IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Kirk B. Smith
1 November 2025

Outline

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

Executive Summary

Summary of Methodologies

- Used SpaceX data with binary targets and mostly logical features.
- Applied standard preprocessing, stratified train/test split, and classical classifiers with GridSearchCV.
- Evaluated models using cross-validation, test accuracy, confusion matrices, and ROC AUC.

Summary of All Results

- Most classifiers had similar test accuracy due to the small test set.
- Confidence intervals were wide, indicating noisy accuracy estimates.
- Model choice should consider error types, business costs, and interpretability.

Introduction

Introduction

- This data science project focuses on predicting the success of SpaceX Falcon 9 first-stage rocket landings.
- SpaceX offers Falcon 9 launches at $62 million, significantly less than competitors who charge over $165 million.
- The cost savings are largely due to SpaceX's ability to reuse the rocket's first stage.

Project Objective

- Assessing the probability of a successful Falcon 9 first-stage landing is essential for calculating the true cost per launch.
- This analysis is valuable for companies aiming to compete with SpaceX for launch contracts.
- The project seeks to uncover actionable business intelligence beyond standard data science analysis.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology involved two steps:
  - SpaceX Falcon Rocket information was downloaded using their public API.
  - Flight details were gathered by web scraping a Wikipedia page with Python tools.
- Data wrangling involved two key steps:
  - Exploratory data analysis (EDA) and determining training labels.
  - Exploratory data analysis (EDA) was performed using visualization and SQL
- Interactive visual analytics were performed using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Standard methods were used to build, tune, and evaluate classification models (presented below).
  - Four models were fully built and tested. These included Logistic Regression, SVM, Decision Tree, and KNN

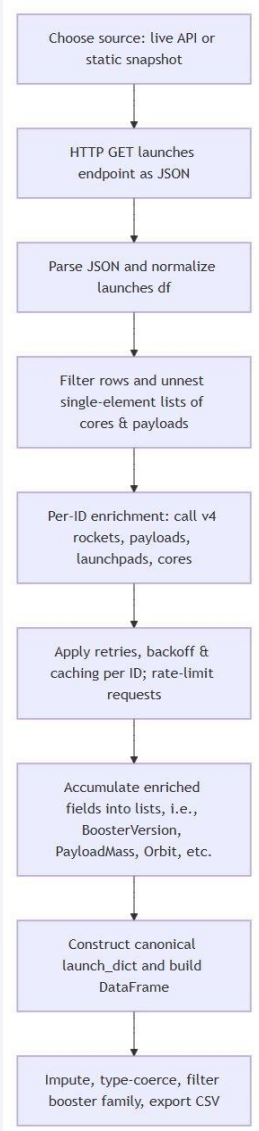
Data Collection – SpaceX API

Data collection with the SpacsX API Pipeline Utilized:

1. **Data Source & Snapshot Strategy:** Used SpaceX API REST calls for live launch data; static JSON snapshot for reproducibility. Snapshots ensure stable analysis; use live API for the latest data.
2. **Ingestion & Schema Normalization:** Loaded JSON via HTTP GET; flattened with pd.json_normalize(). Kept only key columns: rocket, payloads, launchpad, cores, flight number, and date.
3. **Filtering & List Unnesting:** Excluded launches with multiple cores/payloads for consistency. Converted single-item lists to IDs for easier lookups.
4. **Per-ID Data Enrichment:** Fetched details for rockets, payloads, launchpads, and cores via targeted API calls. Added human-readable fields and handled missing data defensively.
5. **Rate Limiting & Caching:** Added delays and retries for API calls; cached responses by ID to reduce load.
6. **Parsing & Type Handling:** Coerced types (e.g., mass to float, dates to datetime); normalized missing values.
7. **Data Assembly:** Compiled enriched attributes into a single dataframe; ensured consistent lengths.
8. **Cleaning & Finalization:** Filtered to target boosters, reset flight numbers, handled missing values, and set types.
9. **Export & Reproducibility:** Exported DataFrame to CSV; archived static data and caches for reproducibility.

SpaceX API Workflow in the left column:

Notebook available on GitHub:



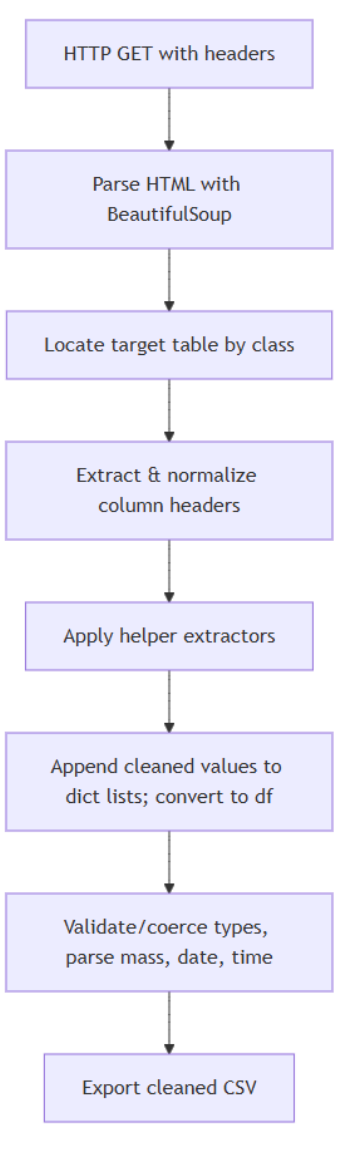
Data Collection – Web Scraping

Web Scraping Pipeline Utilized:

1. **Stable Data:** Used Wikipedia's *oldid* parameter for reproducible, static snapshots.
2. **Polite Scraping:** Custom User-Agent; respect robots.txt; add delays for multiple requests.
3. **Robust Parsing:** Located tables by class; validated headers; cleaned and normalized columns.
4. **Reliable Extraction:** Helper functions handled missing data, normalized values, and filtered valid rows.
5. **Data Output:** Built a Dataframe from cleaned rows; exported to CSV for reproducibility.

Web Scraping Workflow in left column:

Notebook available at GitHub:



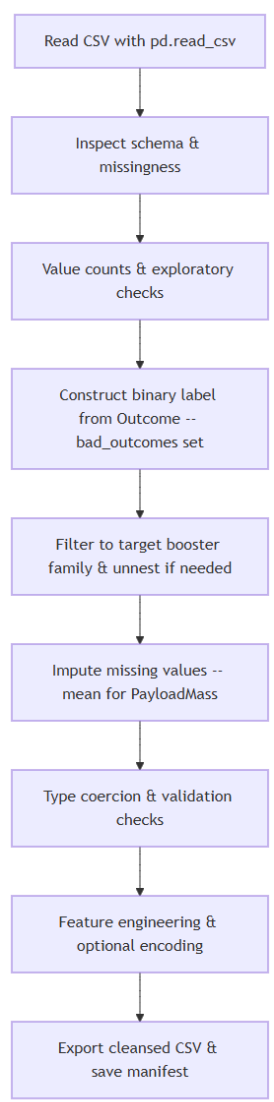
Data Wrangling

Data Wrangling Workflow:

- **Ingestion & Inspection:** Load CSV into Pandas, check top rows, and assess missing values.
- **Column Selection:** Identify and select relevant numerical/categorical features.
- **Exploratory Checks:** Use value counts to review category distributions and spot rare cases.
- **Label Construction:** Map raw outcomes to binary labels for supervised learning.
- **Filtering:** Keep only Falcon 9 launches; reset indices for consistency.
- **Missing Values:** Impute means for continuous features; manage NaNs.
- **Type Validation:** Cast columns to correct types and validate ranges.
- **Feature Engineering:** Create derived features; ensure transformations are reproducible.
- **Persistence:** Export cleaned data and document all steps for reproducibility.

See workflow in the left column:

The Notebook is available on GitHub at



EDA with Data Visualization

During exploratory data analysis of the SpaceX launches dataset, the following charts and artifacts were produced. The visualizations helped identify candidate features for modeling (payload mass, flight number, orbit, launch site), class balance, and potential outliers or missing ranges (for example, some sites having few heavy payload launches). The plots included:

- **FlightNumber vs PayloadMass (catplot)** — a scatter-like categorical plot with hue set to `Class` (landing success). Useful to inspect how payload mass and flight experience relate to landing outcomes and to identify patterns or outliers.
- **FlightNumber vs LaunchSite (catplot)** — compares flight number distribution across launch sites with `Class` hue. Helps surface site-level differences and whether experience (flight number) correlates with success per site.
- **Orbit vs FlightNumber (catplot)** — shows flight number by orbit type with success overlay. Useful to detect orbit-specific patterns in success rates and whether some orbits are more sensitive to experience.
- **PayloadMass vs Orbit (catplot)** — inspects payload distributions across orbit types and overlays success. Helps identify payload–orbit combinations with higher failure risk.
- **Success Rate by Orbit (barplot)** — bar chart of mean `Class` per `Orbit`. Provides a compact summary of which orbits have higher historical success.
- **Success Rate by Year (line plot)** — temporal trend showing how overall launch success evolved year-to-year. Useful to assess improvements over time.
- **One-hot encoded features (`features_one_hot`)** — categorical columns (`Orbit`, `LaunchSite`, `LandingPad`, `Serial`) were encoded and cast to `float64` to prepare a numeric feature matrix for modeling.
- **Note:** Separator lines were drawn between categorical levels (when available) by converting axis yticks to a NumPy array and computing midpoints; this improves visual separation on the categorical plots.

