IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Adrián Valdavida
01/2026

Outline

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

Executive Summary

- In this project, I applied a full data science pipeline: data cleaning, feature engineering, and exploratory analysis, followed by training and tuning several classification models to predict Falcon 9 landing success.
- Overall, the models reached a solid test accuracy, with Logistic Regression, SVM and KNN performing similarly, while the Decision Tree tended to overfit. The analysis also revealed that launch site, orbit type, and payload play a central role in landing success

Introduction

- SpaceX revolutionized the space industry by making Falcon 9 first stages reusable, dramatically cutting launch costs from over $60 million to around $30 million per flight. This project uses historical SpaceX launch data covering flight numbers, launch sites, payload masses, orbits, and landing outcomes to analyze reusability patterns.
- The key question is: which factors most influence whether the first stage successfully lands after separation?
  Specifically, we want to determine the impact of launch site, orbit type, payload mass, and mission characteristics on landing success rates, and build predictive models to estimate future outcomes.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology: Data was collected from SpaceX records and SQL database SPACEXTBL.
  - Describe how data was collected
- Perform data wrangling: Missing values handled and categorical variables one-hot encoded for modeling.
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL: Used visualizations and SQL queries to understand launch patterns and success factors
- Perform interactive visual analytics using Folium and Plotly Dash: Created Folium maps showing launch sites and success markers by location.
- Perform predictive analysis using classification models: Built and tuned Logistic Regression, SVM, KNN, and Decision Tree classifiers.
  - How to build, tune, evaluate classification models

Data Collection

- Two main sources were used: a CSV dataset with Falcon 9 launch records and the SPACEXTBL SQL database containing detailed mission information.

- The process followed these steps:
  - Connected to SQLite database → Loaded SPACEXTBL table
  - Queried launch sites, mission outcomes, and payload data
  - Imported CSV with flight details → Merged datasets
  - Exported to pandas DataFrame for analysis"

Data Collection – SpaceX API

- After the general data collection overview, here are the specific SpaceX API calls used. I connected to the SPACEXTBL SQLite database via Python, queried key tables for mission details, and merged them with the Falcon 9 CSV dataset. The flowchart shows the exact process: database query to DataFrame, then merge. Full code and outputs are in this GitHub notebook for peer review: https://github.com/AValdavida/Test-Repo/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

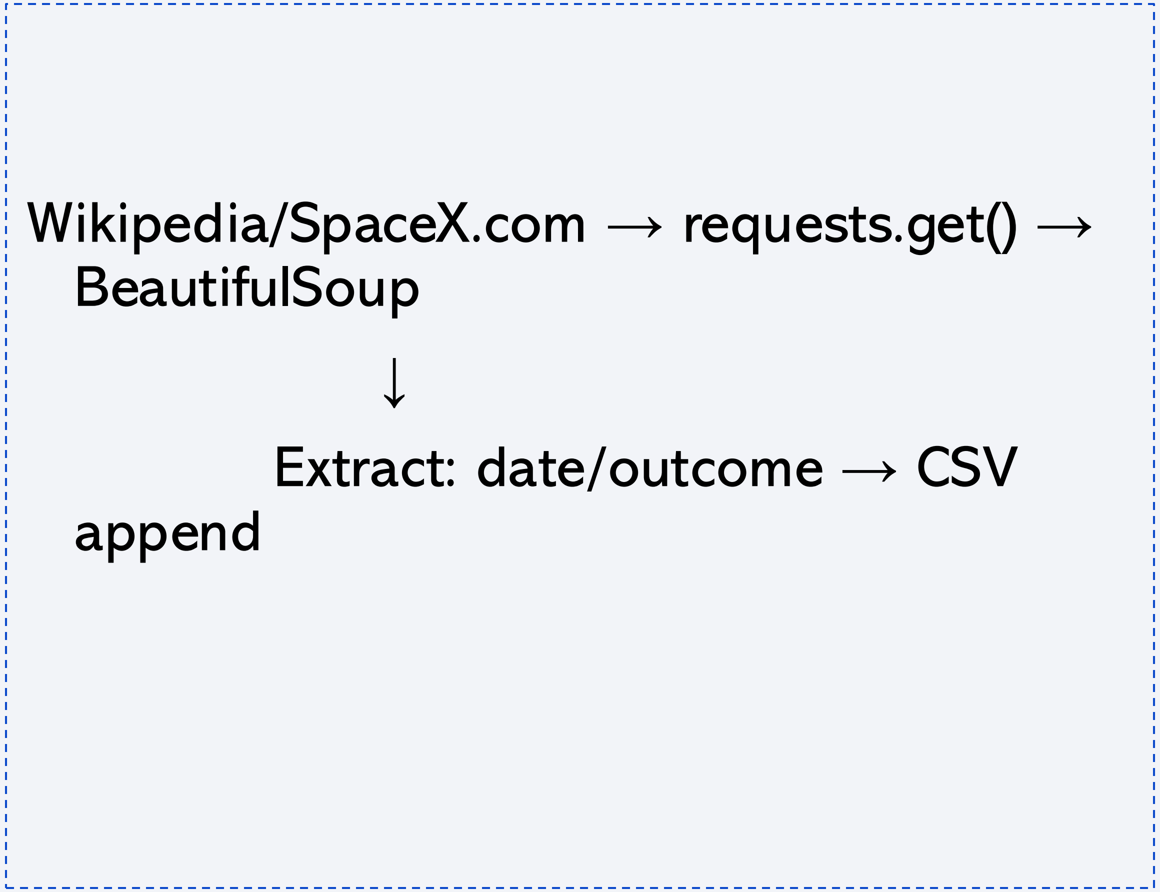
SQLite DB (SPACEXTBL) → SQL Query (pandas.read_sql) → DataFrame launches

↓

CSV Falcon9 → pd.merge() → Final Dataset

Data Collection - Scraping

- Complementing the API data, I scraped additional real-time launch information from reliable sources. Used requests.get() on Wikipedia Falcon 9 page → BeautifulSoup parsed HTML tables → Extracted launch dates, status, and outcomes → Appended to main dataset
- https://github.com/AValdavida/Test-Repo/blob/main/jupyter-labs-webscraping.ipynb



Wikipedia/SpaceX.com → requests.get() → BeautifulSoup

↓

Extract: date/outcome → CSV append

Data Wrangling

- Identified missing PAYLOAD_MASS → Filled with median → Standardized Landing_Outcome → Created binary target (1=Success/0=Failure) → One-hot encoded Launch_Site/Orbit

Raw datasets → Check NaN/missing

↓

Fill median → Parse outcomes

↓

Binary labels → pd.get_dummies()

↓

Clean DataFrame → Modeling ready

- https://github.com/AValdavida/Test-Repo/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- Folium maps: launch sites success by location (geospatial patterns); Bar charts: success rates by orbit/payload (categorical trends); Scatter plots: payload vs success (correlations); Line charts: success evolution over time (temporal patterns)

- https://github.com/AValdavida/Test-Repo/blob/main/edadataviz.ipynb

EDA with SQL

- The SQL queries i performed are:

```
•  %sql DROP TABLE IF EXISTS SPACEXTABLE;

•  %sql create table SPACEXTABLE as select * from SPACEXTBL where Date is not null

•  %sql SELECT * FROM SPACEXTABLE LIMIT (5);

•  %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;

•  %sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%'LIMIT (5);

•  %%sql SELECT SUM(PAYLOAD_MASS__KG_)

•          FROM SPACEXTABLE

•         WHERE Customer = 'NASA (CRS)';

•  %%sql SELECT AVG(PAYLOAD_MASS__KG_)

•          FROM SPACEXTABLE

•         WHERE Booster_Version = 'F9 v1.1';

•  %sql SELECT DISTINCT Landing_Outcome FROM SPACEXTABLE;

•  %%sql SELECT MIN(DATE)

•          FROM SPACEXTABLE

•         WHERE Landing_Outcome = 'Success (ground pad)';
```

```
•  %%sql SELECT Booster_Version,Payload

•          FROM SPACEXTABLE

•         WHERE PAYLOAD_MASS__KG_ BETWEEN '4000' AND '6000'

•          AND Landing_Outcome = 'Success (drone ship)';

•  %sql SELECT DISTINCT Mission_Outcome FROM SPACEXTABLE;

•  %%sql

•  SELECT

•      COUNT(CASE WHEN Mission_Outcome like '%Success%' THEN 1 END) AS
   Successes,

•      COUNT(CASE WHEN Mission_Outcome like '%fail%' THEN 1 END) AS
   Failures

•  FROM SPACEXTABLE;

•  %%sql

•  SELECT DISTINCT Booster_Version

•  FROM SPACEXTABLE

•  WHERE Payload_Mass__kg_ = (

•      SELECT MAX(Payload_Mass__kg_)
```

