IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Tassio Luz Canhadas
August 18, 2025

Outline

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

Executive Summary

Lab 1 – SpaceX Data Collection with APIs

In this first lab, my main goal was to collect raw data directly from APIs in order to create the foundation of my analysis. I worked with the SpaceX API and learned how to request data in JSON format using Python's requests library.

I practiced extracting specific keys from nested JSON structures, such as mission name, launch date, payload mass, booster version, and landing success. Then I cleaned and normalized this information into a Pandas DataFrame, making sure that missing values and inconsistent entries were handled properly.

By the end of this lab, I had successfully built a structured dataset of Falcon 9 launches. This dataset contained the essential features I would need for predictive modeling, such as payload mass, orbit, and landing outcomes. It was the first step in transforming messy real-world data into a usable format.

Executive Summary

Lab 2 – Web Scraping SpaceX Data

After learning how to collect data via APIs, I moved on to web scraping to complement the dataset. APIs are powerful, but they sometimes don’t provide everything I need, so scraping allowed me to capture extra details.

Using BeautifulSoup and requests, I extracted data from Wikipedia tables about SpaceX Falcon 9 launches. I identified the relevant HTML structures, looped through rows, and parsed the text into variables like mission date, launch site, payload, and outcome. I also had to clean this data carefully, since HTML tables often contain links, empty cells, and inconsistencies.

At the end of this process, I combined my scraped dataset with the API dataset. This gave me a richer and more complete source of information, and it also allowed me to validate the accuracy of the data. I now had a solid dataset ready for deeper analysis.

Executive Summary

Lab 3 – Exploratory Data Analysis (EDA) and Visualization

Once I had the data prepared, I turned to exploratory data analysis (EDA) to look for meaningful patterns. For this, I relied on Pandas, Matplotlib, and Seaborn.

I created descriptive statistics, frequency counts, and several visualizations such as histograms, scatter plots, bar plots, and boxplots. These helped me answer questions like: Does payload mass affect landing success? Do different orbit types show different success rates? Do newer booster versions perform better?

Through this analysis, I discovered that payload mass and orbit type are closely related to landing outcomes. Launches into Low Earth Orbit (LEO) were more successful, while launches into Geostationary Transfer Orbit (GTO) had more failures. I also noticed that newer booster versions consistently outperformed older ones, which makes sense given the engineering improvements SpaceX has made. Some orbit categories even showed 100% success rates in my dataset.

Executive Summary

Lab 4 – Launch Site Location Analysis

In the fourth lab, I expanded the analysis into a geospatial perspective. I wanted to know whether the launch site itself could influence the chances of success.

I used Folium to create interactive maps showing the main SpaceX launch sites in Florida and California. I plotted markers for the different launches, including whether the outcome was successful or not. I also calculated distances from launch sites to coastlines and analyzed whether location patterns were significant.

The maps revealed that launches are concentrated at Cape Canaveral, Kennedy Space Center, and Vandenberg Air Force Base. I noticed that all of these sites are close to the coast, which is a deliberate safety measure. I also observed that some sites had higher success rates, which likely reflects better infrastructure and more modern boosters being used at those locations.

This lab showed me that location, while not the only factor, can play a role in outcomes, especially when considered alongside booster technology and payload mass.

Introduction

This project explores the background and context of SpaceX Falcon 9 launches, focusing on the factors that influence first-stage landing success. By combining data from APIs, web scraping, and exploratory analysis, I aim to understand how payload, orbit type, booster version, and launch site affect outcomes. The main problems I want to address are: Which features are most predictive of success, and how can this knowledge improve future launch predictions?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The data was collected from two main sources: the official SpaceX API and publicly available records on Wikipedia. Using Python's requests library, I retrieved launch information in JSON format from the API, which included details such as payload mass, orbit type, booster version, and landing outcomes. In addition, I applied web scraping with BeautifulSoup to extract complementary information from HTML tables on Wikipedia. After collection, the datasets were cleaned and processed. This involved normalizing nested JSON fields, handling missing values, removing duplicates, and converting all data into structured Pandas DataFrames. Finally, I merged both sources to create a comprehensive dataset suitable for exploratory data analysis and visualization.

Methodology

Executive Summary

- Perform data wrangling
  - The collected data was processed through several cleaning and transformation steps to ensure consistency and usability. I handled missing values by either imputing or removing incomplete records, standardized column names, and converted categorical variables into uniform formats. Nested JSON fields were normalized, numerical values were scaled where necessary, and irrelevant or duplicate entries were removed. The final dataset was stored in Pandas DataFrames, making it ready for exploratory data analysis and visualization.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Methodology

Executive Summary

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Methodology

Executive Summary

Perform exploratory data analysis (EDA) using visualization and SQL
Perform interactive visual analytics using Folium and Plotly Dash
Perform predictive analysis using classification models

I performed exploratory data analysis (EDA) using both SQL and visualization techniques to better understand the dataset. Through SQL queries, I summarized categorical and numerical features such as orbit type, booster version, and payload mass. With visualization libraries like Matplotlib and Seaborn, I created scatter plots, boxplots, and bar charts that highlighted how these features influenced landing outcomes. Building on this, I incorporated interactive visual analytics with Folium and Plotly Dash. Folium allowed me to map launch sites and display landing results geographically, while Plotly Dash enabled the creation of dynamic dashboards where filters and comparisons could be explored in real time. Finally, I advanced to predictive analysis by developing classification models. I built models such as Logistic Regression, Decision Trees, and K-Nearest Neighbors, tuned them through cross-validation, and evaluated performance with metrics including accuracy, precision, recall, and F1-score. This complete process demonstrated how structured analysis, interactive visualization, and machine learning can work together to predict Falcon 9 first-stage landing success.

Data Collection

The datasets were collected through two complementary methods: API extraction and web scraping. The process ensured that I obtained structured, reliable, and enriched data for analysis.

Data Collection Process:

API Extraction: SpaceX API → JSON data → Pandas DataFrame

Web Scraping: Wikipedia launch tables → cleaned → Pandas DataFrame

Data Cleaning: remove duplicates, handle missing values, standardize columns

Integration: merge API + web-scraped data → final structured dataset

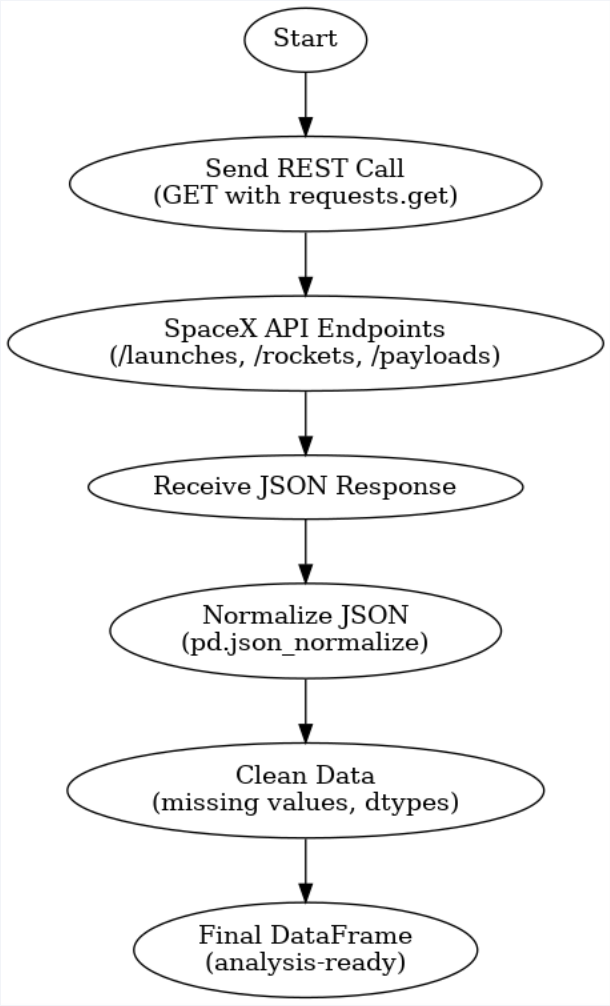
Data Collection – SpaceX API

Key Phrases (SpaceX REST Calls)

- **Base URL:** https://api.spacexdata.com/v4/
  No auth: public endpoints, JSON responses
  Endpoints (GET):

  /launches → core launch records (ids to rockets, payloads, launchpads, ships)
  /rockets → specs (name, stages, reusable, mass, thrust)
  /payloads → mass, orbit, customers
  /launchpads → site name, region, coordinates
  /ships → ASDS/ships metadata (optional)
- **Process:** requests → JSON → pd.json_normalize → join by IDs → clean/types/dates → analysis-ready DataFrame
- **Joins:**
  launch.rocket → rockets._id
  launch.launchpad → launchpads._id
  launch.payloads[] → payloads._id
- **Cleaning:** rename columns, cast dtypes, parse dates, handle missing, deduplicate

