IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

ESSOU Jean-Jéliel Yao
14/05/2025

Outline

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

Executive Summary

- **Summary of Methodologies**:
  In this project, data collection is done through an API and web scraping, followed by data conversion into a Pandas DataFrame for manipulation. Interactive visualizations are created using Plotly Dash for launch records analysis, and a map is built with Folium to assess the proximity of launch sites. Machine learning techniques, including Support Vector Machines (SVM), Classification Trees, and Logistic Regression, are applied to predict the success of Falcon 9 first-stage landings. The data is split into training and testing sets to optimize hyperparameters and identify the most accurate model.
- **Summary of All Results**:
  The project provides interactive insights into Falcon 9 launch data, including visualizations of landing success and launch site proximity. The machine learning models were evaluated based on their performance in predicting landing success, with the best model identified using test data. The results can help optimize Falcon 9 launch processes and support decision-making for future business opportunities.

Introduction

- **Project Background and Context**:
  The project focuses on predicting the success of Falcon 9's first-stage landings by analyzing historical launch data. SpaceX aims to reduce the cost of rocket launches by reusing the first stage, and understanding the factors that influence successful landings is crucial. By predicting landing success, we can provide insights into the efficiency of the rocket, help SpaceX optimize its operations, and offer valuable information to other companies considering launching with SpaceX.
- **Problems You Want to Find Answers**:
  The main problem is predicting whether the first stage of the Falcon 9 will land successfully. Key questions include:
- What factors (e.g., weather conditions, launch site, rocket specifications) influence landing success?
- How can machine learning models accurately predict landing outcomes based on these factors?
- Which machine learning algorithm (SVM, classification trees, or logistic regression) performs best in predicting the success of a Falcon 9 landing?
- Can interactive data visualizations and maps help better understand and analyze launch records and site proximities?

Section 1

Methodology

Methodology

Data Collection
- Used **SpaceX RESTful API** to retrieve structured JSON launch data (mission name, date, payload, rocket ID, landing success, etc.).
- Applied **web scraping** with `BeautifulSoup` and `requests` to extract additional launch details from Wikipedia and SpaceX pages.
- Combined both sources into a unified dataset for further processing.

Data Wrangling
- Converted JSON and HTML data into **Pandas DataFrames**.
- Removed redundant and irrelevant columns.
- Handled **missing values** (imputation or removal depending on context).
- Converted categorical variables into numerical (e.g., one-hot encoding).
- Standardized column formats (dates, booleans, etc.)

Exploratory Data Analysis (EDA)
- Used `matplotlib`, `seaborn`, and `plotly.express` for initial insights.
- Analyzed distributions of launch success by
- Identified correlations and trends to select relevant features.

Interactive Visual Analytics
- Built an interactive **dashboard** using **Plotly Dash**:
- Filters for launch site and payload range.
- Pie charts and scatter plots for landing success.
- Created a **Folium map** to visualize launch site locations and proximity.

Data Modeling
- Selected key features for prediction: payload mass, orbit, launch site, booster version, etc.
- Split data into **training and test sets** (e.g., 80/20).
- Trained multiple models:
  Logistic Regression
  Support Vector Machines (SVM)
  Classification Trees (Decision TreeClassifier)

Model Evaluation
- Used **cross-validation** and **GridSearchCV** to tune hyperparameters.
- Evaluated models on test set using:
- **Accuracy**
- **F1-score**
- **Confusion Matrix**
- Selected the best-performing model based on test accuracy and generalization.

Data Collection

Data Collection Process

The data collection for this project involved two main sources: **RESTful API access** and **web scraping**.

Key Steps in the Process:

Accessing the SpaceX API

- A RESTful API provided access to structured JSON data.
- Launch data including mission name, launch date, payload, rocket ID, and landing outcome was retrieved.

Web Scraping Additional Data

- Web scraping techniques were used to extract complementary information from SpaceX's official launch pages and Wikipedia.
- Tools like `BeautifulSoup` and `requests` were used to parse HTML content.

Data Integration and Cleaning

- Collected JSON and HTML data were converted into a unified Pandas DataFrame.
- Missing values were handled, redundant columns removed, and data types standardized.

Data Collection – SpaceX API

https://github.com/jeanJeliel/Projet_Falcon.git

Part I
Send a GET request to the SpaceX REST API to retrieve launch data.
Save the response as a `.json` file, then **convert the JSON data into a Pandas DataFrame** for analysis.

```
spacex_url="https://api.spacexdata.com/v4/launches/past"

response = requests.get(spacex_url)
```

```
# Use json_normalize method to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Part II
Apply custom logic to clean the data .
Create predefined lists to store specific data fields.
Use custom functions to extract and append data to these lists.
Finally, **combine the lists into a dictionary** and convert it into a structured dataset.

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
# Call getBoosterVersion
getBoosterVersion(data)

# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(data)
```

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

Part III
Build a Pandas DataFrame using the dictionary that contains the cleaned and organized data.

```
# Create a data from launch_dict
df = pd.DataFrame.from_dict(launch_dict)
```

Part IV
Filter the DataFrame to include only launches of the **Falcon 9** rocket.
Reset the `FlightNumber` column to ensure it is sequential.
Fill missing values in `PayloadMass` with the **mean** of that column.

```
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
```

```
# Calculate the mean value of PayloadMass column and Replace the np.nan values with its mean value
data_falcon9 = data_falcon9.fillna(value={'PayloadMass': data_falcon9['PayloadMass'].mean()})
```

Data Collection – Scraping

https://github.com/jeanJeliel/Projet_Falcon.git

Part I
Send a request to the static URL and store the HTML response in a variable.

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

```
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
data = response.text
```

Part II
Create a BeautifulSoup object from the HTML response, then extract all tables present in the page.

```
soup = BeautifulSoup(data, 'html5lib')
html_tables = soup.find_all('table')
```

Part III
Extract all column header names from the tables retrieved from the HTML page.

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if(name != None and len(name) > 0):
        column_names.append(name)
```

Part IV
Use the extracted column names as dictionary keys, and apply custom functions and logic (see Appendix) to populate the dictionary with parsed launch table data.

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelevant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Data Wrangling

Data Wrangling Process

- **Standardized column names** for consistency and readability.
- **Filtered the dataset** to focus only on relevant Falcon 9 launches.
- **Handled missing values**, replacing `PayloadMass` nulls with the column's **mean value**.
- **Parsed mission outcome values** (e.g., `True Ocean`, `False RTLS`) to identify **successful vs. unsuccessful landings**.
- **Defined training labels** by assigning:
  - 1 for successful landings (`True Ocean`, `True RTLS`, `True ASDS`)
  - `0` for failed landings (`False Ocean`, `False RTLS`, `False ASDS`)
- **Converted categorical variables** (e.g., `Orbit`, `LaunchSite`) into numerical using **one-hot encoding**.
- **Reset index and flight numbers** to maintain a clean, sequential structure.
- **Prepared final cleaned dataset** for EDA and model training.

Data Analysis

Load Space X dataset, from last section.

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577366	28.561857
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577366	28.561857
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008	-80.577366	28.561857
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011	-80.577366	28.561857

Identify and calculate the percentage of the missing values in each at

```
df.isnull().sum()/len(df)*100
```

```
FlightNumber      0.000000
Date              0.000000
BoosterVersion    0.000000
PayloadMass       0.000000
Orbit             0.000000
LaunchSite        0.000000
Outcome           0.000000
Flights           0.000000
GridFins          0.000000
Reused            0.000000
Legs              0.000000
LandingPad       28.888889
Block             0.000000
ReusedCount       0.000000
Serial            0.000000
Longitude         0.000000
Latitude          0.000000
dtype: float64
```

Identify which columns are numerical and categorical:

```
df.dtypes
```

```
FlightNumber        int64
Date               object
BoosterVersion     object
PayloadMass       float64
Orbit              object
LaunchSite         object
Outcome            object
Flights             int64
GridFins             bool
Reused               bool
Legs                 bool
LandingPad         object
Block             float64
ReusedCount         int64
Serial             object
Longitude         float64
Latitude          float64
dtype: object
```

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