EDA with SQL

- The SQL queries i performed are:

```
%%sql

SELECT

    substr(Date,6,2) AS Month,

    Landing_Outcome,

    Booster_Version,

    Launch_Site

FROM SPACEXTABLE

WHERE substr(Date,0,5)='2015'

  AND (Landing_Outcome LIKE '%drone ship%'

     OR Landing_Outcome LIKE '%False drone ship%')

  AND Landing_Outcome LIKE '%fail%';
```

https://github.com/AValdavida/Test-Repo/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

```
%%sql
SELECT Landing_Outcome,
COUNT(*) as count
FROM SPACEXTABLE
WHERE substr(Date,0,11)>='2010-06-04'
AND substr(Date,0,11)<='2017-03-20'
GROUP BY Landing_Outcome
ORDER BY count DESC;
```

Build an Interactive Map with Folium

- folium.Circle (black outline, radius=1000m) + folium.Marker (DivIcon labels) for 4 sites (CCAFS LC-40/SLC-40, KSC LC-39A, VAFB SLC-4E); MarkerCluster (red=failed/green=success launches); PolyLine (red, weight=1) + DivIcon distance markers to coast/rail/highway/city.. Circles visualize launch volume by site; Markers provide GPS precision; Lines measure infrastructure accessibility (coast proximity for recovery ships, rail/highways for booster transport, population for safety buffers).
- Circles/Markers locate sites precisely with popups; MarkerCluster differentiates success (green)/failure (red) launches; PolyLines + distance labels quantify proximities revealing optimal site criteria: coast recovery, rail/highway logistics, population safety buffers.
- https://github.com/AValdavida/Test-Repo/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Pie chart (success/failure rates by Launch Site dropdown); Scatter plot (Payload Mass vs Outcome, colored by Booster Version); Range Slider (Payload 0-10000+kg); Cross-filtering via callbacks updating both charts live.
- Dropdown filters sites revealing success patterns (CCAFS SLC-40 highest rate); Slider tests payload-success correlation (optimal 4-6k kg range); Scatter shows Booster Version impact; Live updates enable hypothesis testing: site+payload+booster combinations.
- https://github.com/AValdavida/Test-Repo/blob/main/spacex_dash_app_Con_Preguntasyrespuestas.py

Predictive Analysis (Classification)

- Logistic Regression, SVM, KNN: accuracy 0.83 (consistent confusion matrices); Decision Tree: 0.66 (overfit). SVM selected as best performer due to balanced precision/recall across classes despite tied accuracy.
- Clean DataFrame → 80/20 Train/Test Split

  ↓
- 4 Classifiers → Cross-Validation

  ↓
- Confusion Matrix: [LR/SVM/KNN: 0.83 | DT: 0.66]

  ↓
- **SVM Production** (best precision on both classes)
- https://github.com/AValdavida/Test-Repo/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

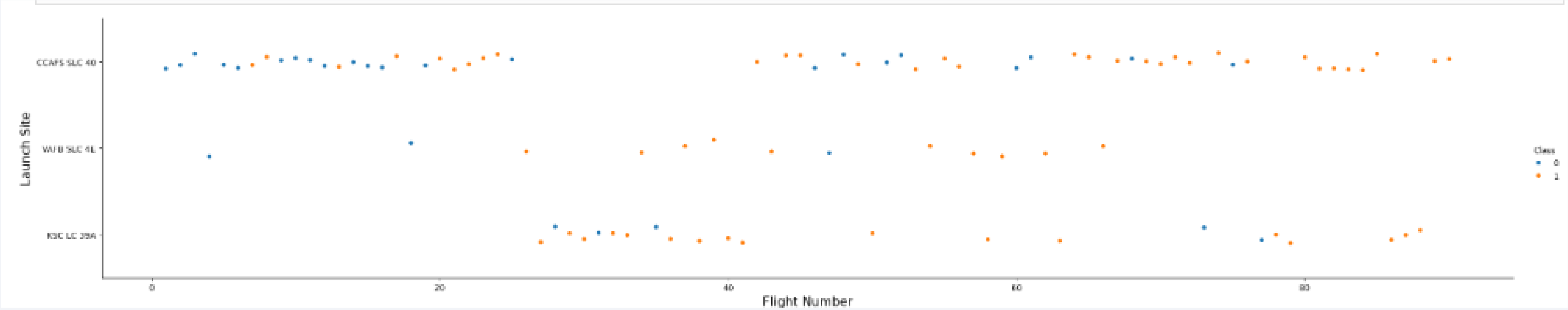
- CCAFS SLC-40 highest success rate; Optimal payload 4-6k kg; Success improving over time; Sites near coast (0.51km avg) with rail/highway access.
- Folium: CircleMarkers + PolyLines to coast/rail (screenshots); Dash: Site dropdown → Pie chart success + Payload slider → Scatter outcomes.
- LR/SVM/KNN: 83% accuracy (confusion matrix); Decision Tree: 66% (overfit); SVM best precision/recall balance; Top features: PayloadMass + LaunchSite.

Análisis	Key Insight	Visual Tool
EDA	CCAFS SLC-40 >90% success	Folium maps + Bars
Interactive	Payload 4-6kkg optimal	Dash dropdown/slider
Predictive	SVM 83% accuracy	Confusion Matrix