The notebook is available on GitHub at:

EDA with SQL

The EDA with SQL portion of the analysis involved 11 SQL tasks comprising of these statements:

1. `SELECT DISTINCT Launch_Site FROM SPACEXTABLE;` — (Task 1) List unique launch sites present in the dataset.
2. `SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;` — (Task 2) Show the first 5 records for launch sites starting with 'CCA' to inspect sample rows and schema.
3. `SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer = "NASA (COTS)";` — (Task 3) Compute the total payload mass carried by missions for the NASA (COTS) customer group.
4. `SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version LIKE "F9 v1.1%";` — (Task 4) Compute the average payload mass for boosters matching F9 v1.1.
5. `SELECT DISTINCT Date FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)' ORDER BY Date ASC LIMIT 1;` — (Task 5) Find the earliest date with a successful ground-pad landing.
6. `SELECT MIN(PAYLOAD_MASS__KG_) FROM SPACEXTABLE;` — Find the minimum payload mass in the table (used as a quick data-range check).
7. `SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND Payload_Mass__KG_ > 4000 AND Payload_Mass__KG_ < 6000 ORDER BY Booster_Version ASC;` — (Task 6) List booster versions that achieved drone-ship successes for payloads between 4000 and 6000 kg.
8. Mission outcome counts (Task 7):
   - `SELECT COUNT(Mission_Outcome) AS Success_Count FROM SPACEXTABLE WHERE Mission_Outcome LIKE '%Success%';` — count of outcomes containing 'Success'.
   - `SELECT COUNT(Mission_Outcome) AS Failure_Count FROM SPACEXTABLE WHERE Mission_Outcome NOT LIKE '%Success%';` — count of non-success outcomes.
   - `SELECT Mission_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTABLE GROUP BY Mission_Outcome ORDER BY Outcome_Count DESC;` — distribution of each mission outcome value.
9. `SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);` — (Task 8) Find booster versions that carried the maximum payload (using a subquery).
10. `SELECT substr(Date, 6,2) AS Month, substr(Date, 0,5) AS Year, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%Failure%' AND Landing_Outcome LIKE '%drone ship%' AND substr(Date, 0, 5) = '2015';` — (Task 9) List failure drone-ship landing records in 2015 with month, booster version and launch site (SQLite substr used to extract month/year).
11. `SELECT DISTINCT COUNT(Landing_Outcome) AS Outcome_Count, Landing_Outcome FROM SPACEXTABLE WHERE Date Between '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome Order BY COUNT(Landing_Outcome) DESC;` — (Task 10) Rank landing outcome counts within the given date range in descending order.

Build an Interactive Map with Folium

Objects (maps, markers, circles, etc.) constructed:

1. **Data ingest and selection:** Read `spacex_launch_geo.csv` and selected relevant columns (`Launch Site`, `Lat`, `Long`, `class`).. Grouped by `Launch Site` to extract a single coordinate per site for site-level markers.

2. **Create a Folium map object:** Initialize map and set the center and zoom for interactive exploration.

3. **Add highlighted circles for launch sites:** Used `folium.Circle([lat, long] to draw a visible area around each site. Circles emphasize the site footprint and are helpful when comparing distances or coverage areas.

4. **Add textual markers for site labels**: Added markers with `DivIcon` to show the site name at the coordinate. These labels make it easy to identify which marker corresponds to which launch site when zooming and panning.

5. **Add marker clusters for launch records:** Created a `MarkerCluster` and added per-launch markers colored by class (green for success, red for failure). Marker clustering avoids overplotting when many launches share the same coordinates and makes patterns easier to see.

6. **Use MousePosition for interactive coordinate capture:** Added `MousePosition` so that hovering showed exact lat/long values for points of interest (coastlines, cities, railways) when selecting proximity targets.

7. **Compute distances between coordinates:** Implemented `calculate_distance` (Haversine formula) to compute kilometers between a launch site and a selected proximity (coastline, city, railway). This quantitative measure supports objective proximity analysis.

8. **Add distance markers and polylines:** Plotted a `folium.Marker` showing the numeric distance (as an icon label) and drew a connecting line. Polylines and labeled distances make spatial relationships explicit and are useful for reporting or visual inspection.

Why add these markers:

- Markers and labels: identify exact locations and provide human-readable context (site names, distances).
- Circles: visually emphasize the area around a site and help compare proximity to nearby features at the same map scale.
- MarkerCluster: reduces visual clutter when many launches share coordinates and surfaces aggregate patterns of success/failure.
- MousePosition: enables interactive coordinate capture for distance calculations.
- Polylines + distance labels: provide clear, measurable connections between a launch site and nearby geographic features (coastlines, cities, railways), enabling quantitative proximity analysis.

The notebook is available for download on GitHub at

Build a Dashboard with Plotly Dash

The elements developed for the Dashboard include:

1. Dropdown (id="site-dropdown")

- Purpose: let the user select either "ALL" launch sites or a specific site.
- Why included: comparing sites side-by-side (ALL) or drilling into one site (specific) supports both high-level and focused analysis workflows.

2. RangeSlider (id="payload-slider")

- Purpose: allow selection of a payload-mass interval to filter launches.
- Why included: payload mass is a continuous predictor of landing success so interactive filtering helps reveal payload-dependent patterns and reduces visual clutter for dense datasets.
- UX detail: updatemode="mouseup" triggers updates after the user finishes moving the handles, which avoids excessive re-rendering during dragging.

3. Pie chart (id="success-pie-chart")

- Shows: (a) For all sites: total successful launches per site, or (b) for one site: success vs failure counts.
- Why: Pie charts quickly show site contributions or, for a single site, the success/failure split.

4. Scatter chart (id="success-payload-scatter-chart")

- Plots Payload Mass (kg) vs landing outcome, with Booster Version Category for color grouping.
- Why: Scatter plots reveal relationships between payload and outcome; color shows booster interactions.
- Hover data (Flight Number, Launch Site) enables detailed inspection without leaving the chart.

Interaction design (how components work together):

- The Dropdown controls both charts: “ALL” shows overall site success; selecting a site focuses both charts for detailed analysis.
- The RangeSlider filters the scatter chart by payload, helping test if heavier payloads affect recovery rates and which boosters handled them.

The notebook is available for download on GitHub at

Predictive Analysis (Classification)

Methods/Workflow utilized for predictive analysis:

- **Data:** the notebook loads the SpaceX datasets and assigns a binary target `Y` (Class) and feature matrix `X`. The feature set contains mostly logical (0/1) variables and a normalized continuous feature.
- **Preprocessing:** numeric data are standardized using StandardScaler (the pipeline scales features before training). Logical features remain binary unless explicitly transformed.
- **Split:** a stratified train/test split is used (test_size=0.2, random_state=42); the resulting test set is small (~18 samples), which impacts the stability of test-set estimates.
- **Modeling:** a range of classical classifiers are trained and tuned via GridSearchCV (Logistic Regression, SVM, Decision Tree, KNN, etc.). For each model, the code searches modest hyperparameter grids and selects the best CV parameters.
- **Evaluation:** models are compared using cross-validated accuracy (CV), test-set accuracy, confusion matrices, and ROC AUC where probabilistic outputs exist. Additional diagnostics include cross-val summaries, bootstrap 95% CIs for test accuracy, and a disagreement table showing per-sample prediction differences.
- The notebook is available for download from GitHub at:

Results

Summary of EDA Results

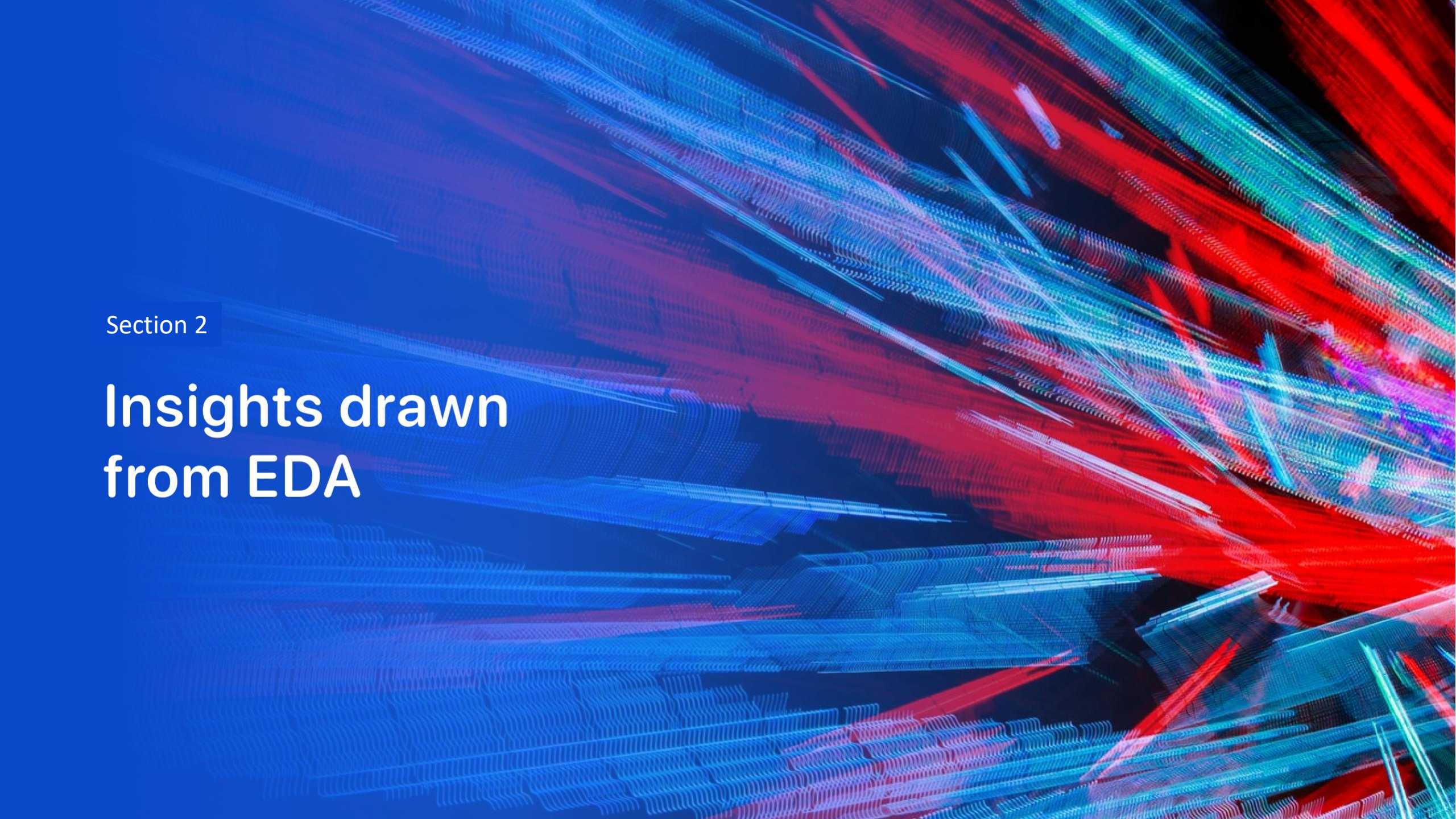
- EDA on the SpaceX dataset produced plots showing key relationships.
- Plots included catplots, barplots, and a line plot to visualize mission outcomes by variables like orbit, site, location, and flight number.
- These visualizations helped identify features linked to mission outcomes for the classification model.
- Section 2, below, presents the plots with explanatory notes.

Summary of EDA with SQL results

- The SQL phase of EDA revealed key insights on SpaceX Falcon launches, including success rates, launch site details, payload statistics, and outcomes for specific landing platforms.
- Section 2 below, starting with slide 23, presents the SQL outputs along with the SQL code used to query the output.

Summary of Classification Results

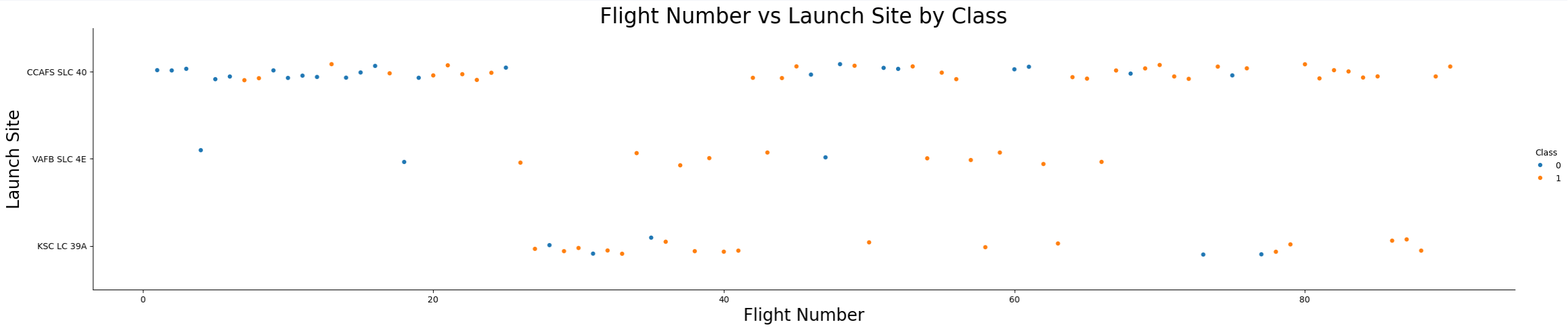
- Many classifiers produce similar test-set accuracies on the small hold-out set. This is expected: with only ~18 test samples, each prediction moves accuracy by ~5–6%, so different models can appear to tie by chance.
- Cross-validation and bootstrap diagnostics indicate considerable variability. Wide confidence intervals imply the observed test accuracies are noisy estimates of true performance.
- Confusion matrices expose different error types (false positives vs false negatives). Choice of a preferred model should consider these trade-offs alongside accuracy/ROC metrics and business cost of each error type.
- Tree-based models feature importances (plotted in the notebook) that help with interpretation; linear models provide coefficient directionality. These insights can guide feature selection and further modeling refinements.
- Recommendations: prefer cross-validated performance and bootstrap CIs to single test-set accuracy; if possible, enlarge the test set, use repeated CV, or run paired tests between top models to assess significance before choosing a production model.


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

The Flight Number vs. Launch Site plot shows the following:

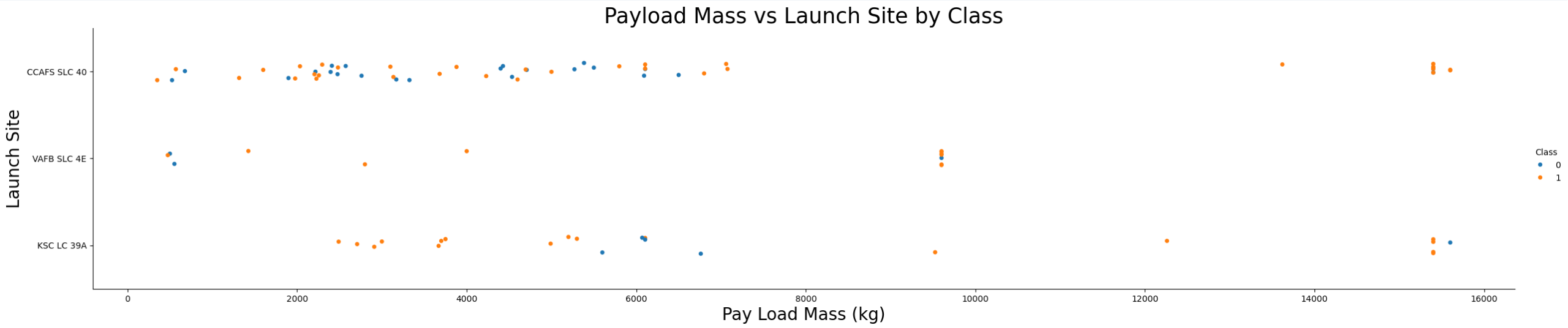
1. The three launch sites evaluated are listed on the vertical axis.
2. Flight numbers in ascending order are along the horizontal axis.
3. Color coding of does shows Blue for a failed mission and Red for a successful mission.
4. As indicated by the color division, missions have become more likely to succeed over time.
5. Also notable is that the different launch sites have different activity levels over time



