success rate for GTO, where both successful and unsuccessful outcomes are observed.



Launch Success Yearly Trend

We can observe that the success rate since 2013 kept increasing till 2020.



All Launch Site Names

Display the names of the unique launch sites in the space mission

SQL Query:

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE
```

Description:

We can find the names of the unique launch sites using **DISTINCT**

I used this query to retrieve **unique launch site names** from the **SPACEXTABLE**.

Breakdown:

- **SELECT**: Specifies the column to retrieve, which is **Launch_Site** in this case.
- **DISTINCT**: Ensures that only **unique** (non-duplicate) values are returned. If a launch site appears multiple times in the table, it will be shown only once in the result.
- **FROM SPACEXTABLE**: Indicates the table (**SPACEXTABLE**) from which the data is being selected.
- **%sql**: This is a **magic command** often used in **Jupyter Notebooks** to run SQL queries within a cell.

Query Result:

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

Launch Site Names Begin with 'CCA'

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

SQL Query:

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

Description:

This SQL query to retrieve up to **5 rows** from the **SPACEXTABLE** where the **Launch_Site** contains the substring "CCA".

Query Result:

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

Total Payload Mass

Display the total payload mass carried by boosters launched by NASA (CRS)

SQL Query:

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)'
```

Description:

I used this SQL query to calculate the **total payload mass** (in kilograms) for all launches conducted for **NASA (CRS)** from the **SPACEXTABLE**.

Breakdown:

- **%sql**: A **magic command** used in notebook environments (like Jupyter) to run SQL queries inside a code cell.
- **SELECT SUM(PAYLOAD_MASS__KG_)**:
- **SUM(...)** is an **aggregate function** that returns the **total** (sum) of values in the specified column.
- **PAYLOAD_MASS__KG_** is the column representing the **mass of the payload** in kilograms.
- **FROM SPACEXTABLE**: Specifies the table from which data is being retrieved.
- **WHERE customer = 'NASA (CRS)'**: Filters the records to include only those where the customer is 'NASA (CRS)', which refers to NASA's Commercial Resupply Services missions.

Query Result:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

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

SQL Query:

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1'
```

Description:

I used this SQL query calculates the **average payload mass** (in kilograms) for all launches that used the **'F9 v1.1' booster version**, based on data from the **SPACEXTABLE**.

Breakdown:

- **%sql**: A **magic command** used in notebook environments (like Jupyter) to run SQL queries inside a code cell.
- **SELECT AVG(PAYLOAD_MASS__KG_)**:
- **AVG(...)** is an **aggregate function** that computes the average of the values in the specified column.
- **PAYLOAD_MASS__KG_** is the column representing the **mass of the payload** in kilograms.
- **FROM SPACEXTABLE**: Specifies the table from which data is being retrieved.
- **WHERE Booster_Version = 'F9 v1.1'**: Filters the rows to include only those where the booster version is 'F9 v1.1' (a specific version of the Falcon 9 rocket).

Query Result:

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

List the date when the first successful landing outcome in ground pad was achieved.

SQL Query:

```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)'
```

Description:

I used this SQL query to retrieve the **earliest date** on which a rocket achieved a **successful landing on a ground pad**, based on the data in the `SPACEXTABLE`.

Breakdown:

- `%sql`: A **magic command** used in notebook environments (like Jupyter) to run SQL queries inside a code cell.
- `SELECT MIN(Date)`:
- `MIN(...)` is an **aggregate function** that returns the **earliest (minimum) value** in the `Date` column.
- `Date` refers to the column storing the **launch or landing date**.
- `FROM SPACEXTABLE`: Specifies the table from which data is being retrieved.
- `WHERE Landing_Outcome = 'Success (ground pad)'`: Filters the records to include only those where the landing outcome was a **successful landing on a ground pad**.

Query Result:

MIN(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

SQL Query:

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE (Landing_Outcome = 'Success (drone ship)'  
AND (PAYLOAD_MASS_KG_ > 4000 AND PAYLOAD_MASS_KG_ < 6000))
```

Description:

I used this SQL query to retrieve the **booster versions** used in SpaceX launches where the **landing was successful on a drone ship** and the **payload mass was between 4,000 and 6,000 kg**.

Breakdown:

- `%sql`: A **magic command** used in notebook environments (like Jupyter) to run SQL queries inside a code cell.
- `SELECT Booster_Version`: Specifies that the query will return the **booster version** used in the selected launches.
- `FROM SPACEXTABLE`: Specifies the table from which data is being retrieved.
- `WHERE Landing_Outcome = 'Success (ground pad)'`: Only include launches where the rocket **successfully landed on a drone ship**.
- `(PAYLOAD_MASS_KG_ > 4000 AND PAYLOAD_MASS_KG_ < 6000)`: Only include those launches where the **payload mass was greater than 4,000 kg but less than 6,000 kg**.

Query Result:

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

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

SQL Query:

```
%sql SELECT SUM(CASE WHEN Mission_Outcome = 'Success' THEN 1 ELSE 0 END) AS  
Success_Count, SUM(CASE WHEN Mission_Outcome = 'Failure (in flight)' THEN 1 ELSE  
0 END) AS Failure_Count FROM SPACEXTABLE;
```

Description:

I used this SQL query to count the number of **successful missions** and **in-flight mission failures** from the `SPACEXTABLE` by using conditional aggregation.

Breakdown:

- `%sql`: A **magic command** used in notebook environments (like Jupyter) to run SQL queries inside a code cell.
- `SELECT`: Begins the query and specifies the columns to return.
- `SUM(CASE WHEN Mission_Outcome = 'Success' THEN 1 ELSE 0 END) AS Success_Count`: Counts the total number of missions where the outcome was marked as **'Success'**.
- `SUM(CASE WHEN Mission_Outcome = 'Failure (in flight)' THEN 1 ELSE 0 END) AS Failure_Count`: Counts the total number of missions that ended in **'Failure (in flight)'**.

Query Result:

Success_Count	Failure_Count
98	1

Boosters Carried Maximum Payload

List all the booster_versions that have carried the maximum payload mass. Using a subquery.

SQL Query:

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

Description:

I used this SQL query to find booster version(s) used in the launch that carried the **heaviest payload** recorded in the `SPACEXTABLE`.

Breakdown:

- `%sql`: A **magic command** used in notebook environments (like Jupyter) to run SQL queries inside a code cell.
- `SELECT Booster_Version`: Retrieves the **booster version(s)** associated with the condition below.
- `FROM SPACEXTABLE`: Specifies the table to query.
- `WHERE PAYLOAD_MASS__KG_ = (...)`: Filters the data to only include rows where the payload mass is equal to the **maximum payload mass** in the entire table.
- `(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)`: Finds the **maximum value** in the `PAYLOAD_MASS__KG_` column. This value is used to match the row(s) with the heaviest payload.

