IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

OUTLINE

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

EXECUTIVE SUMMARY

Summary of Methodologies

- **Data collection:** Pulled launch/landing data from the SpaceX API and scraped supplemental tables (e.g., orbit, landing type/site) from Wikipedia.
- **Data wrangling:** Cleaned/typed fields, parsed dates, handled missing values, one-hot encoded categories, and engineered the **landing_success** label.
- **EDA (visual & SQL):** Computed site/orbit/year success rates and payload stats (SQL); explored temporal trends and payload×orbit patterns (matplotlib/plotly).
- **Interactive analytics:** Built a **Folium** map (sites colored by outcome + proximity overlays) and a **Plotly Dash** app (site pies, payload-vs-success scatter).
- **Predictive modeling:** Compared **Logistic Regression, SVM, k-NN,** and **Decision Tree** with cross-validation; evaluated on a held-out test set using **Accuracy, F1**, and confusion matrices.

EXECUTIVE SUMMARY

Summary of all Results

- **Data coverage:** 2010-06-04 to 2020-11-05
- **Launch counts:** CCAFS LC-40 = 26; CCAFS SLC-40 = 7; KSC LC-39A = 13; VAFB SLC-4E = 10
- **Key EDA insights:** success rate increases over time; varies by orbit and payload; launch sites are coastal and set back from major cities
- **Interactive outputs:** Folium outcome-colored maps; Plotly Dash pies and payload vs outcome scatter
- **Best model:** Logistic Regression
- **Test accuracy:** 0.778
- **Test F1:** 0.846
- **Confusion matrix (LogReg):** TN = 3, FP = 3, FN = 1, TP = 11 (rows = Actual, columns = Predicted; success = landed)

INTRODUCTION: PROJECT BACKGROUND AND CONTEXT

- SpaceX Falcon 9 first-stage landings enable reusability, which drives down launch costs and improves cadence.
- The dataset (as provided by the course) covers Falcon 9 missions from 2010-06-04 to 2020-11-05, with engineered features ready for EDA and modeling.
- Prior labs built the pipeline: data collection (SpaceX API + Wikipedia), wrangling/encoding, SQL + visual EDA, and interactive analytics (Folium map, Plotly Dash).
- This capstone focuses on turning those artifacts into a concise analysis and a predictive model for landing outcomes.

INTRODUCTION: PROBLEMS YOU WANT TO FIND ANSWERS TO

- **Which factors most influence landing success?** (e.g., payload mass, booster flights/reuse, orbit, launch site)
- **How does landing success vary by site and orbit over time?**
- **Given mission characteristics, can we reliably predict a successful landing?**
- **What trade-offs exist (precision vs. recall) when classifying success vs. failure?**
- **How can interactive tools (map/dashboard) help operations quickly assess mission risk and drivers?**

Section 1

Methodology

Data Collection Methodology:

- Pulled launch/landing records from the SpaceX REST API (JSON), including mission date, payload, orbit, site, and landing outcome.
- **Scraped Wikipedia** tables to supplement/verify orbit, landing type/site details when missing in the API.
- Stored raw pulls with timestamps; normalized nested JSON to tabular format (one row per launch).

METHODOLOGY

Perform Data Wrangling:

- **Cleaning**: fixed types, parsed dates, standardized site/orbit names, handled missing/unknowns.
- **Feature engineering**: created the binary label (landing_success); derived numeric features (e.g., payload mass) and selected categorical drivers (orbit, launch site).
- **Encoding & readiness**: one-hot encoded categoricals; preserved column names for traceability; saved the engineered dataset (course-provided dataset_part_3.csv

METHODOLOGY

Perform Exploratory Data Analysis (EDA) Using Visualization and SQL:

- **SQL EDA:** computed success rates by year, orbit, and launch site; aggregated payload statistics; validated counts.
- **Visual EDA:** trend lines for success rate by year; bar charts by orbit/site; payload vs outcome patterns; site distribution counts.

Perform interactive visual analytics using Folium and Plotly Dash:

- **Folium map:** plotted launch pads with outcome coloring; added proximity context (nearest coast, road, rail, city).
- **Plotly Dash:** interactive filters (site, payload range, orbit); pie charts (success share by site) and scatter (payload vs outcome).

Perform Predictive Analysis Using Classification Models

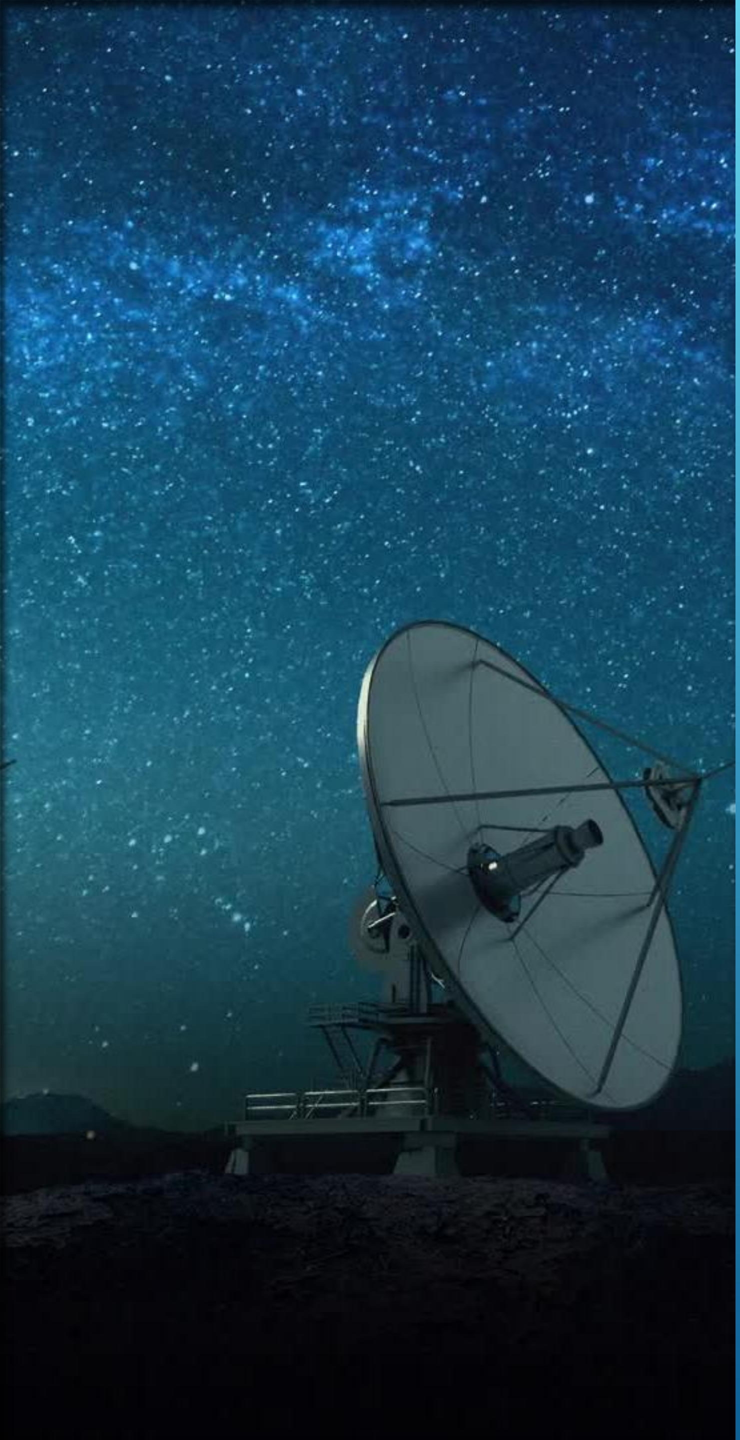
- **Models compared**: Logistic Regression, SVM, k-NN, Decision Tree
- **Data split:** stratified train/test; no leakage (transformations fit only on train).

METHODOLOGY

How To Build, Tune, Evaluate Classification Models:

- **Preprocessing**: scale numeric features (StandardScaler), one-hot encode categoricals (OneHotEncoder) via a ColumnTransformer inside a Pipeline.
- **Tuning**: GridSearchCV with cross-validation (k-fold) optimizing F1; consistent grids per model.
- **Evaluation**: report F1 and Accuracy on the test set; show confusion matrices; (optional) report ROC AUC; document best hyperparameters.
- **Selection & interpretation:** pick best by F1; review errors (FP/FN); compute permutation importance to highlight top drivers

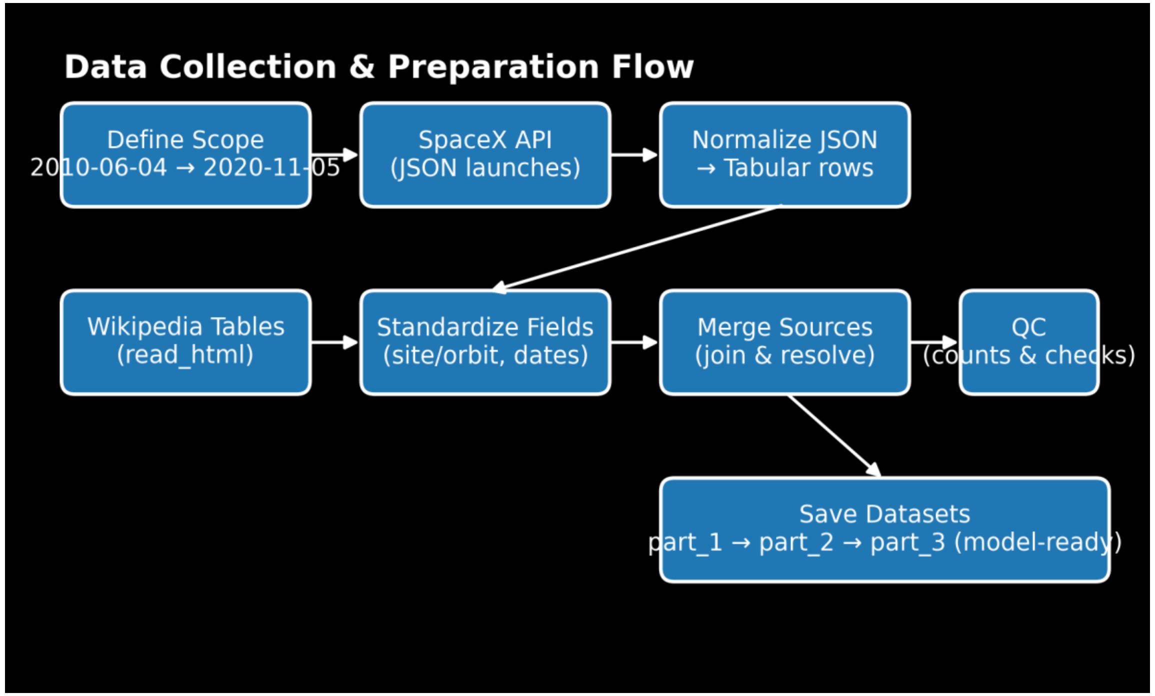
METHODOLOGY



- **Primary source (API):** Queried the SpaceX REST API for launch records (mission date, payload, orbit, launch site, landing outcome).
- **Supplemental source (web tables)**: Scraped Wikipedia launch/landing tables to fill or verify orbit and landing-type/site details.
- **Normalization:** Flattened nested JSON → tabular rows (one row per launch); standardized site/orbit names and date formats.
- **Versioned artifacts:** Saved intermediate outputs (e.g., dataset_part_1.csv → raw/normalized; dataset_part_2.csv → cleaned/merged; dataset_part_3.csv → model-ready).Quality checks: Row counts by year/site; cross-check API vs. Wikipedia; spot-check sample missions.