Payload vs. Launch Site

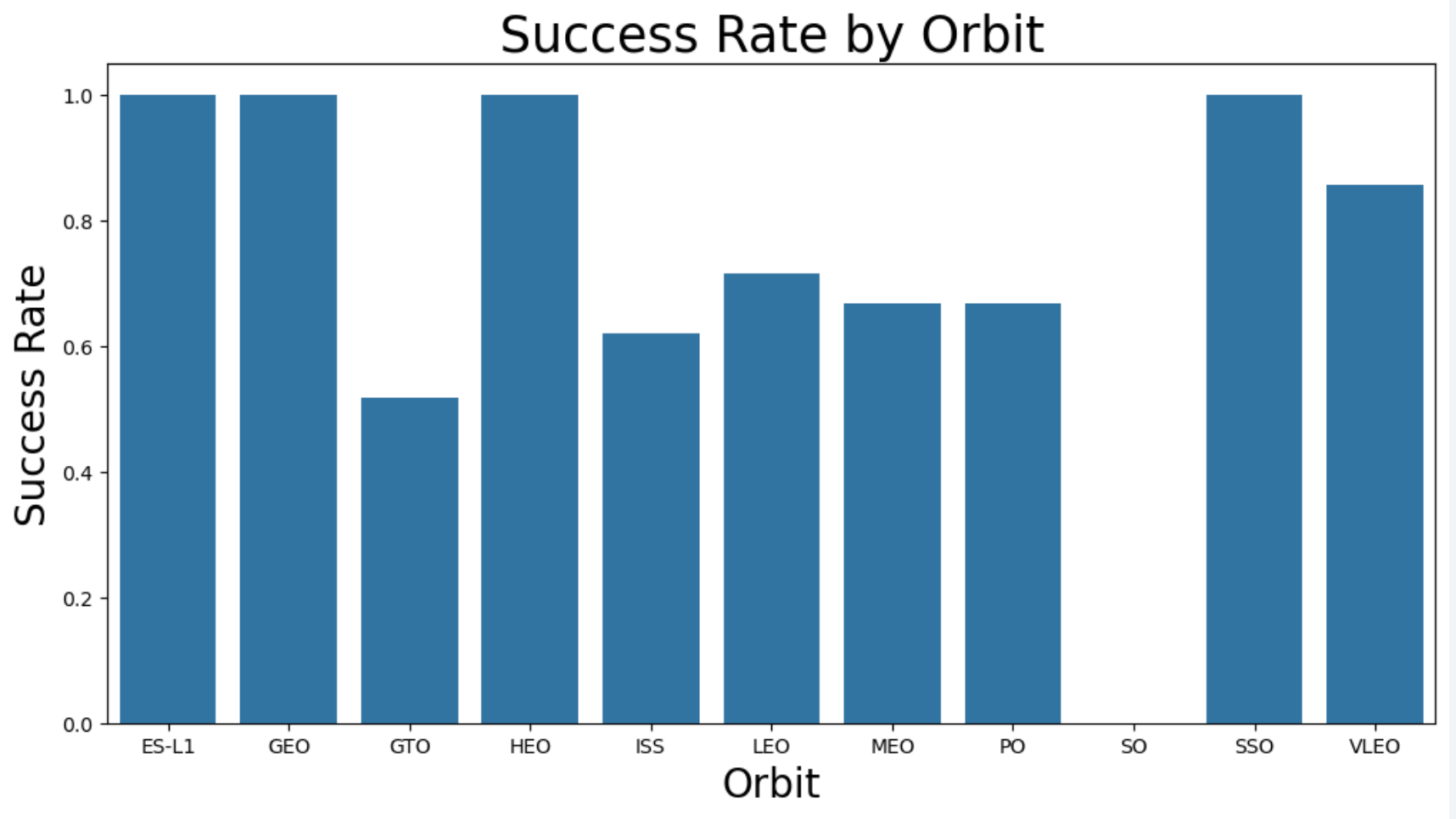
The Payload Mass vs. Launch Site by Class plot shows the following:

1. The three launch sites evaluated are listed on the vertical axis.
2. Pay Load Mass (kg) is shown on the horizontal axis.
3. Color coding of does shows Blue for a failed mission and Red for a successful mission.
4. As indicated by the color division, missions carrying above 8,000 kg are more likely to succeed than smaller payloads
5. As was true in the plot of site by flight number, the CCAFS SLC40 site carried the most tonnage and also the most flights.



Success Rate vs. Orbit Type

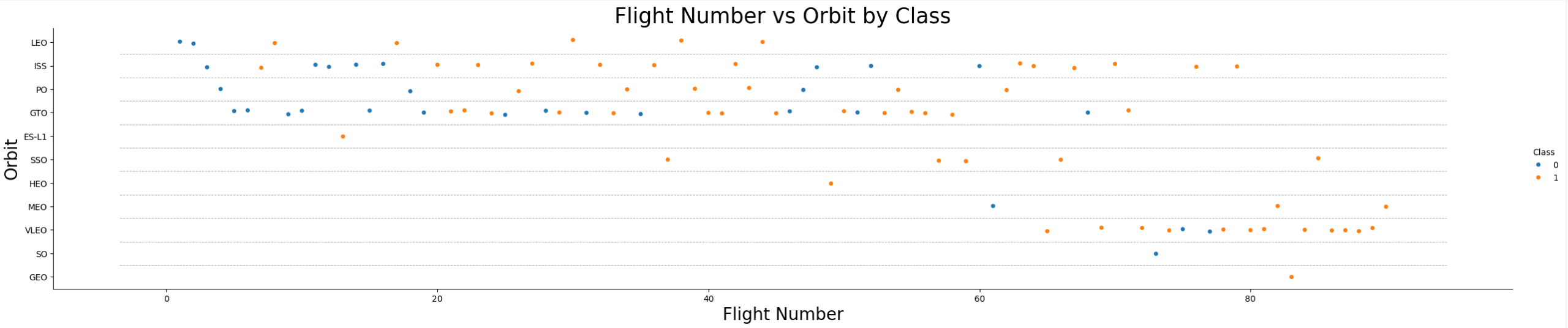
- The success rate is shown in the vertical column.
- The 11 possible orbits for Falcon rockets are shown on the horizontal axis.
- The chart indicates that four orbits, ES-L1, GEO, HEO, and SSO, have 100 percent mission success rates.
- No SO orbit missions have succeeded, hence it's 0.0 score.
- The remaining orbits, GTO, ISS, LEO, MEO, PO, and VLEO, range from 55 percent to 90 percent success rates.



Flight Number vs. Orbit Type

The Flight Number vs. Orbit Type plot shows the following:

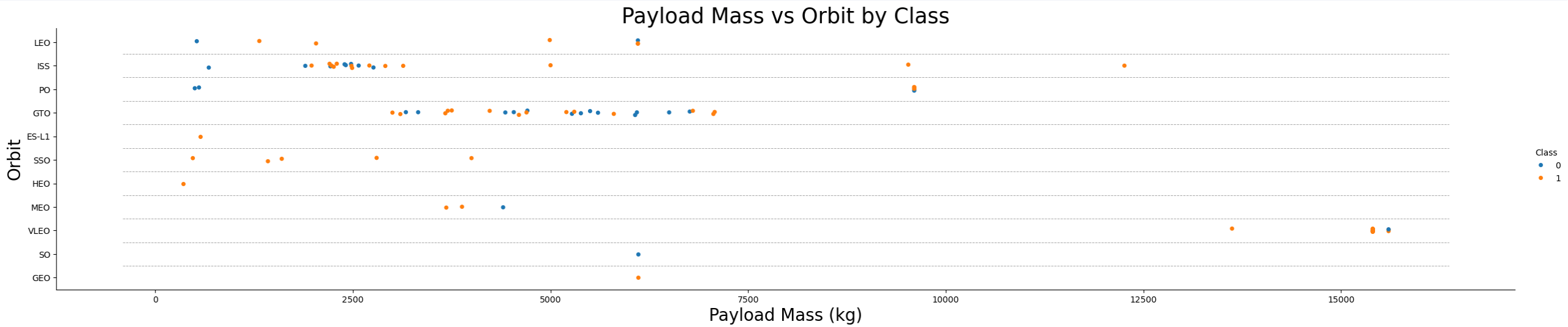
1. The Orbit type is shown in the vertical axis.
2. The flight number is shown on the horizontal axis.
3. Color coding of does shows Blue for a failed mission and Red for a successful mission.
4. As indicated by the dot patterns in the chart, several orbits gained prominence over time, especially the VELO orbit.
5. Relating back to the Success Rate by Orbit chart, the SO orbit (which had a 0.0 score) has only been used once.



Payload vs. Orbit Type

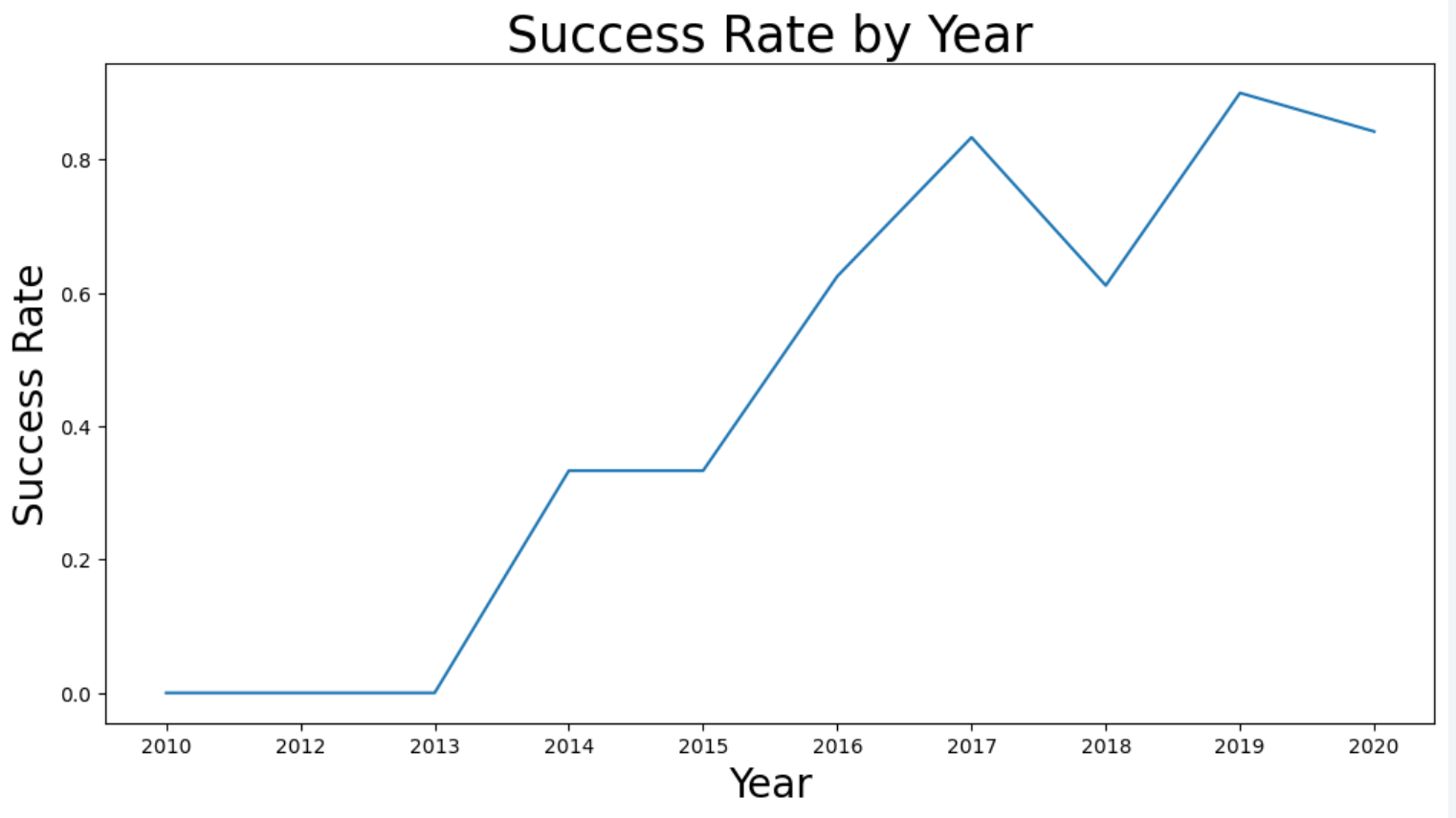
The Payload vs. Orbit Type plot shows the following:

1. The Orbit type is shown in the vertical axis.
2. The Payload Mass (kg) is shown on the horizontal axis.
3. Color coding of does shows Blue for a failed mission and Red for a successful mission.
4. As indicated by data patterns, the VELO orbit is preferred for the heaviest payloads.
5. The GTO Orbit is preferred for payloads between 2700kg to 7500kg; ISS is most common for <2500kg payloads.



Launch Success Yearly Trend

- The success rate is shown in the vertical column.
- Years are shown on the horizontal axis.
- The chart indicates overall growth in the success rate from 2010 to 2020.
- There were no years between 2013 and 2017 in which the success rate declined.
- Following the dip in 2018, the success rate rebounded and hit a new high in 2019.
- Despite a modest decline in 2020, the year ranks as the second highest success rate since 2010.



All Launch Site Names

Launch Sites

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

Obtained via the following SQL query:

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

Launch Site Names Begin with 'CCA'

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

The records were obtained via the following SQL query:

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

Total Payload Mass

Total payload mass carried by boosters from NASA:

```
525 kg
```

The finding was obtained via the following SQL query:

```
%%sql SELECT SUM(PAYLOAD_MASS__KG_)
    FROM SPACEXTABLE
    WHERE Customer = "NASA (COTS)";
```

Average Payload Mass by F9 v1.1

Average payload mass carried by booster version F9 v1.1:

2534.66 kg

The finding was obtained via the following SQL query:

```
%%sql SELECT AVG(PAYLOAD_MASS__KG_)
    FROM SPACEXTABLE
    WHERE Booster_Version LIKE "F9 v1.1%";
```

First Successful Ground Landing Date

The date of the first successful landing outcome on a ground pad:

2012-12-22

The finding was obtained via the following SQL query:

```
%%sql SELECT DISTINCT Date FROM SPACEXTABLE
    WHERE Landing_Outcome = 'Success (ground pad)'
    ORDER BY Date ASC
    LIMIT 1;
```

Successful Drone Ship Landing with Payload between 4000 and 6000

Boosters that have successfully landed on a drone ship and had a payload mass greater than 4,000 but less than 6,000

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

Obtained via the following SQL query:

```
%%sql SELECT DISTINCT Booster_Version FROM SPACEXTABLE
    WHERE Landing_Outcome = 'Success (drone ship)' AND
            Payload_Mass__KG_ > 4000 AND Payload_Mass__KG_ < 6000
    ORDER BY Booster_Version ASC;
```

Total Number of Successful and Failure Mission Outcomes

Successful Missions:

100

Failed Missions:

1

Obtained via the following SQL queries:

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

Boosters Carried Maximum Payload

Boosters that carried maximum payloads:

```
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
```

Obtained via the following SQL queries:

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

2015 Launch Records

Records of failure landing outcomes in drone ships, listing booster versions, launch sites, and the month for 2015:

Month	Year	Landing_Outcome	Booster_Version	Launch_Site
01	2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Obtained via the following SQL queries:

```
%%sql SELECT substr(Date, 6,2) AS Month,
      substr(Date, 0,5) AS Year,
      Landing_Outcome,
      Booster_Version,
      Launch_Site
FROM SPACEXTABLE
WHERE Landing_Outcome LIKE '%Failure%'
    AND Landing_Outcome LIKE '%drone ship%'
    AND substr(Date, 0, 5) = '2015';
```

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

Records of failure landing outcomes in drone ships, listing booster versions, launch sites, and the month for 2015:

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

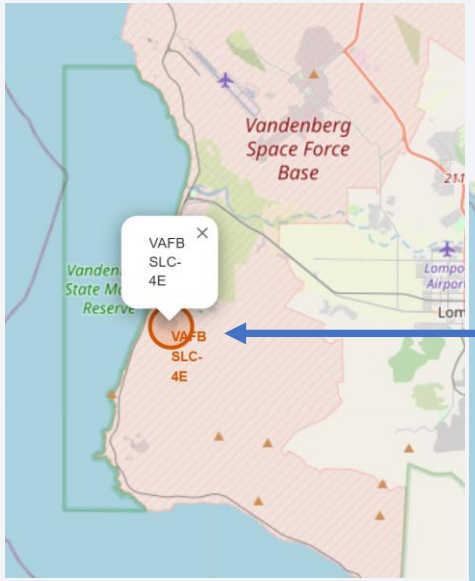
Obtained via the following SQL queries:

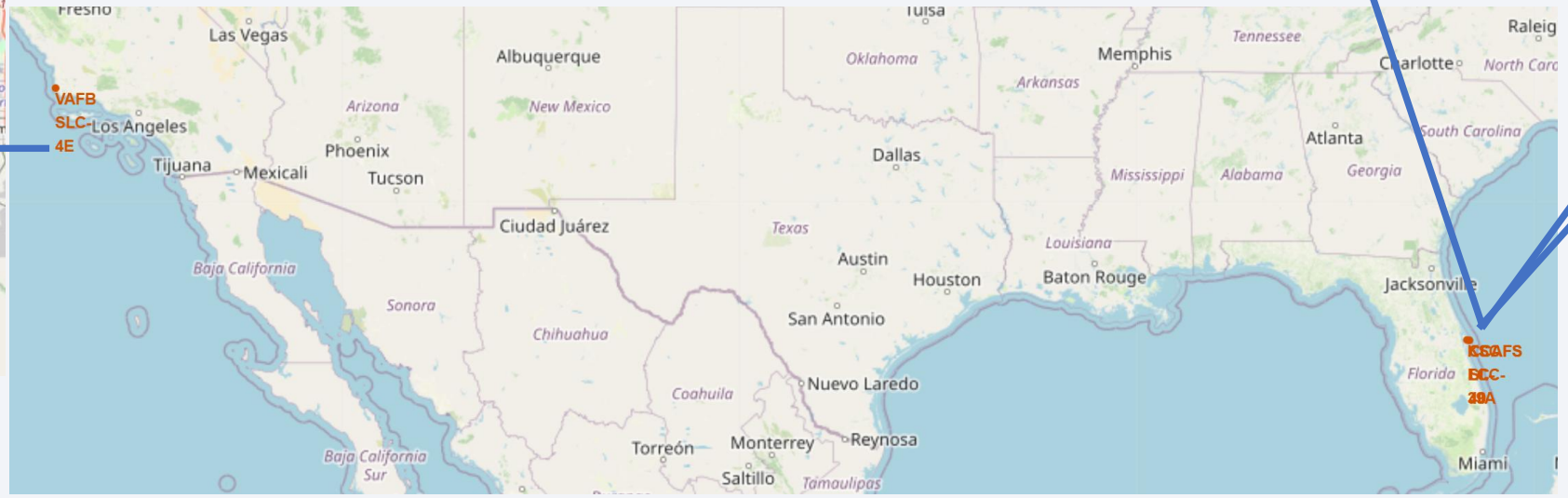
```
%%sql SELECT DISTINCT COUNT(Landing_Outcome) AS Outcome_Count,
          Landing_Outcome FROM SPACEXTABLE
   WHERE Date Between '2010-06-04' AND '2017-03-20'
   GROUP BY Landing_Outcome
   Order BY COUNT(Landing_Outcome) DESC;
```

Section 3
Launch Sites
Proximities Analysis

Folium Map of U.S. Launch Site Studies

- The fragment of the U. S. map shows the three Florida and one California launch sites studied.
- The detailed image to the left shows the California site near Vandenberg Space Force Base.
- The inset image to the right shows the three Florida launch sites located near Cape Canaveral.