Query Result:

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

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.

SQL Query:

```
%sql SELECT strftime('%m', Date) AS Month_Number, CASE strftime('%m', Date) WHEN '01' THEN 'January' WHEN '02' THEN 'February' WHEN '03' THEN 'March' WHEN '04' THEN 'April' WHEN '05' THEN 'May' WHEN '06' THEN 'June' WHEN '07' THEN 'July' WHEN '08' THEN 'August' WHEN '09' THEN 'September' WHEN '10' THEN 'October' WHEN '11' THEN 'November' WHEN '12' THEN 'December' END AS Month_Name, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE Landing_Outcome = 'Failure (drone ship)' AND strftime('%Y', Date) = '2015'
```

Description: I used this SQL query to retrieve detailed information about drone ship landing failures in the year 2015, and also extracts and formats the month of each failure.

Breakdown: This SQL query, executed using the `%sql` magic command in a notebook environment like Jupyter, extracts data from the `SPACEXTABLE` specifically for launches in the year 2015 where the landing outcome was a failure on a drone ship. It uses `strftime('%m', Date) AS Month_Number` to extract the numerical month from the `Date` column and a `CASE` statement to convert that number into a full month name (e.g., `'01'` becomes `'January'`). The query also retrieves the `Landing_Outcome`, `Booster_Version` (the rocket model used), and `Launch_Site` (the location of the launch) to provide more context for each failed landing. The `WHERE` clause ensures that only records with a landing outcome of `'Failure (drone ship)'` and a launch year of 2015 are included in the result.

Query Result:

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

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

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

SQL Query:

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

Description: I used this SQL query to analyze the **number of different landing outcomes** for SpaceX missions that occurred **between June 4, 2010, and March 20, 2017**.

Breakdown:

- **SELECT Landing_Outcome, COUNT(*) AS Outcome_Count:** Retrieves the **type of landing outcome** (e.g., success, failure, no attempt) from the Landing_Outcome column. Uses **COUNT(*)** to count the **number of times** each unique landing outcome occurred. Renames this count as **Outcome_Count**.
- **FROM SPACEXTABLE:** Specifies the source table containing launch and landing data.
- **WHERE Date BETWEEN '2010-06-04' AND '2017-03-20':** Filters the records to include **only launches within the specified date range**.
- **GROUP BY Landing_Outcome:** Groups the results by each unique landing outcome so that the **COUNT(*)** applies to each group.
- **ORDER BY Outcome_Count DESC:** Sorts the output in **descending order**, so the most frequent landing outcomes appear first.

Query Result:

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

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

Section 3

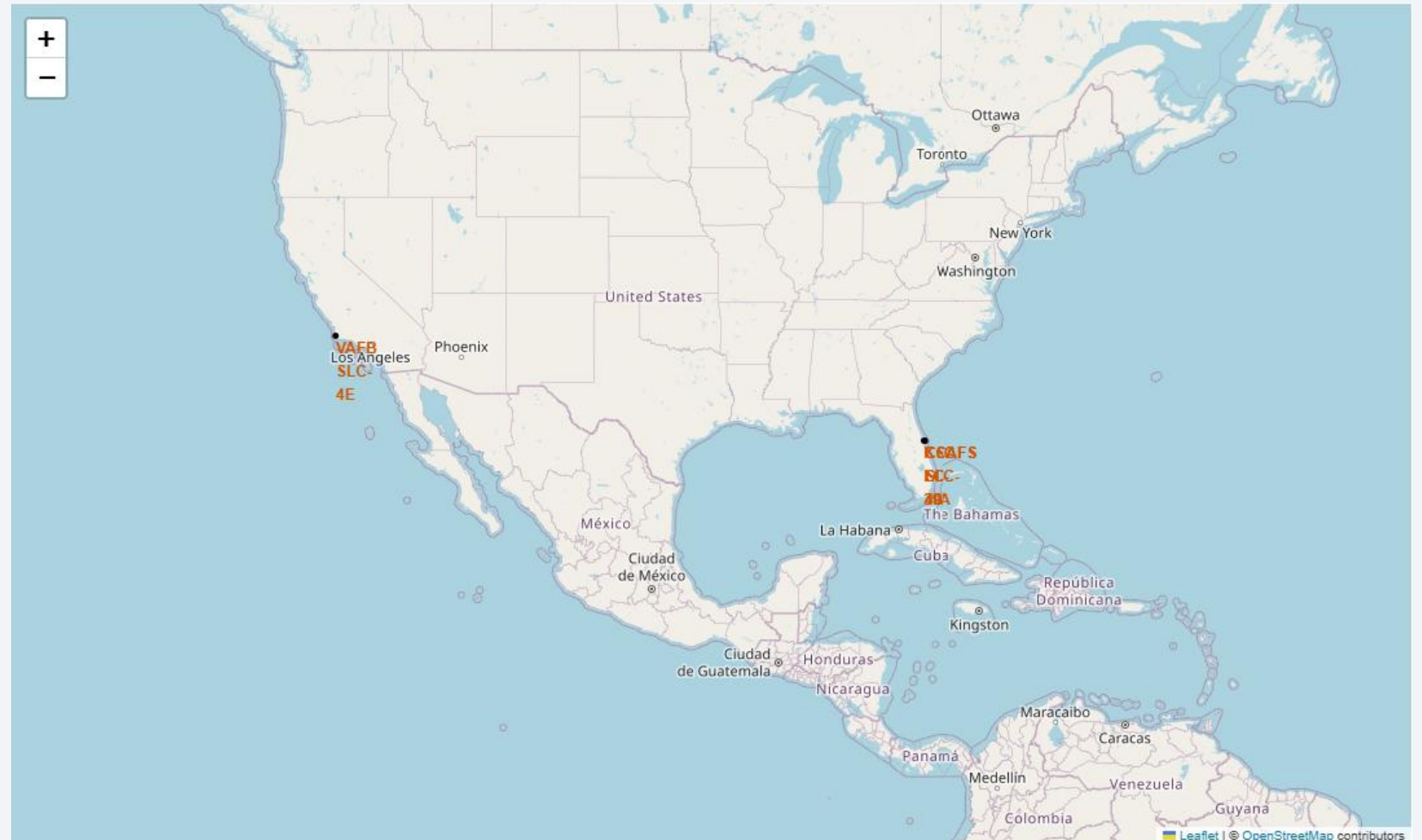
Launch Sites Proximities Analysis

Mapping SpaceX Launch Sites Using Folium

The SpaceX launch sites are located **near the coasts of the United States**, specifically in the Florida and California regions.


A folium map is created using the `folium.Map()` function to display all launch sites. Each site's location is marked using its latitude and longitude by adding `folium.Circle()` and `folium.Marker()` objects to the map.