DATA COLLECTION

DATA COLLECTION FLOW CHART



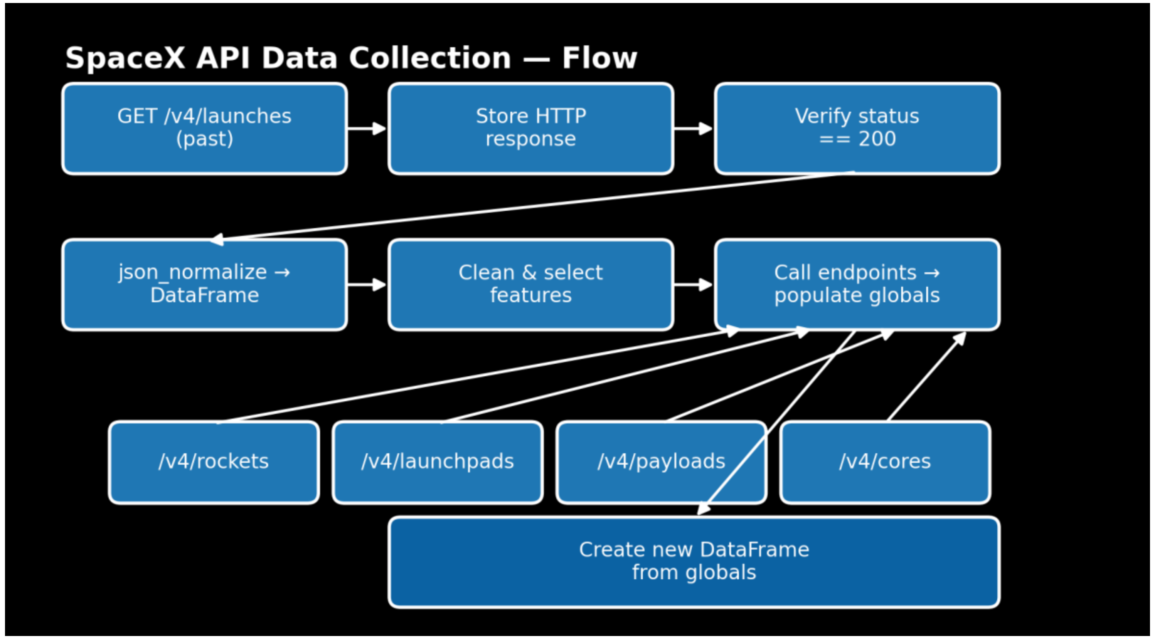
DATA COLLECTION – SPACEX API

Workflow

- Use GET request to get the launches in the past – We used static data in the exercise to make the data consistent though we also hit SpaceX API as an exercise
- Store the response in a response object
- Verify the response had a 200 status code
- Use Pandas function .json_normalize on the responses json to turn the response into a dataframe
- We proceeded to clean the data to keep the features we wanted
- Used custom functions to hit the following endpoints to populate global variables:
  - /V4/rockets
  - /V4/launchpads
  - /v4/payloads
  - /v4/cores
- Create A Custom Dataframe using global variables

Github URL: [Jupyter Notebook]

DATA COLLECTION – SPACEX API FLOW CHART

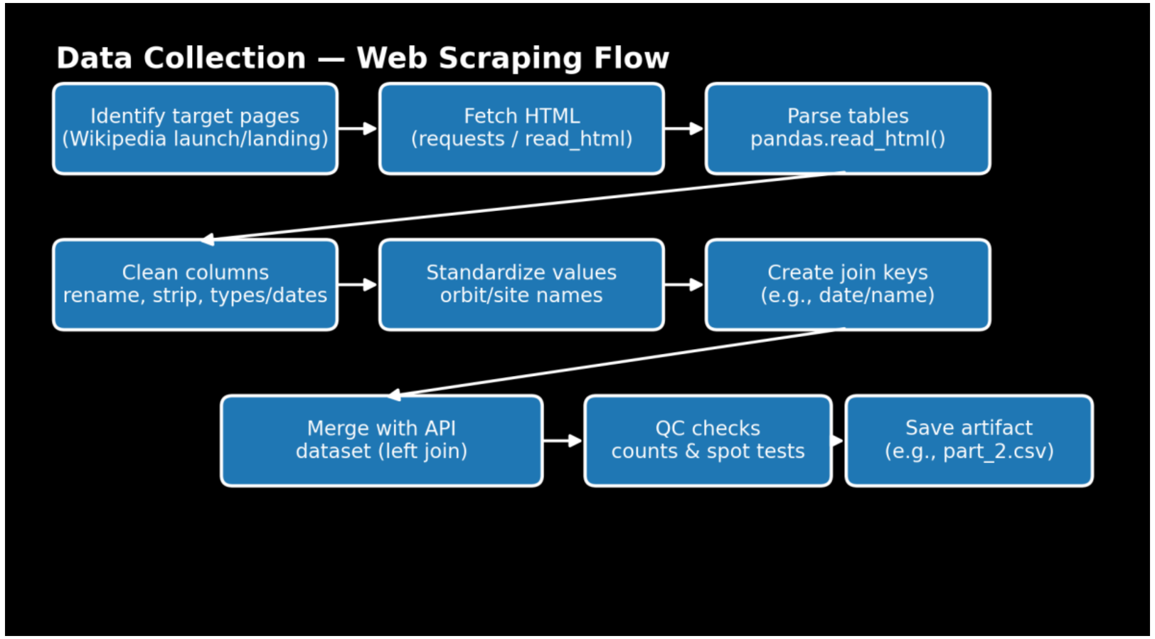


DATA COLLECTION - SCRAPING

- **Target pages**: Wikipedia launch/landing tables for Falcon 9.
- **Fetch**: requests or pandas.read_html() to load HTML tables.
- **Parse**: pandas.read_html() → list of DataFrames; select relevant table(s).
- **Clean**: rename columns, strip whitespace, fix data types and dates.
- **Standardize**: normalize orbit and launch site names to match API fields.
- **Keys**: create join keys (e.g., date/name) for merging.
- **Merge**: left-join with the API dataset; resolve conflicts.QC: validate counts by year/site; spot-check missions.

Github URL: Jupyter Notebook

DATA COLLECTION – WEB SCRAPPING FLOW CHART



DATA WRANGLING

- **Unify sources**: merge the SpaceX API table with the scraped Wikipedia tables (orbit / landing details) on date / mission keys; resolve naming conflicts.
- **Clean fields:** strip/standardize text; normalize launch site and orbit names; parse date to datetime; coerce payload mass to numeric.
- **Handle missing:** fill or flag unknowns (e.g., “NA/Unknown”); drop rows that lack essential fields after merge.
- **Engineer target:** create the binary landing_success label used for modeling (success = 1, otherwise 0).
- **Select features for modeling:** keep core drivers (e.g., PayloadMass(kg), Flights/FlightNumber, Orbit, LaunchSite); drop IDs, free text, and post-event columns used only for EDA.
- **Encode & scale:** one-hot encode categoricals (Orbit_, LaunchSite_), keep numerics as floats; scale numerics later in the model pipeline.
- **Quality checks:** verify row counts by year/site, inspect sample joins, confirm value ranges (e.g., payload mass) and unique levels (orbits/sites).

Github URL: Jupyter Notebook

EDA WITH DATA VISUALIZATION

EDA with Data Visualizations - What we plotted & why

- **Success rate by year (line chart):** to see the overall trend and learning curve over time.
- **Success rate by orbit (bar chart):** to compare outcomes across mission profiles (e.g., GTO vs LEO/SSO).
- **Launch counts by site (bar chart):** to understand site usage and provide context for success comparisons.
- **Folium launch-site map (outcome-colored):** to show geographic distribution of sites and visually encode outcomes.
- **Folium proximity overlay (city/road/rail/coast):** to highlight environmental and logistical context around a selected pad.
- **Dash pie (success share by site):** to quickly compare success proportion across sites interactively.
- **Dash scatter (payload mass vs. outcome with slider):** to inspect how payload relates to landing success and enable "what-if" filtering.

Github URL: [Jupyter Notebook]

EDA WITH SQL

EDA with SQL - What we queried & why

- **Success rate by launch site:** compare outcomes across pads to spot site-level differences.
- **Success rate by orbit:** assess how mission profile (LEO/SSO/GTO/…) relates to landing outcomes.
- **Success trend by year:** check improvement over time (learning curve, hardware maturity).
- **Payload statistics by orbit:** understand payload differences that may affect success probability.
- **Launch distribution by site (counts):** provide context/sample sizes for comparisons.
  - Result: CCAFS LC-40 = 26; CCAFS SLC-40 = 7; KSC LC-39A = 13; VAFB SLC-4E = 10.
- **Date coverage (min/max):** define analysis scope and completeness.
  - Result: 2010-06-04 to 2020-11-05.
- **Takeaways:** SQL aggregates confirmed visual EDA—success improves over time, varies by orbit and site, and payload mass distributions differ across orbits.

Github URL: [Jupyter Notebook]

BUILD AN INTERACTIVE MAP WITH FOLIUM

What we built:

- **Outcome-colored launch pads:** markers for CCAFS LC-40, CCAFS SLC-40, KSC LC-39A, and VAFB SLC-4E colored by landing outcome (success/fail).
- **Popups & tooltips:** quick view of site name, date(s), orbit(s), payload mass, and outcome counts.
- **Proximity context (overlay):** lines/labels to nearest coast, road, rail, and city for a selected site (e.g., CCAFS LC-40).
- **Base tiles:** clean basemap (e.g., OpenStreetMap) for geographic context; zoom controls enabled.

Why we used it:

- **Geospatial insight:** shows that all pads are coastal and generally set back from major cities (noise/safety rationale).
- **Operational context:** proximity to roads/rail highlights logistics constraints (e.g., limited rail near CCAFS pads).
- **Exploratory speed:** hover/click to connect place → mission attributes → outcomes without flipping between tables.
- **Storytelling:** pairs naturally with EDA charts and Dash snapshots to triangulate site, payload, and orbit effects.

Key Takeaways:

- All four launch sites are near the coast and away from major urban centers.
- CCAFS LC-40 and CCAFS SLC-40 are separate pads on the same barrier island; rail access in this area is limited.
- The geography supports EDA findings: site usage differs, and environment/logistics likely contribute to outcome variability.

Github URL: [Jupyter Notebook]

BUILD A DASHBOARD WITH PLOTLY DASH

What we built

- **Single-page Dash app** to explore Falcon 9 landing outcomes interactively.
- **Live filters for Launch Site and Payload range** (slider) to update all charts together.

Key components

- **Pie - All Sites:** success share across all pads (high-level snapshot).
- **Pie - Selected/Best Site:** success share for the chosen pad (site-level drill-down).
- **Scatter - Payload vs. Outcome:** points show missions; filter by payload to see how mass relates to landing success.

Why we used it

- **Fast "what-if" analysis:** change site/payload and instantly see how success rates shift.
- **Bridges EDA ↔ modeling:** helps verify patterns the classifier exploits (e.g., payload and site effects).
- **Audience-friendly:** clean visuals for non-technical stakeholders; interactive exploration during reviews.

Notable insights from our dashboard

- Site matters: success share differs by pad (confirming EDA counts and rates).
- Payload matters: heavier payloads show lower success bands in several orbits.
- Filter synergy: combining site + payload slider highlights pockets where the landing probability changes most.

Github URL: Python File (Note a Notebook)

PREDICTIVE ANALYSIS (CLASSIFICATION)

Summary of methodologies

- **Framing:** Predict Falcon 9 first-stage landing success using the feature set from the lab (including operational indicators such as Legs and LandingPad).
- **Features & prep:** Numeric features standardized; categorical features one-hot encoded per the lab; no manual feature removal beyond the lab steps.
- **Train/test split:** Stratified 80/20.
- **Models compared:** Logistic Regression, SVM, k-NN, Decision Tree, each tuned with cross-validation (lab grids).
- **Evaluation:** Test-set Accuracy and F1, plus confusion matrices; ROC-AUC where applicable.