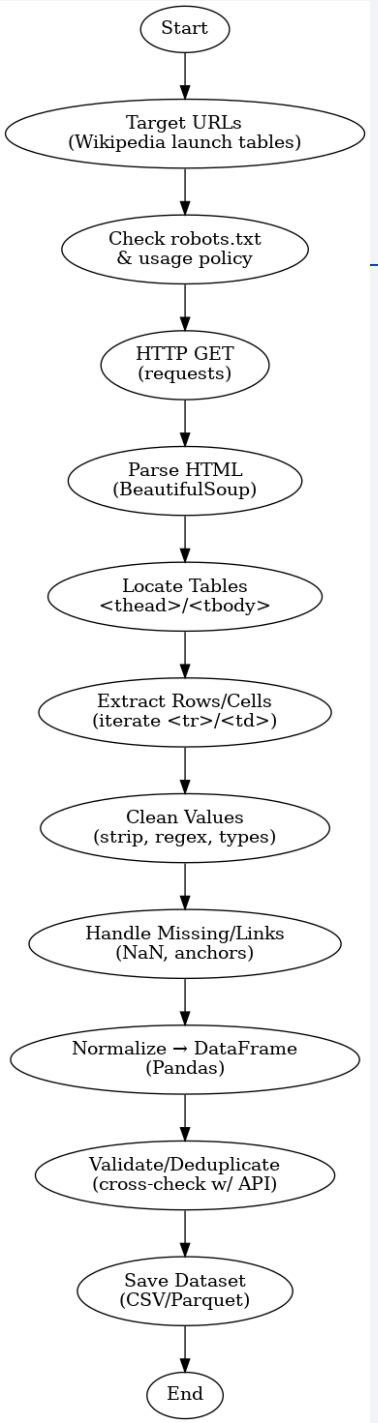
Link:
https://github.com/tassioluz/SpaceXLauches/tree/main



Data Collection - Scraping

- **Key Phrases — Web Scraping**
- **Target URLs:** Wikipedia Falcon 9 launch tables
- **Compliance:** check robots.txt, usage policy
- **Request:** requests.get() (HTTP GET)
- **Parse:** BeautifulSoup(html, "lxml")
- **Locate:** <table>, <thead>, <tbody>
- **Extract:** iterate <tr> / <td> → rows
- **Clean:** strip text, regex, type casting
- **Handle:** missing values, links/anchors
- **Normalize:** to Pandas DataFrame
- **Validate:** deduplicate, cross-check with API
- **Persist:** save CSV / Parquet

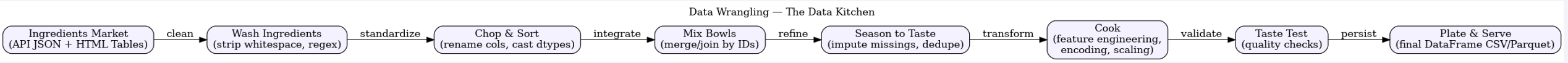
Link:
https://github.com/tassioluz/SpaceXLauches/tree/main

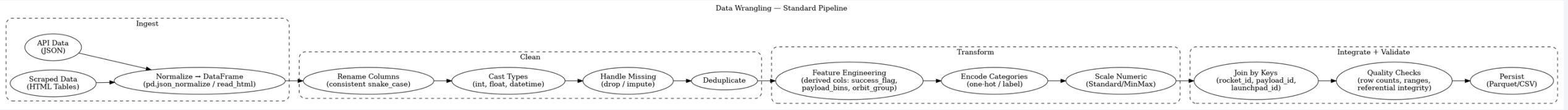


Data Wrangling

Key Phrases — Data Wrangling
Normalize to DataFrame: pd.json_normalize, pd.read_html
Column standardization: rename to snake_case
Type casting: int, float, datetime (pd.to_datetime)
Missing values: drop / impute (mean/median/constant)
Deduplication: drop_duplicates
Feature engineering: success flag, payload bins, orbit groups
Categorical encoding: one-hot / label encoding
Scaling: Standard/MinMax (when modeling)
Integration: joins by rocket_id, payload_id, launchpad_id
Quality checks: row counts, ranges, referential integrity
Persist: CSV/Parquet

Link Github:
https://github.com/tassioluz/SpaceXLauches/tree/main





EDA with Data Visualization

Bar Charts
Usage: Compared counts of successful vs. failed landings by orbit type and booster version.
Why: Bar charts are effective for categorical comparisons, letting me see which categories had higher success rates.
Scatter Plots
Usage: Payload mass vs. landing outcome; payload mass vs. orbit type.
Why: Scatter plots show correlations and trends between numerical and categorical variables. This helped me identify how heavier payloads influenced landing outcomes.
Boxplots
Usage: Payload mass distributions across orbit types and booster categories.
Why: Boxplots highlight variability, median values, and outliers. I used them to see which orbits had consistent payload ranges and how they related to success.
Histograms
Usage: Distribution of payload masses across launches.
Why: Histograms allowed me to observe frequency patterns, like whether launches clustered at certain payload weights.
Stacked/Grouped Bar Charts
Usage: Success rates segmented by booster version or launch site.
Why: Helped visualize how newer booster technology performed compared to older ones.
Folium Maps (Interactive)
Usage: Mapped launch sites and landing outcomes.
Why: Geospatial visualization clarified the geographical patterns — e.g., why launches occur near coastlines and how location ties to outcomes.
Plotly Dash (Interactive Dashboards)
Usage: Created dynamic filtering and drill-down dashboards.
Why: Allowed interactive exploration of data, making it easy to test different feature combinations and view results in real time.

Link GitHub: https://github.com/tassioluz/SpaceXLauches/tree/main

EDA with SQL

Created analysis table
Built SPACEXTABLE from the raw SPACEXTBL, filtering out rows with NULL dates.
Task 1 – Unique launch sites
Selected distinct values from "Launch_Site" to list all different launch locations.
Task 2 – Launches starting with “CCA”
Retrieved 5 records where launch sites begin with "CCA".
Task 3 – NASA (CRS) payload mass
Computed the total payload mass carried by boosters for customer 'NASA (CRS)'.
Task 4 – Average payload mass for F9 v1.1
Calculated the average payload mass where "Booster_Version" = 'F9 v1.1'.
Task 5 – First successful ground pad landing
Used MIN(Date) to find the earliest date of a "Success (ground pad)" landing.
Task 6 – Drone ship successes with medium payloads
Listed distinct booster versions with "Success (drone ship)" landings and payload mass between 4000–6000 kg.
Task 7 – Mission outcomes summary
Counted records grouped by "Mission_Outcome" to show totals of success vs. failure.
Task 8 – Maximum payload booster versions
Used a subquery to find boosters carrying the maximum payload mass.
Task 9 – Failures in 2015 (drone ship)
Queried records with "Landing_Outcome" like 'Failure (drone ship)%' in year 2015.
Converted month numbers with strftime('%m') into month names via a CASE expression.
Task 10 – Ranked landing outcomes (2010–2017)
Counted outcomes between '2010-06-04' and '2017-03-20', ordered by frequency in descending order.

Build an Interactive Map with Folium

Folium Map Objects and Their Purpose

Markers

What: Added folium.Marker() objects at each launch site location.

Why: To visually identify where launches occurred. Each marker served as a simple, clickable point with additional info in the popup.

Circle / CircleMarker

What: Plotted folium.Circle() or CircleMarker() around launch sites.

Why: Circles made the sites more visible and emphasized their geographic distribution. Radius/size sometimes represented additional attributes (like number of launches).

Lines / Polylines

What: Used folium.PolyLine() to connect launch sites with nearby reference points (such as the coast or landing zones).

Why: Helped illustrate spatial relationships, for example the distance from a launch site to the ocean or to a recovery location.

Popups / Tooltips

What: Attached text popups to markers and circles with information like site name, outcome, or success count.

Why: Allowed interactive exploration, so hovering or clicking on a map object revealed extra context about that launch site.

Clustered Markers (optional in lab extensions)

What: MarkerCluster to group multiple launches at the same or nearby sites.

Why: To avoid clutter when many markers overlapped, while still allowing detailed inspection on zoom.

Link GitHub: https://github.com/tassioluz/SpaceXLauches/tree/main

Build a Dashboard with Plotly Dash

Dashboard Plots, Graphs, and Interactions

Bar Charts

What: Bar charts comparing launch outcomes (success/failure) across sites and booster versions.

Why: They provided a clear way to compare categorical variables, letting me quickly see which sites or boosters performed better.

Pie Charts

What: Pie charts showing proportions of successful vs. failed landings.

Why: Useful for displaying overall success distribution at a glance, highlighting the balance between outcomes.

Scatter Plots

What: Scatter plots of payload mass vs. launch success.

Why: Allowed me to visualize the relationship between numerical (payload) and categorical (success/failure) data, showing whether heavier payloads influenced landing outcomes.

Interactive Dropdowns

What: Dropdown filters to select launch sites or booster versions.

Why: Enabled dynamic exploration of the dataset, so I could focus on a specific site or booster and instantly update the plots.

Range Slider

What: Slider to adjust payload mass range for filtering scatter plots.

Why: Helped in interactively studying how different payload weight ranges affect landing success, without writing new queries.