These elements are added using the `add_child()` method. The map reveals that three launch sites—**CCAFS LC-40, CCAFS SLC-40, and KSC LC-39A**—are clustered on the eastern coast, close to one another. In contrast, the VAFB SLC-4E site is located on the western coast of the U.S.

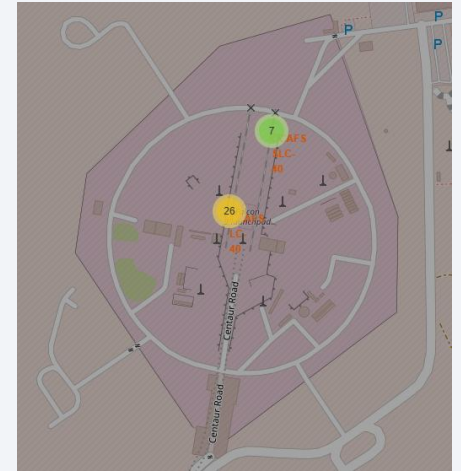


Color Labeled Markers for Launch Sites Using Folium

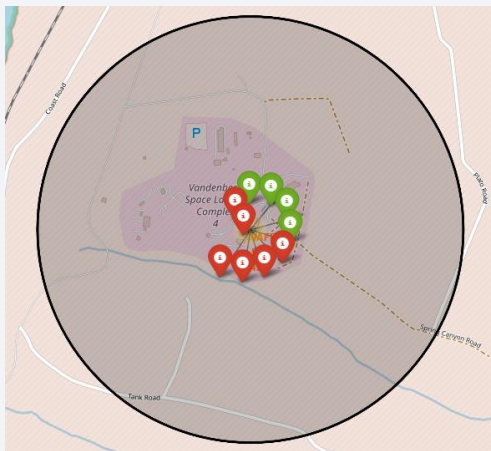
 Green Marker shows successful launches

 Red Marker shows failures

These screenshots clearly indicate that **KSC LC-39A** has the highest probability of success.



VAFB SLC-4E



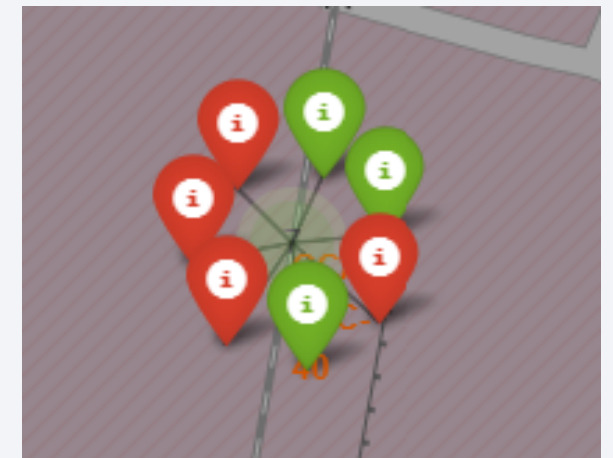
KSC LC-39A



CCAFS LC-40



CCAFS SLC-40



Launch Site Proximity to Landmarks Using Folium

Q: Are launch sites in close proximity to railways?

Ans: Yes, Nearest railway is at 1.29 km.

Q: Are launch sites in close proximity to highways?

Ans: Yes, Nearest highway is at 0.6 km.

Q: Are launch sites in close proximity to coastline?

Ans: Yes, Nearest coastline is at 0.86 km.

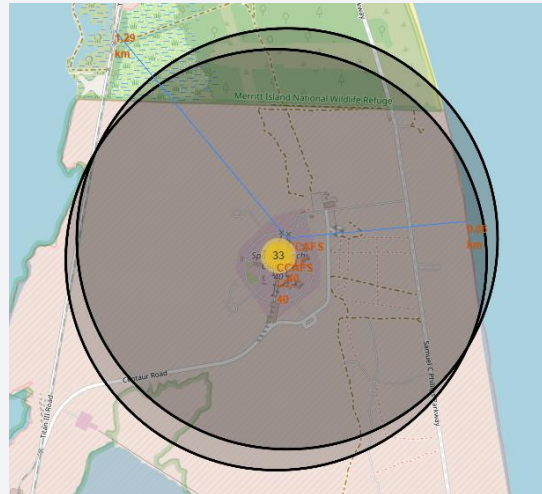
Q: Do launch sites keep certain distance away from cities?

Ans: No, Nearest city from the launch site is at a far distance of 53.82 km.