Github URL: Jupyter Notebook

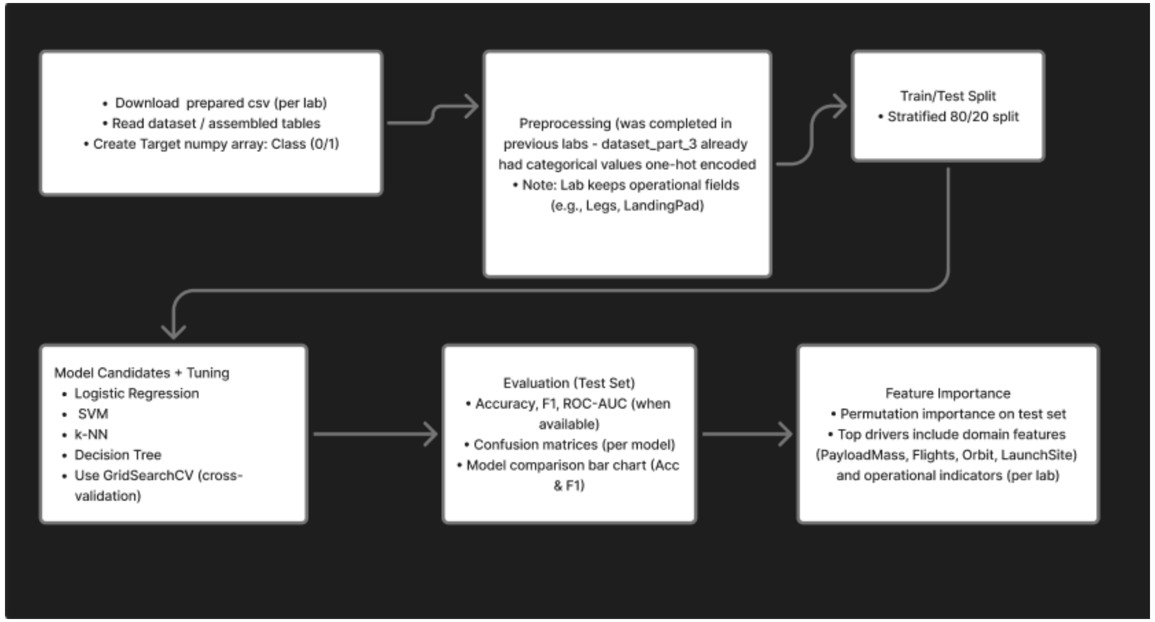
PREDICTIVE ANALYSIS (CLASSIFICATION)

Summary of all results

- **Best model (by F1):** Logistic Regression — Accuracy 0.778, F1 0.846 (test).
- **Confusion matrix (Success = landed):** TN=3, FP=3, FN=1, TP=11 ⇒ high recall for successes; some failures predicted as successes.
- **Model comparison:** Several models showed identical test accuracy on this small test set; F1 distinguishes performance.
- **Feature importance (as trained in the lab):** High importance surfaces on operational/proxy features (e.g., Legs_True, LandingPad_*) along with domain drivers (e.g., Orbit_*, LaunchSite_*, PayloadMass, Flights).
- **Interpretation:** these operational fields correlate strongly with outcomes in the dataset and can dominate importance—this is expected given the lab's feature set.
- Github URL: Jupyter Notebook

PREDICTIVE ANALYSIS

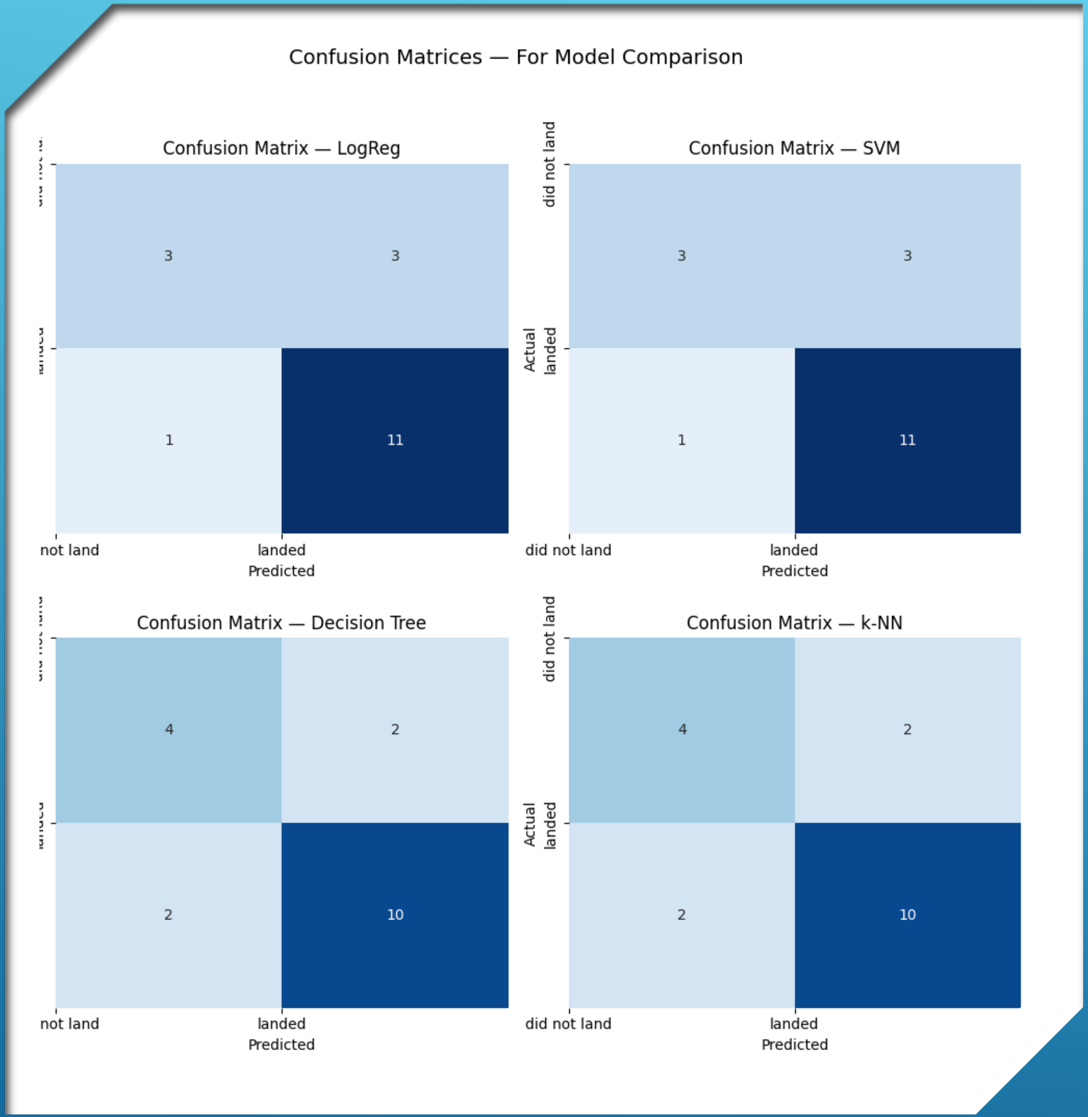
MODEL CREATION FLOWCHART



RESULTS
EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis

- **Launch distribution:** CCAFS LC-40: 26, CCAFS SLC-40: 7, KSC LC-39A: 13, VAFB SLC-4E: 10.
- **Trend:** Falcon 9 success rate increases over time (see "Success Rate by Year").
- **By orbit:** Success rates vary by orbit (e.g., some orbits show lower success) — consistent with mission profile difficulty.
- **Payload relation:** Higher payload masses tend to reduce success probability in some orbits (see payload vs outcome scatter).
- **Geospatial context:** Launch pads are coastal and set back from cities; two pads share the same island (CCAFS LC-40 & SLC-40). (Folium map + proximity table).



RESULTS

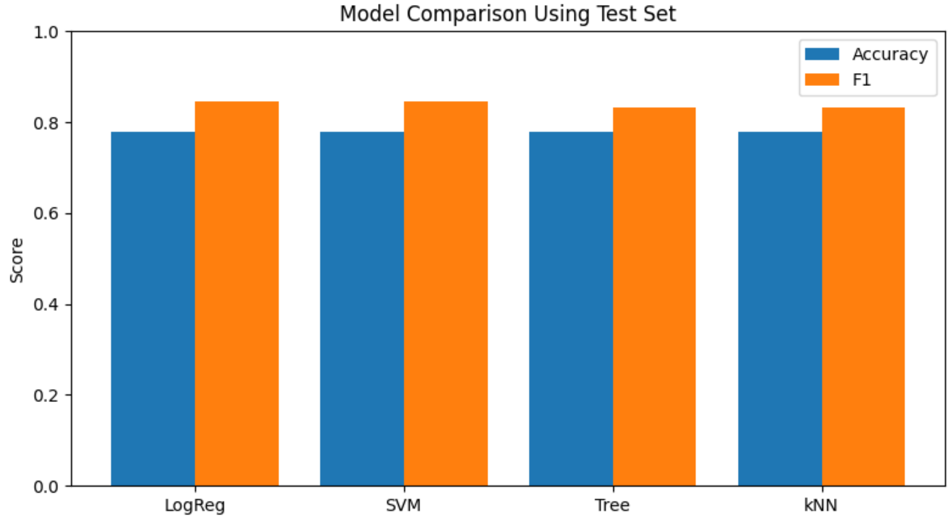
CONFUSION MATRIX FOR MODEL COMPARISON

RESULTS

MODEL COMPARISON FOR F1 AND ACCURACY

	model	test_accuracy	test_f1
0	LogReg	0.777778	0.846154
1	SVM	0.777778	0.846154
2	Tree	0.777778	0.833333
3	kNN	0.777778	0.833333

Best model by F1: LogReg | Accuracy=0.778, F1=0.846



RESULTS

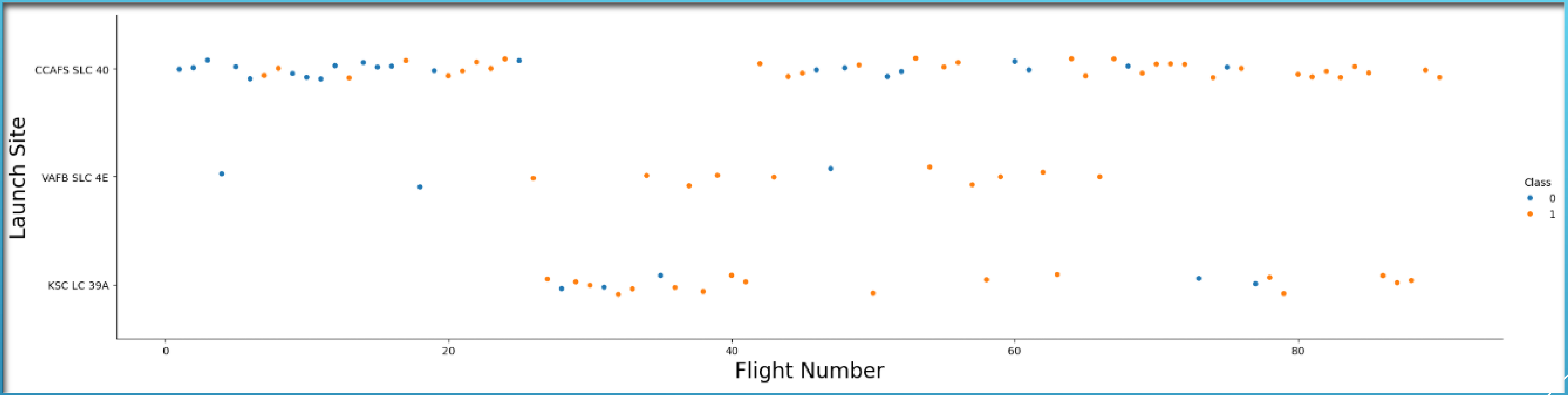
FEATURE IMPORTANCE MEAN AND STD

	feature	importance_mean	importance_std
82	Legs_True	0.006154	0.015689
81	Legs_False	0.006154	0.015689
78	GridFins_True	0.006154	0.015689
77	GridFins_False	0.006154	0.015689
48	Serial_B1028	0.000000	0.000000
54	Serial_B1035	0.000000	0.000000
53	Serial_B1034	0.000000	0.000000
52	Serial_B1032	0.000000	0.000000
51	Serial_B1031	0.000000	0.000000
50	Serial_B1030	0.000000	0.000000