Section 2

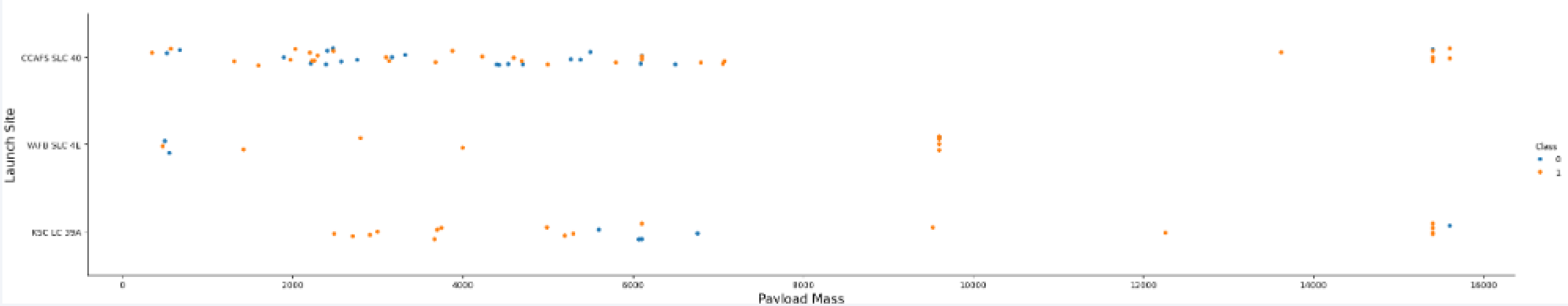
Insights drawn from EDA

Flight Number vs. Launch Site



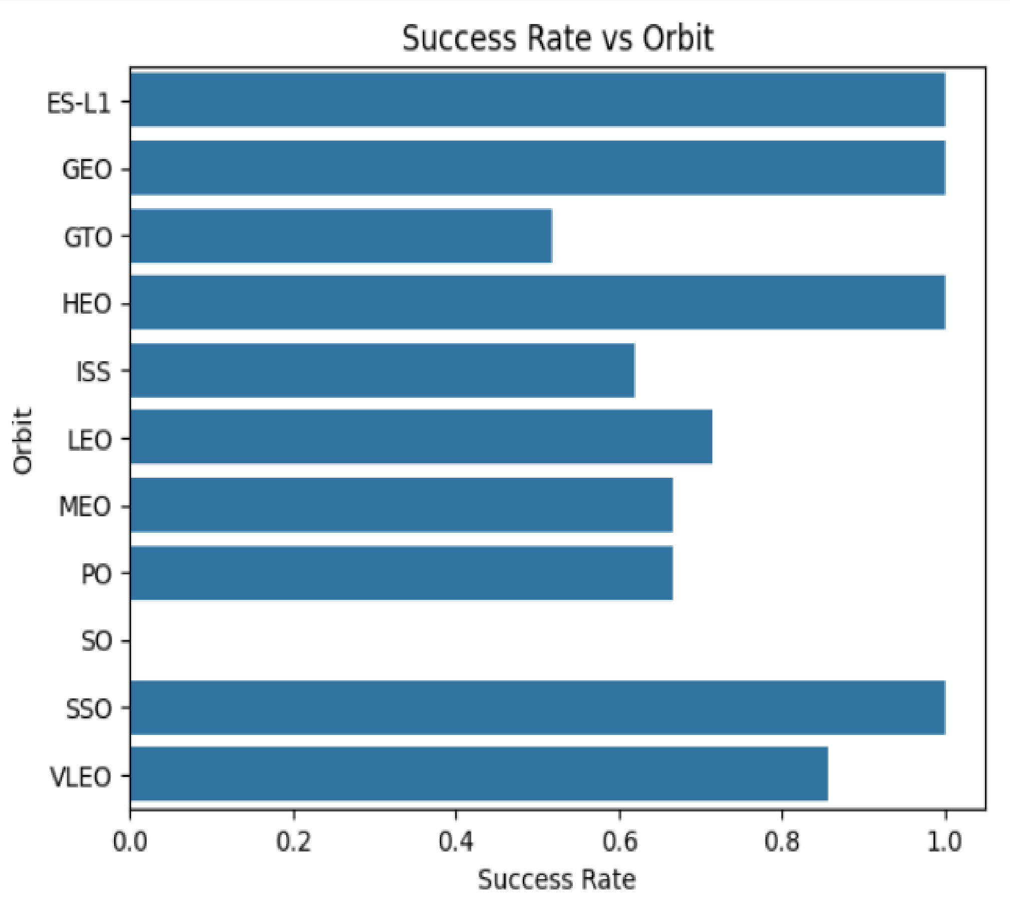
- It can be observed that the LaunchSite "CCAFS SLC 40" is the most used, no matter the class.

Payload vs. Launch Site



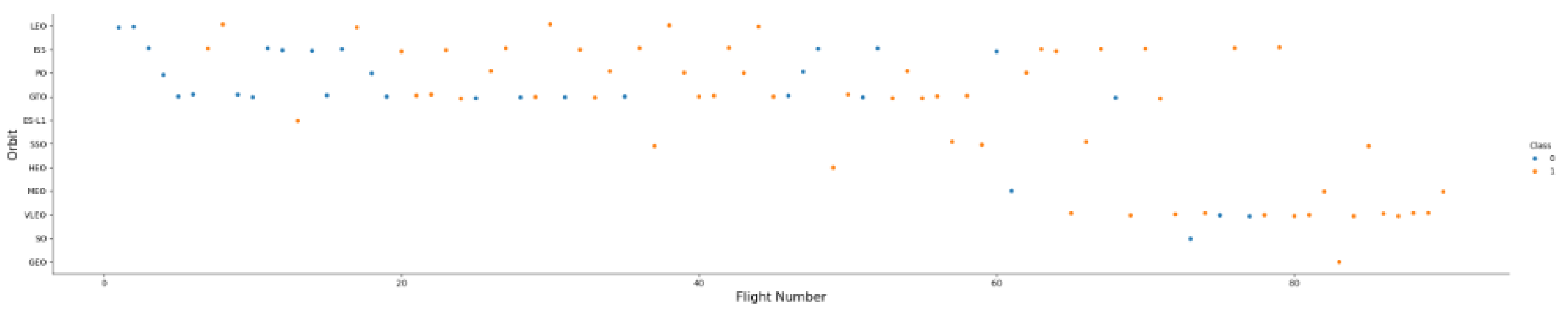
- Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type



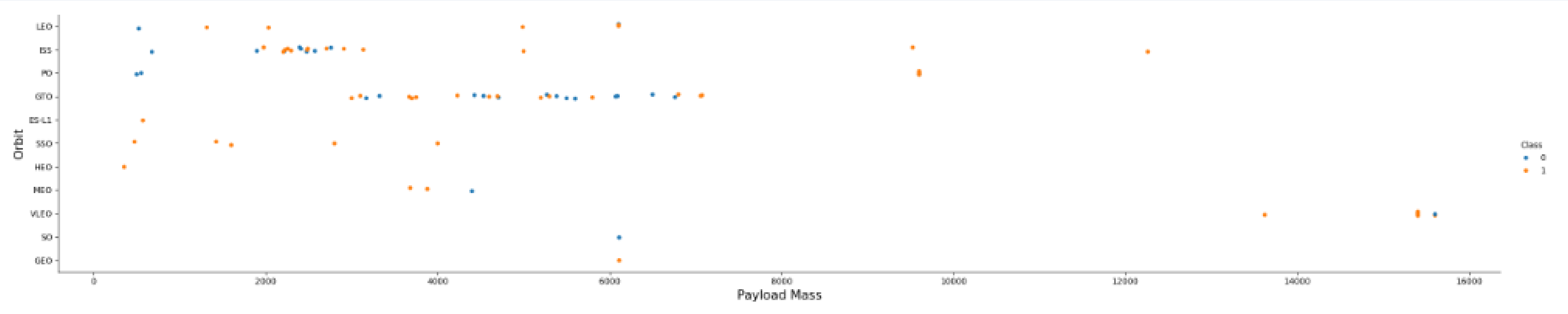
- The orbits with the highest success rate are ES-L1, GEO, HEO and SSO

Flight Number vs. Orbit Type



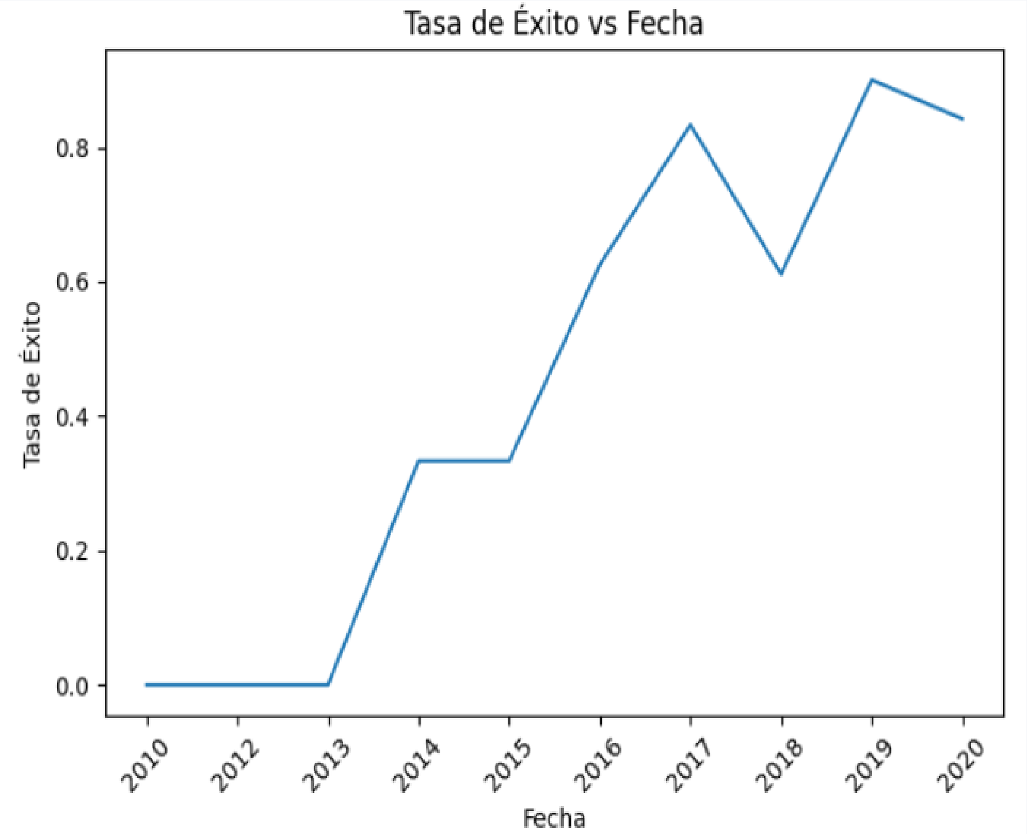
- You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