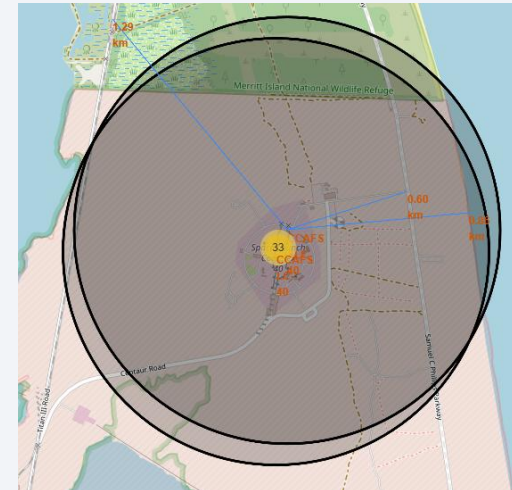
Distance to Coastline



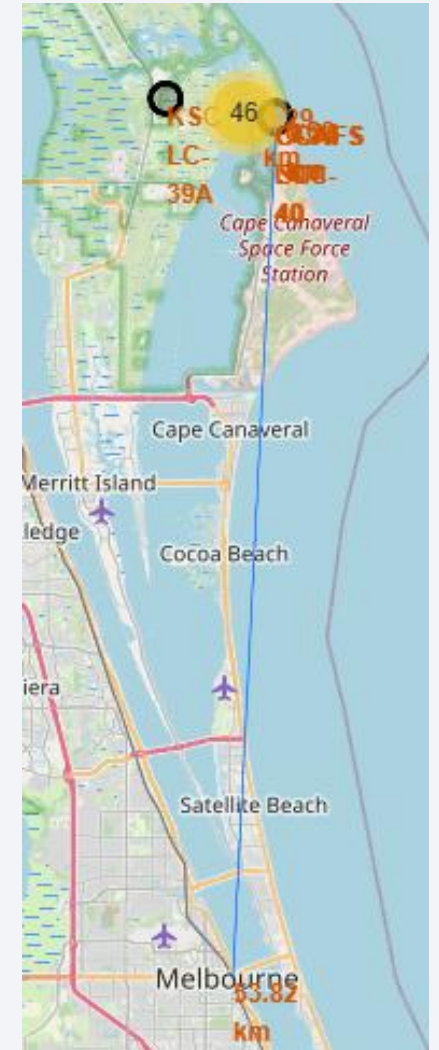
Distance to Railway Line



Distance to Highway



Distance to Nearest City - Melbourne



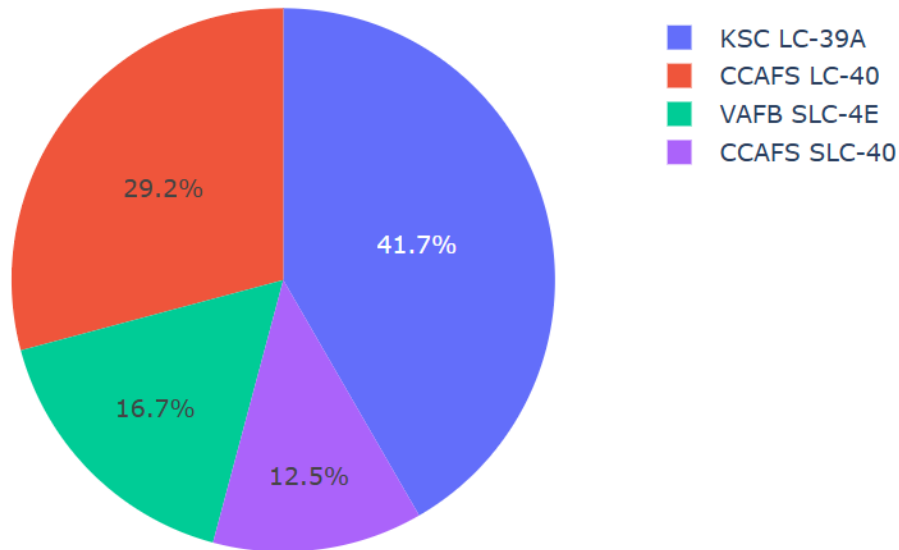


Section 4

Build a Dashboard with Plotly Dash

Launch Success Percentage by Site (Pie Chart Dashboard)

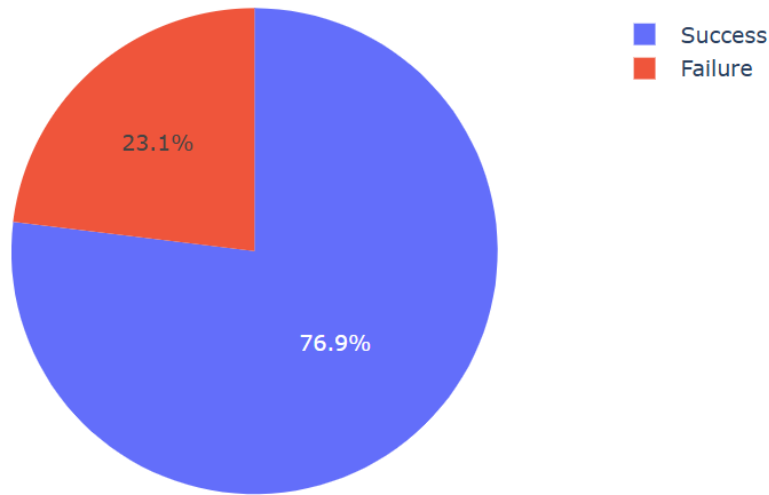
Total Successful Launches by Site



The graph under the “All Sites” option indicates that **KSC LC-39A** has the highest success rate among all launch sites, while **CAFS SLC-40** has the lowest. It is evident that KSC LC-39A recorded the greatest number of successful launches compared to the other sites.

Launch Site with Highest Success Ratio – KSC LC-39A

Success vs. Failure for site KSC LC-39A



KSC LC-39A achieved a success rate of **76.9%** and a failure rate of **23.1%**, making it the launch site with the highest success ratio, as shown in the screenshot. In the pie chart, the **success percentage** is represented in **purple**, while the failure percentage is shown in **red**. The legend indicates the “class” field, where **0 represents failure** and **1 represents success**.

Launch Outcome by Payload Range – Scatter Plot Analysis

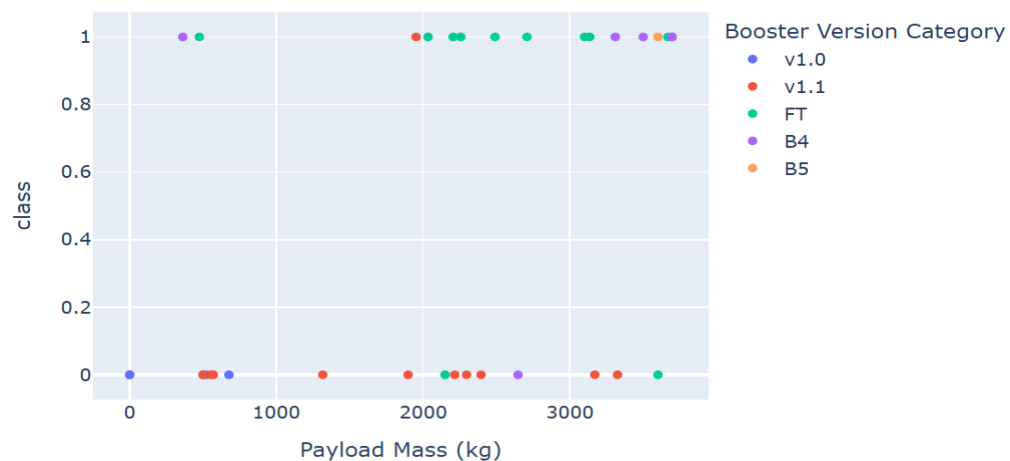
The data indicates that **lower payload masses tend to have higher success rates** compared to heavier payloads. The scatter plot is generated based on the input selected from the dropdown and the payload mass range defined by the slider. It illustrates the **correlation between payload mass and launch success rate** across all sites, while also displaying **booster version classifications**.

When the slider is set from **0 to 10,000 kg**, it includes **all recorded launches**. In this range, **booster version FT** shows the **highest number of successful launches**, while **version V1.1** accounts for the **most failures**. For payloads **greater than 2,500 kg**, **booster version FT** leads both in successes (**7 launches**) and failures (**6 launches**). There are **very few launches above 5,000 kg**, with **FT** still being the most successful version in this range. For payloads over **7,500 kg**, only **two launches** are recorded—one success and one failure, both using **booster version B4**. The **highest success rate** is observed for payloads **under 5,000 kg**. The **maximum payload mass launched** was **9,600 kg**.