Link GitHub: https://github.com/tassioluz/SpaceXLauches/tree/main

Predictive Analysis (Classification)

Model Development Process (Key Phrases)

Data Preparation

Cleaned dataset, selected relevant features (payload mass, orbit, booster version, launch site).

Split Data

Divided into training and test sets to avoid overfitting.

Build Models

Implemented Logistic Regression, Decision Tree, K-Nearest Neighbors, and SVM.

Evaluate Models

Used accuracy, precision, recall, and F1-score to measure performance.

Hyperparameter Tuning

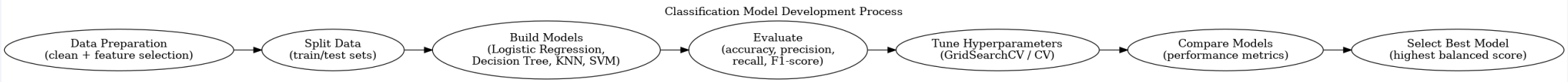
Applied GridSearchCV and cross-validation to optimize parameters (e.g., k for KNN, max_depth for Decision Trees).

Compare Models

Compared performance metrics across all models.

Select Best Model

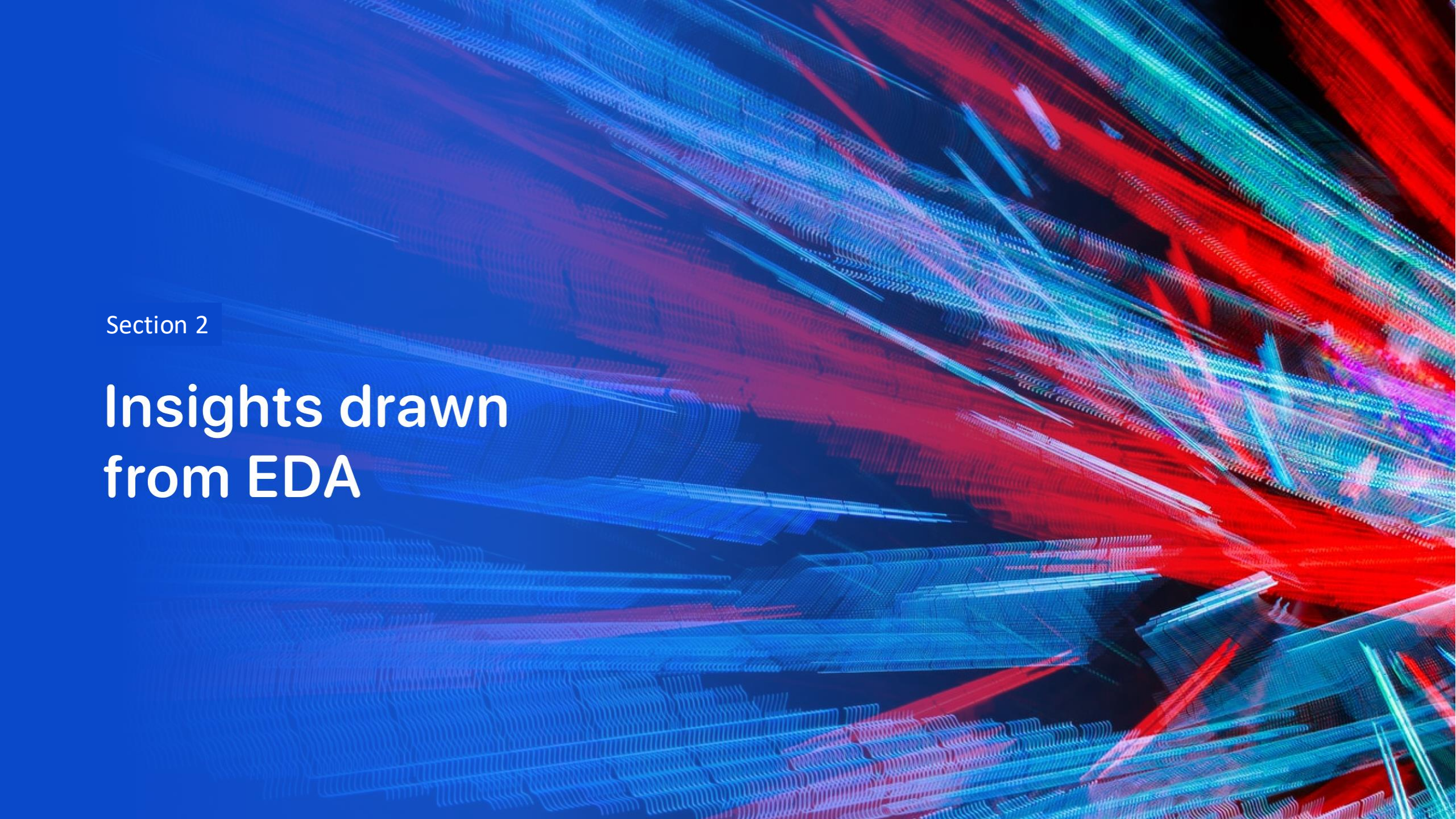
Chose the model with the most balanced and consistent performance.



Link GitHub: https://github.com/tassioluz/SpaceXLauches/tree/main

Results

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


Section 2
Insights drawn from EDA

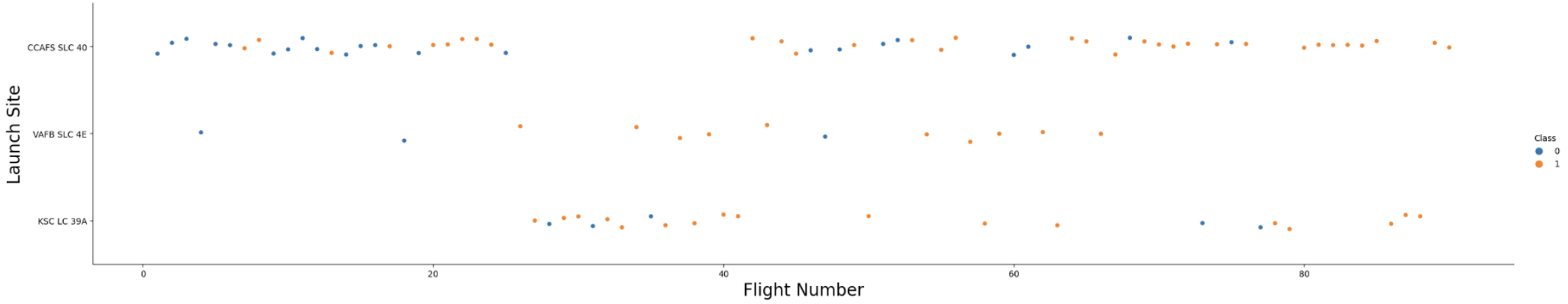
Flight Number vs. Launch Site

TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

In [5]:

```
sns.catplot(data=df, x="FlightNumber", y="LaunchSite", hue="Class", aspect=5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

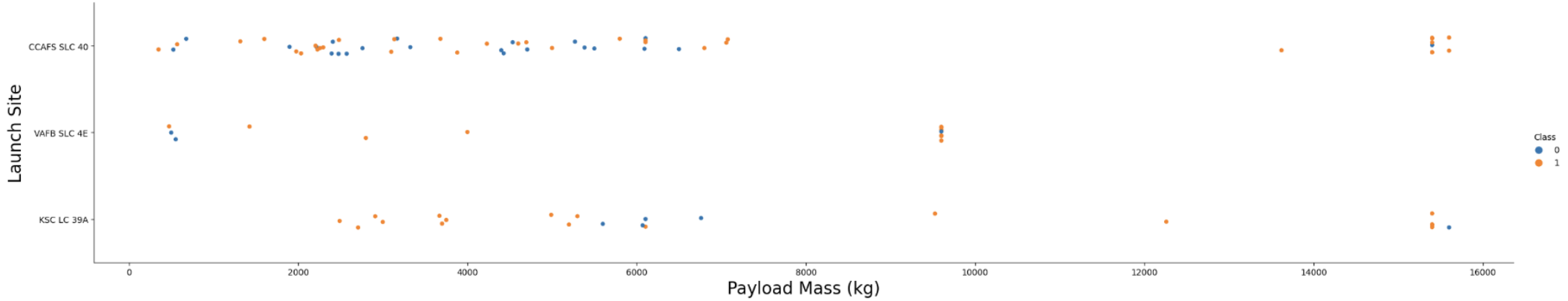


Payload vs. Launch Site

TASK 2: Visualize the relationship between Payload Mass and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

In [6]:
```
sns.catplot(data=df, x="PayloadMass", y="LaunchSite", hue="Class", aspect=5)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```



Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

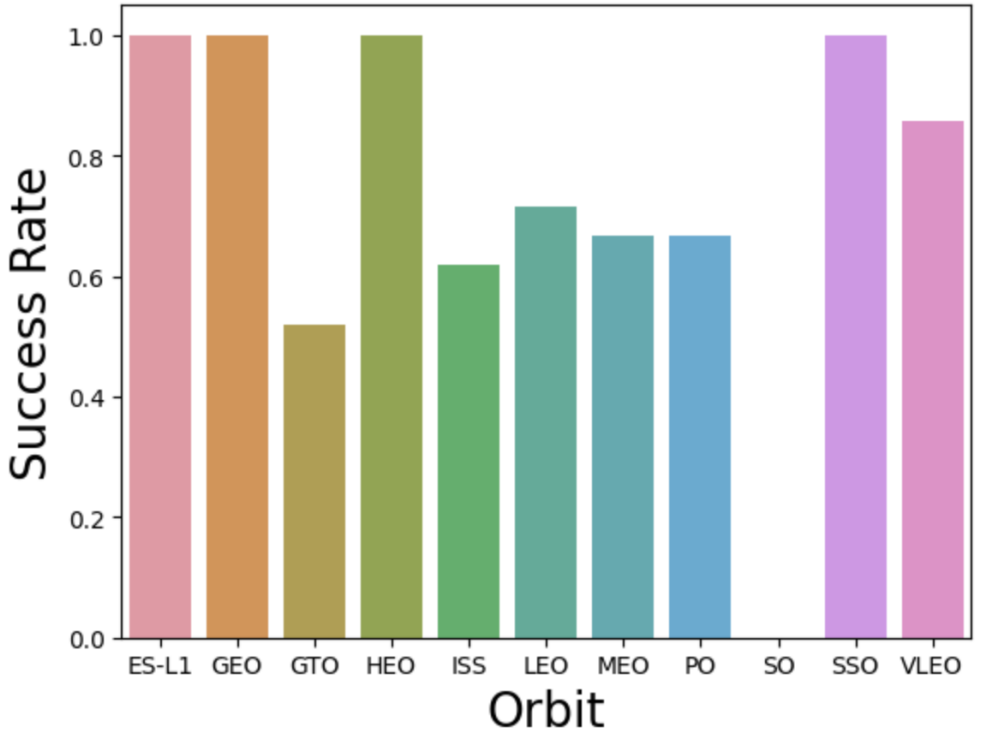
TASK 3: Visualize the relationship between success rate of each orbit type

Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the sucess rate of each orbit

In [7]:

```
df_group = df.groupby('Orbit')["Class"].mean().reset_index()
sns.barplot(data=df_group, x="Orbit", y="Class")
plt.xlabel("Orbit", fontsize=20)
plt.ylabel("Success Rate", fontsize=20)
plt.show()
```

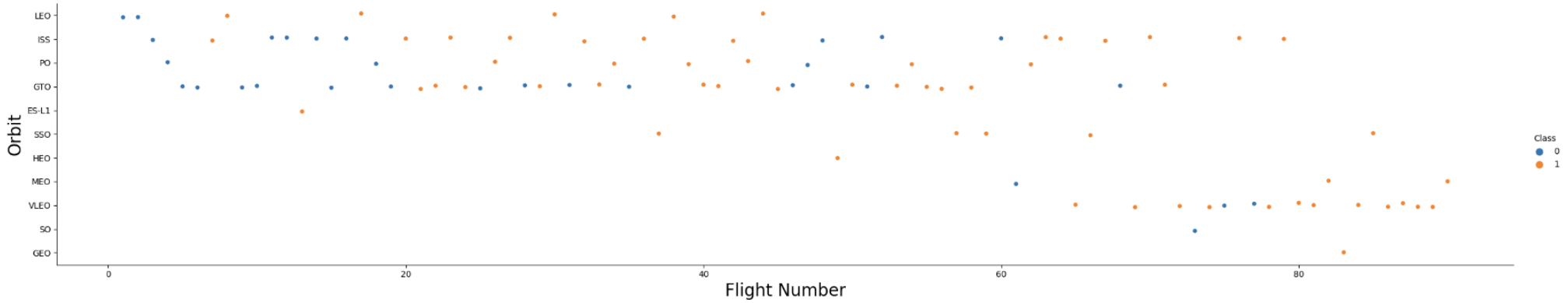


Flight Number vs. Orbit Type

TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

In [8]:
```
sns.catplot(data=df, x="FlightNumber", y="Orbit", hue="Class", aspect=5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Orbit", fontsize=20)
plt.show()
```



You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

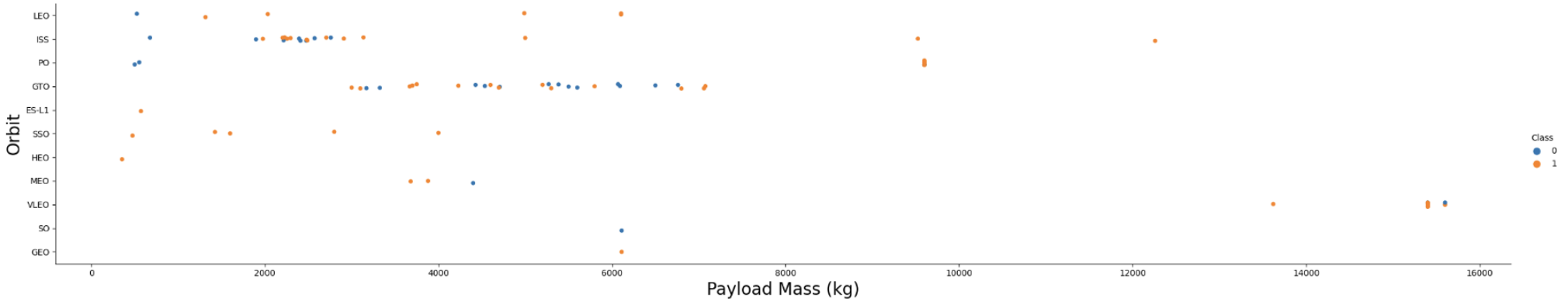
Payload vs. Orbit Type

TASK 5: Visualize the relationship between Payload Mass and Orbit type

Similarly, we can plot the Payload Mass vs. Orbit scatter point charts to reveal the relationship between Payload Mass and Orbit type

In [9]:

```
sns.catplot(data=df, x="PayloadMass", y="Orbit", hue="Class", aspect=5)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Orbit", fontsize=20)
plt.show()
```



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend

TASK 6: Visualize the launch success yearly trend

You can plot a line chart with x axis to be `Year` and y axis to be average success rate, to get the average launch success trend.

The function will help you get the year from the date:

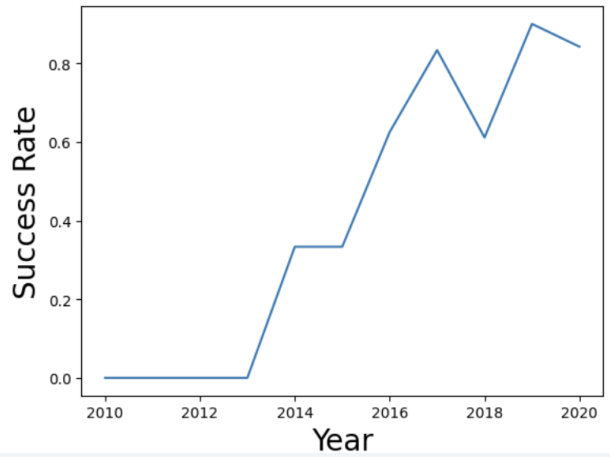
In [10]:
```
# Extract the year from the Date column and replace it with the year

# Convert the Date column to datetime and extract the year
df['Date'] = pd.to_datetime(df['Date']).dt.year
df.head()
```

Out[10]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad
0	1	2010	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN
1	2	2012	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN
2	3	2013	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN
3	4	2013	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN
4	5	2013	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN

In [11]:
```
df_year = df.groupby('Date')["Class"].mean().reset_index()
plt.figure()
plt.plot(df_year['Date'], df_year['Class'])
plt.xlabel('Year', fontsize=20)
plt.ylabel('Success Rate', fontsize=20)
plt.show()
```



All Launch Site Names

SQL Query: %sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;

CCAFS LC-40 - **Cape Canaveral**

KSC LC-39A - **Kennedy Space Center**

VAFB SLC-4E - **Vandenberg Air Force Base**

This query retrieves the distinct values from the Launch_Site column. The result shows that Falcon 9 launches took place from three different sites: **Cape Canaveral (CCAFS LC-40)**, **Kennedy Space Center (KSC LC-39A)**, and **Vandenberg Air Force Base (VAFB SLC-4E)**. These reflect the main operational launch pads used by SpaceX during the dataset period.

Launch Site Names Begin with 'CCA'

Task 2

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

In [11]:
```
%sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;
```

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

Out[11]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (p
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 (p
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

This query uses the LIKE 'CCA%' pattern to filter launch sites that begin with "CCA".
The result confirms that several missions in the dataset were launched from **Cape Canaveral (CCAFS LC-40)**, which was heavily used during the early Falcon 9 program.