```
Orbit
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
HEO       1
ES-L1     1
SO        1
GEO       1
Name: count, dtype: int64
```

https://github.com/jeanJeliel/Projet_Falcon.git

EDA with Data Visualization

- **Categorical Plot (Catplot):**
  Plotted: `FlightNumber` vs. `LaunchSite`
  Purpose: To observe the distribution of flight numbers across different launch sites and identify usage frequency or patterns at each site.

- **Scatter Plot:**
  Plotted: `PayloadMass (kg)` vs. `LaunchSite`, with hue set to landing `class` (success/failure)
  Purpose: To explore the relationship between payload mass, launch location, and landing outcome.

- **Bar Chart:**
  Plotted: Orbit types vs. success rate
  Purpose: To analyze and compare the landing success rates across different orbit types (e.g., LEO, GEO).

- **Scatter Plot:**
  Plotted: `FlightNumber` vs. `Orbit`, with hue set to landing `class`
  Purpose: To study how mission orbits evolved over time and whether certain orbits are more likely to lead to successful landings.

- **Line Chart:**
  Plotted: Year vs. average success rate
  Purpose: To visualize the trend in launch success over the years and evaluate improvement in SpaceX's landing technology.

https://github.com/jeanJeliel/Projet_Falcon.git

EDA with SQL

https://github.com/jeanJeliel/Projet_Falcon.git

Summary of SQL Queries Executed

- **Retrieved all unique launch site names** from the dataset to identify the different locations used for Falcon 9 missions.
- **Displayed 5 records** where the launch site name starts with `'CCA'` to filter missions launched from Cape Canaveral locations.
- **Calculated the total payload mass** carried by boosters launched under **NASA (CRS)** missions.
- **Computed the average payload mass** for missions using **booster version F9 v1.1** to understand its performance.
- **Identified the first date** when a **successful landing occurred on a ground pad**.
- **Listed booster names** that successfully landed on a **drone ship** and carried a **payload between 4000 and 6000 kg**.
- **Counted the total number** of **successful and failed mission outcomes** to assess overall mission performance.
- **Identified booster versions** that **carried the maximum payload mass**, highlighting top-performing configurations.
- **Retrieved failed drone ship landings in 2015**, along with booster versions and launch site names, for specific failure analysis.
- **Ranked the count of different landing outcomes** (e.g., "Success (ground pad)", "Failure (drone ship)") **between 2010-06-04 and 2017-03-20**, in **descending order** to evaluate landing trends and performance over time.

Build an Interactive Map with Folium

Folium Map Building Summary and Purpose

1. **Initialize Map Centered on NASA Johnson Space Center**
   Action: Created a `folium.Map` centered at Houston, Texas.
   Purpose: To serve as the initial view and reference location for launching the map experience.

2. **Add Circles and Markers for Each Launch Site**
   Action: Used `folium.Circle` and `folium.Marker` to highlight and label each launch site from the `launch_sites` DataFrame.
   Purpose: To visualize and distinguish the different launch locations.

3. **Check Geographical Questions via Map Exploration**
   Explored:
   Whether launch sites are near the **Equator**.
   Whether launch sites are close to the **coast**.
   Purpose: To analyze site positioning strategy in relation to trajectory efficiency and safety.