Section 2

Insights drawn from EDA

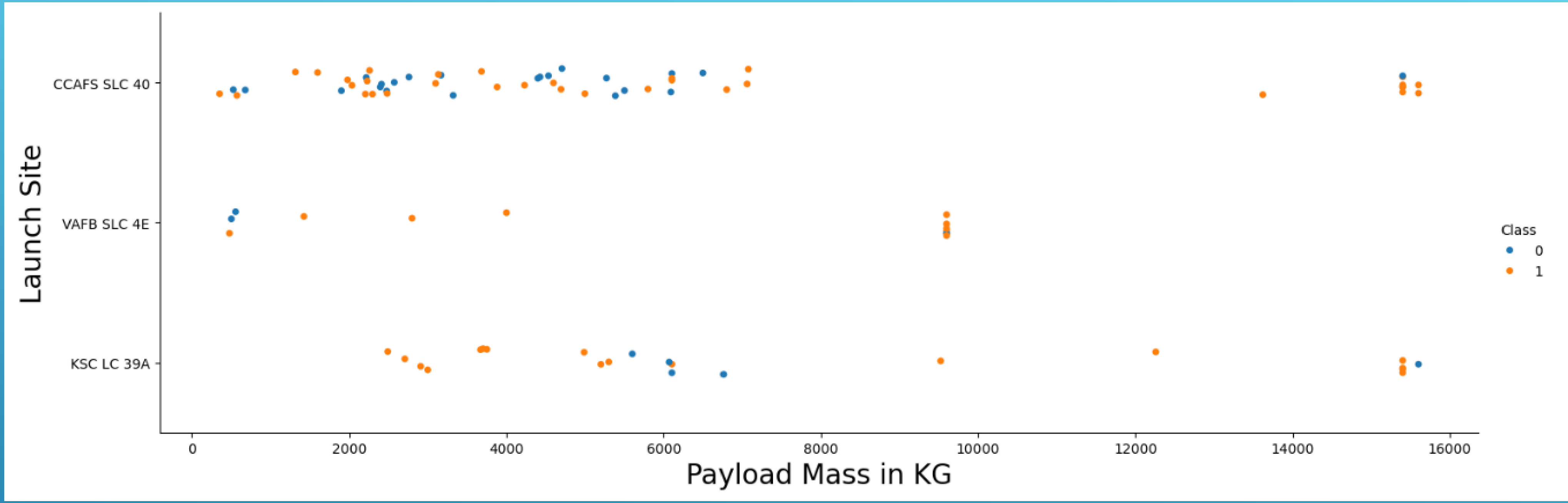
FLIGHT NUMBER VS. LAUNCH SITE



Observations:

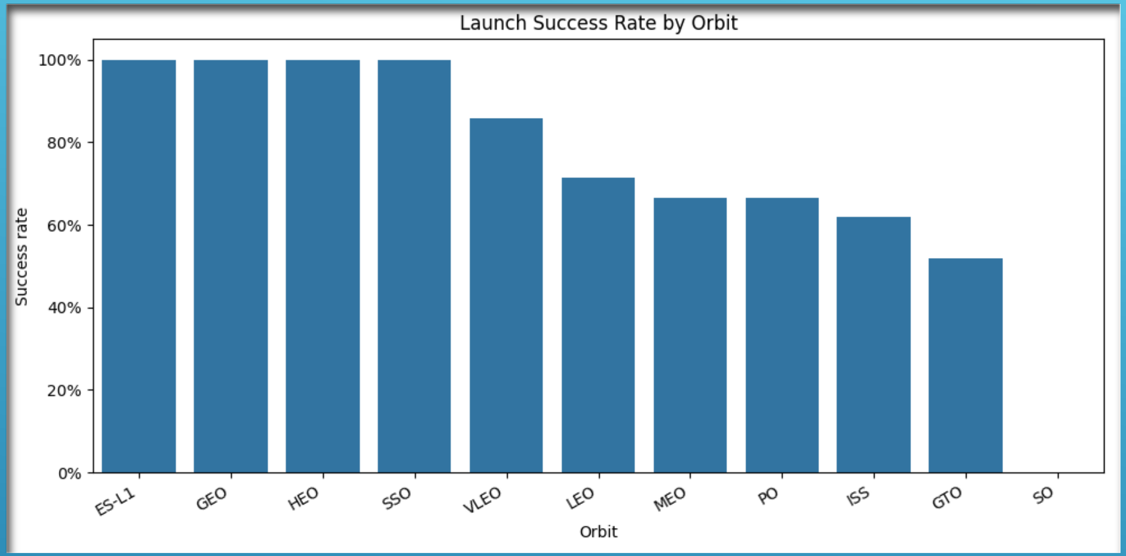
- KSC LC 39A has most failures earlier
- CCAFS SLC 40 has the most flights
- Most failures are in earlier flights

PAYLOAD VS. LAUNCH SITE



Observations:

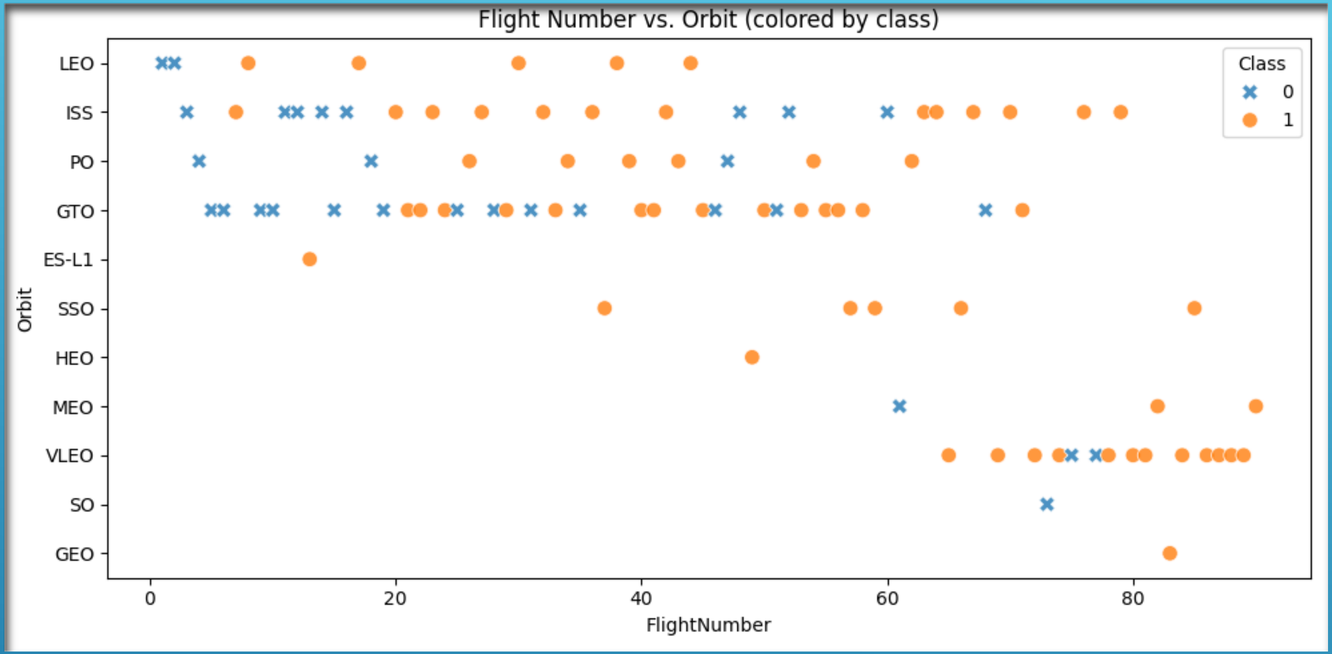
- VAFB SLC 4E has not launched any rockets over 10k kg
- Most of the Payload Mass is < 8k kg
- Successes are higher in > 8k kg



Observations:

- ES-L1, GEO, HEO, SSO have 100% Success Rate
- SO has the lowest success rate

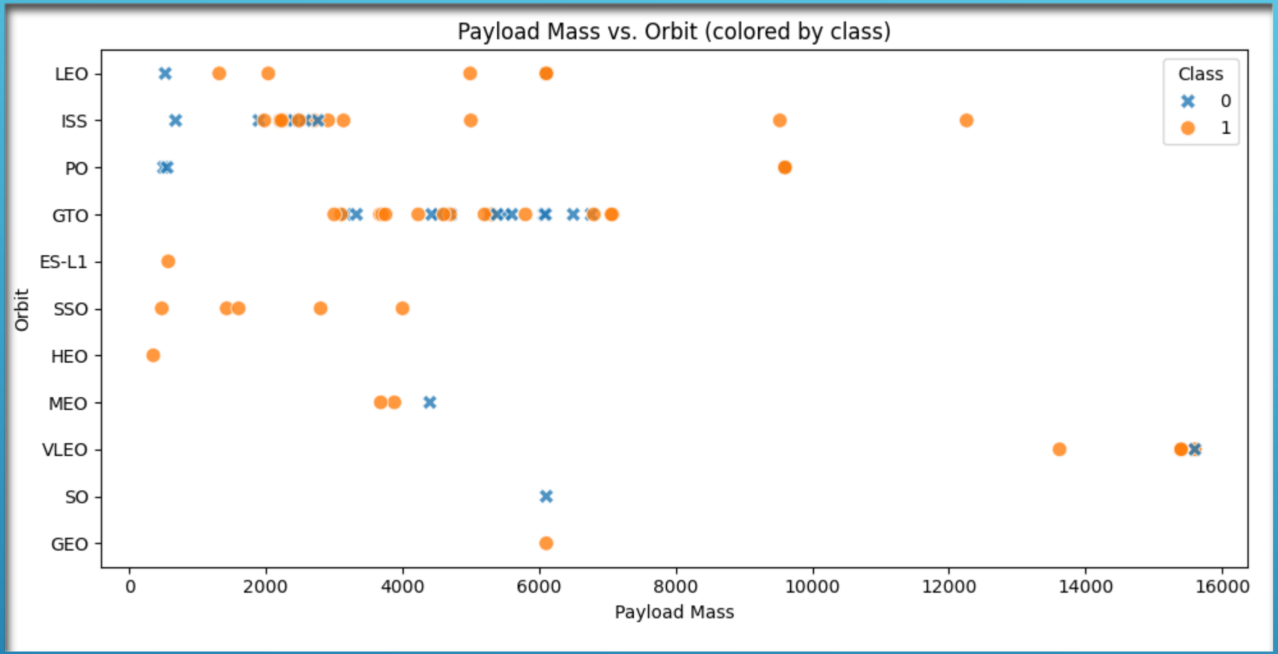
SUCCESS RATE VS. ORBIT TYPE



Observations:

- Successes increase as flight number increases
- The four orbits with 100% success have fewer flights
- SO that had the lowest success only had a single flight