Total Payload Mass

Task 3

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

In [12]:
```
%sql SELECT SUM("Payload_Mass__kg_") AS total_payload_kg FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)';
```

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

Out[12]:

total_payload_kg
45596

Average Payload Mass by F9 v1.1

Task 4

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

In [13]:
```
%sql SELECT AVG("Payload_Mass__kg_") AS avg_payload_kg FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';
```

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

Out[13]:

avg_payload_kg
2928.4

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

In [14]:
```
%sql SELECT MIN("Date") AS first_success_ground_pad FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground p
```

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

Out[14]:

first_success_ground_pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

In [15]:
```
%sql SELECT DISTINCT "Booster_Version" FROM SPACEXTABLE WHERE "Landing_Outcome" LIKE 'Success (drone ship)%' AND
```

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

Out[15]:

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

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

In [16]:
```
%sql SELECT "Mission_Outcome", COUNT(*) AS cnt FROM SPACEXTABLE GROUP BY "Mission_Outcome" ORDER BY cnt DESC;
```

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

Out[16]:

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

Boosters Carried Maximum Payload

Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

In [17]:
```
%sql SELECT DISTINCT "Booster_Version" FROM SPACEXTABLE WHERE "Payload_Mass__kg_" = (SELECT MAX("Payload_Mass__kg
```

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

Out[17]:

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

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

In [20]:
```
%%sql
SELECT CASE strftime('%m',"Date")
         WHEN '01' THEN 'January' WHEN '02' THEN 'February' WHEN '03' THEN 'March'
         WHEN '04' THEN 'April'   WHEN '05' THEN 'May'      WHEN '06' THEN 'June'
         WHEN '07' THEN 'July'    WHEN '08' THEN 'August'   WHEN '09' THEN 'September'
         WHEN '10' THEN 'October' WHEN '11' THEN 'November' WHEN '12' THEN 'December'
       END AS Month,
       "Landing_Outcome", "Booster_Version", "Launch_Site"
FROM SPACEXTABLE
WHERE substr("Date",1,4)='2015' AND "Landing_Outcome" LIKE 'Failure (drone ship)%'
ORDER BY strftime('%m',"Date");
```

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

Out[20]:

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

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

Task 10

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

In [22]:
```
%%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.
```

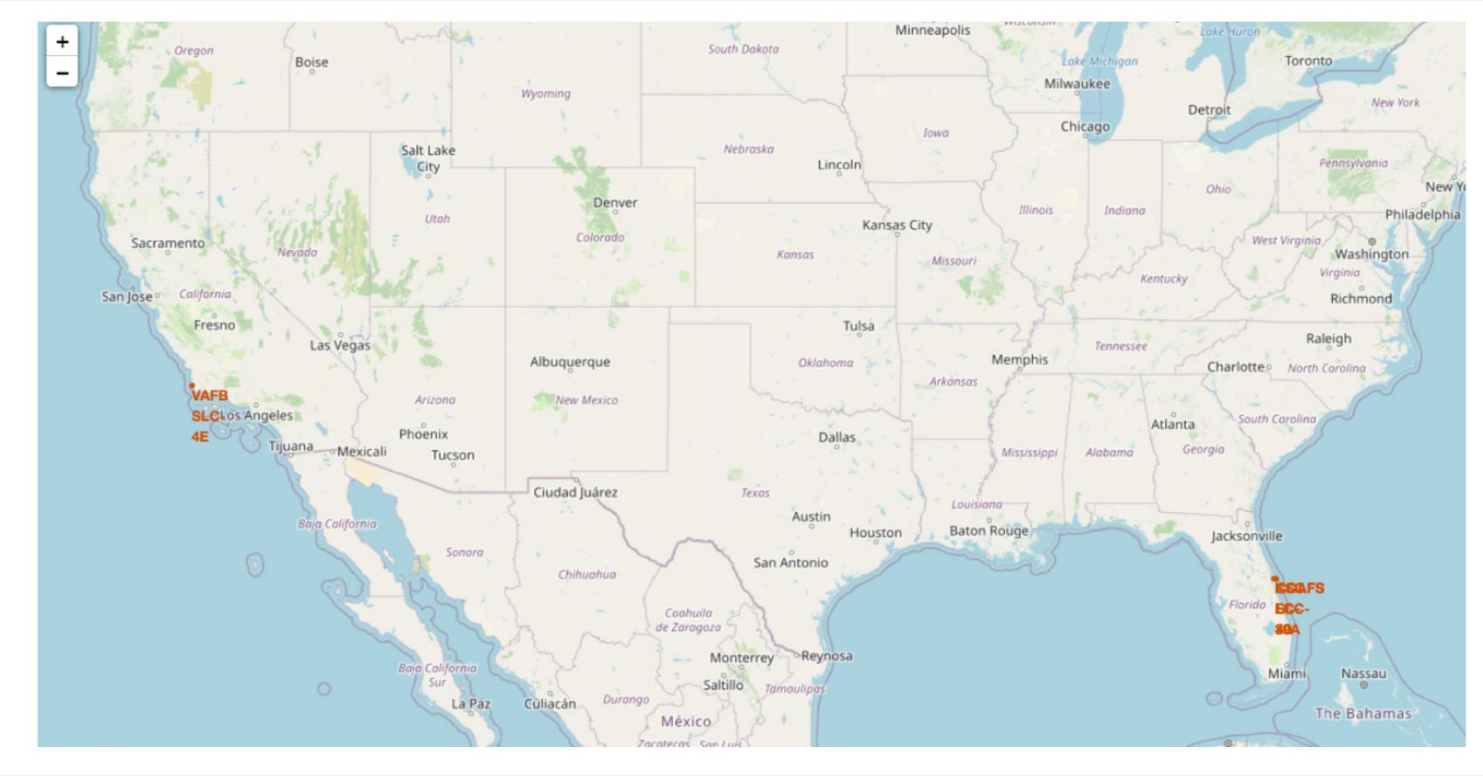
Out[22]:

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

Global SpaceX Launch Sites (Falcon 9)



The global distribution of Falcon 9 launch sites. Markers identify the primary pads used in the dataset: **CCAFS LC-40** and **KSC LC-39A** in Florida, and **VAFB SLC-4E** in California.

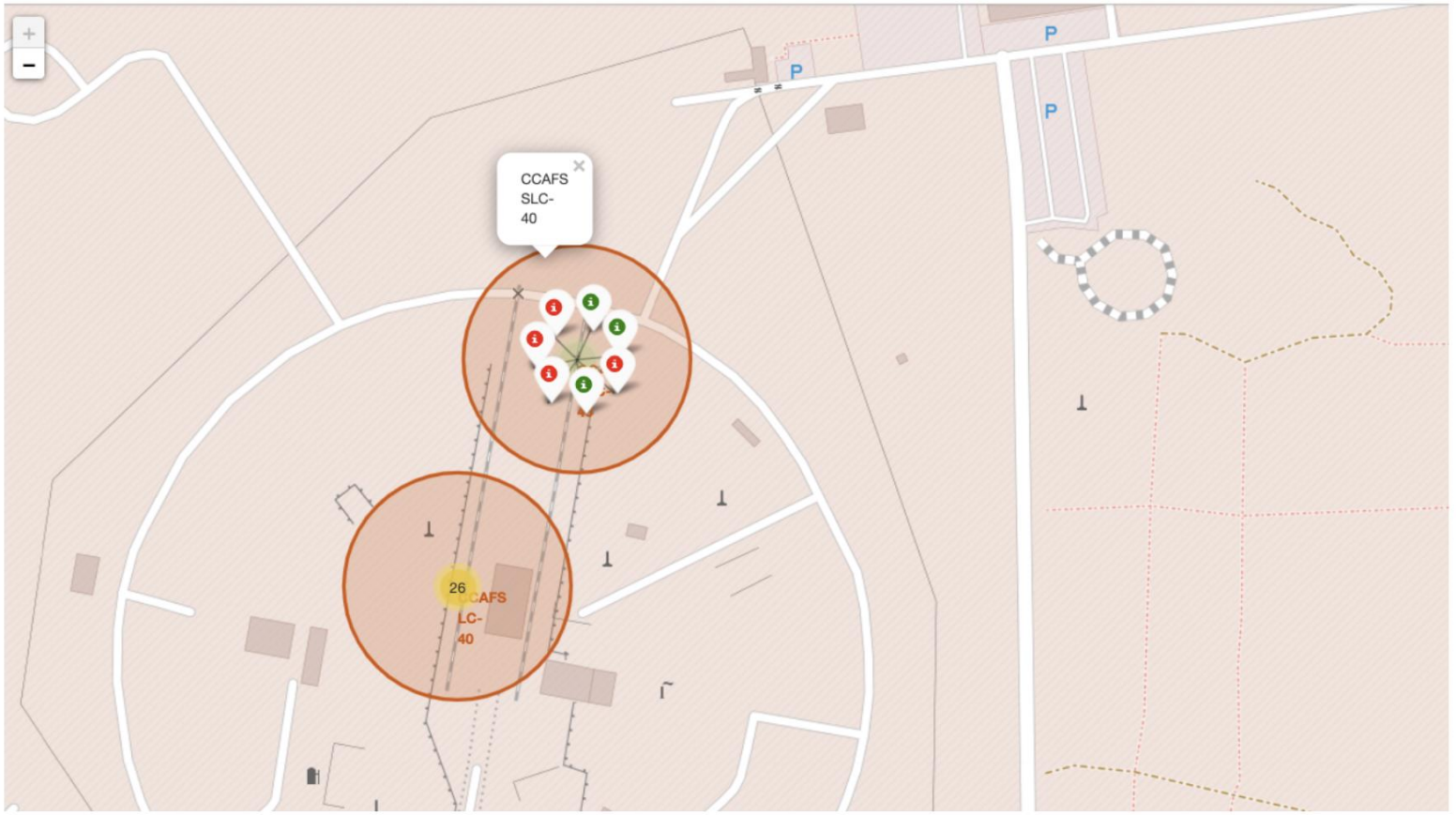
Why it matters: The clustering on the U.S. coasts reflects safety (downrange over ocean) and mission profiles (e.g., Vandenberg for polar/Sun-sync orbits). Seeing all sites together clarifies SpaceX's operational footprint and helps relate geographic factors to outcomes analyzed elsewhere in the project.