4. **Create a MarkerCluster for Launch Records**
   Action:
   Created a new column `marker_color` in `spacex_df` to assign **green** for success (`class = 1`) and **red** for failure (`class = 0`).
   Added `folium.Marker` objects to a `MarkerCluster`, colored accordingly.
   Purpose: To simplify visualization of multiple overlapping markers and allow easy identification of sites with high or low success rates.

5. **Add MousePosition Plugin**
   Action: Enabled mouse coordinate tracking using `folium.plugins.MousePosition`.
   Purpose: To interactively retrieve coordinates of nearby features like **railways**

Build a Dashboard with Plotly Dash

Plots/Graphs Added

- **Pie Chart (`success-pie-chart`)**:
  Displays the **distribution of successful vs failed launches**.
  - When *"All Sites"* is selected → shows total success count per launch site.
  - When a *specific site* is selected → shows the success vs failure distribution for that site.

- **Scatter Plot (`success-payload-scatter-chart`)**:
  Shows the **relationship between Payload Mass and Launch Outcome**.
  - Payload Mass is on the X-axis.
  - Launch Outcome (Success = 1, Failure = 0) is on the Y-axis.
  - Colored by Orbit type to visualize additional patterns.

User Interactions Added

- **Dropdown (`launch site selector`)**:
  Allows users to select:
  All launch sites
  Individual launch sites (e.g., CCAFS, VAFB)
- **Payload Range Slider**:
  Enables filtering launches by payload mass (kg), dynamically updating the scatter plot.

Why These Plots and Interactions Were Added

- To **visually analyze launch performance** across different sites and payload ranges.
- The **dropdown** offers flexibility to compare overall and site-specific success rates.
- The **slider** helps investigate how **payload weight impacts success probability**.
- The **interactive design** makes the dashboard intuitive and insightful for both technical and non-technical users.

Predictive Analysis (Classification)

1. Data Preparation

2. Data Splitting

Training and Testing Data Split: Split the dataset into **training** (80%) and **test** (20%) sets using **train_test_split** to ensure unbiased evaluation.

3. Model Selection and Hyperparameter Tuning

- **Support Vector Machine (SVM)**:Trained an SVM model with different kernel types (linear, rbf) and tuned the **C** and **gamma** hyperparameters using **GridSearchCV**.
- **Classification Trees**:
  Trained a Decision Tree classifier and fine-tuned hyperparameters like **max_depth**, **min_samples_split**, and **min_samples_leaf** using **GridSearchCV** to avoid overfitting.
- **Logistic Regression**:
  Trained a Logistic Regression model and tuned the **regularization strength** (C) hyperparameter.

4. Model Evaluation

- **Cross-validation**:

Performed **k-fold cross-validation** to evaluate model performance and reduce variance.

- **Accuracy and Metrics**:

Used accuracy, precision, recall, and F1-score to evaluate model performance on both training and test data.
Created confusion matrices to visualize true positives, false positives, true negatives, and false negatives.

5. Model Comparison and Selection

- **Test Data Evaluation**:

Evaluated the models on the **test dataset** to check which model performed best in predicting the success of the first-stage landing.

- **Best Model Selection**:

Compared performance metrics for each model.