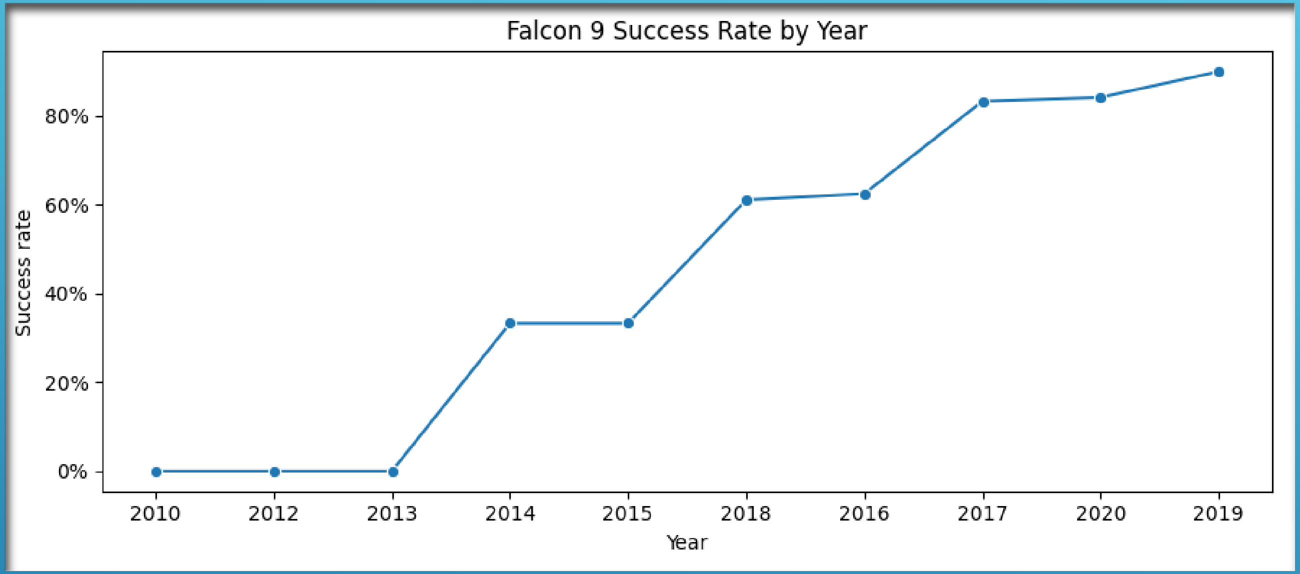
FLIGHT NUMBER VS. ORBIT TYPE



Observations:

- GTO fall between > 2K kg < 8K kg payload
- Almost all orbits are under 8k payload
- The only SO flight was a failure with 6K kg payload

PAYLOAD VS. ORBIT TYPE



Observations:

- The success rate since 2013 has kept increasing till 2020

LAUNCH SUCCESS YEARLY TREND

```
[14]: %sql SELECT DISTINCT "Launch_Site" F
```

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

[14]:

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

ALL LAUNCH SITE NAMES - SQL

- **Launch Site Names:**
  - CCAFS LC-40
  - VAFB SLC-4E
  - KSC LC-39A
  - CCAFS SLC-40
- **Query:** SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE
- Use Distinct to get the unique values in the Launch_Site column in the SPACEXTABLE

LAUNCH SITE 5 NAMES BEGIN WITH 'CCA'

- **Query:** `SELECT * FROM SPACEXTABLE WHERE "Launch_SITE" LIKE "CCA%" LIMIT 5`
  - Use LIKE keyword to filter results where Launch_SITE column starts with “CCA” the % is a wildcard.
  - LIMIT keyword limits the result set to 5 records

```
%sql SELECT * FROM SPACEXTABLE WHERE "Launch_SITE" LIKE "CCA%" LIMIT 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2

TOTAL PAYLOAD MASS

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS 'Total Payload (KG) for NASA (CRS)' FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)'
```

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

Total Payload (KG) for NASA (CRS)
45596

- **Query:** SELECT SUM(PAYLOAD_MASS__KG_) AS 'Total Payload (KG) for NASA (CRS)' FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)’
- Use SUM keyword to total results of PAYLOAD_MASS__KG_ column
- Use AS to rename result column
- Use WHERE Customer = ‘NASA (CRS) to filter results to NASA

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS 'Average Payload Mass in (KG) carried by booster F9 v1.1' FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1%'
 * sqlite:///my_data1.db
Done.
```

Average Payload Mass in (KG) carried by booster F9 v1.1
2534.6666666666665

AVERAGE PAYLOAD MASS BY F9 V1.1

- **Query:** SELECT AVG(PAYLOAD_MASS__KG_) AS 'Average Payload Mass in (KG) carried by booster F9 v1.1' FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1%’
- Use AVG keyword to average results of PAYLOAD_MASS__KG_ column
- Use AS to rename the resulting column
- Use WHERE Booster_Version LIKE ‘F9 v1.1%” to filter results to the correct booster version

FIRST SUCCESSFUL GROUND LANDING DATE

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

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2015-12-22	1:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

- **Query:** `SELECT* FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)' AND Date = ( SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)');`
- Use WHERE Landing_Outcome = 'Success (ground pad) to filter results
- Use subquery for addition WHERE Date is equal to the MIN(Date)
- Note Date was already in ISO format YYYY-MM-DD

SUCCESSFUL DRONE SHIP LANDING WITH PAYLOAD BETWEEN 4000 AND 6000

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

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

Booster_Version	PAYLOAD_MASS__KG_
F9 FT B1022	4696
F9 FT B1026	4600
F9 FT B1021.2	5300
F9 FT B1031.2	5200

- **Booster Versions**
  - **F9 FT B1022**
  - **F9 FT B1026**
  - **F9 FT B1021.2**
  - **F9 FT B1031.2**
- **Query:** `SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND CAST(PAYLOAD_MASS__KG_ AS INTEGER) > 4000 AND CAST(PAYLOAD_MASS__KG_ AS INTEGER) < 6000`
- Use WHERE Landing_Outcome = 'Success (drone ship) to filter results
- Use AND for additional filter on PAYLOAD_MASS__KG_ casting as INT as > 4000 and then another AND to filter < 6000

TOTAL NUMBER OF SUCCESSFUL AND FAILURE MISSION OUTCOMES

```
%%sql
SELECT DISTINCT(TRIM(Mission_Outcome)) AS 'Total Mission Outcomes'
FROM SPACEXTABLE;
SELECT
  SUM(CASE WHEN TRIM(Mission_Outcome) IN ('Success',
                                'Success (payload status unclear)') THEN 1 ELSE 0 END) AS 'Total Success',
  SUM(CASE WHEN TRIM(Mission_Outcome) IN ('Failure (in flight)') THEN 1 ELSE 0 END) AS 'Total Failure'
FROM SPACEXTABLE;
```

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

Total Success	Total Failure
100	1

- **Total Success: 100**
- **Total Failures: 1**
- **Query:** SELECT DISTINCT(TRIM(Mission_Outcome)) AS 'Total Mission Outcomes'FROM SPACEXTABLE;SELECT SUM(CASE WHEN TRIM(Mission_Outcome) IN ('Success', 'Success (payload status unclear)') THEN 1 ELSE 0 END) AS 'Total Success', SUM(CASE WHEN TRIM(Mission_Outcome) IN ('Failure (in flight)') THEN 1 ELSE 0 END) AS 'Total Failure'FROM SPACEXTABLE;
- Used Trim due to extra spaces in the data impacting totals for the Mission_Outcome column
- Used SUM to total columns on records that are considered successes
- Used SUM to total columns on records that are considered failures
- Used AS keyword to rename the columns to more legible values

```
%%sql SELECT DISTINCT(Booster_Version), PAYLOAD_MASS__KG_
    FROM SPACEXTABLE
    WHERE PAYLOAD_MASS__KG_ =
    (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
    ORDER BY Booster_Version
```

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

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

BOOSTERS CARRIED MAXIMUM PAYLOAD

- **Booster Versions**
  - **F9 B5 B1048.4, F9 B5 B1048.5, F9 B5 B1049.4, F9 B5 B1049.5, F9 B5 B1049.7, F9 B5 B1051.3, F9 B5 B1051.4, F9 B5 B1051.6, F9 B5 B1056.4, F9 B5 B1058.3, F9 B5 B1060.2, F9 B5 B1060.3**
- **Query:** `SELECT DISTINCT(Booster_Version), PAYLOAD_MASS__KG_ FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE) ORDER BY Booster_Version`
- Use DISTICT to get unique Booster Versions
- Use subquery to filter on the MAX PAYLOAD_MASS__KG_
- Orderd by Booster_Version for legibility

2015 LAUNCH RECORDS

THE FAILED LANDING_OUTCOMES IN DRONE SHIP, THEIR BOOSTER VERSIONS, AND LAUNCH SITE NAMES FOR IN YEAR 2015

```
%%sql
SELECT
  CASE substr(Date, 6, 2)
    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, 0, 5) = '2015' -- year = 2015
  AND TRIM(Landing_Outcome) = 'Failure (drone ship)' -- failure on drone ship
ORDER BY CAST(substr(Date, 6, 2) AS INTEGER)
```

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

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

- **Booster Versions**
  - **F9 v1.1 B1012**
  - **F9 v1.1 B1015**
- **Query:** `SELECT CASE substr(Date, 6, 2) 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_SiteFROM SPACEXTABLEWHERE substr(Date, 0, 5) = '2015' AND TRIM(Landing_Outcome) = 'Failure (drone ship)' ORDER BY CAST(substr(Date, 6, 2) AS INTEGER)`
- Use CASE and substr on the Date field to get the Month out of the ISO Date format (two month digits) to convert to the appropriate month name
- Use substr to get the Year value to compare it to 2015
- Order by the Date month number casting as an Integer

RANK LANDING OUTCOMES

BETWEEN 2010-06-04 AND 2017-03-20

```
%%sql
SELECT
  TRIM(Landing_Outcome) AS landing_outcome,
  COUNT(*)              AS outcome_count
FROM SPACEXTABLE
WHERE date(Date) BETWEEN '2010-06-04' AND '2017-03-20'
  AND Landing_Outcome IS NOT NULL
GROUP BY TRIM(Landing_Outcome)
ORDER BY outcome_count DESC, landing_outcome;
```

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

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

- **Landing Outcomes**
  - **No attempt 10**
  - **Failure (drone ship) 5**
  - **Success (drone ship) 5**
  - **Controlled (ocean) 3**
  - **Success (ground pad)3**
  - **Failure (parachute) 2**
  - **Uncontrolled (ocean) 2**
  - **Precluded (drone ship) 1**
- **Query:**