Folium Map with Colored Launch Outcome Markers

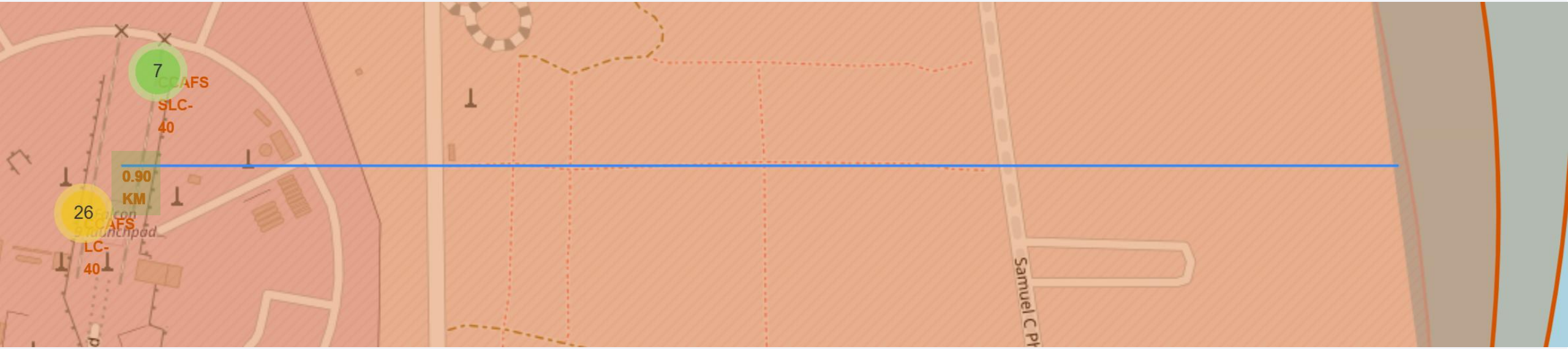
The image shows a detailed view of the folium map showing the location of the KSC LC-39A launch site.

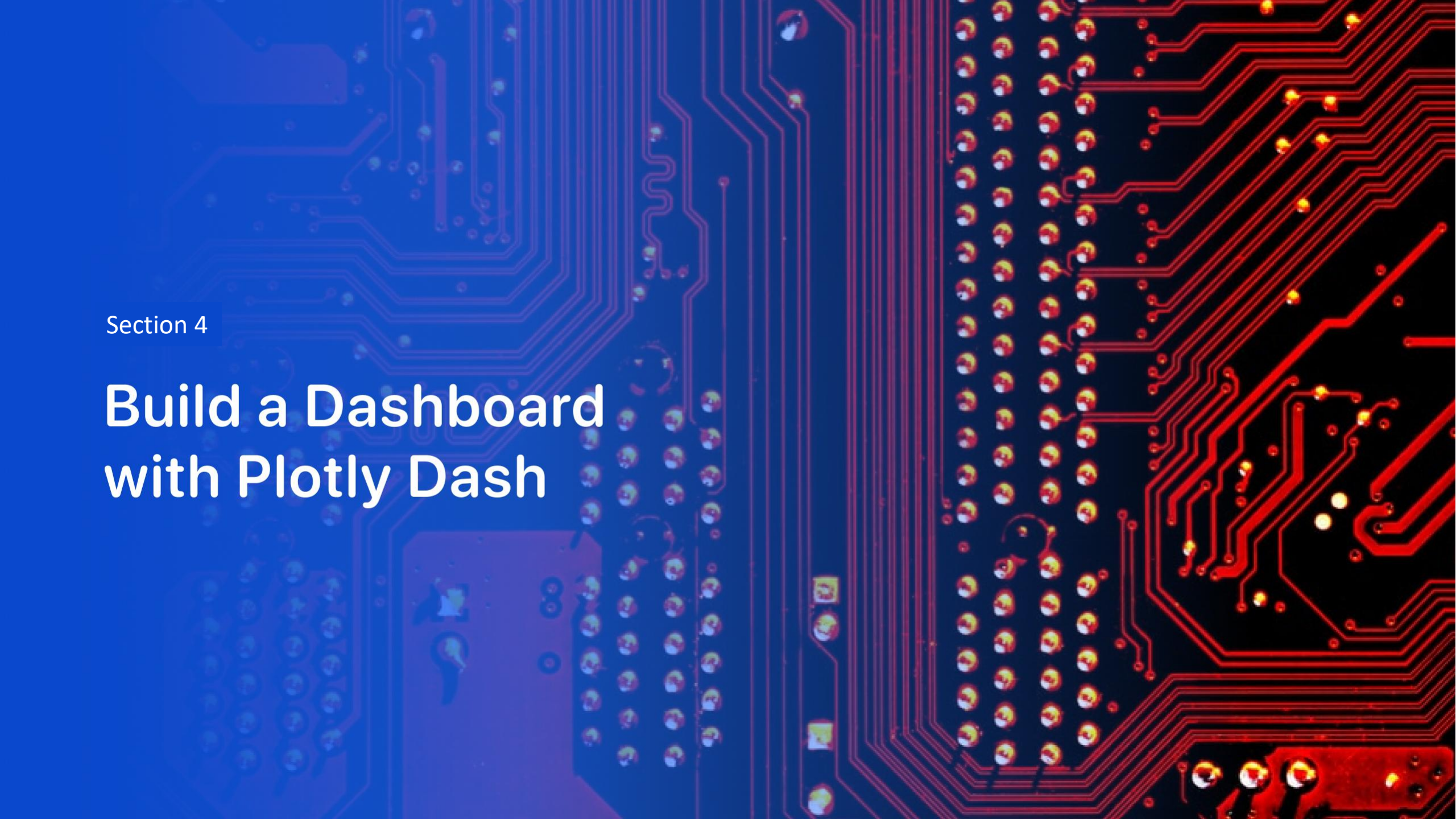
The colored markers show the history of launch outcomes at the site, with green for success and red for failure.

The marker on the inside of the spiral is the first launch at the site. They are presented sequentially thereafter.

Folium Map with Distance Marking Line

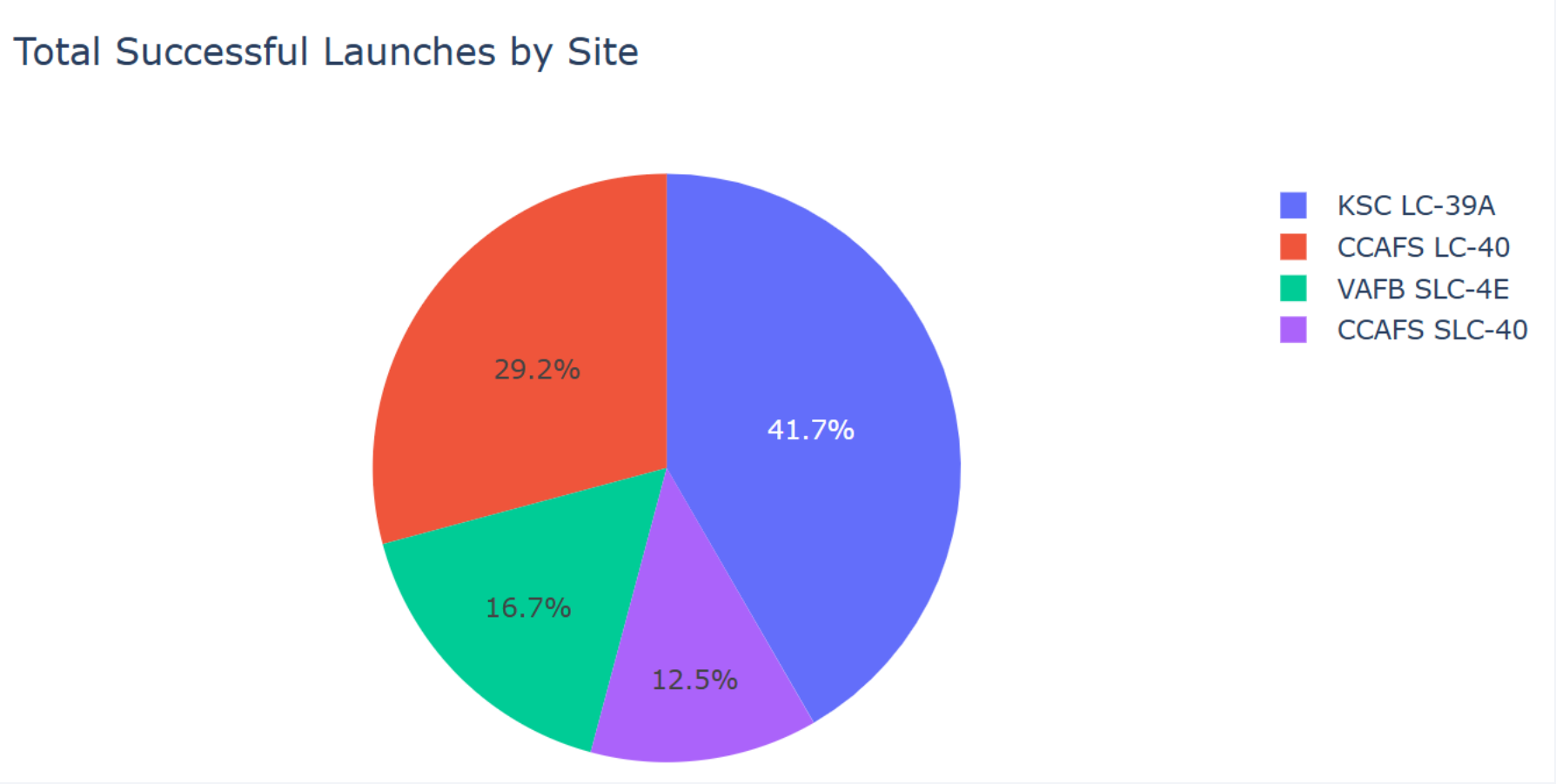
The image shows the midpoint between two of the Florida launch sites and the nearest coastline. The 90 km distance is within both sites' safety margin, suggesting that the adjoining coastline will have to be closed for a minimum of 10km to civilian boating during launches. This detail shows Folium's built-in capacity to integrate with analytic code in generate map features.