Payload range (Kg):



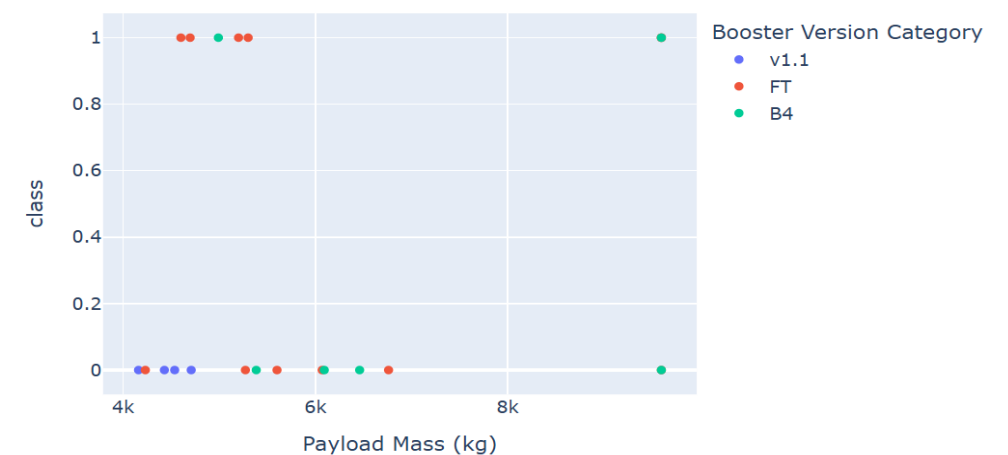
Correlation Between Payload and Success for All Sites



Payload range (Kg):



Correlation Between Payload and Success for All Sites



Section 5

Predictive Analysis (Classification)

KNN Model Performance

```
GridSearchCV
GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'p': [1, 2]})
  best_estimator_: KNeighborsClassifier
                    KNeighborsClassifier(n_neighbors=10, p=1)
                      KNeighborsClassifier
                        KNeighborsClassifier(n_neighbors=10, p=1)
```

Interpretation & Insights

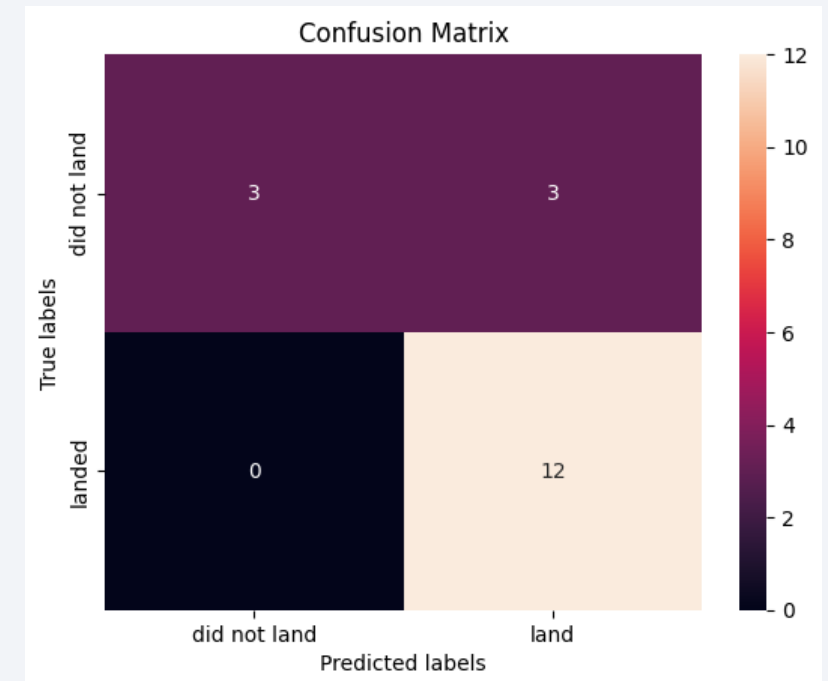
- 1) **True Positives (TP = 12):** The model correctly predicted 12 cases where the rocket **did land**.
- 2) **True Negatives (TN = 3):** The model correctly predicted 3 cases where the rocket **did not land**.
- 3) **False Positives (FP = 3):** The model incorrectly predicted 3 cases as **landed** when the rocket **did not land**.
- 4) **False Negatives (FN = 0):** The model did not make any incorrect predictions **of did not land** when the rocket had actually landed.

Key Observations

The model has **high recall** for the "landed" class, meaning it correctly identifies almost all rockets that actually landed. However, it misclassifies some rockets that **did not land** as having landed (3 false positives), indicating a **bias toward predicting landings**. There are **no false negatives**, which is beneficial in cases where missing a successful landing is more critical than falsely predicting one. Depending on the business case (e.g., safety or cost), the **false positives** may or may not be acceptable.

Performance Metrics Overview

- **Accuracy** = $(TP + TN) / \text{Total} = (12 + 3) / 18 = 83.3\%$
- **Precision** (for "land") = $TP / (TP + FP) = 12 / (12 + 3) = 80\%$
- **Recall** (for "land") = $TP / (TP + FN) = 12 / (12 + 0) = 100\%$
- **F1 Score** = $2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) = 88.9\%$



Logistic Regression Model Performance

```
GridSearchCV
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                        'solver': ['lbfgs']})
└─ best_estimator_:
    LogisticRegression
    └─ LogisticRegression
        └─ LogisticRegression(C=0.01)
```

Interpretation & Insights

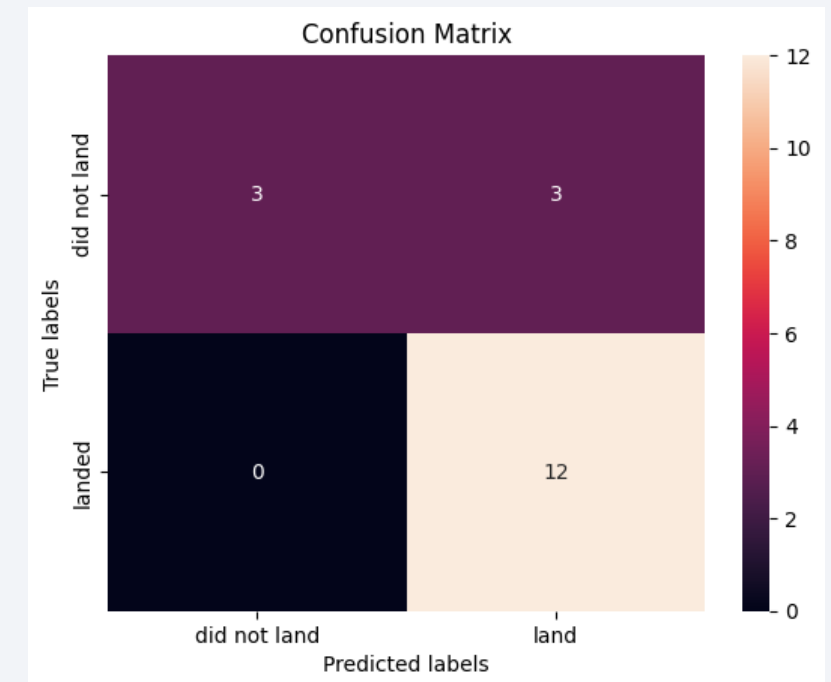
- 1) **True Positives (TP = 12):** The model correctly predicted 12 cases where the rocket **did land**.
- 2) **True Negatives (TN = 3):** The model correctly predicted 3 cases where the rocket **did not land**.
- 3) **False Positives (FP = 3):** The model incorrectly predicted 3 cases as **landed** when the rocket **did not land**.
- 4) **False Negatives (FN = 0):** The model did not make any incorrect predictions **of did not land** when the rocket had actually landed.