```
SELECT  TRIM(Landing_Outcome) AS landing_outcome, COUNT(*) AS outcome_count FROM SPACEXTABLEWHERE date(Date) BETWEEN '2010-06-04' AND '2017-03-20' AND Landing_Outcome IS NOT NULLGROUP BY TRIM(Landing_Outcome)ORDER BY outcome_count DESC, landing_outcome;
```

- TRIM the Landing_Outcome due to the extra spaces
- Renamed selected columns using AS to make the results more readable
- Use BETWEEN for the target dates
- Group By the Outcomes
- Use Count(*) to count the totals of the grouped Landing_Outcome

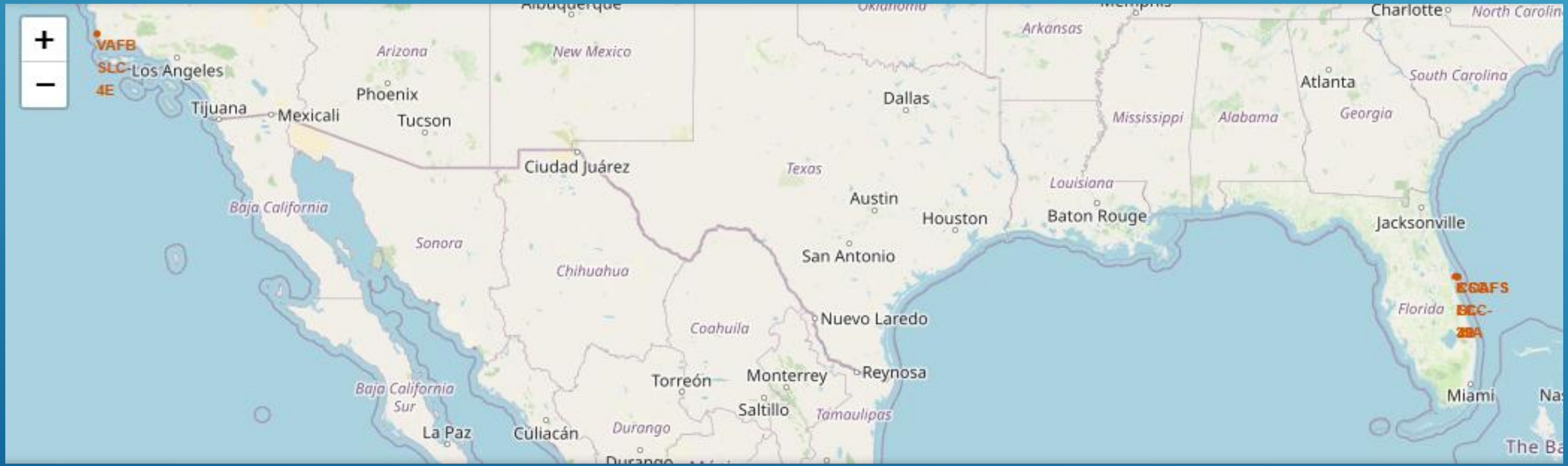
Section 3

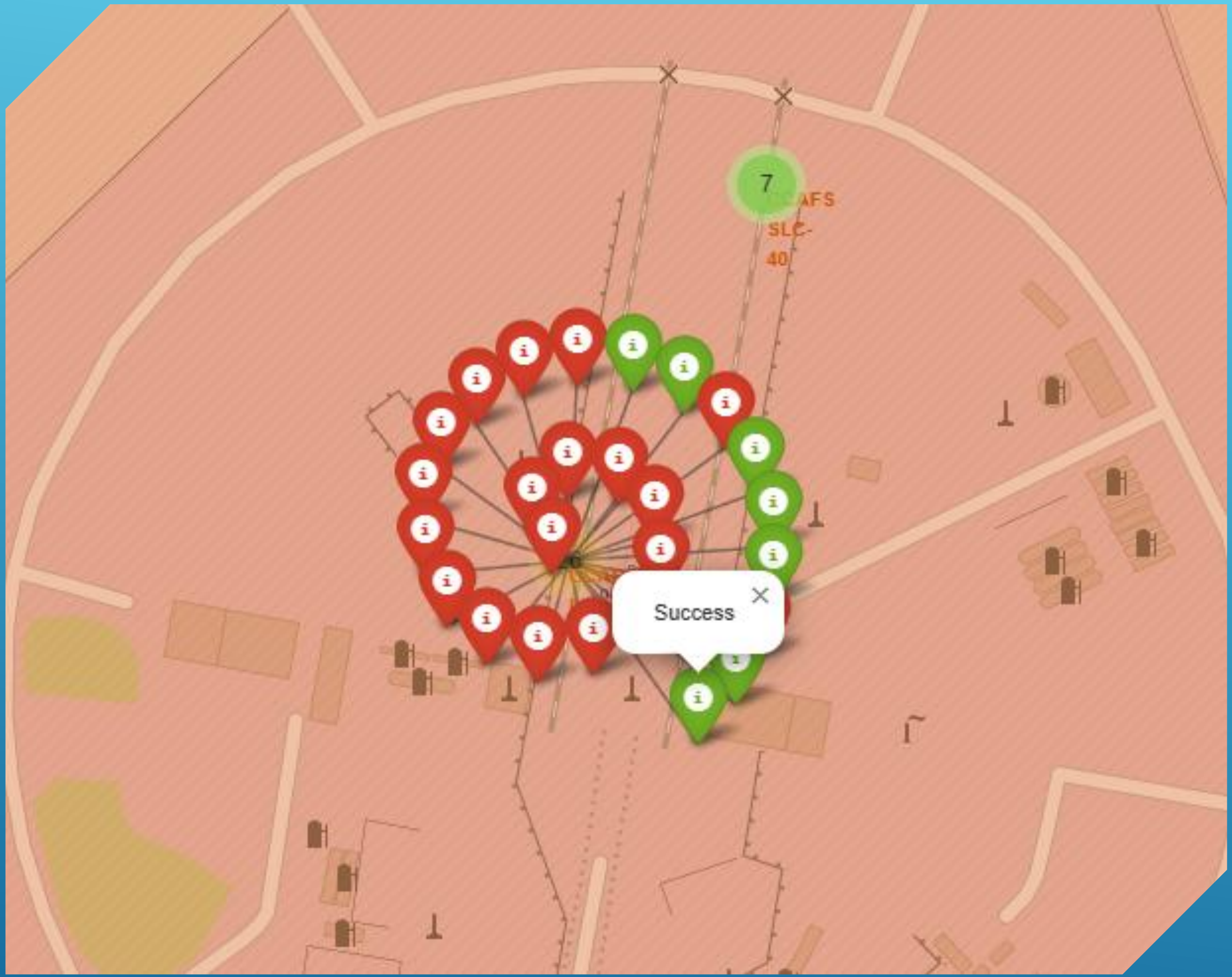
Launch Sites Proximities Analysis

FOLIUM MAP SPACEX LAUNCH ALL SITES

Important Elements and Findings

- Four total sites
- Sites are on coasts
- Sites on each coast
- Sites are in the southern U.S.
- Two sites are on the same island

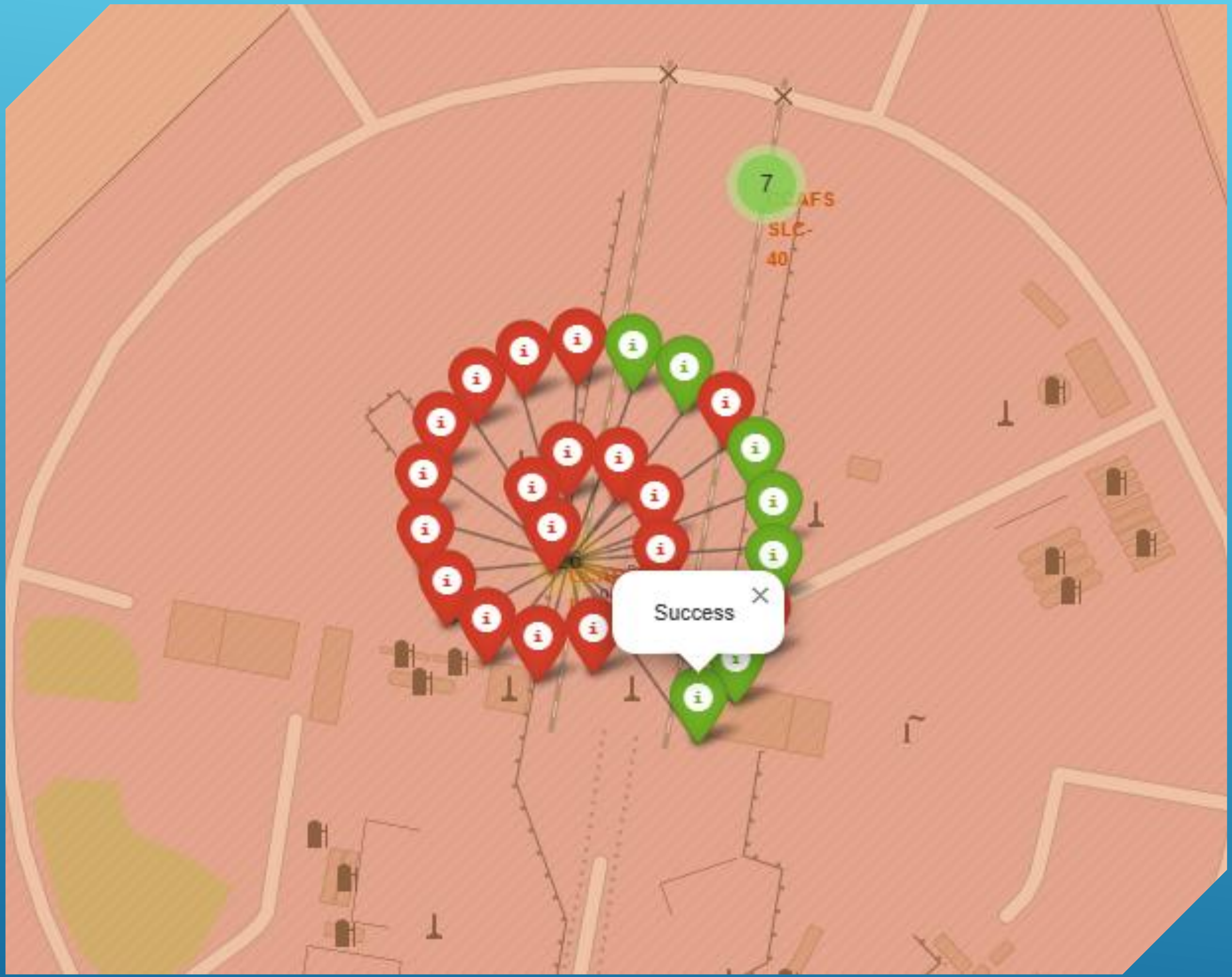




FOLIUM MAP
CCAFS SLC-40
COLOR LABELS LAUNCHES

Important Elements and Findings

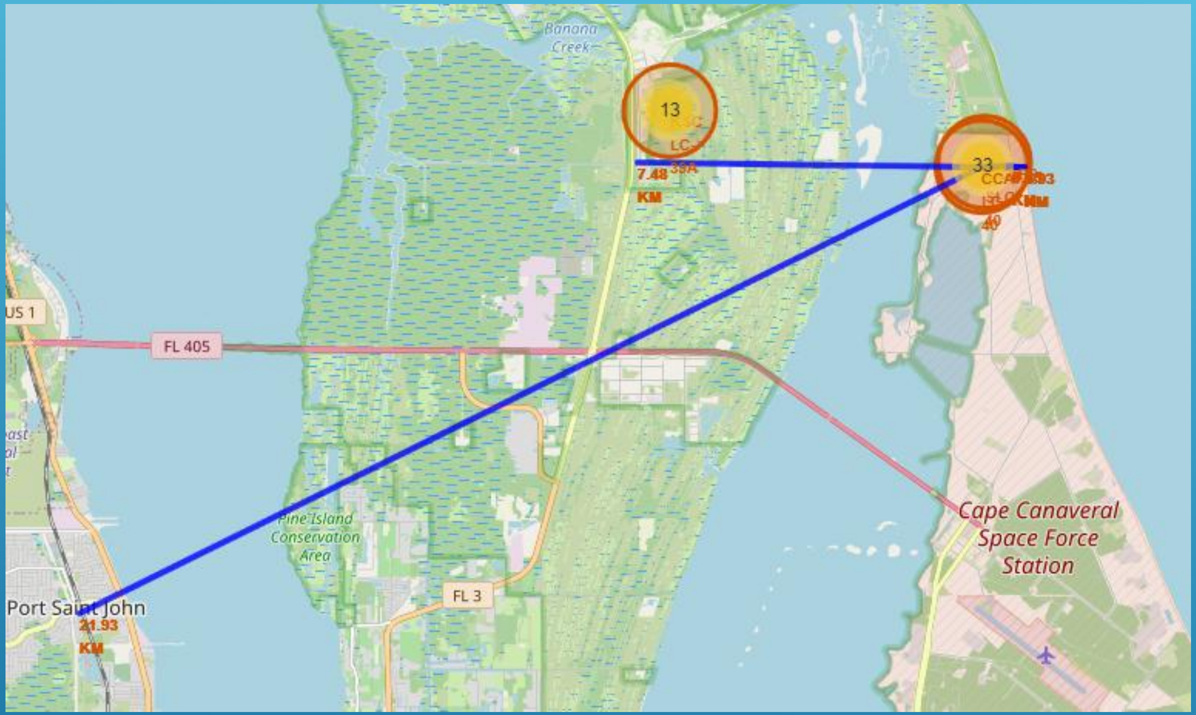
- More Failures than Success
- 19 Failures to 7 Successes



FOLIUM MAP
CCAFS SLC-40
COLOR LABELS LAUNCHES

Important Elements and Findings

- More Failures than Success
- 19 Failures to 7 Successes



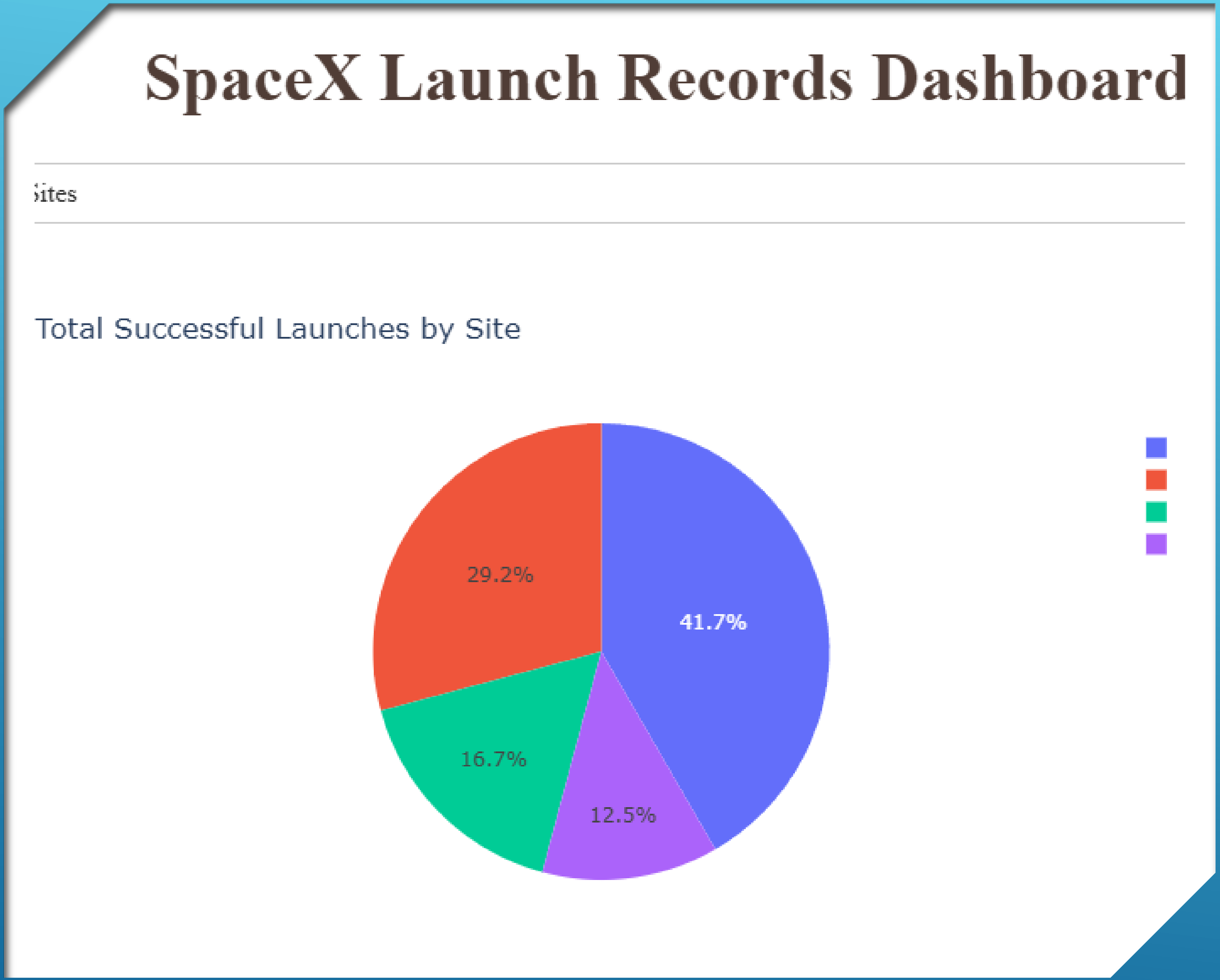
FOLIUM MAP
CCAFS SLC-40
PROXIMITY MARKERS