Success/failed launches for each site on the map

On the Folium map, each launch site is marked with outcomes color-coded by success or failure. **Cape Canaveral (CCAFS LC-40)** has the highest density of markers, reflecting that it hosted many of the early Falcon 9 flights, with a mix of successes and some failures in its initial missions. **Kennedy Space Center (KSC LC-39A)** shows mostly successful launches, since it was added later after the technology matured. **Vandenberg (SLC-4E)** has fewer markers overall, but like KSC its outcomes are predominantly successful, because launches from this site are mission-specific and occurred later in the program.

In summary, the map shows that failures are clustered mainly in the earlier Cape Canaveral missions, while later launches across all sites are largely successful. This illustrates both **the geographic spread of launches** and **the steady improvement in reliability** as Falcon 9 evolved.

Success/failed launches for each site on the map



From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.

Distances between a launch site to its proximities

Screenshot-ready map

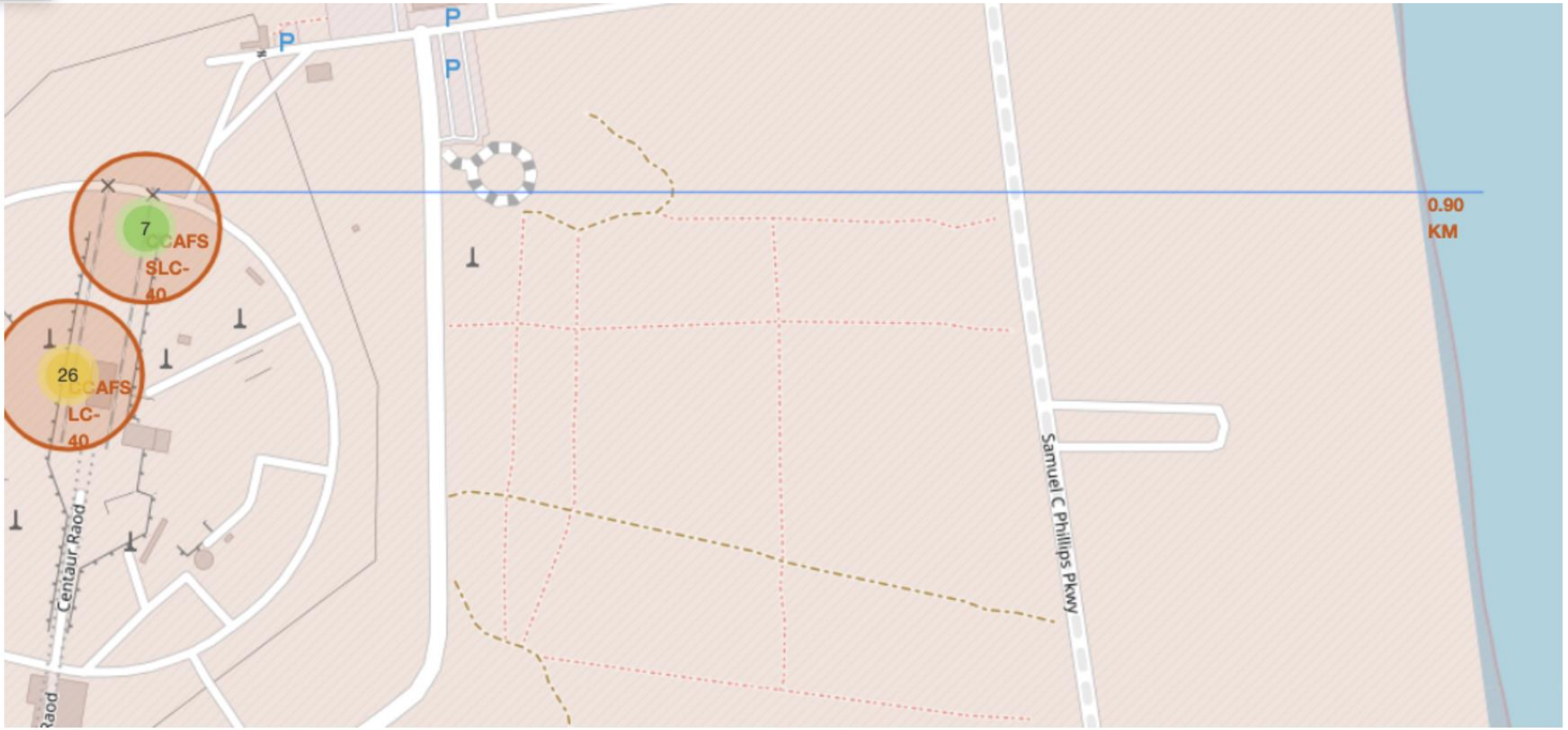
Open this file in your browser and take the screenshot with all markers and lines visible:
KSC LC-39A Proximities Map

What's on the map:
A **blue marker** for the launch site (**KSC LC-39A**).
Three **proximity points**:
Nearest Coastline (Atlantic) — distance shown in the line tooltip.
Highway (Kennedy Pkwy SR-3) — distance shown in the line tooltip.
Railway (KSC Industrial RR) — distance shown in the line tooltip.
Polylines from the site to each point; hover any line to see the **distance in km**.
The map auto-fits to include all points.

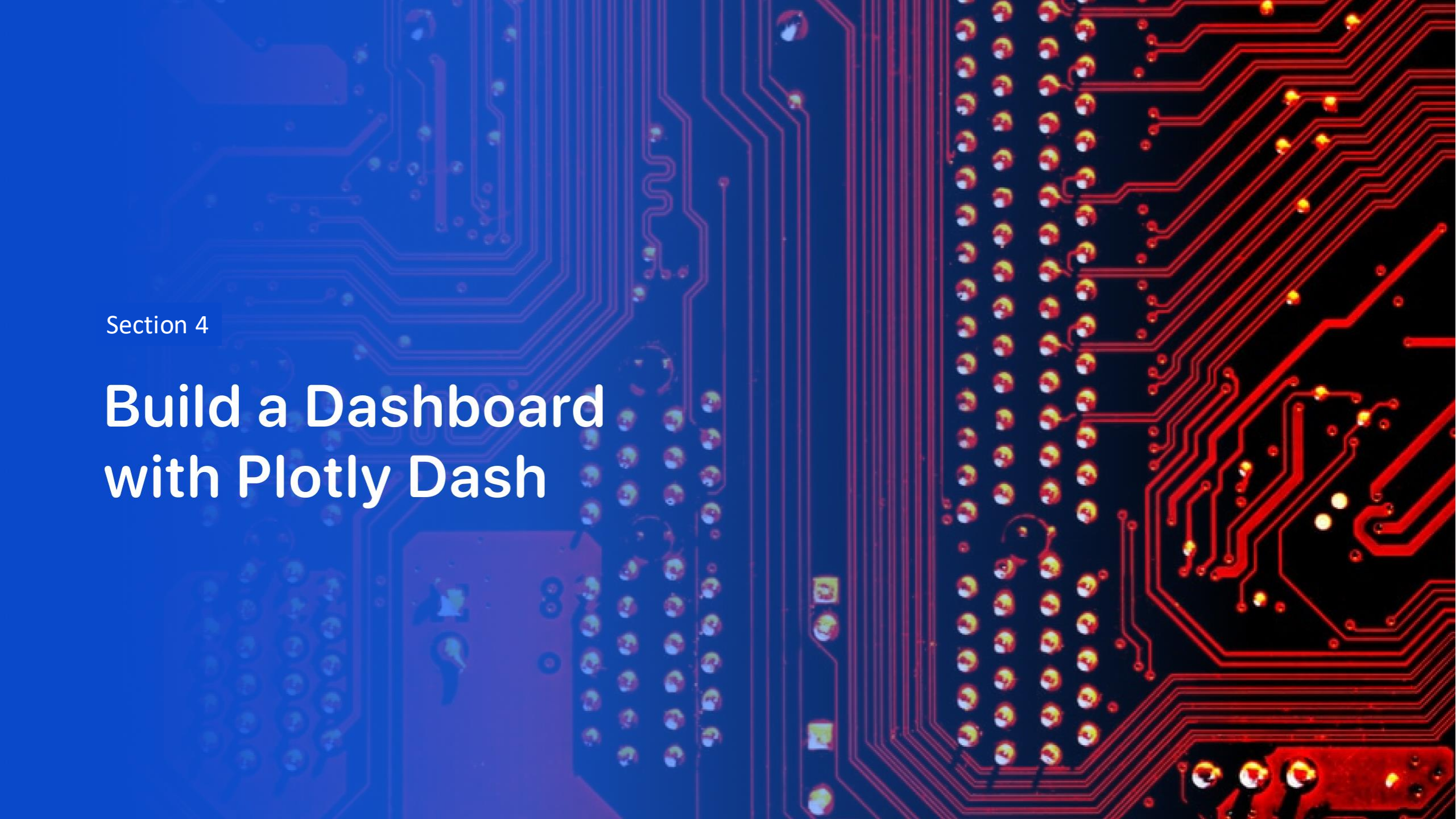
Distances between a launch site to its proximities