Key Observations

The model has **high recall** for the "landed" class, meaning it correctly identifies almost all rockets that actually landed. However, it misclassifies some rockets that **did not land** as having landed (3 false positives), indicating a **bias toward predicting landings**. There are **no false negatives**, which is beneficial in cases where missing a successful landing is more critical than falsely predicting one. Depending on the business case (e.g., safety or cost), the **false positives** may or may not be acceptable.

Performance Metrics Overview

- **Accuracy** = $(TP + TN) / \text{Total} = (12 + 3) / 18 = 83.3\%$
- **Precision** (for "land") = $TP / (TP + FP) = 12 / (12 + 3) = 80\%$
- **Recall** (for "land") = $TP / (TP + FN) = 12 / (12 + 0) = 100\%$
- **F1 Score** = $2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) = 88.9\%$



Decision Tree Model Performance

```
GridSearchCV
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                          'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                          'max_features': ['sqrt', 'log2', None],
                          'min_samples_leaf': [1, 2, 4],
                          'min_samples_split': [2, 5, 10],
                          'splitter': ['best', 'random']})
best_estimator_: DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=10, max_features='sqrt',
                      min_samples_leaf=2, min_samples_split=10,
                      splitter='random')
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=10, max_features='sqrt',
                      min_samples_leaf=2, min_samples_split=10,
                      splitter='random')
```

Interpretation & Insights

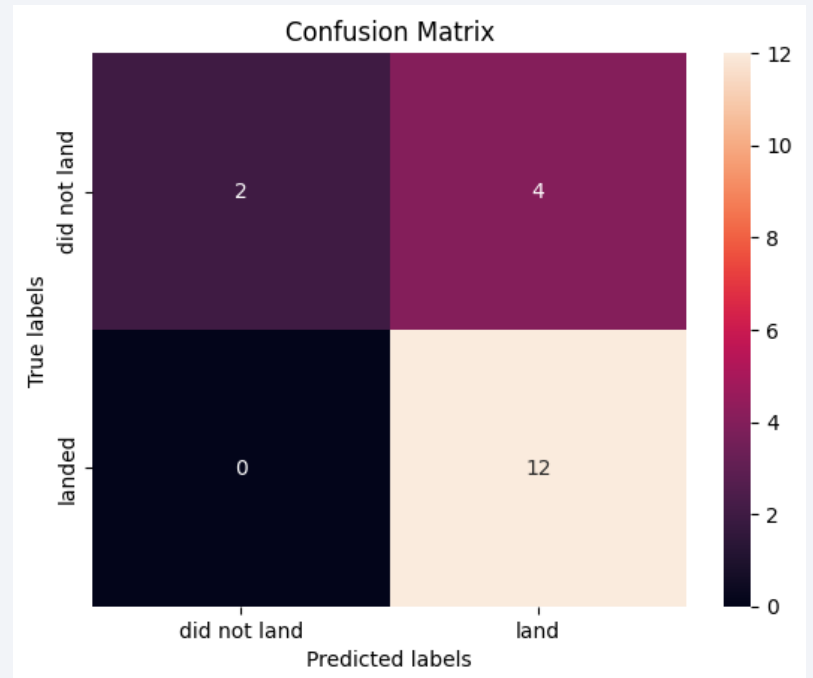
- 1) **True Positives (TP = 12)**: The model correctly predicted 12 cases where the rocket **did land**.
- 2) **True Negatives (TN = 2)**: The model correctly predicted 2 cases where the rocket **did not land**.
- 3) **False Positives (FP = 4)**: The model incorrectly predicted 4 cases as **landed** when the rocket **did not land**.
- 4) **False Negatives (FN = 0)**: The model did not make any incorrect predictions **of did not land** when the rocket had actually landed.

Key Observations

The model has **excellent recall** for predicting landings, meaning it detects all true landings. However, **precision is lower**, due to 4 false positives—cases where the model predicted a landing incorrectly. **False negatives are zero**, which is positive for use cases where missing a real landing is critical. The model appears to be **over-predicting landings**, possibly favouring the "landed" class more often.

Performance Metrics Overview

- **Accuracy** = $(TP + TN) / \text{Total} = (12 + 2) / 18 = 77.8\%$
- **Precision** (for "land") = $TP / (TP + FP) = 12 / (12 + 4) = 75\%$
- **Recall** (for "land") = $TP / (TP + FN) = 12 / (12 + 0) = 100\%$
- **F1 Score** = $2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) = 85.7\%$



SVM Model Performance

```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
                                     1.00000000e+03]),
                         'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
                                         1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})

best_estimator_: SVC
SVC(C=np.float64(1.0), gamma=np.float64(0.03162277660168379), kernel='sigmoid')

SVC
SVC(C=np.float64(1.0), gamma=np.float64(0.03162277660168379), kernel='sigmoid')
```

Interpretation & Insights

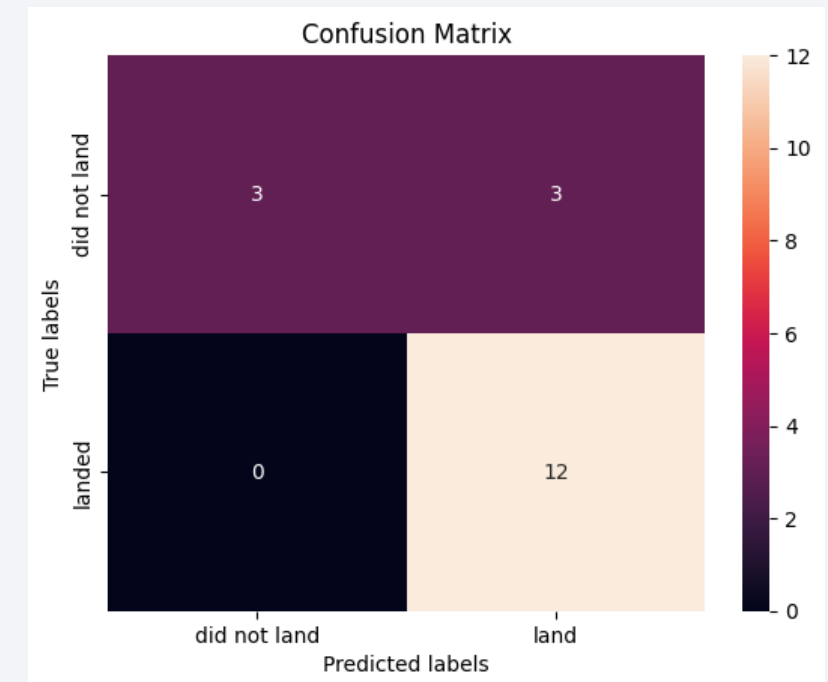
- 1) **True Positives (TP = 12)**: The model correctly predicted 12 cases where the rocket **did land**.
- 2) **True Negatives (TN = 3)**: The model correctly predicted 3 cases where the rocket **did not land**.
- 3) **False Positives (FP = 3)**: The model incorrectly predicted 3 cases as **landed** when the rocket **did not land**.
- 4) **False Negatives (FN = 0)**: The model did not make any incorrect predictions **of did not land** when the rocket had actually landed.