Payload vs. Orbit Type



- 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



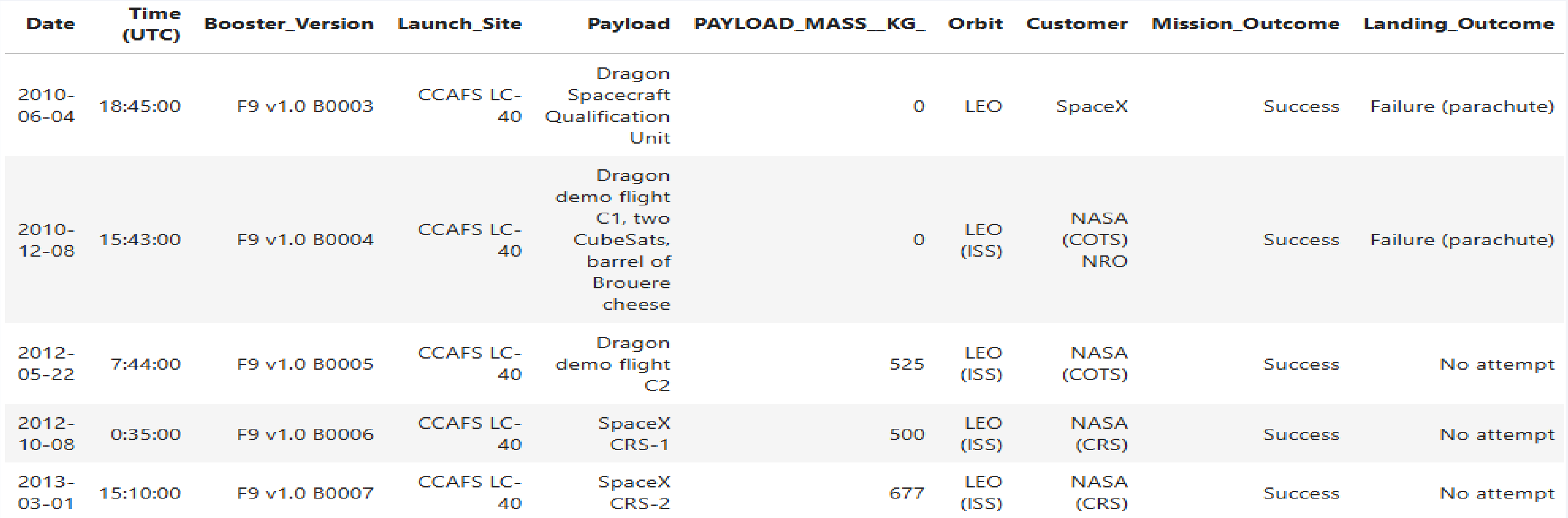
Can observe that the success rate since 2013 kept increasing till 2020

All Launch Site Names

- The names of the unique launch sites are: CCAFS LC-40, VAFB SLC-4E, KSC LC-39A, CCAFS SLC-40
- The query: SELECT DISTINCT Launch_Site FROM SPACEXTABLE → 4 unique sites; CCAFS SLC-40 most frequent (success leader); sqlite://my_data1.db → Direct SpaceX API data access.

Launch Site Names Begin with 'CCA'

- CCAFS LC-40: 5 early missions (2010-2013); F9 v1.0 boosters B0003-B0007; Dragon qualification + CRS-1/2; All mission success, landings: 2x parachute failure, 3x no attempt.
- SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5 → CCAFS LC-40 dominates early Falcon 9 program; Early landing attempts failed (parachutes); Shows site evolution from Dragon demos to operational CRS missions.



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

Total Payload Mass

- The query: SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)'
- Results: NASA (CRS): 45,596 kg total payload across Dragon CRS missions to ISS.
- This sustained Falcon 9 development enabling reusable landing tech. Primary customer driving launch cadence.

Average Payload Mass by F9 v1.1

- The query: SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)'
- Results: NASA CRS: 2928.4 kg avg payload per mission
- Average payload mass per CRS mission to ISS; Reveals typical Dragon cargo capacity; Consistent ~2-3t per flight pattern.

First Successful Ground Landing Date

- The query: SELECT MIN(DATE) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)'
- Results: 2015-12-22
- Falcon 9 Full Thrust first ground pad success (Orbcomm OG2); Marked reusability breakthrough; 5 years from first launch (2010) to landing success.

Successful Drone Ship Landing with Payload between 4000 and 6000

- The query: SELECT Booster_Version,Payload WHERE PAYLOAD_MASS__KG_ BETWEEN '4000' AND ''6000 AND Landing_Outcome='Success (drone ship)'
- Results:

Booster	**Payload**
F9 FT B1022	JCSAT-14
F9 FT B1026	JCSAT-16
F9 FT B1021.2	SES-10
F9 FT B1031.2	SES-11/EchoStar 105

- Optimal payload range (4-6k kg) drone ship landings; Falcon 9 FT Block 3 boosters; SES/JCSAT telecom sats.

Total Number of Successful and Failure Mission Outcomes

- The query: %%sql SELECT

COUNT(CASE WHEN Mission_Outcome like '%Success%' THEN 1 END) AS Successes,

COUNT(CASE WHEN Mission_Outcome like '%fail%' THEN 1 END) AS Failures

FROM SPACEXTABLE;

- Results:

Successes	Failures
100	1

- 99.0% mission success rate; Single failure in SPACEXTABLE dataset; Demonstrates Falcon 9 operational maturity.

Boosters Carried Maximum Payload

- The query: %%sql SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE Payload_Mass__kg_ = (SELECT MAX(Payload_Mass__kg_) FROM SPACEXTABLE);

- Results:

```
                                  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
```

- 12 unique F9 B5 Block 5 boosters carried maximum payloads; Latest generation handles heaviest loads reliably.

2015 Launch Records

- The query: %%sql SELECT substr(Date,6,2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr(Date,0,5)='2015' AND (Landing_Outcome LIKE '%drone ship%' OR Landing_Outcome LIKE '%False drone ship%') AND Landing_Outcome LIKE '%fail%';
- Results:

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

- 2 early drone ship failures (Jan/Apr 2015); F9 v1.1 boosters B1012/B1015 from CCAFS LC-40; Offshore recovery tech still maturing.