Section 4
Build a Dashboard with Plotly Dash

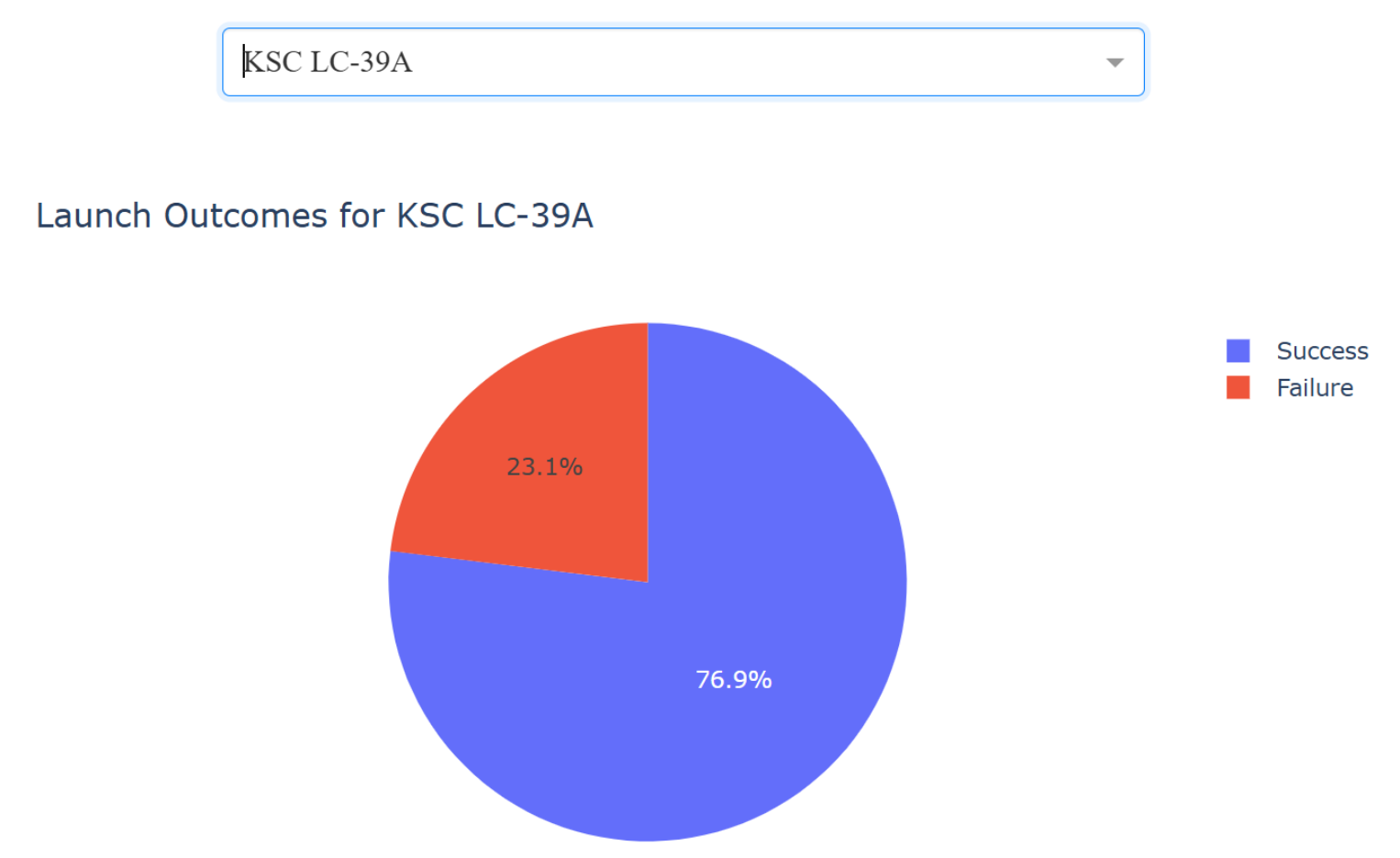
Dashboard Success Pie Chart for All Launches

The image shows a screenshot of a dashboard detail developed as part of this project. The chart shows the percentage of successful launches per site.



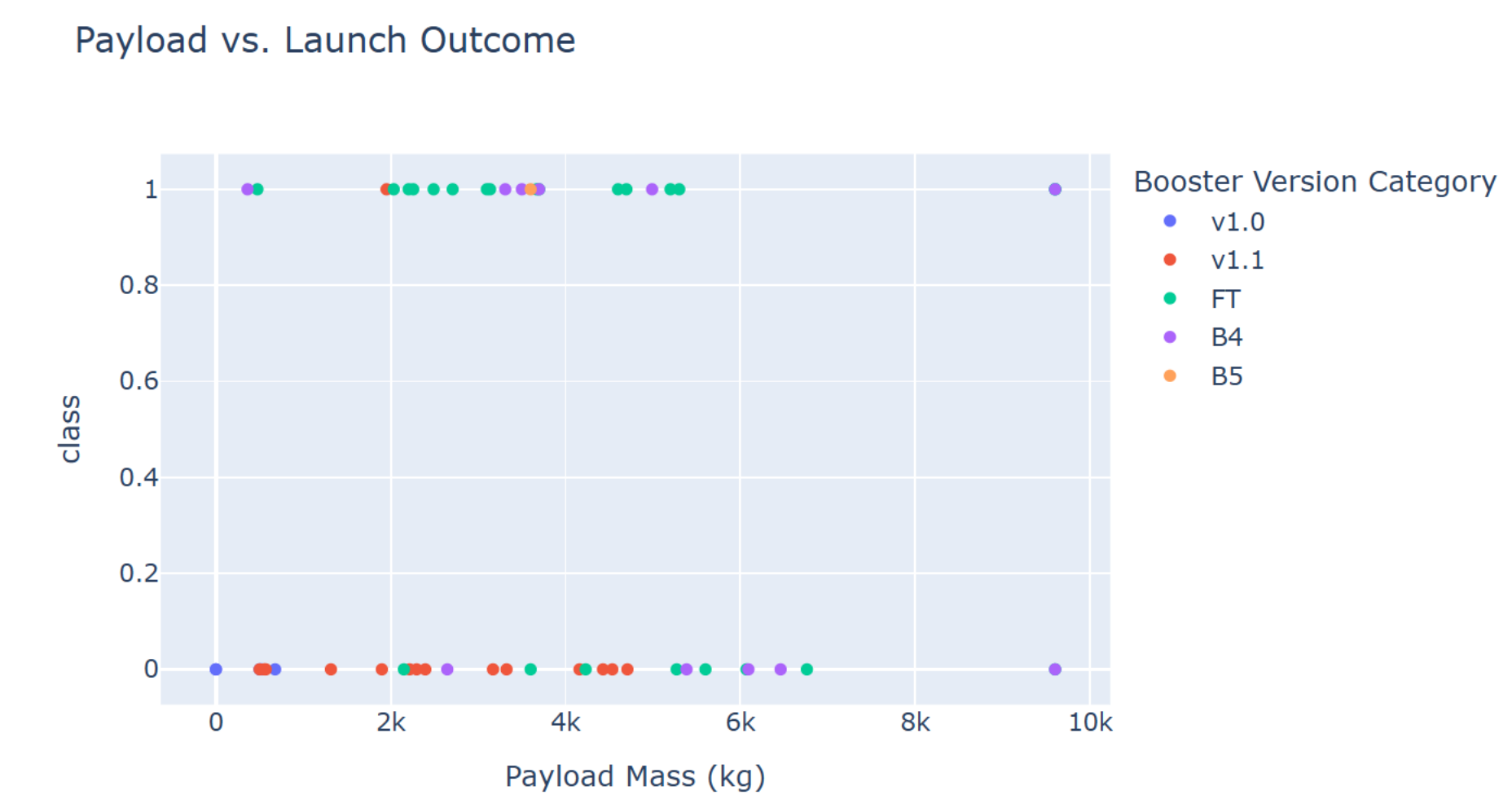
Dashboard Success Pie Chart for Most Successful Launch Site

This dashboard detail screenshot shows a chart of the KSC LC-39A launch site's mission outcomes via a pie chart. With a 76.9% success rate, the KSC LC-39A site had the highest success rate among the sites studied.