Key Observations

The model has **high recall** for the "landed" class, meaning it correctly identifies almost all rockets that actually landed. However, it misclassifies some rockets that **did not land** as having landed (3 false positives), indicating a **bias toward predicting landings**. There are **no false negatives**, which is beneficial in cases where missing a successful landing is more critical than falsely predicting one. Depending on the business case (e.g., safety or cost), the **false positives** may or may not be acceptable.

Performance Metrics Overview

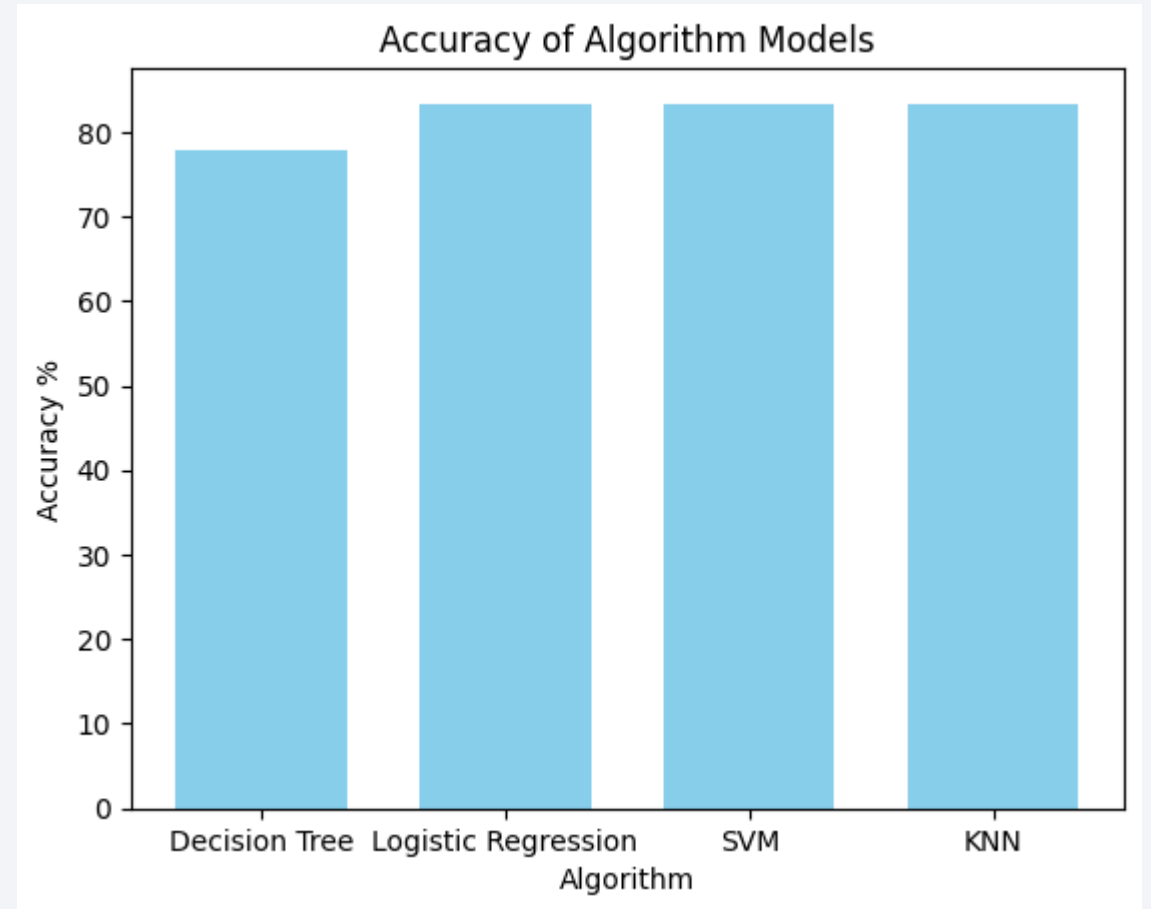
- **Accuracy** = $(TP + TN) / \text{Total} = (12 + 3) / 18 = 83.3\%$
- **Precision** (for "land") = $TP / (TP + FP) = 12 / (12 + 3) = 80\%$
- **Recall** (for "land") = $TP / (TP + FN) = 12 / (12 + 0) = 100\%$
- **F1 Score** = $2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) = 88.9\%$



Comparative Accuracy Analysis of Models

This bar chart displays the **accuracy of each machine learning model** evaluated.