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

Landing_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

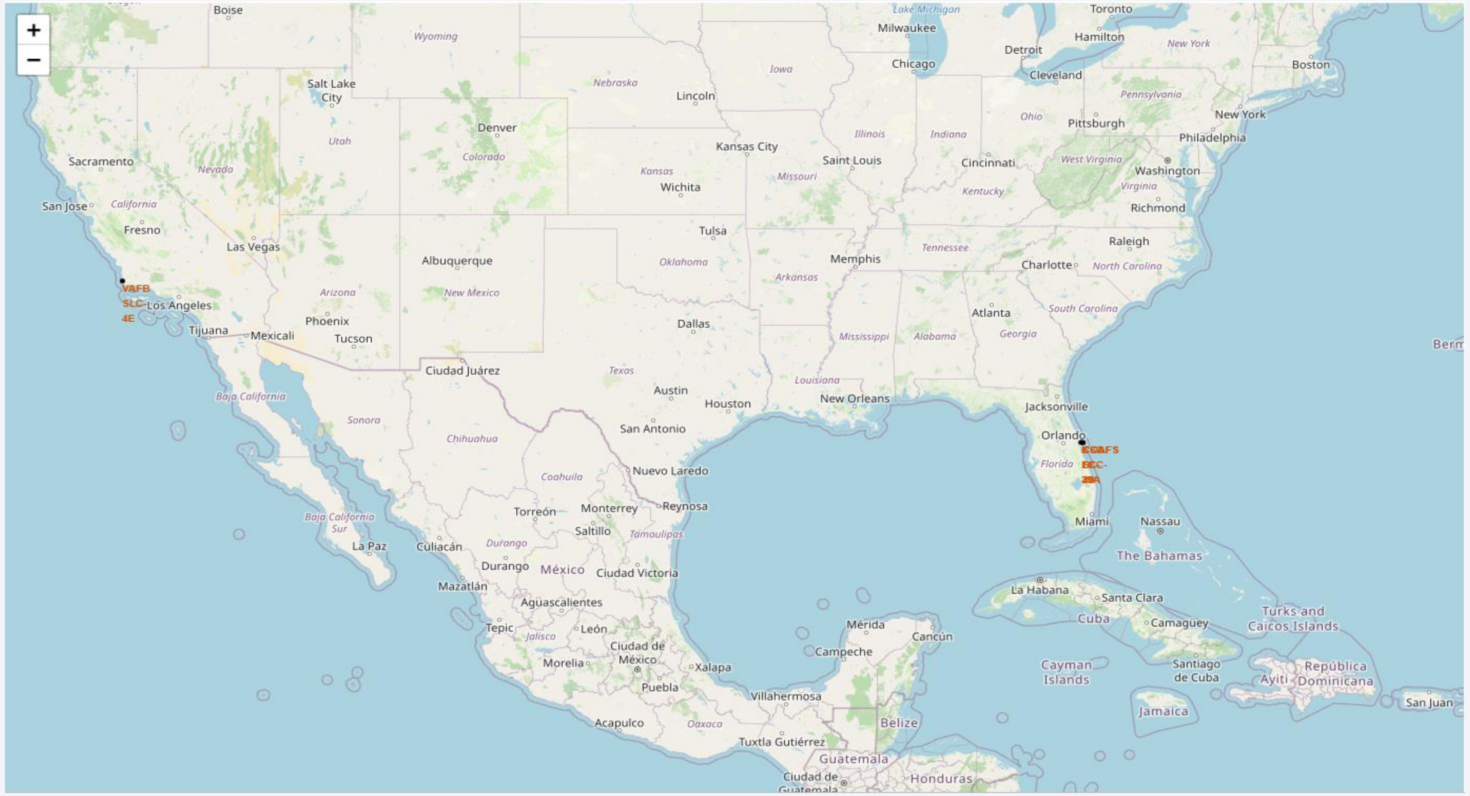
Early era (2010-2017):

- 32% no attempts (10/31)
- Drone ship: 50% success (5/10)
- 3 ground pad = reusability breakthrough

Section 3

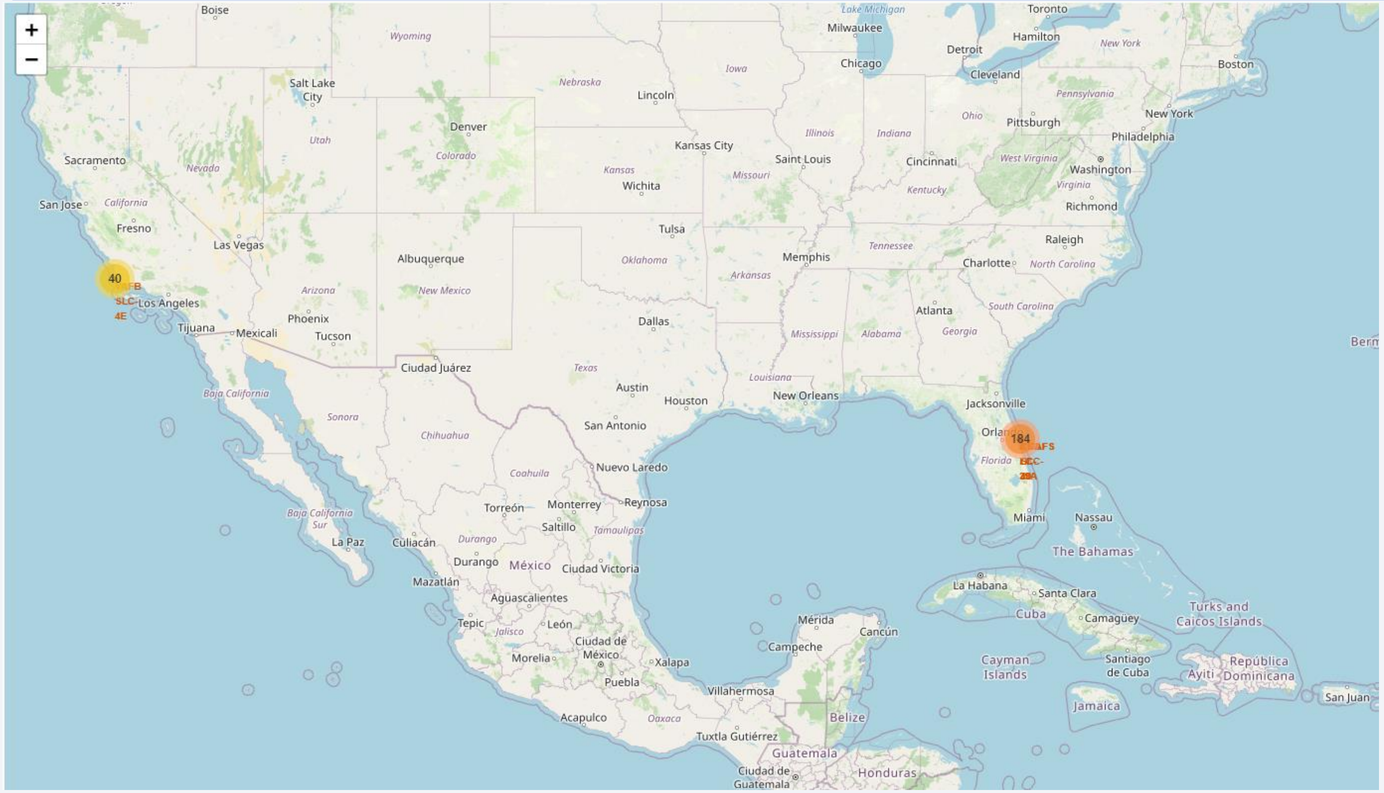
Launch Sites Proximities Analysis

Launch sites Map



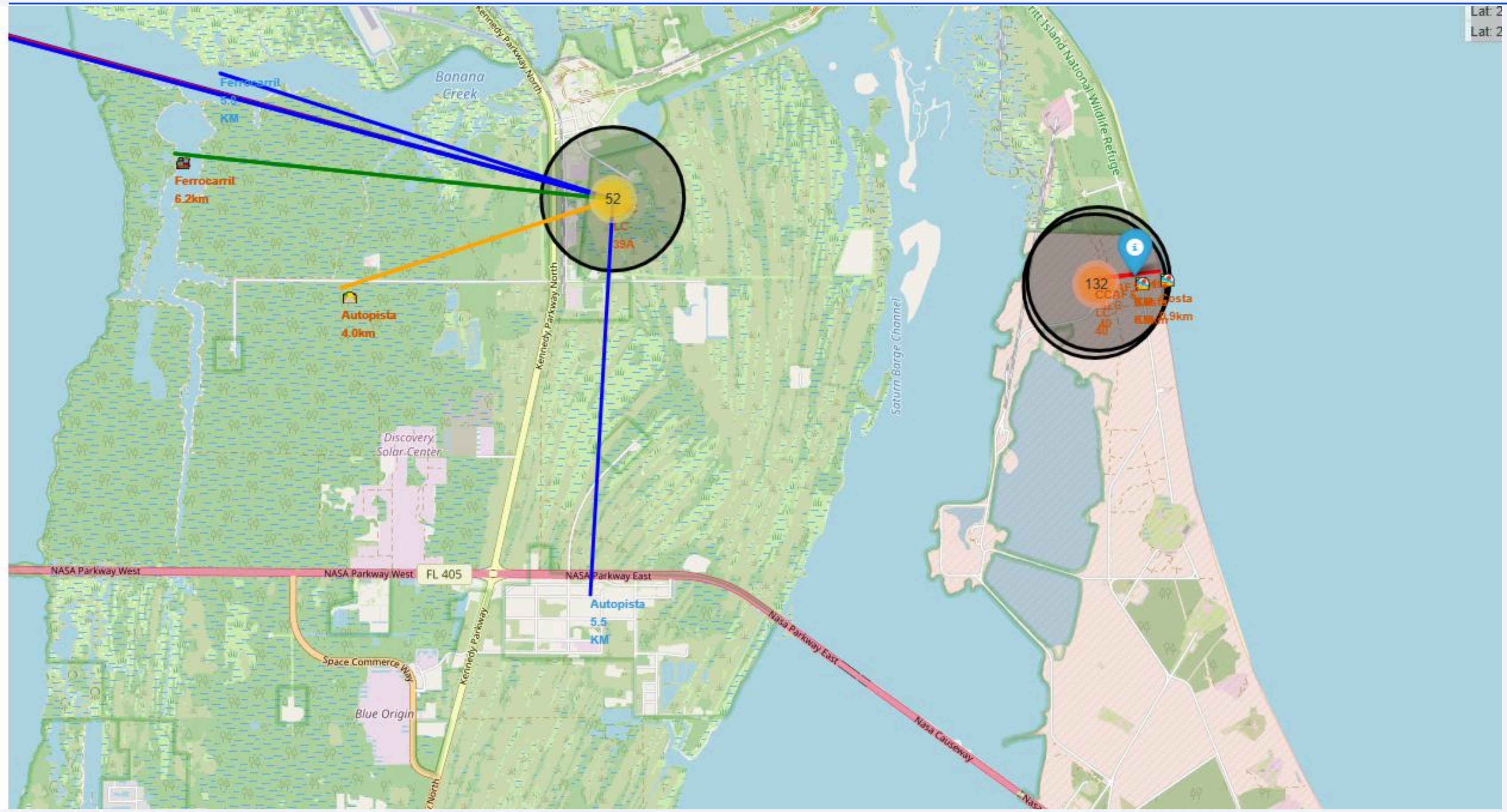
Launch sites aren't in proximity to the Ecuator line but yes they are very close to the coast. Also sites must be in a place with no residences o living areas, only launch infrastructures, near to water so all the energy displayed can't be dangerous for human life and where a hypothetical malfunction and subsequent explosion of the rocket would cause the least amount of damage.