Selected the model with the highest **accuracy** and **F1-score** as the best model for prediction.

6. Model Improvement

- **Hyperparameter Optimization**:

Used **GridSearchCV** and **RandomizedSearchCV** for exhaustive and randomized hyperparameter search to improve model performance.

- **Feature Engineering**:

Potential additional features were considered to enhance model accuracy, such as combining features or adding interaction terms.

- **Regularization and Pruning**:

Applied regularization (e.g., L2 for Logistic Regression) and pruning for Decision Trees to avoid overfitting.

7. Final Model

- After tuning, the model with the **best performance on the test data** was selected and finalized for predicting Falcon 9 first-stage landing success.

This process ensures the best-performing classification model is chosen based on careful evaluation and optimization of hyperparameters.

https://github.com/jeanJeliel/Projet_Falcon.git

Results

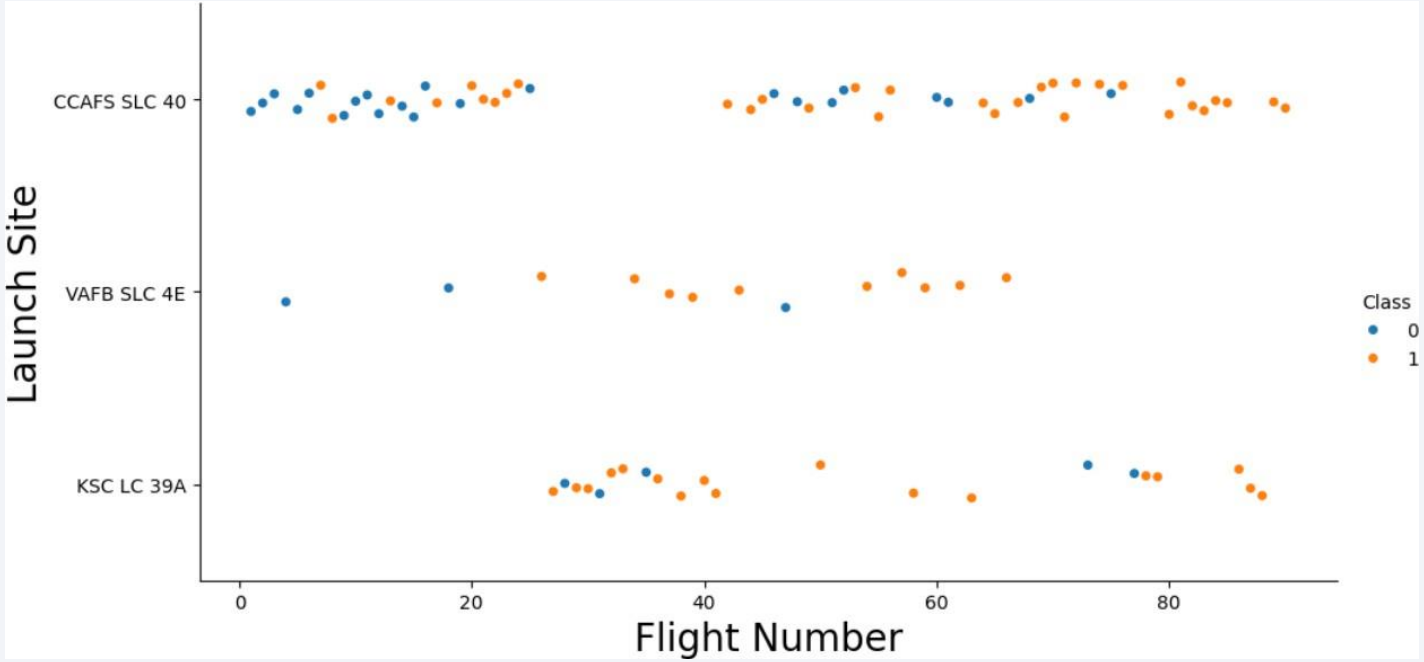
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

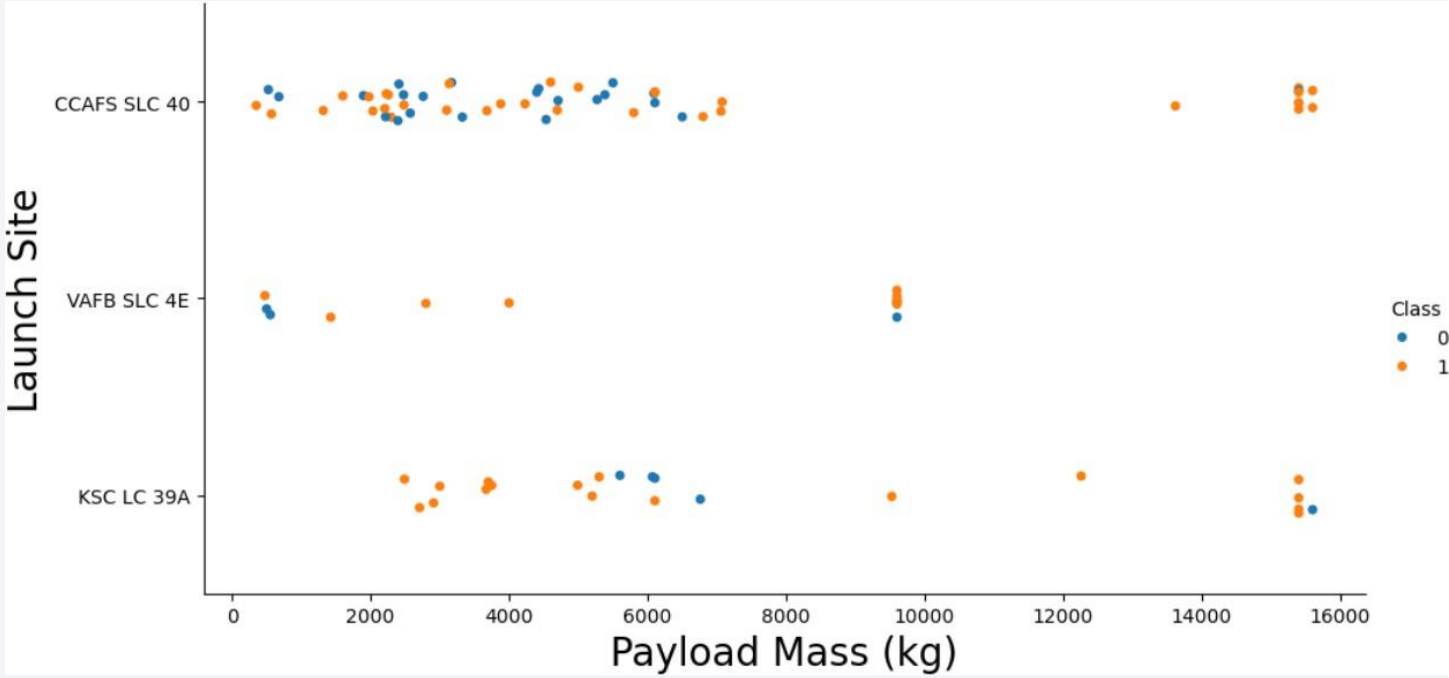
Flight Number vs. Launch Site

As the number of flights increases, the launch success rate also improves. Early flights (FlightNumber < 30), especially from CCAFS SLC 40 and VAFB SLC 4E, were generally unsuccessful. In contrast, KSC LC 39A did not host early flights, resulting in a higher success rate. Overall, after around FlightNumber 30, most launches tend to be successful (Class = 1).



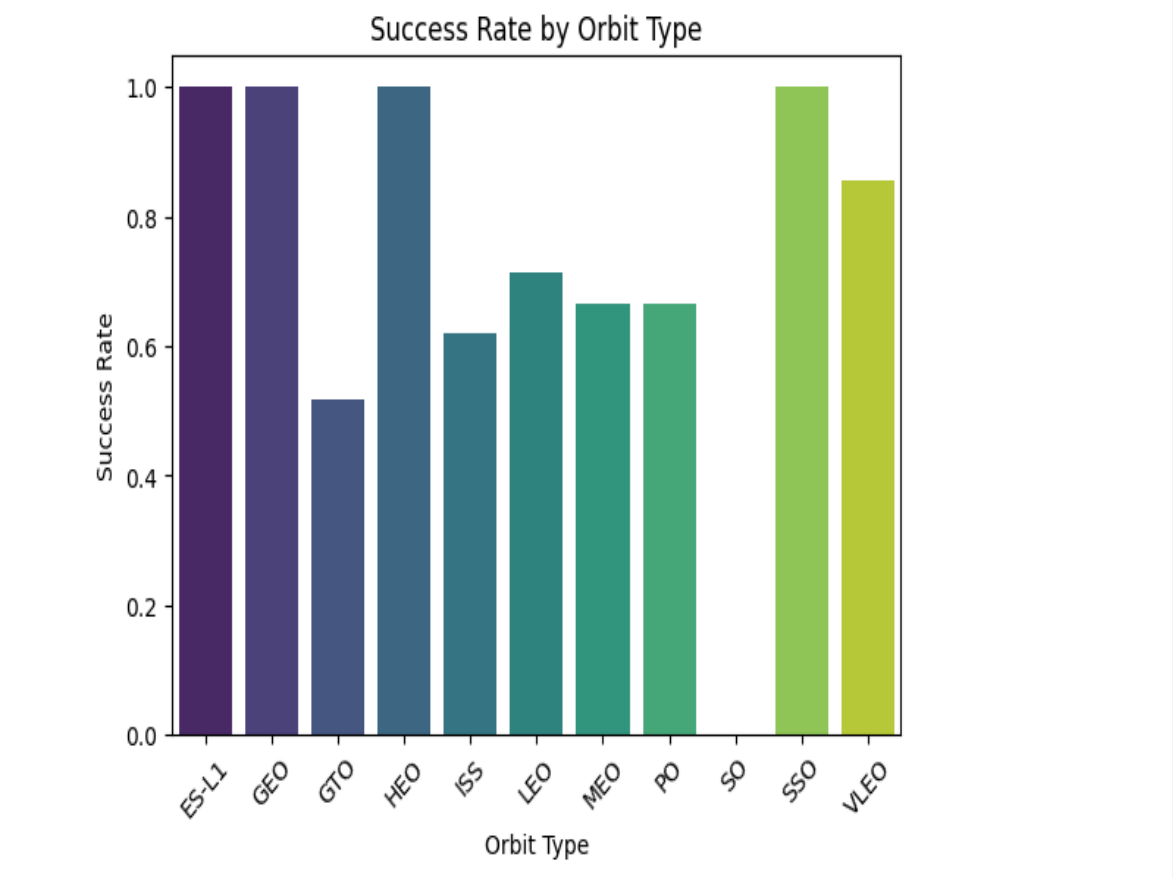
Payload vs. Launch Site

Above a payload mass of approximately 7000 kg, unsuccessful landings are rare, though data for such heavy launches is limited. There is no strong correlation between payload mass and success rate across launch sites. All sites launched a range of payloads, with CCAFS SLC 40 handling mostly lighter payloads, aside from a few outliers.



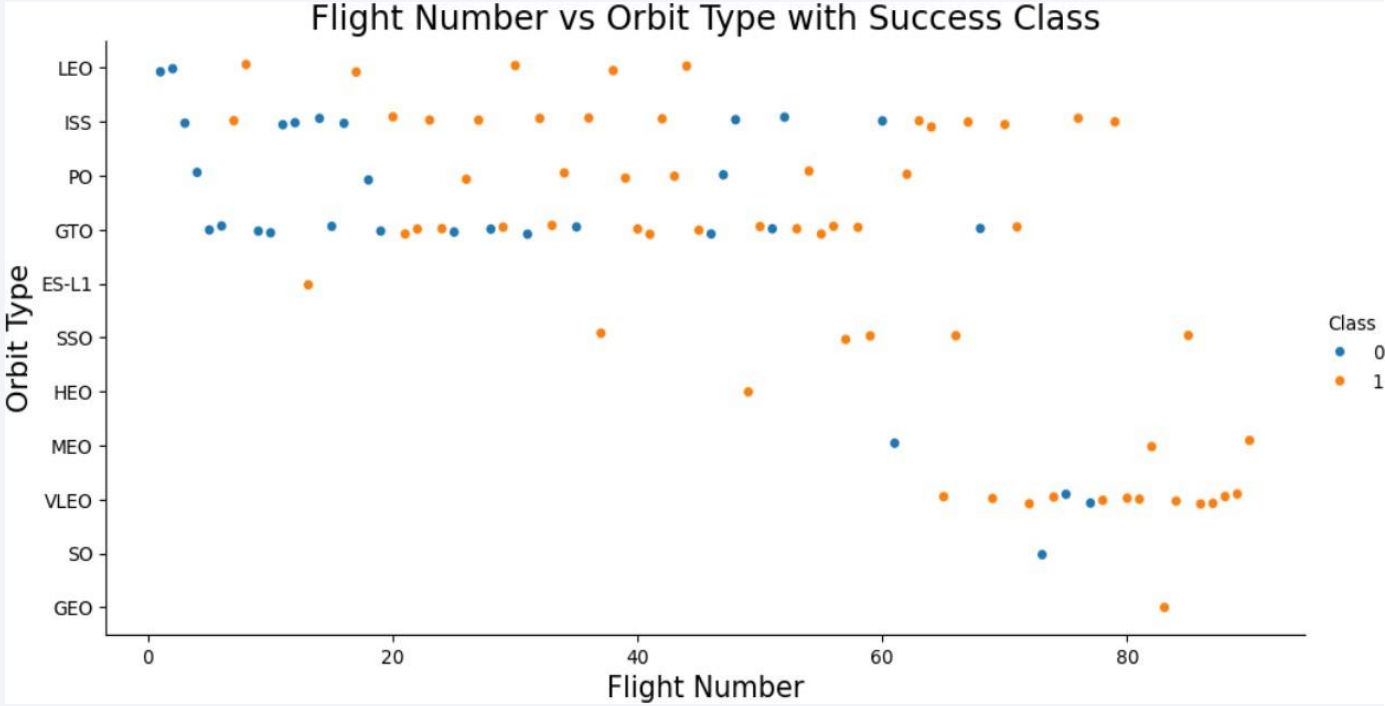
Success Rate vs. Orbit Type

The Success Rate vs. Orbit Type bar chart reveals that ES-L1, GEO, HEO, and SSO orbits have a 100% success rate, while the SO (Heliocentric Orbit) has the lowest success rate at 0%.



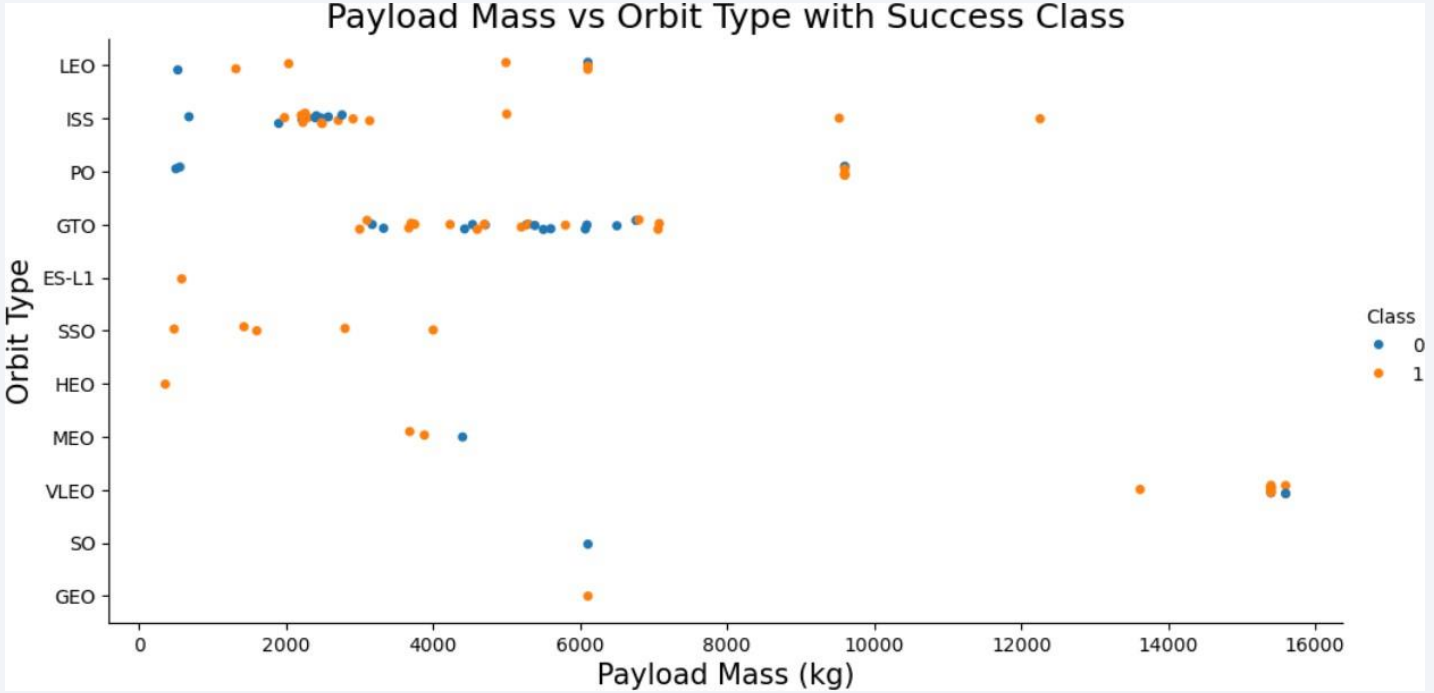
Flight Number vs. Orbit Type

The scatter plot of Orbit Type vs. Flight Number highlights that the perfect success rates for GEO, HEO, and ES-L1 are due to only one flight into each of these orbits, while the 100% success rate for SSO is more meaningful with five successful launches. There is little correlation between flight number and success rate for GTO. Overall, higher flight numbers tend to correspond with increased success rates, especially for LEO, where failures occurred mostly in early launches.



Payload vs. Orbit Type

The scatter plot of Orbit Type vs. Payload Mass reveals that heavier payloads tend to be more successful in PO, ISS, and LEO orbits, though PO has limited data. For GTO, no clear relationship is observed between payload mass and success rate. VLEO launches are linked to heavier payloads, which aligns with expectations.



Launch Success Yearly Trend

Between 2010 and 2013, all landings failed with a 0% success rate. From 2014 onward, success rates improved overall, despite slight drops in 2018 and 2020. Since 2016, the success rate consistently remained above 50%.

All Launch Site Names

The **`SELECT DISTINCT`** statement is used to retrieve all unique launch site names from the dataset, eliminating any duplicate entries.