Your updated map with distance line should look like the following screenshot:

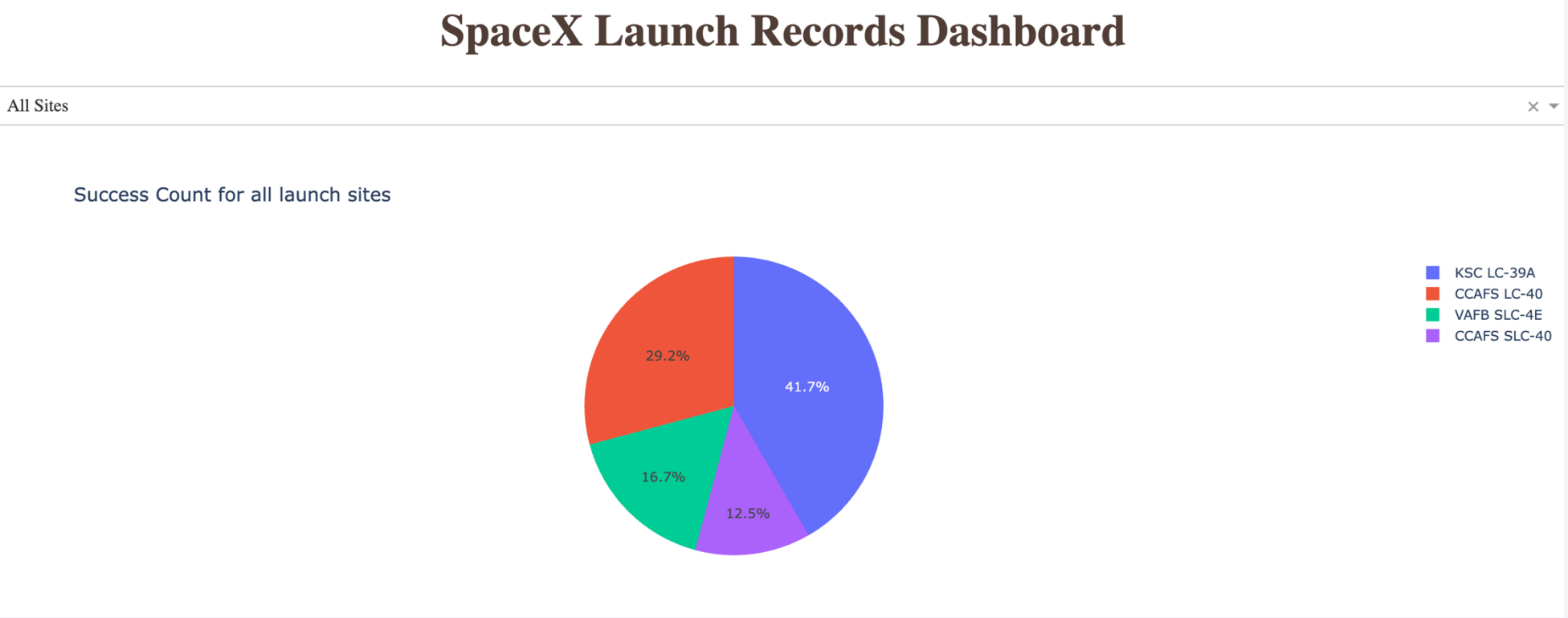
107 KB



TODO: Similarly, you can draw a line betwee a launch site to its closest city, railway, highway, etc. You need to use `MousePosition` to find the their coordinates on the map first

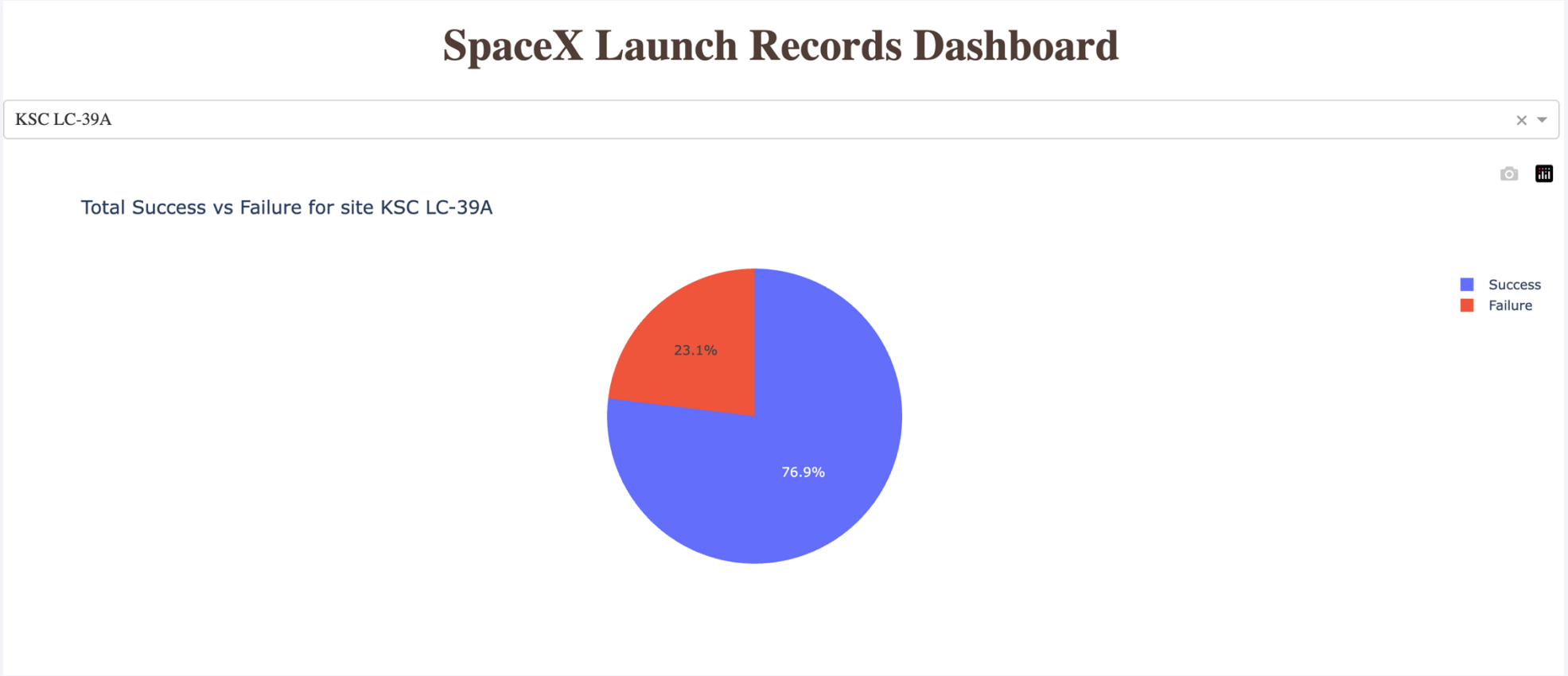

Section 4
Build a Dashboard with Plotly Dash

Success count for all launch sites



The pie chart shows the distribution of successful launches across all SpaceX launch sites, highlighting that most successes came from the Florida sites (CCAFS LC-40 and KSC LC-39A), with fewer from VAFB SLC-4E and CCAFS SLC-40.

Highest launch site success ratio



The launch site with the highest launch success ratio is **KSC LC-39A**, and the corresponding pie chart shows the breakdown of **success vs. failure** outcomes for launches from that site. The chart makes it clear that **the vast majority of launches from KSC LC-39A were successful**, with only a small portion representing failures, confirming it as the most reliable site in the dataset.

Payload Range

The dashboard shows that most successful launches came from KSC LC-39A and CCAFS LC-40. The scatter plot indicates that successes are concentrated in the 2,000–6,000 kg payload range, with newer booster versions (FT, B4, B5) achieving higher success rates, while older versions (v1.0, v1.1) had more failures. Overall, success rates improved as payloads increased and booster technology advanced.