Success/Fail Launch map

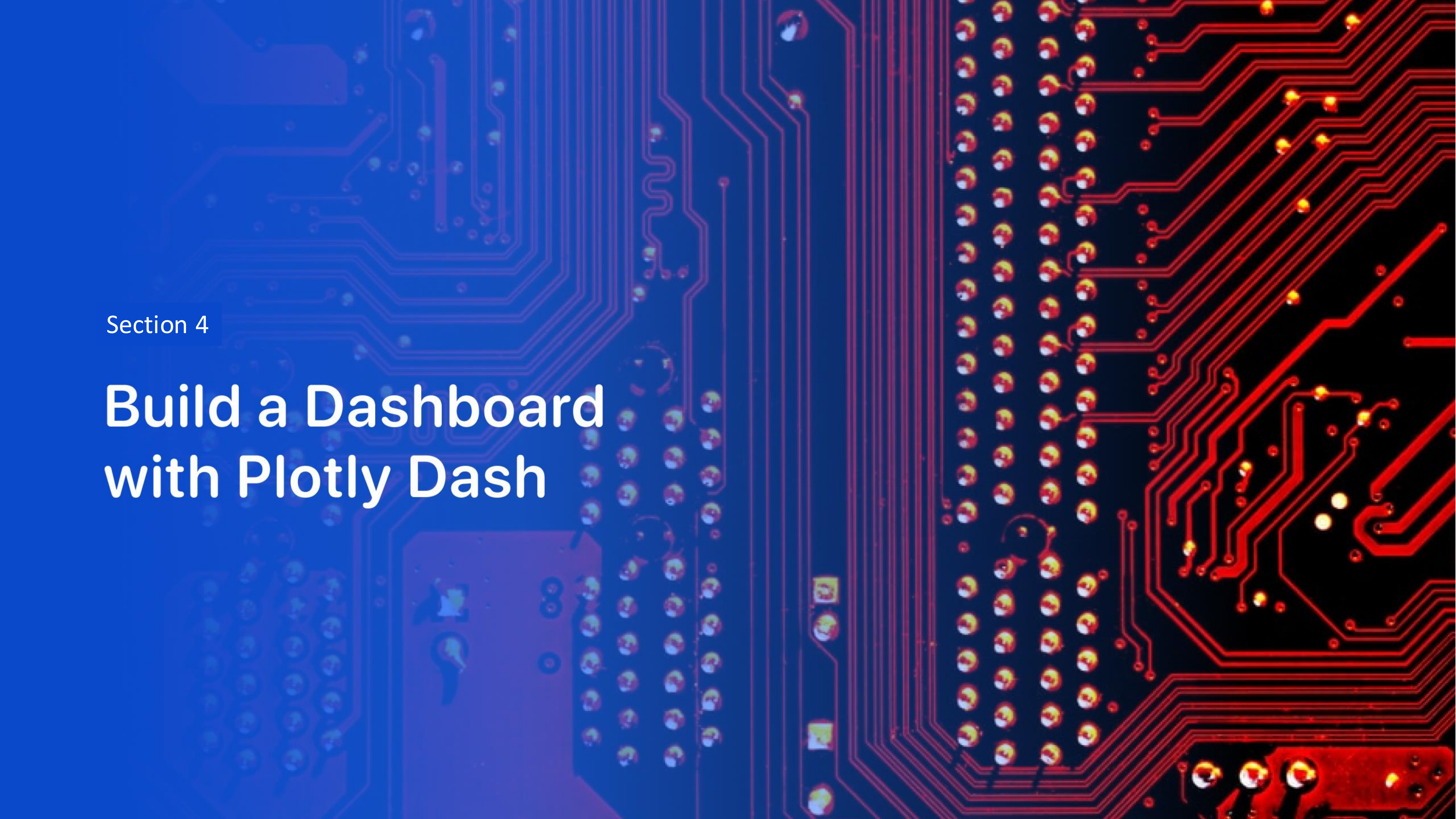


- The site with most success is KSC LC-39A.

Proximity map



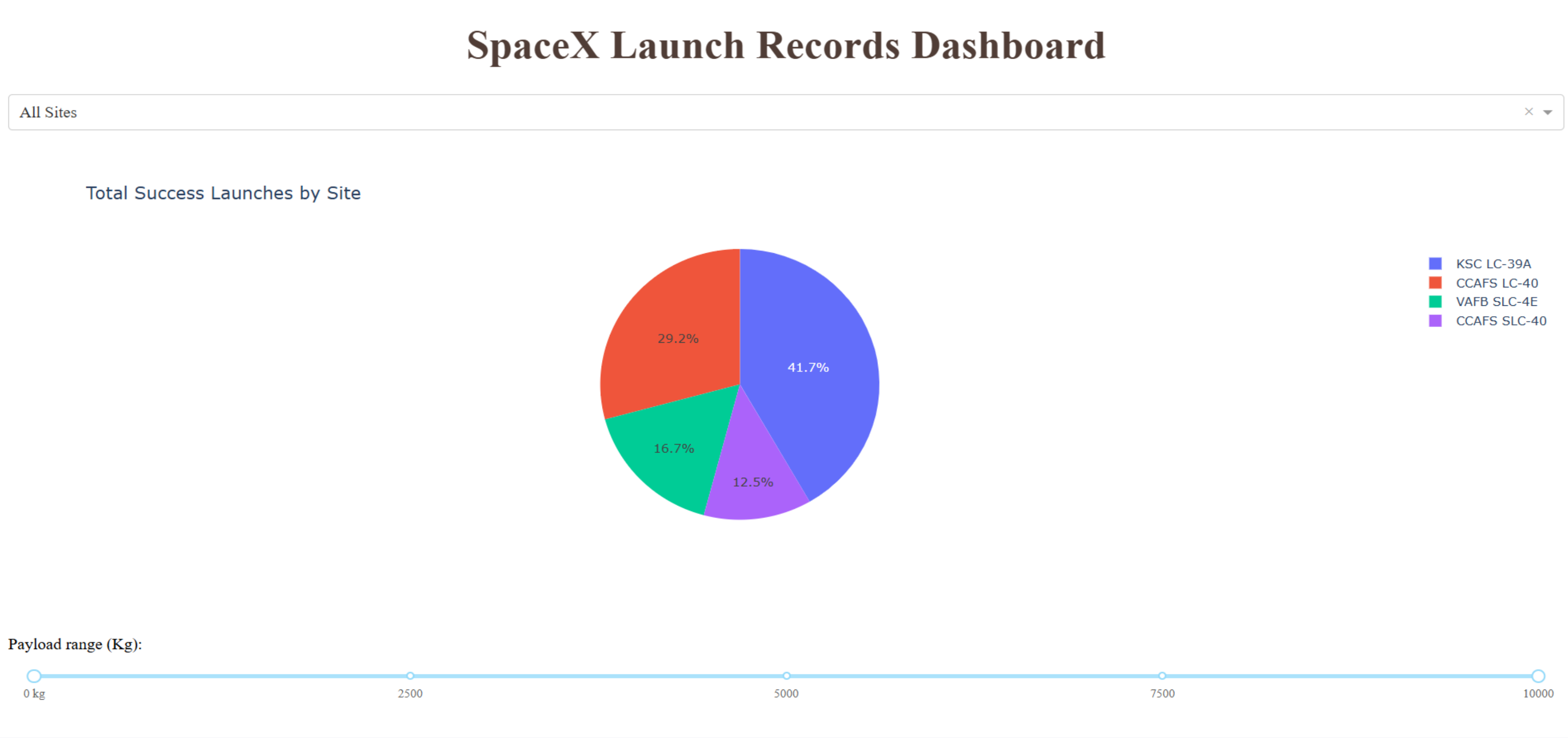
Any location or human mobility route is kilometers away, while the coast is always the closest.