```
]: %sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;

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

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

Launch Site Names Begin with 'CCA'

The **LIKE** clause with a wildcard is used to filter and return records where the launch site name starts with "CCA", and **LIMIT 5** restricts the output to the first 5 matching entries.

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

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

The **SUM** function combined with the **WHERE** clause calculates the total payload mass by aggregating values only for rows where the booster was launched by NASA.

Task 3

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

```
[18]: %sql SELECT SUM("PAYLOAD_MASS__KG_") AS Total_PayloadMass FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)';
```

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

[18]:

Total_PayloadMass
45596

Task 4

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

```
[19]: %sql SELECT AVG("PAYLOAD_MASS__KG_") AS Avg_PayloadMass FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';
```

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

[19]:

Avg_PayloadMass
2928.4

Average Payload Mass by F9 v1.1

The **AVG** function combined with the **WHERE** clause calculates the average payload mass by averaging the values only for rows where the booster version is F9 v1.1.

First Successful Ground Landing Date

The **MIN** function combined with the **WHERE** clause is used to find the earliest date on which the first successful landing outcome occurred on a ground pad by selecting the minimum date where the landing was successful and occurred on a ground pad.

Task 5

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

Hint:Use min function

```
[22]: %sql SELECT MIN("Date") AS First_Successful_Landing FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)';
 * sqlite:///my_data1.db
Done.
```

[22]:

First_Successful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

By using numerical operators in the WHERE clause, you can filter the dataset to list the names of boosters that successfully landed on a drone ship and had a payload mass greater than 4000 but less than 6000, ensuring the conditions for both landing success and payload mass are met.

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

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

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

Total Number of Successful and Failure Mission Outcomes

By using the **COUNT** function and **GROUP BY** with the `Mission Outcome`, you can calculate the total number of successful and failed mission outcomes, grouping the results by their respective outcomes (success or failure).

List the total number of successful and failure mission outcomes

```
%sql SELECT Mission_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTABLE GROUP BY Mission_Outcome;
```

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

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

Boosters Carried Maximum Payload

Using a **WHERE** clause combined with a subquery and the **MAX** function allows you to list the names of the boosters that have carried the maximum payload mass. The subquery finds the maximum payload mass, and the **WHERE** clause ensures that only boosters with that payload mass are selected.

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

```
[25]: %sql SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
 * sqlite:///my_data1.db
Done.
```

[25]:

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

Using the **SUBSTR** function on the date field creates a "Month" column by extracting the month from the date. Then, by filtering the data with the **WHERE** clause on both the `date` and `Landing Outcome` fields, you can specifically list the failed landing outcomes on the drone ship, along with their booster versions and launch site names for the year 2015. This allows for precise filtering based on both the month and landing status.

```
%sql SELECT SUBSTR(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site
FROM SPACEXTABLE WHERE SUBSTR(Date, 1, 4) = '2015' AND Landing_Outcome = 'Failure (drone ship)';
```

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

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

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

Using **COUNT(*)** on the Landing_Outcome field in combination with the WHERE function on Date filters the data within the specified date range. The **GROUP BY** clause is applied on Landing_Outcome to group the results by each outcome (such as "Failure (drone ship)" or "Success (ground pad)"). Finally, **ORDER BY** is used on the Outcome_Count to rank the landing outcomes in descending order, providing a clear view of the most frequent outcomes within the specified date range between 2010-06-04 and 2017-03-20.

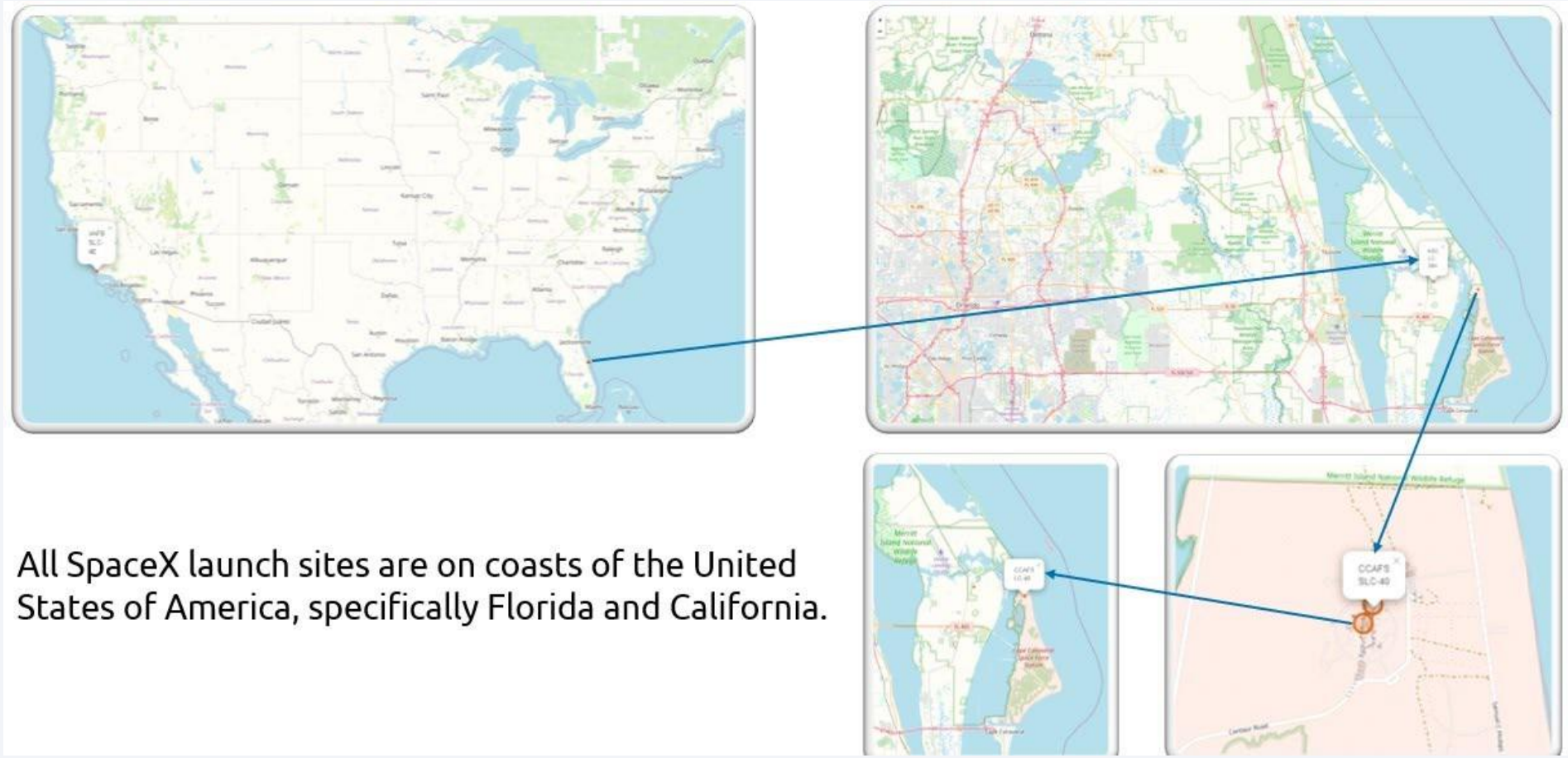
```
%sql SELECT Landing_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
    GROUP BY Landing_Outcome ORDER BY Outcome_Count DESC;
```