Dashboard Section Showing Payload Mass vs. Launch Outcomes

This dashboard detail screenshot shows a scatter plot with outcome class (success = 1, failure = 0) vs. payload mass (kg) by booster version, colored by category.

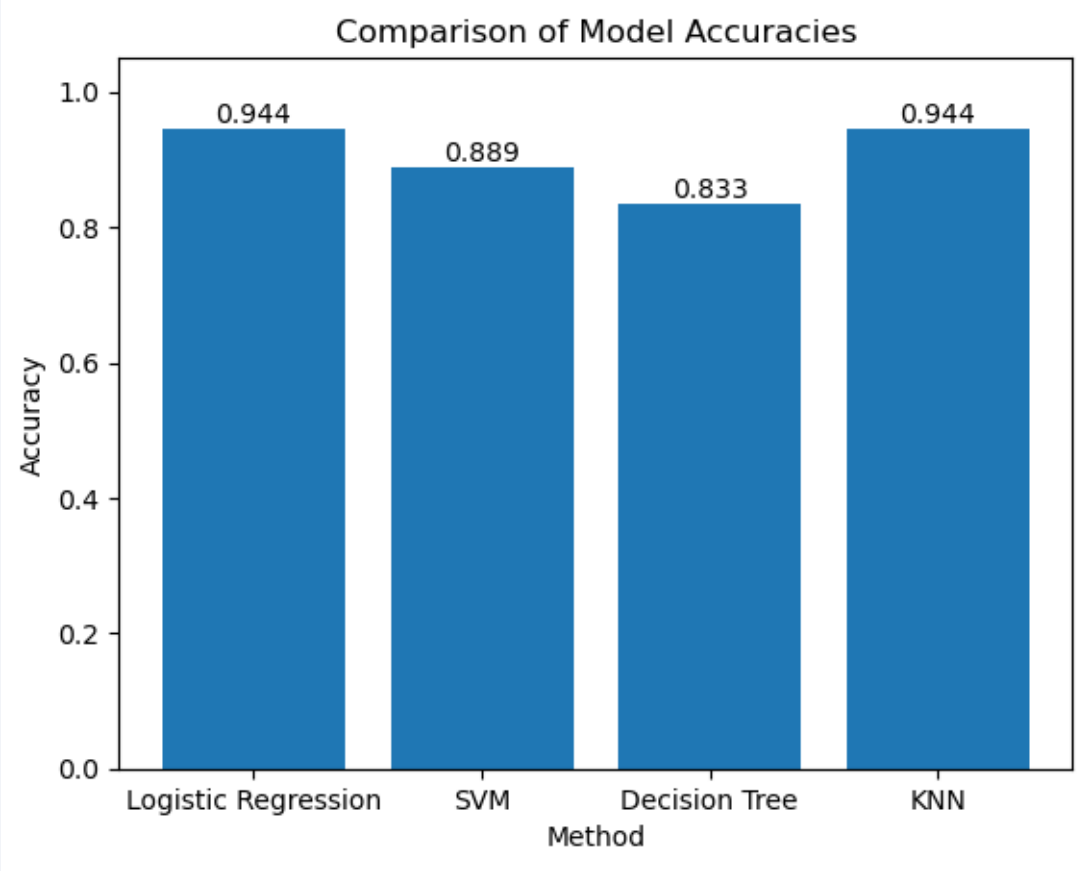


Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The Logistic Regression model tied with the KNN model, each with an accuracy of 0.944.

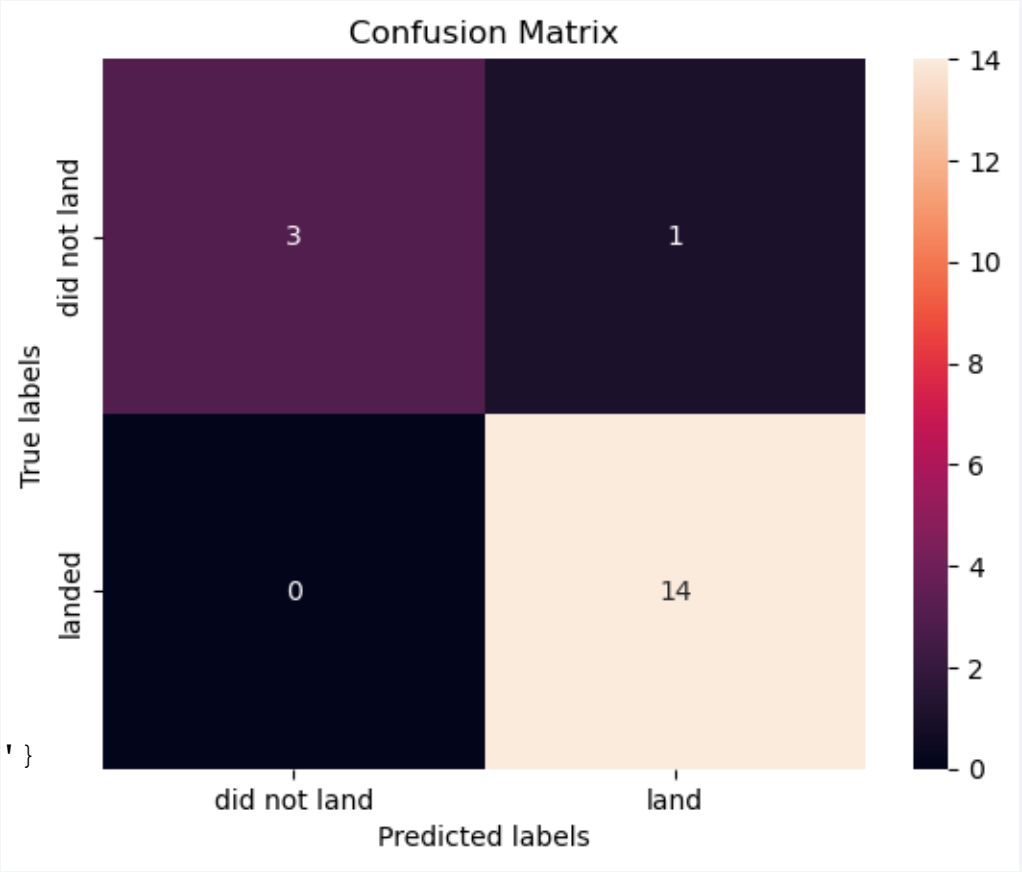


Confusion Matrix

- The confusion matrix for the best-performing model, Logistic Regression, shows a test accuracy of 0.944, using the best hyperparameter tuning derived for the model via cross-validation.
- The classification report for this model showed an overall best score of 0.803 via the best hyperparameter tuning.

class	precision	recall	f1-score	support
0	1.00	0.75	0.86	4
1	0.93	1.00	0.97	14
accuracy			0.94	18
macro avg	0.97	0.88	0.91	18
weighted avg	0.95	0.94	0.94	18

```
best_score_ (cv): 0.8035714285714285
best_params_: {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
```



Conclusions

- This project, especially in its thoroughness, offered the learners an excellent opportunity to clarify and utilize the wide range of material covered in the course.
- The fundamental study question regarding supporting price setting in a bid for booster rockets is a tantalizing topic and, based on my prior experience in industry, is on point with how analytics is used to support critical decisions by non-technically oriented senior staff.
- The test set used via the recommended 80/20 split is very small (18 samples). Switching to a 70/30 split didn't add many additional test cases (6 additional samples): Likely due to the constraint of a small test set, the results indicate:
  - Accuracy is coarse: each single prediction changes accuracy by ~1/18 (~5.6%).
  - Different models can easily produce the same accuracy by agreeing on the majority of easy cases. In fact, three of mine originals did. That dropped to two when I changed the random seed number—evidence of how frail small sample sets can be.
  - Measured test accuracy is noisy; I used cross-validation, bootstrap confidence intervals, and additional metrics to better assess performance. What I achieved was helpful in interpreting the results: this being evidence that the bootstrapping at n=2000 and even n=10000 showed wide confidence intervals for accuracy, with the 95% CI for the KNN model running 0.6667 to 1.0, which is very wide given a bootstrap of n=5000.

Appendix

- All of the code developed to produce this report is available via my GitHub account.
- The landing page for this project is: https://github.com/smithmum/final-project
- All of the notebooks can be downloaded as a zip file via:
  https://github.com/smithmum/final-project/archive/refs/heads/main.zip
- Each of the notebooks and scripts is in a folder named to identify the project phase:

EDA with SQL: https://github.com/smithmum/final-project/tree/main/EDA_with_SQL
EDA with dataviz: https://github.com/smithmum/final-project/tree/main/EDA_with_dataviz
Dashboard with Plotly & Dash: https://github.com/smithmum/final-project/tree/main/dashboard_with_Plotly_Dash
Data collection with scraping: https://github.com/smithmum/final-project/tree/main/data_Collection-web_scraping
Data collection with API: https://github.com/smithmum/final-project/tree/main/data_collection-SpaceX_API
Data wrangling: https://github.com/smithmum/final-project/tree/main/data_wrangling
Map with Folium: https://github.com/smithmum/final-project/tree/main/interactive_map_with_Folium
Capstone report PDF: https://github.com/smithmum/final-project/tree/main/ds-capstone-coursera-kbs.pdf

Thank you!