Section 4

Build a Dashboard with Plotly Dash

All sites pIECHART

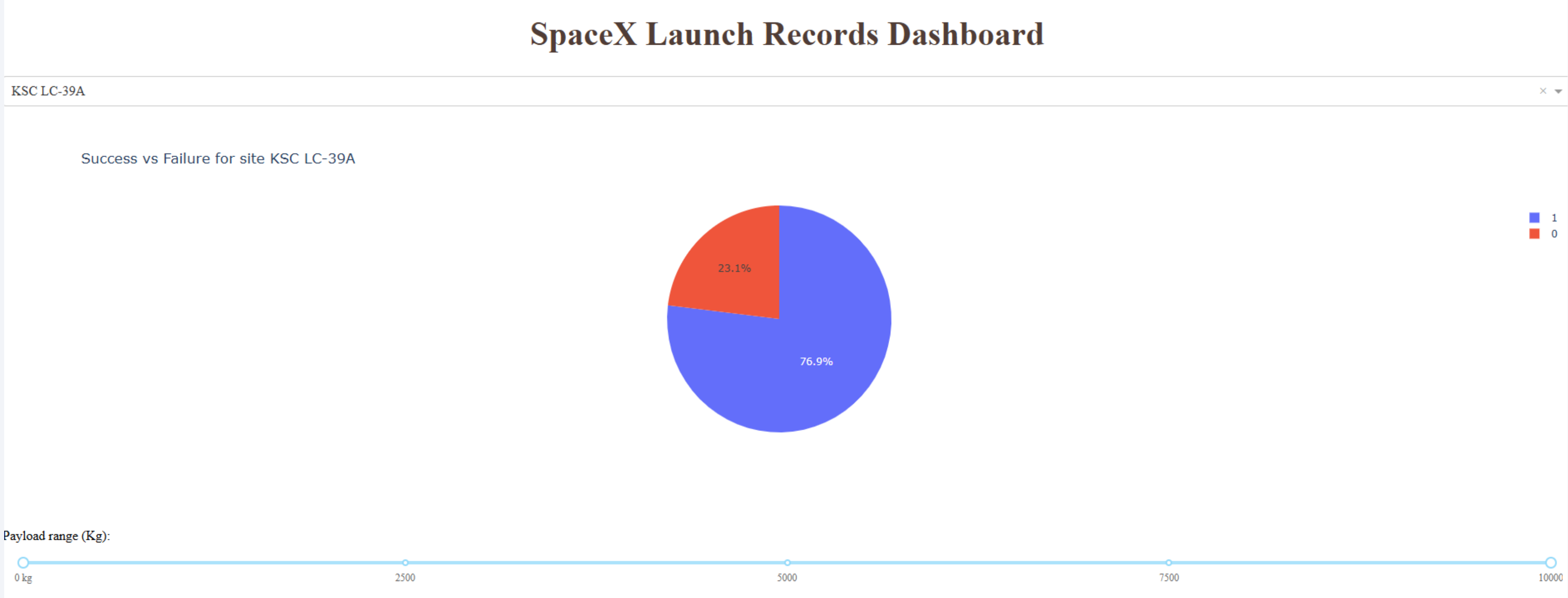


SpaceX Launch Records Dashboard

All Sites

- The site "KSC LC-39A" is the most successful one

KSC LC-39A Launch site success rate

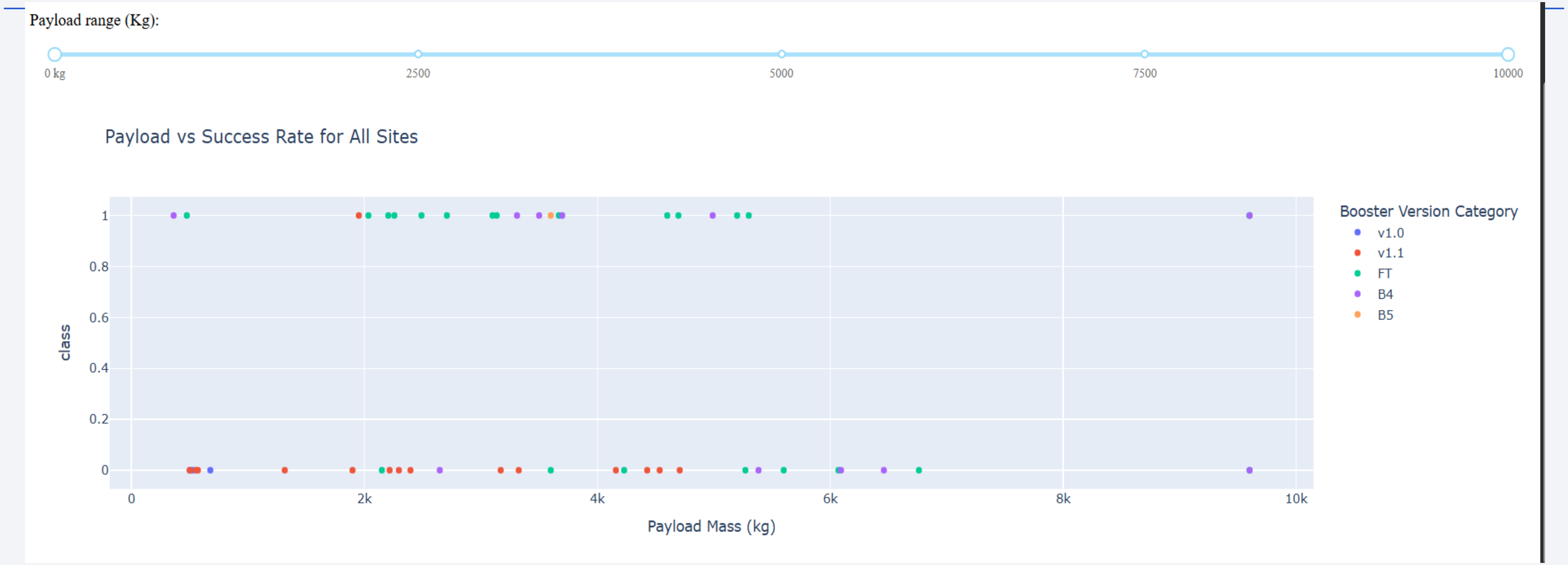


SpaceX Launch Records Dashboard

KSC LC-39A

- More than 75% launches are successful in this site, the best site so far.