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

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

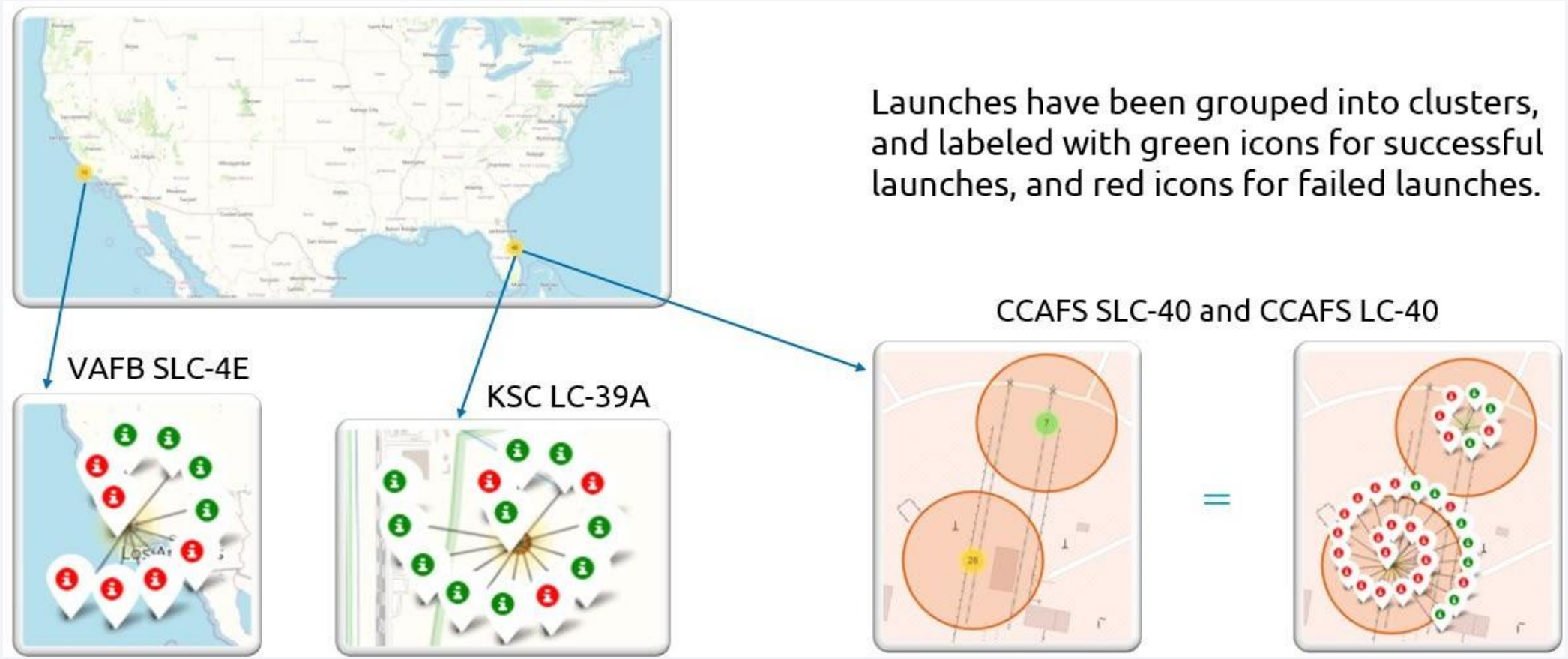

Section 3
Launch Sites
Proximities Analysis

Geographical Analysis of Launch Sites



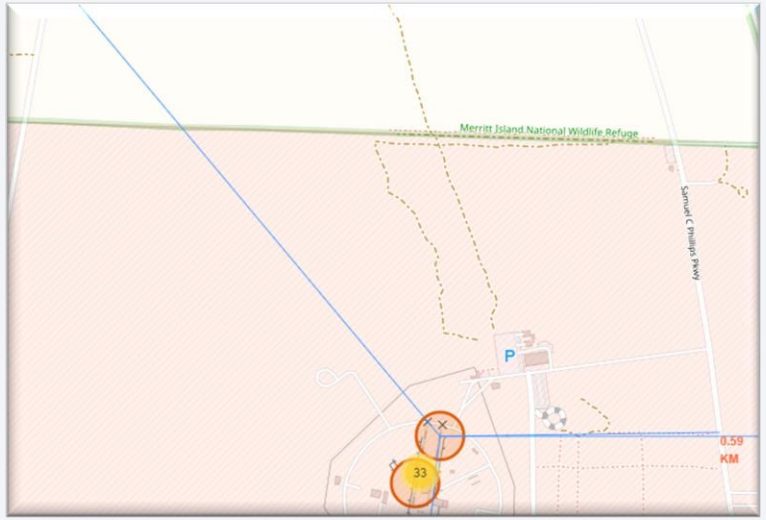
All SpaceX launch sites are on coasts of the United States of America, specifically Florida and California.

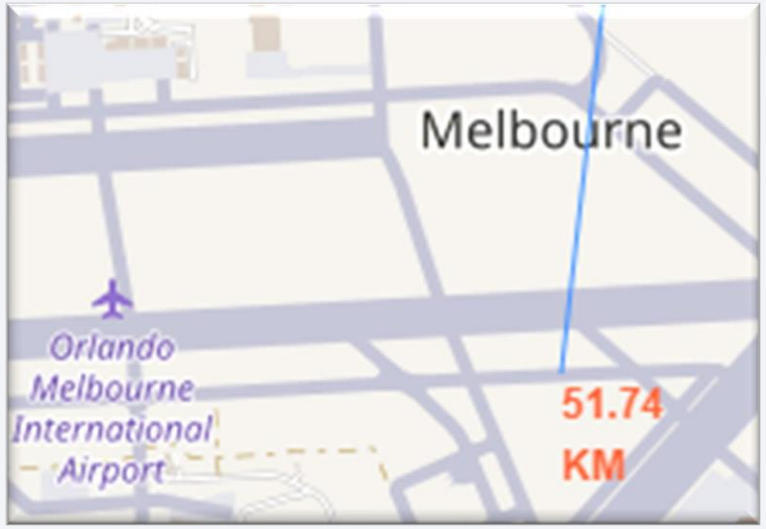
Site-wise Launch Outcome Analysis

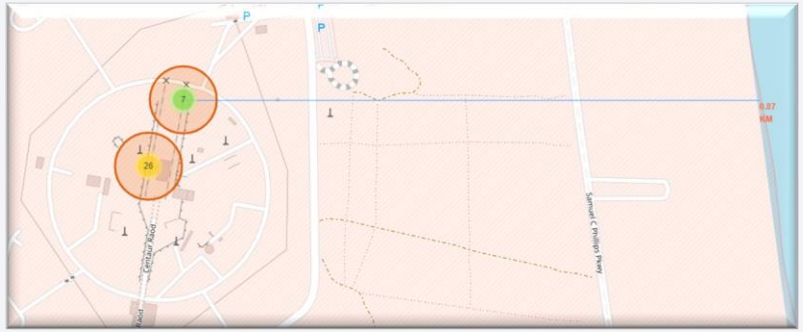


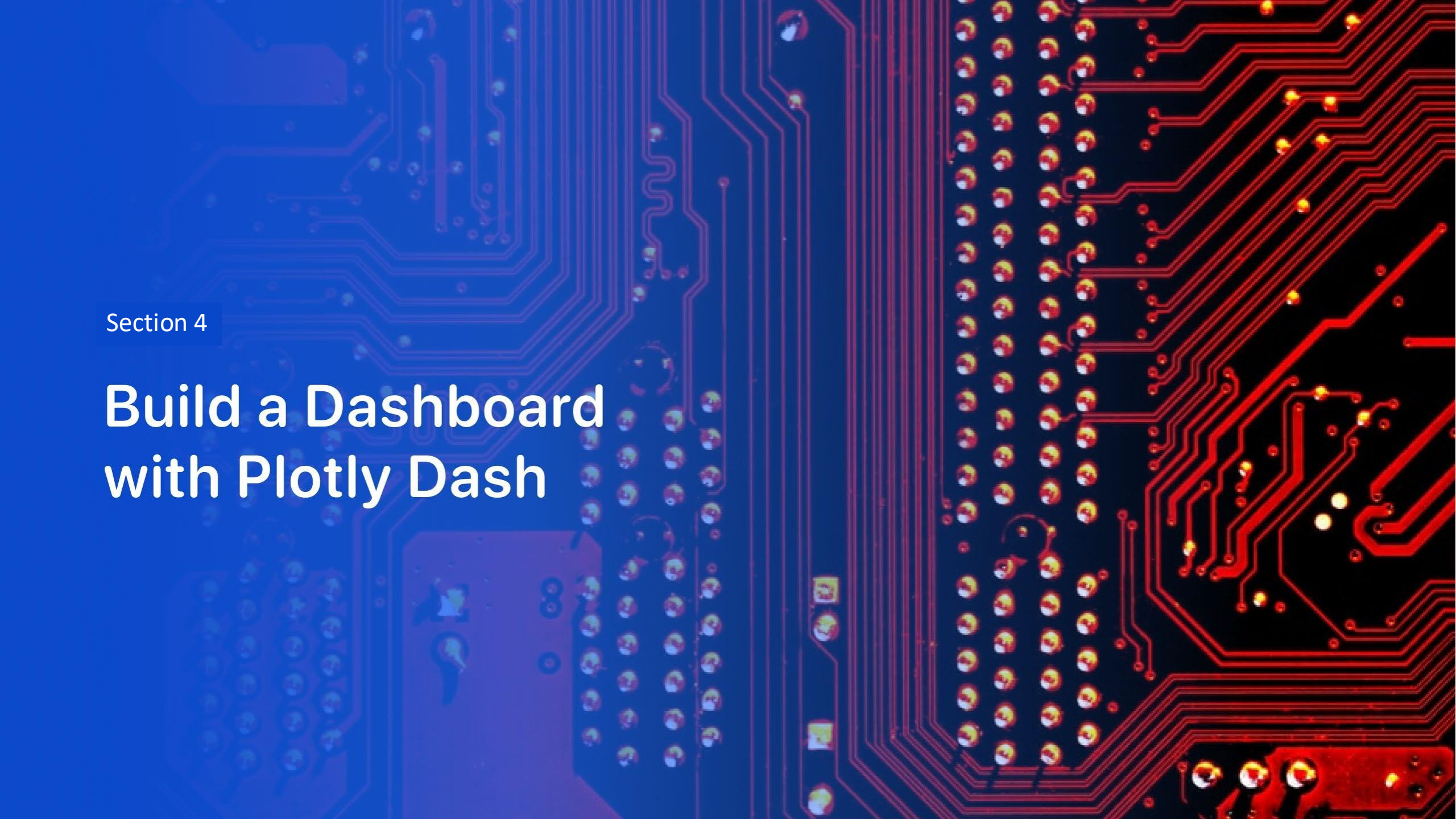
Geospatial Analysis of Launch Site Location