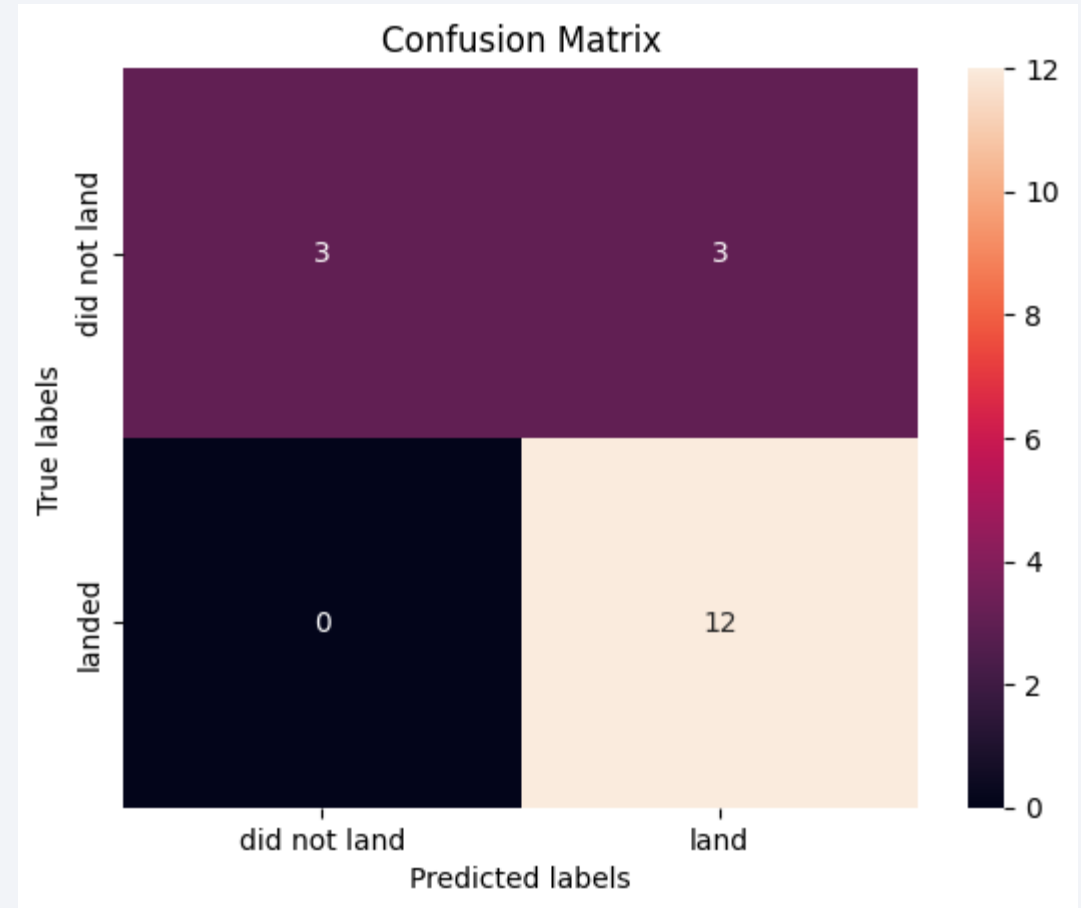
Logistic Regression, SVM, and KNN all achieved the **same accuracy of 83%**, while the **Decision Tree** model recorded the **lowest accuracy at ~78%**.



Confusion Matrix of Best Performing Model

Out of the four models, **Logistic Regression, SVM, and KNN** produced the same confusion matrix.

- There are **3 false positives**, where the model predicted a landing, but the rocket did not land.
- There are **0 false negatives**, meaning the model never predicted a failure when it actually landed.
- The matrix also shows **3 true negatives**, where the model correctly predicted a failed landing.
- And **12 true positives**, where the model correctly predicted a successful landing.

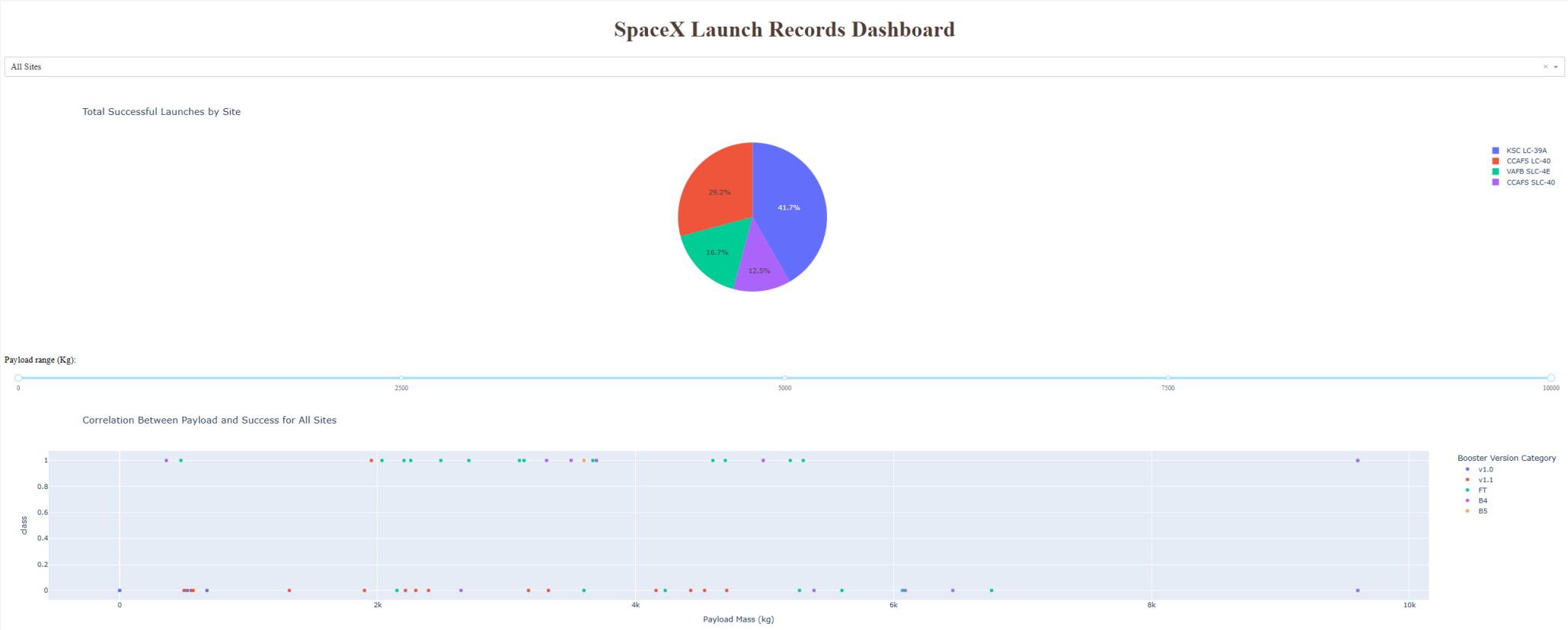


Conclusions

- **Launch Performance:** KSC LC-39A demonstrated the **highest success rate (76.9%)**, while CCAFS SLC-40 had the **most launches** but lower success.
- **Payload Insights:** Heavier payloads (>9,000 kg) generally correlate with **higher success rates**, especially from **KSC LC-39A** and **CCAFS SLC-40**. However, **no direct relationship** between payload mass and success was found across all orbits.
- **Orbit Trends:** ES-L1, GEO, HEO, and SSO achieved **100% success**, while **VLEO** showed potential as an **emerging orbit type**.
- **Geographic Factors:** Launch sites are **close to coastlines, railways, and highways**, but **far from cities** to ensure safety and accessibility.
- **Success Over Time:** There has been a **consistent increase in success rates since 2013**, with **no failures after flight number 79**.
- **Model Accuracy:** Logistic Regression, SVM, and KNN models achieved the **highest accuracy (83%)**, with reliable predictions and **no false negatives**.
- **Overall**, the study highlights **KSC LC-39A as the most reliable launch site**, increasing payload capabilities, and a positive trajectory in launch success due to improved technology and site strategies.

Appendix

Plotly Dashboard



Thank you!