Scatter Plot: Payload vs. Launch Outcome

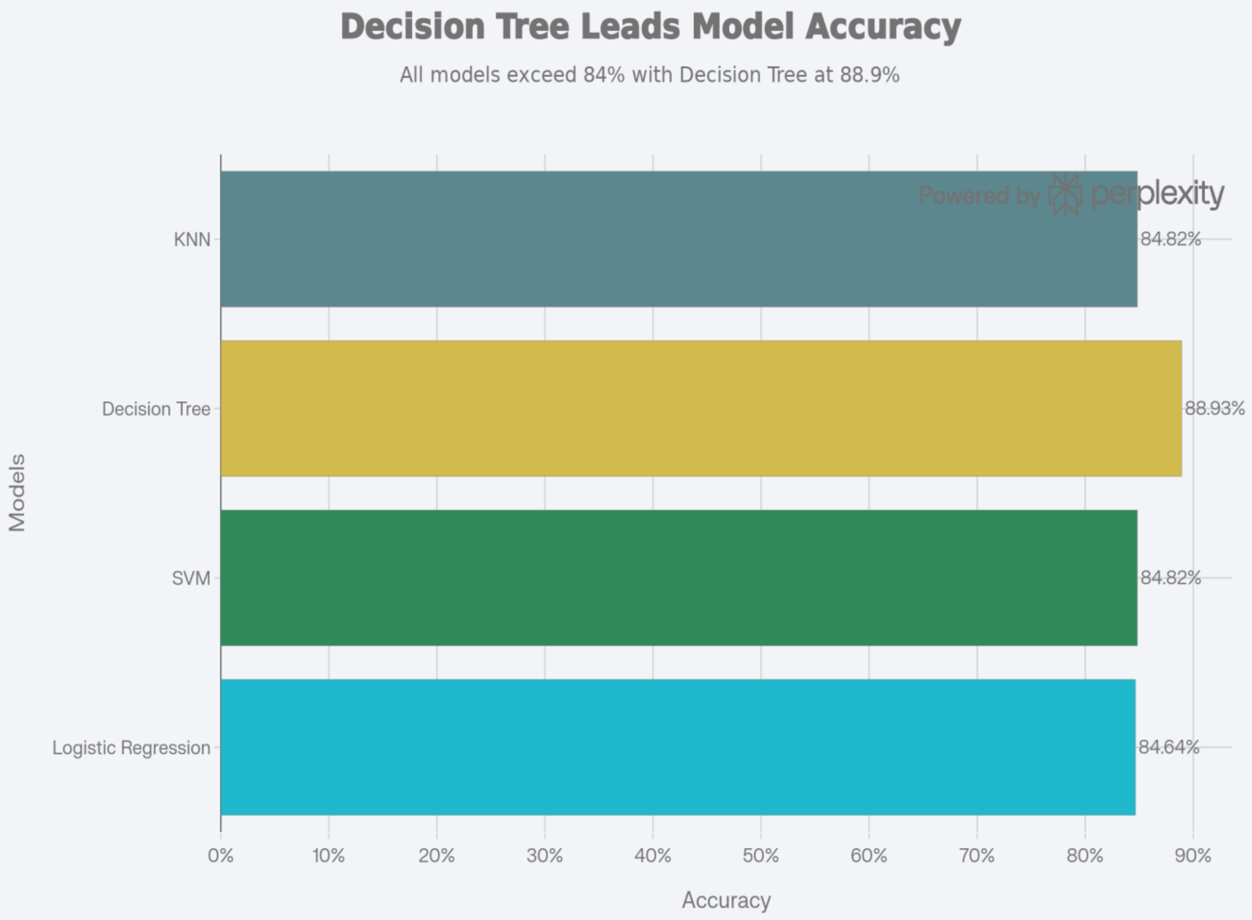


- The highest launch success rate is for 0-5000 kg Payload, which means the worst rate is for 5000-10000 kg Payload.
- Also the booster version with the highest launch rate is 'FT'.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



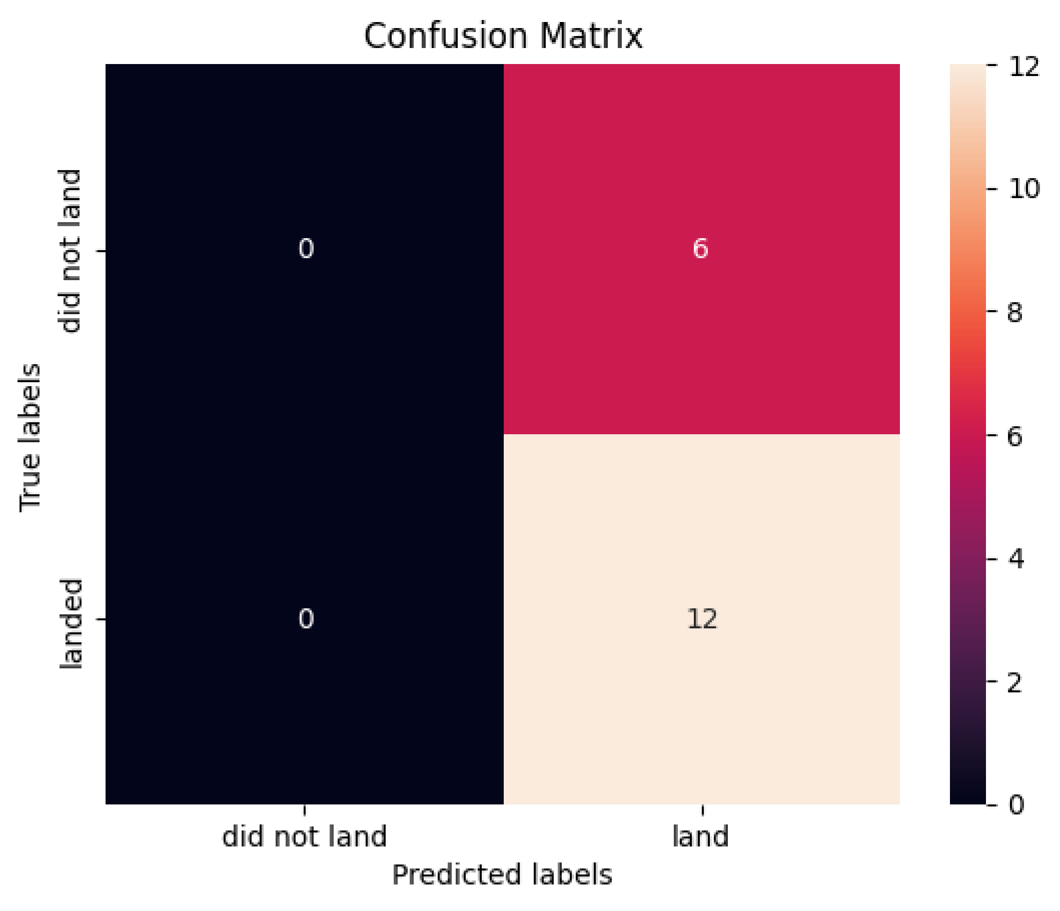
Decision Tree Leads Model Accuracy

All models exceed 84% with Decision Tree at 88.9%

- Decision Tree has the best accuracy on training set

Confusion Matrix

- The confusion matrix yields worse accuracy than the other models because the number of samples in the test set is small; this model improves accuracy as the number of samples increases.
- Acurracy: 66%



Conclusions

- The project demonstrates a complete data science pipeline that reveals key patterns in Falcon 9 reusability.
- CCAFS SLC-40 leads in success rates (>90%), optimal payloads of 4-6k kg correlate with higher landing success, and accuracy improves over time since 2013.
- Dominant factors include Launch Site, Orbit Type (high success in LEO and SSO), and Payload Mass, validated through EDA using Folium, Dash, SQL, and visualizations.
- On the training dataset, the Decision Tree model achieved the highest accuracy at 88%, though it dropped to 66% on test data due to overfitting. Logistic Regression, SVM, and KNN models reached 83% test accuracy with better generalization, and SVM excelled due to balanced precision/recall across classes.

Appendix

- All files used to make this presentation are stored here: https://github.com/AValdavida/Test-Repo
- To explore and analyze the data, both Python and SQL have been used with the necessary visualizations, already shown in this presentation.
- An example of an SQL query used through Python:

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

Appendix

- All files used to make this presentation are stored here: https://github.com/AValdavida/Test-Repo
- Machine learning has been used to determine which model is best for data prediction.
- A sample/summary of the code (python) used in the ML analysis process is shown below:
  - from sklearn.ensemble import DecisionTreeClassifier
  - from sklearn.model_selection import train_test_split
  - X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
  - dt = DecisionTreeClassifier().fit(X_train, y_train) # Achieved 88% train accuracy
  - print(f"Train Acc: {dt.score(X_train, y_train):.2f},
  - Test Acc: {dt.score(X_test, y_test):.2f}")

Thank you!