This launch site is strategically positioned due to its proximity to key infrastructure and geographical features. It is situated just 0.59 km from a major highway, 1.29 km from a railway line, and 0.87 km from the coastline. Additionally, it lies within approximately 50 km of the nearest city. These factors combined indicate that the site offers logistical advantages for transportation, accessibility, and support operations, making it an efficient and well-considered location for launch activities



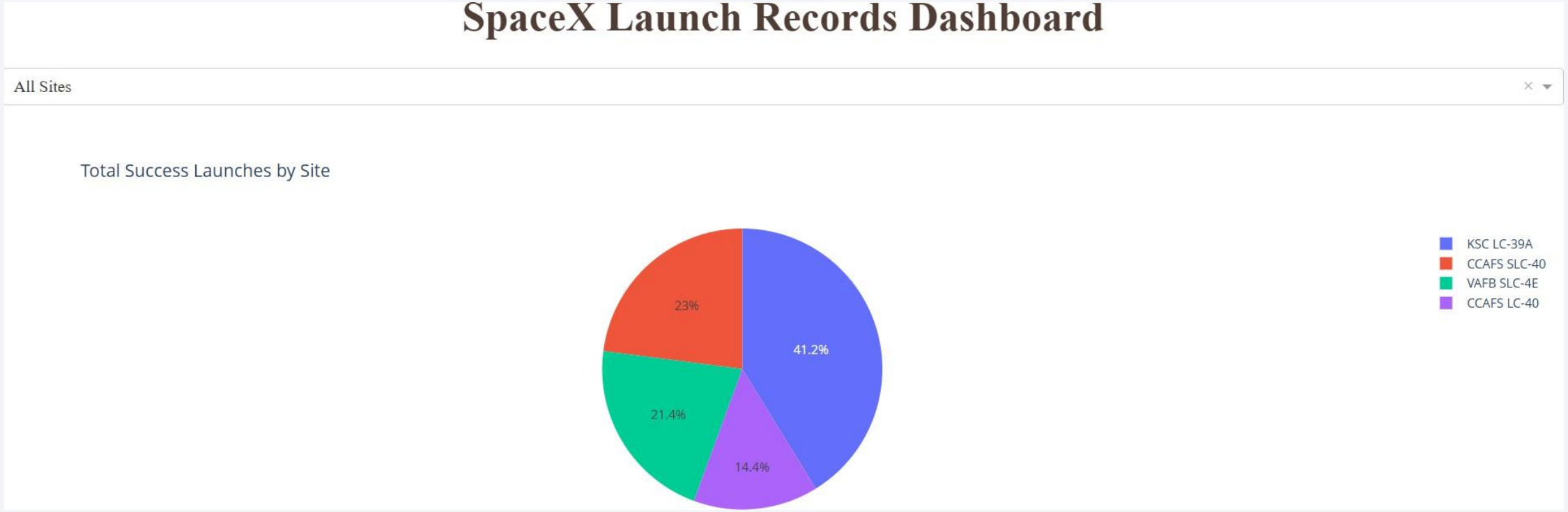




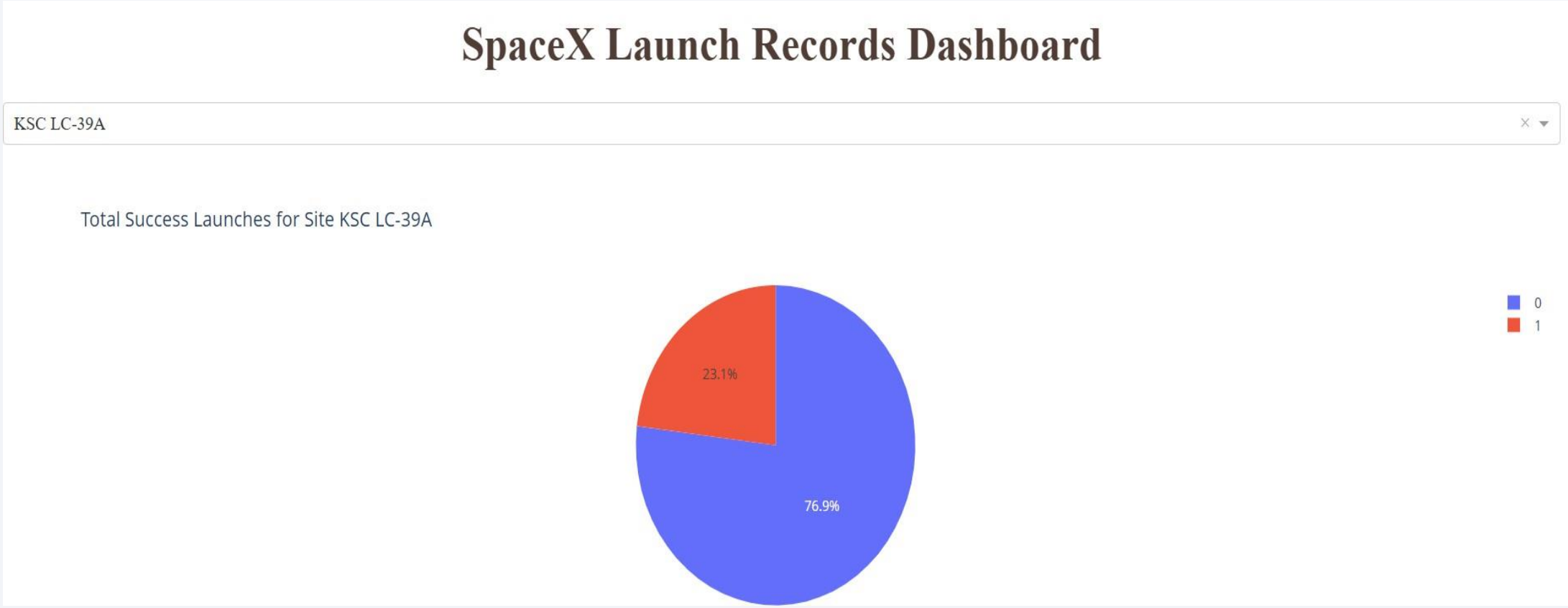

Section 4
Build a Dashboard with Plotly Dash

Comparative Analysis of Launch Success Rates Across All Sites

KSC LC-39A had the most successful launches, with 41.2% of the total successful launches. And VAFB SLC-4E, the less with 21,4%