Important Elements and Findings

- **Coastline: 0.93 km**
- **Major City (Port Saint John): 21.93 km**
- **Rail: 7.48 km**
- **Road: 0.65 km**

Section 4

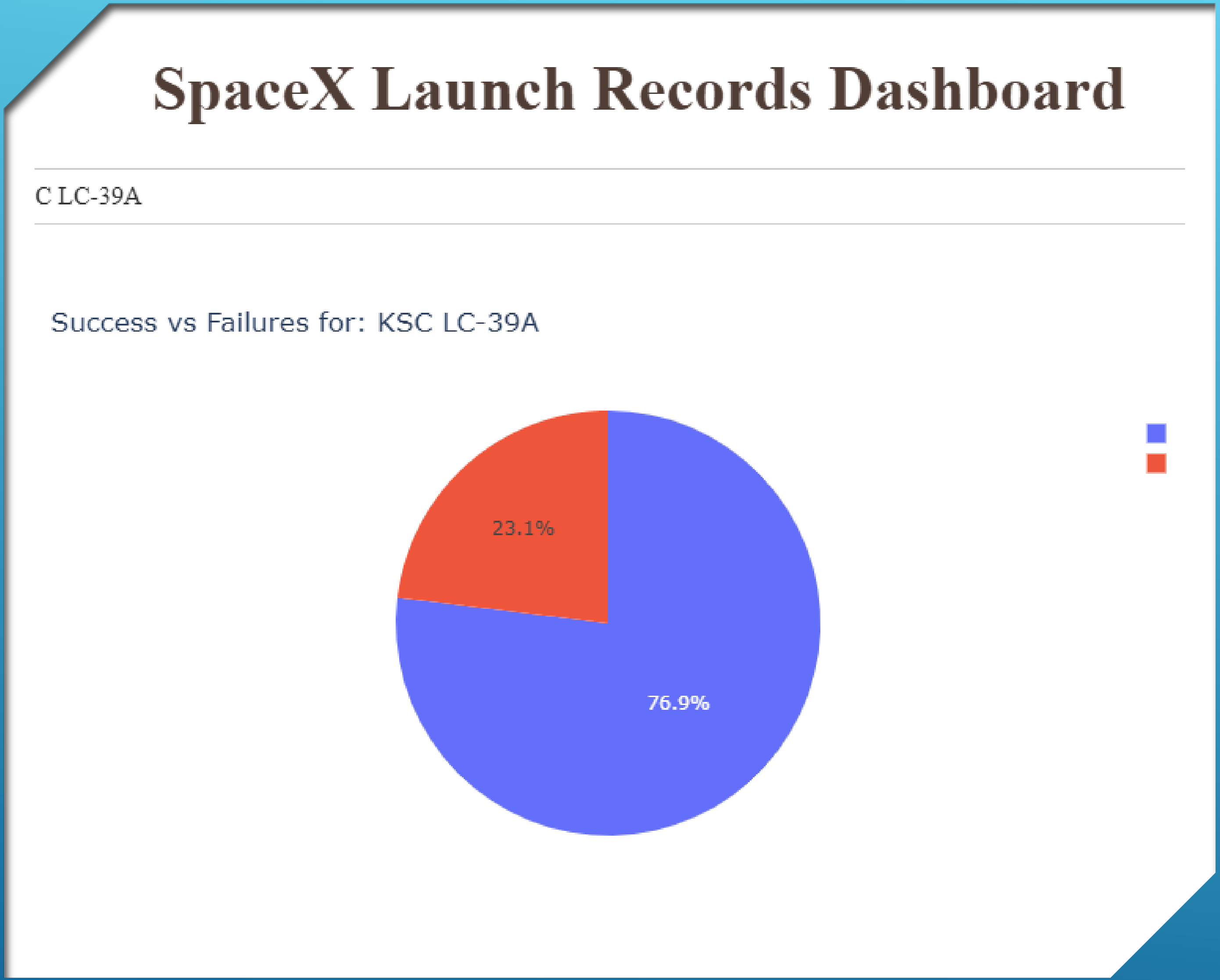
Build a Dashboard with Plotly Dash



SPACEX LAUNCH RECORDS DASHBOARD – ALL SITES

Important Elements and Findings

- Four total sites
- **KSC LC-39A:** Highest total successes at 41.7%
- **CCAFS SLC-40**: Lowest total successes at 12.5%

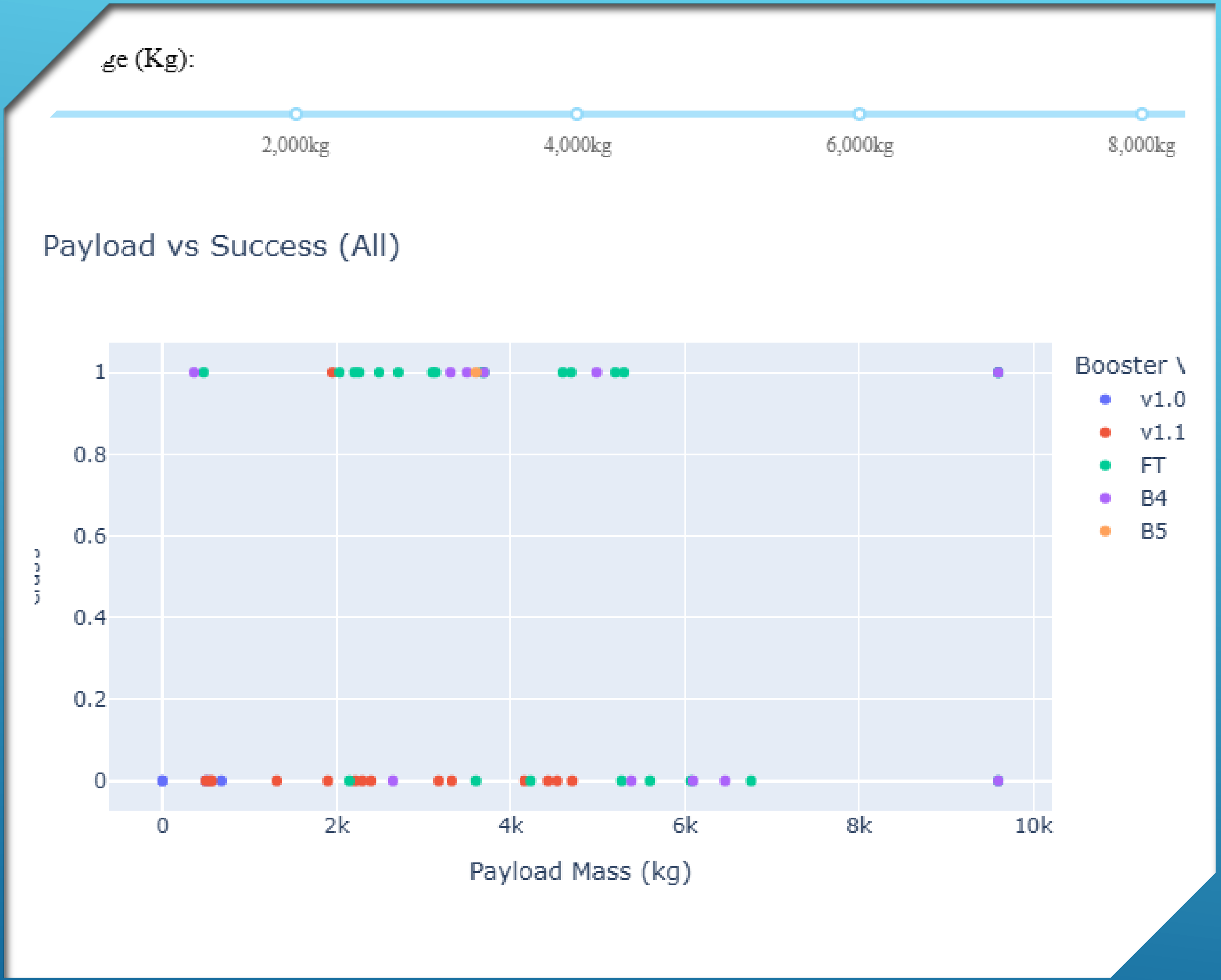


SPACEX LAUNCH RECORDS DASHBOARD – MOST SUCCESSFUL

Important Elements and Findings

- **Success:** 76.9%
- **Failure**: 23.1%

PAYLOAD VS LAUNCH OUTCOME – ALL SITES

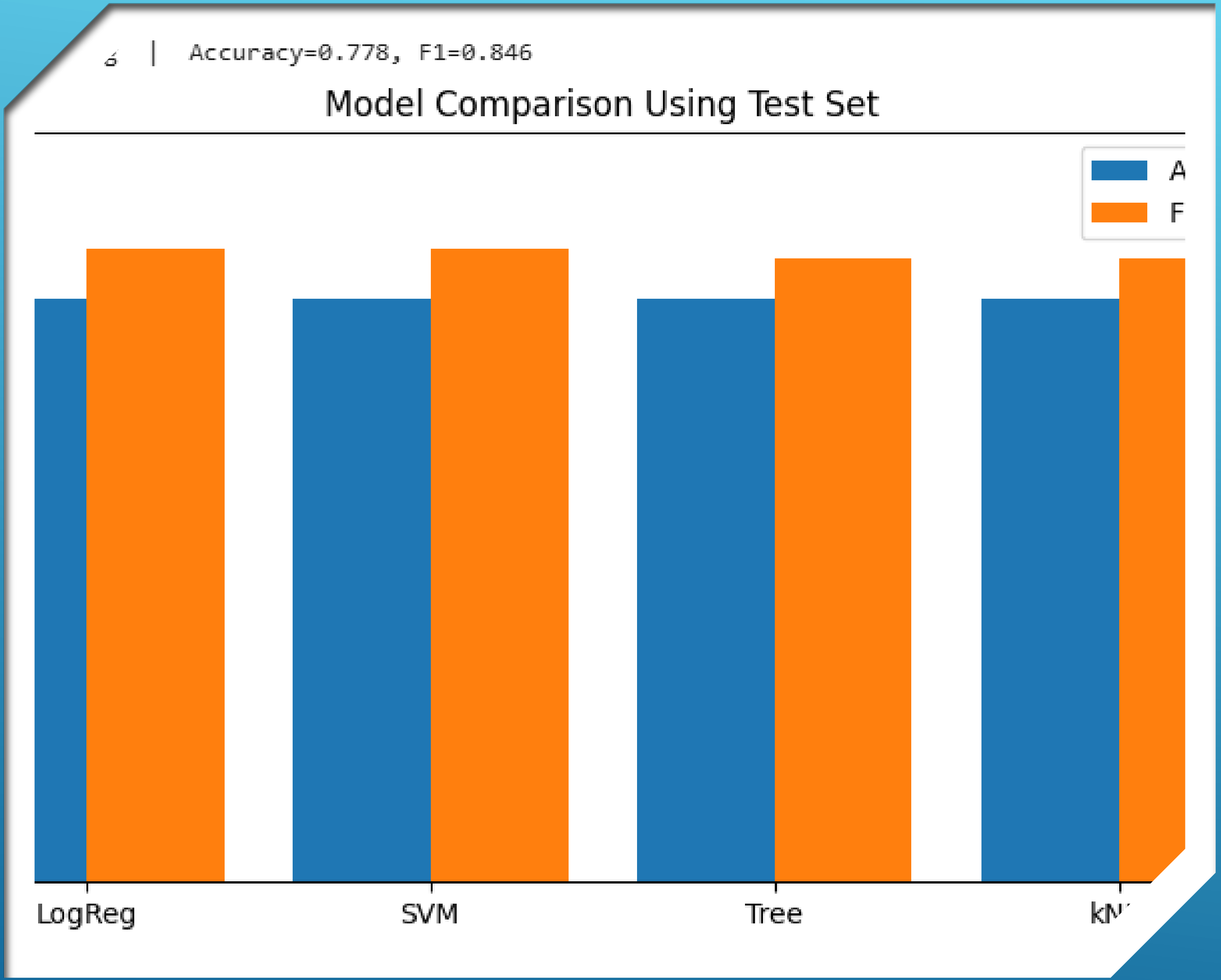


Important Elements and Findings

- Five Booster Versions Compared
- **B5:** Only 1 launch, which was a success, so the success rate is 100%
- **v1.0:** 100% failure rate
- **v1.1**: Mostly failures (1 success)
- Payload did not impact success significantly

Section 5

Predictive Analysis (Classification)

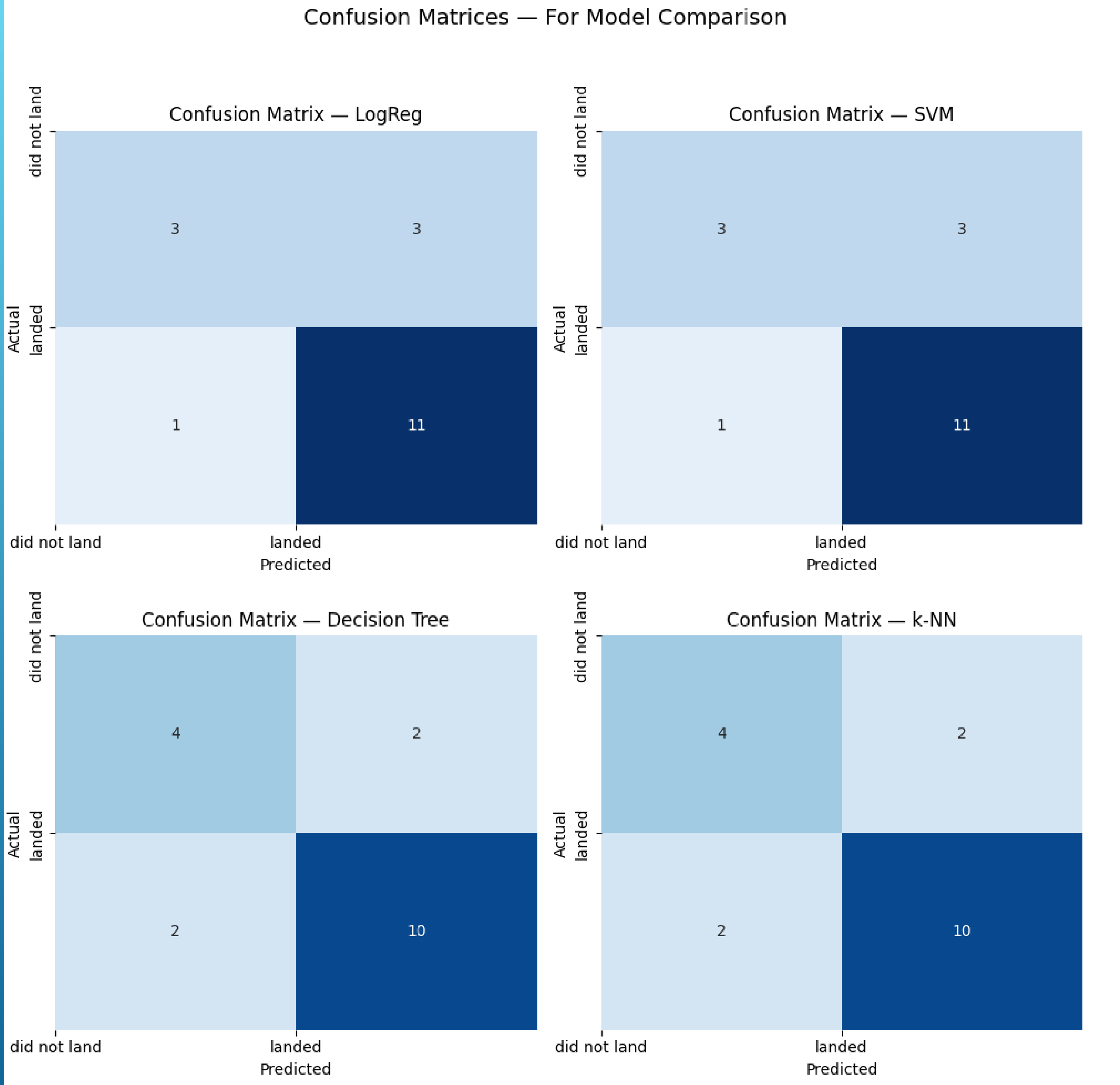


CLASSIFICATION ACCURACY

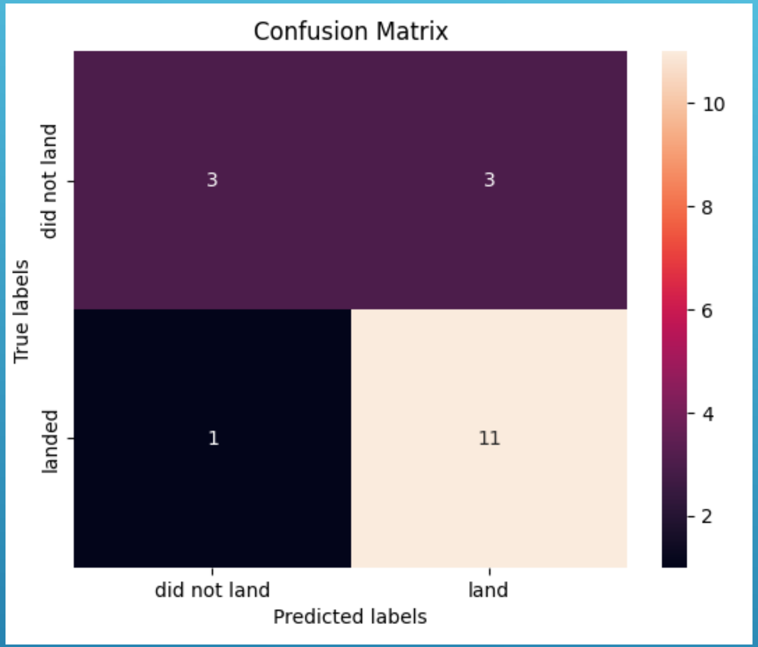
- **Best model by F1:** LogReg
- **Accuracy** = 0.778 (all models had the same test accuracy)
- **F1** = 0.846

CONFUSION MATRIX

All Models



CONFUSION MATRIX



Best Performing: LogReg