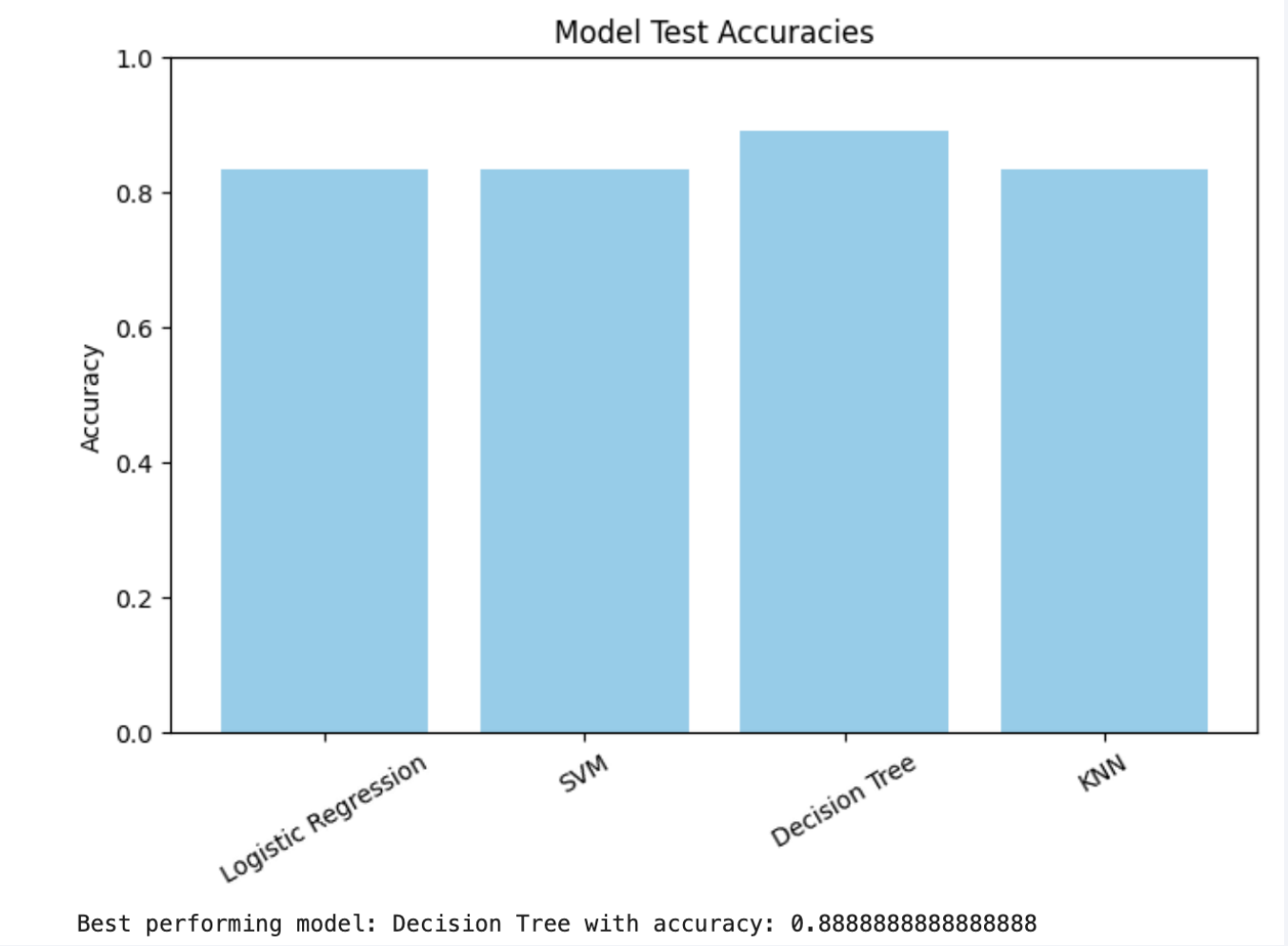
Section 5

Predictive Analysis (Classification)

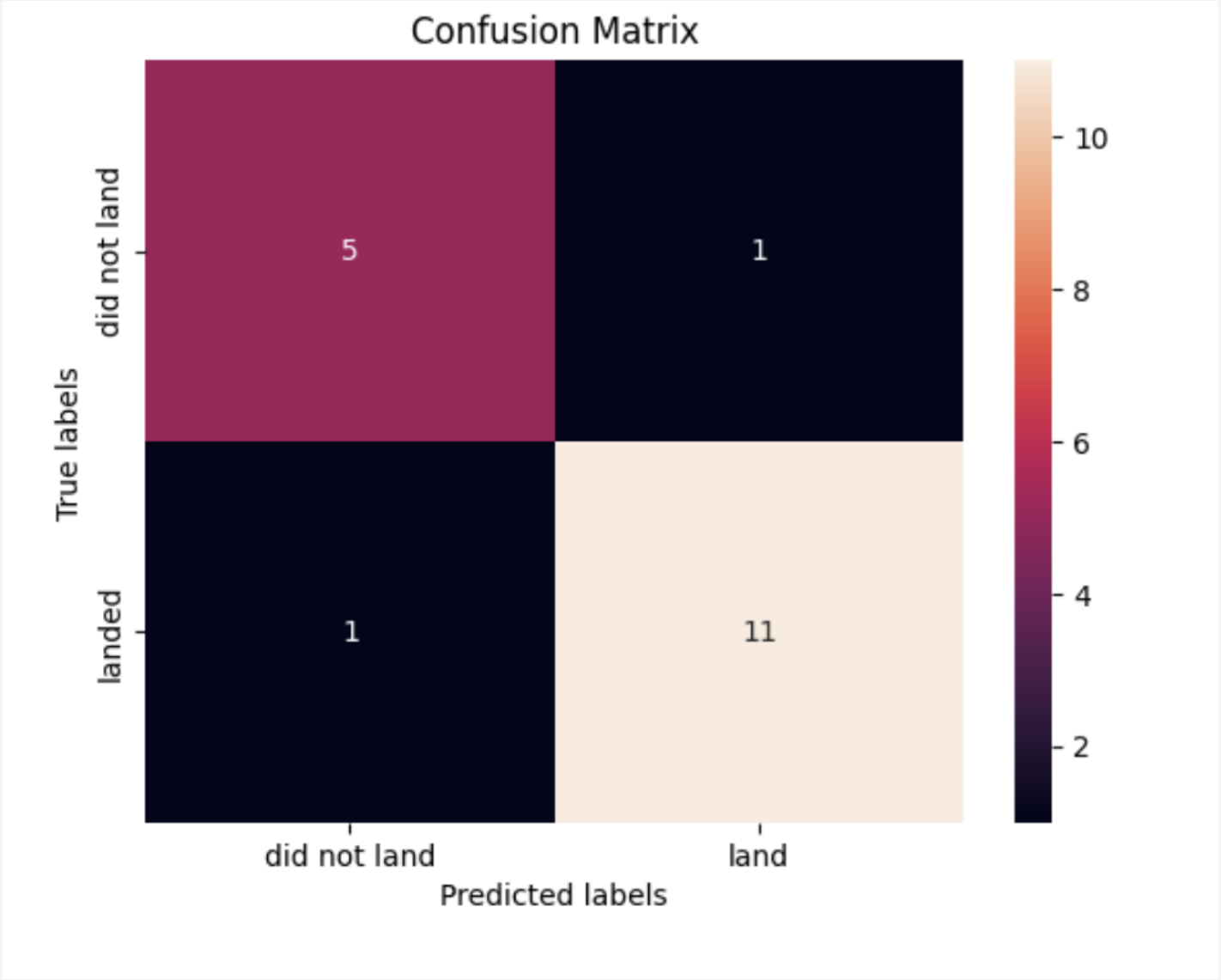
Classification Accuracy

Logistic Regression Test Accuracy: 0.8333
SVM Test Accuracy: 0.8333
Decision Tree Test Accuracy: 0.8889
KNN Test Accuracy: 0.8333

Best performing method on the test set:
Decision Tree



Confusion Matrix



The **Decision Tree** achieved the highest classification accuracy among all models on the test set. This suggests that the tree was best at capturing the underlying relationships in the data. Unlike Logistic Regression or SVM, which make more rigid assumptions about how the classes are separated, a Decision Tree can flexibly split the feature space into smaller, more specific regions. That flexibility allows it to handle both linear and non-linear patterns in the dataset, which likely explains why it performed better for predicting Falcon 9 first-stage landings.

Conclusions

1. Data Collection is Diverse but Essential
Using both **REST API calls** and **web scraping**, we built a complete dataset on Falcon 9 launches. This showed how combining multiple sources ensures reliability and completeness, which is crucial for downstream analysis.

2. Visualization Highlights Performance Patterns
Through **exploratory data analysis (EDA)**, scatter plots, bar charts, and histograms revealed key trends: launch successes increased over time, heavier payloads became more reliable, and newer booster versions had higher success rates.

3. Geospatial and Interactive Dashboards Add Insight
By applying **Folium maps** and **Plotly Dash dashboards**, we uncovered spatial patterns (e.g., concentration of launches at Florida sites) and created interactive views of success vs. failure. These tools helped stakeholders visualize operational reliability at different sites and payload ranges.

4. Machine Learning Confirms Reliability Gains
Building and evaluating models (Logistic Regression, SVM, Decision Tree, KNN) showed that predictive accuracy is strong. Logistic Regression and Decision Trees often performed best, confirming that **SpaceX's success rate can be reliably predicted** based on payload, site, booster version, and orbit.a

Thank you!