Sites Ranked by Launch Success Rate



Payload Mass and Launch Outcome: A Comparative Visualization

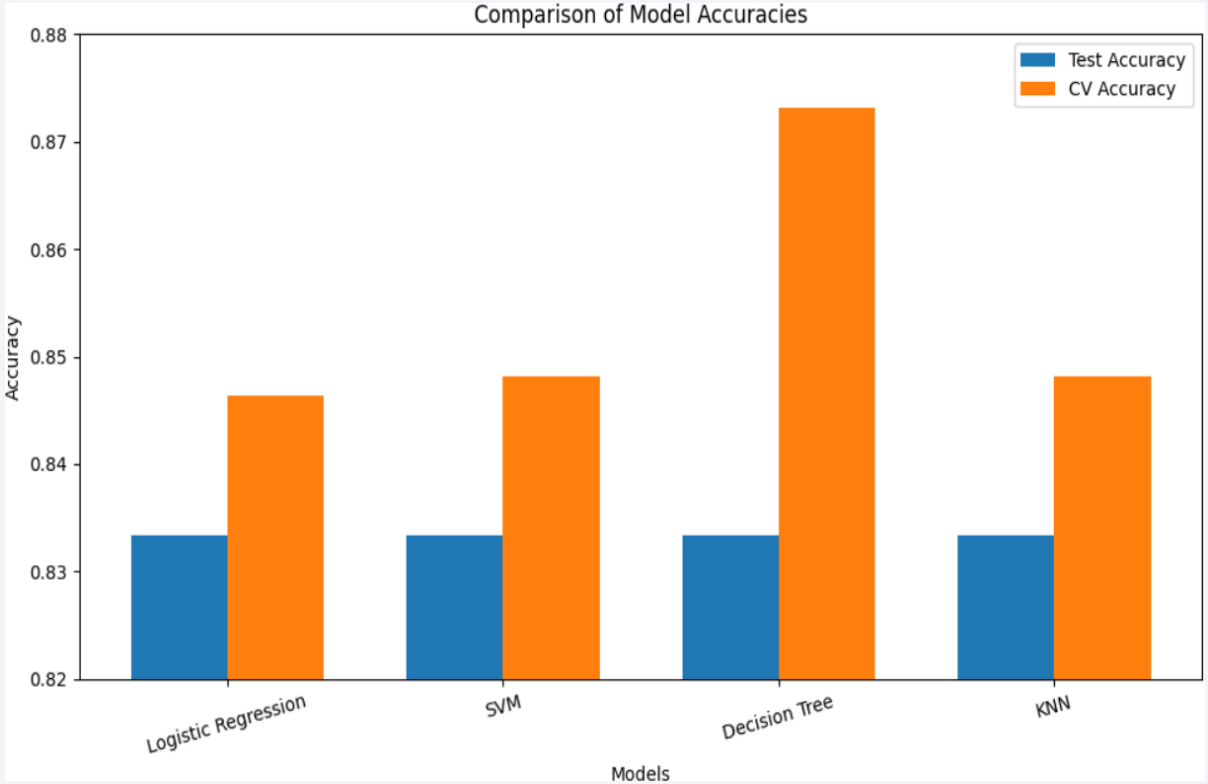
- Plotting the launch outcome vs. payload for all sites uncovers a delineation near 4000 kg, and thus the data has been split into 2 ranges:
  - 0 – 4000 kg (low payloads)
  - 4000 – 10000 kg (massive payloads)
- We can observe that the success for massive payloads is lower than that for low payloads.
- Certain booster types (v1.0 and B5) have not been launched with massive payloads.

Section 5

Predictive Analysis (Classification)

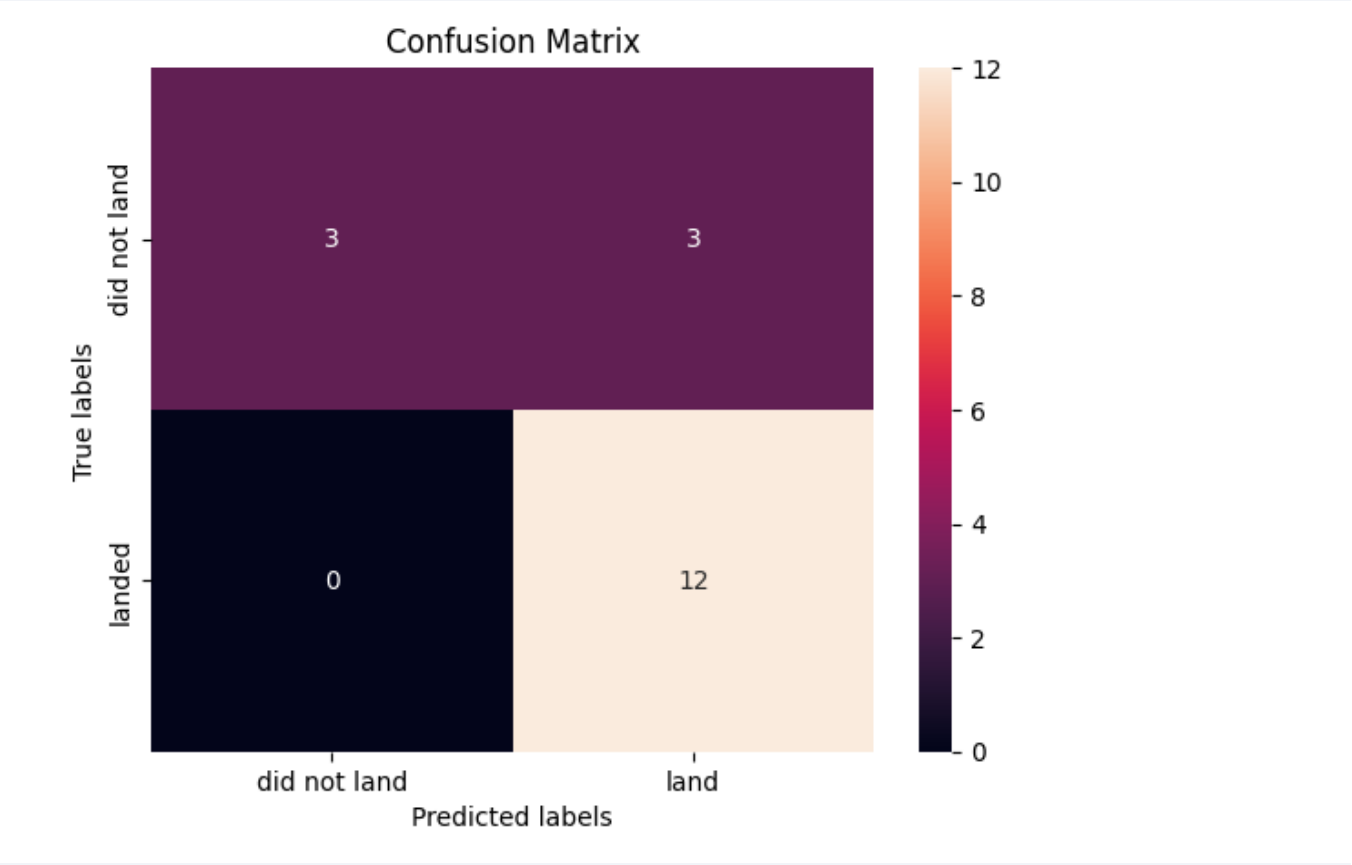
Classification Accuracy

Although all models produced identical accuracy scores on the test set, the Decision Tree Classifier stood out by achieving the highest cross-validation accuracy (0.8732). This indicates that, during model selection, it demonstrated better generalization to unseen data. As a result, it can be considered the most robust and reliable model among the four classifiers evaluated.



Confusion Matrix

The Decision Tree model delivered the best performance overall. The corresponding confusion matrix and detailed analysis confirm that it correctly classifies all landing outcomes. This absence of misclassification is particularly valuable in terms of recall, as it ensures that successful landings are consistently identified without false negatives.



Conclusions

Over time, a clear trend emerges linking operational experience to launch success. As the number of Falcon 9 flights increased, so did the success rate at launch sites. Early missions, particularly those before 2013, were largely unsuccessful — notably, between 2010 and 2013, all landing attempts failed. However, post-2013 marked a turning point, with a noticeable improvement in landing outcomes, and from 2016 onward, SpaceX consistently achieved over a 50% success rate, demonstrating maturation in launch and landing capabilities.

Payload mass was another influential factor. Heavier payloads, typically over 4,000 kg, tended to correlate with lower success rates. That said, orbit type also played a key role in determining outcomes. Orbits such as ES-L1, GEO, and HEO recorded a 100% success rate, although this is based on a single mission each. More impressively, the SSO (Sun-Synchronous Orbit) achieved a flawless success rate across five launches, indicating reliability over multiple missions.

Furthermore, orbit types such as Polar Orbit (PO), the International Space Station (ISS), and Low Earth Orbit (LEO) showed better performance with heavier payloads, suggesting optimized procedures or more robust mission planning for these trajectories. VLEO (Very Low Earth Orbit) missions, typically associated with larger payloads, followed a pattern that aligns with expectations due to lower altitude constraints.
From a site-level perspective, the Kennedy Space Center Launch Complex 39A (KSC LC-39A) stood out. It not only accounted for the highest number of successful launches (41.2% of all successes) but also recorded the best success ratio, with 76.9% of its launches ending in successful landings — highlighting it as SpaceX's most reliable site.

Finally, among the various classification models tested, the **Decision Tree classifier** emerged as the most effective. It achieved the highest cross-validation accuracy and demonstrated strong generalization capabilities. Notably, it correctly classified all landing outcomes on the test set, making it a robust and reliable model, especially critical in operational scenarios where minimizing misclassification is vital.

Thank you!