```
LogReg
Unique predictions: (array([0, 1], dtype=int64), array([ 4, 14]))
              precision    recall  f1-score   support

           0      0.750     0.500     0.600         6
           1      0.786     0.917     0.846        12

    accuracy                          0.778        18
   macro avg      0.768     0.708     0.723        18
weighted avg      0.774     0.778     0.764        18

Confusion matrix:
 [[ 3  3]
 [ 1 11]]
```

- TP (actual success, predicted success) = 11
- TN (actual fail, predicted fail) = 3
- FP (actual fail, predicted success) = 3
- FN (actual success, predicted fail) = 1

CONCLUSIONS

Key takeaways

- **End-to-end pipeline delivered:** Data ingestion (API + tables), wrangling, SQL & visual EDA, Folium map, Dash dashboard, and supervised classification.
- **Consistent EDA story:** Success rate improves over time; outcomes vary by orbit and payload; sites are coastal and set back from cities.
- **Model outcome:** Logistic Regression performed best by F1 on the test set (Accuracy 0.778, F1 0.846), with high recall for successes.
- **Feature signals (lab-trained):** Operational indicators (e.g., Legs, LandingPad) rank highly, alongside domain drivers (PayloadMass, Flights, Orbit, LaunchSite).
- **Practical implication:** Use the Dash dashboard + model outputs to inform pre-launch risk reviews and scenario planning.

Limitations & considerations

- Small test set (≈18) → identical accuracies across models are common; rely on F1 and confusion matrices for differentiation.
- Operational indicators in the lab feature set can dominate importance; this reflects correlations present in the dataset.

APPENDIX

A. Data Sources & Coverage

- **Sources:** SpaceX API; Wikipedia tables (launch/landing details).
- **Coverage:** 2010-06-04 → 2020-11-05 (post-cleaning).
- **Target:** landing_success (0 = did not land, 1 = landed).

B. Data Dictionary (key fields)

- **PayloadMass (kg):** launch payload mass (numeric).
- **Flights**: booster reuse count / flight number (numeric).
- **Orbit**: mission orbit (categorical; one-hot encoded).
- **LaunchSite:** launch pad (categorical; one-hot encoded).
- **(Lab-operational).** Legs, LandingPad, etc. retained per assignment.

C. Github Repo

- Repo Link

Thank you!