SpaceX Falcon 9 Landing Prediction

IBM Data Science Capstone | Aleksy Kucy

Agenda

- Business problem & why landing matters
- Data sources + wrangling pipeline
- EDA & SQL: what the data is trying to tell us
- Maps & dashboard: location and payload patterns
- Machine learning models + final recommendation
- Conclusions & next steps

Executive summary (what we learned)

Goal: predict whether the first stage will land successfully (binary classification).

Key EDA hint: success rates differ by orbit, launch site, and (to a point) payload mass.

Best model in this run: Decision Tree — Test Accuracy ≈ 0.9444 (CV ≈ 0.8750).

Practical takeaway: if you know site + orbit + payload, you already know a lot — the model just formalises it.

(And yes, GridSearch can be a marathon. We made it a sprint.)

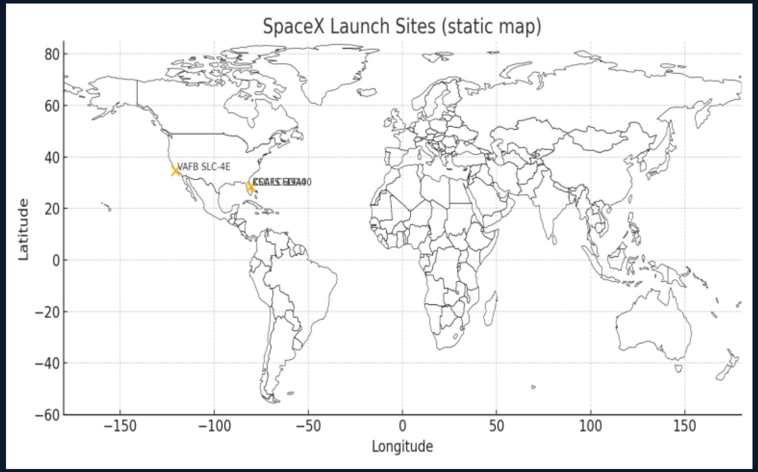
Business problem

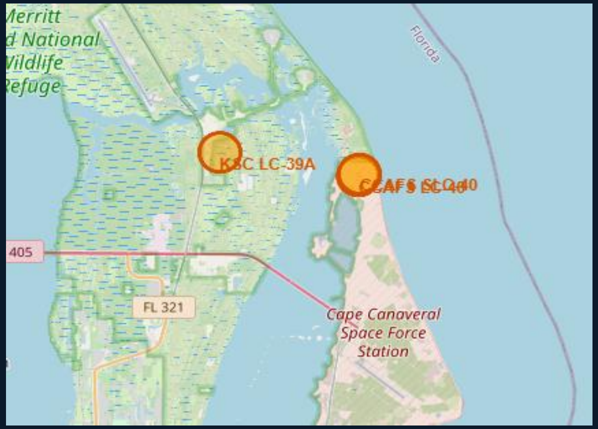
Falcon 9's first stage is the expensive part; reusability depends on safe landing.

If we can predict landing success in advance, we can:

- plan recovery operations and risk
- estimate mission cost / price
- choose mission profiles more rationally

So the question is simple (and slightly brutal): will it stick the landing?





Data sources

SpaceX launch records collected via API (launches, payloads, outcomes).

Supplemented by public web data (e.g., launch site info) and curated tables used in the labs.

Target label (Class): 1 = successful landing, 0 = otherwise.

Features used downstream: launch site, orbit, payload mass, flight number, and reuse-related attributes.

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...	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

Methodology (pipeline)

1) Data Collection (API & web)
2) Data Wrangling (clean, select features, define Class)
3) EDA (visual + statistical patterns)
4) SQL exploration (targeted questions)
5) Interactive analysis (maps + dashboard)
6) ML modelling (LR, SVM, DT, KNN) + evaluation

Data collection

Pulled launch data programmatically (API): missions, rockets, payloads, outcomes.

Joined with additional fields (launch site name, coordinates).

Sanity checks: duplicates, missing payload masses, inconsistent outcomes.

Output: a consistent table ready for wrangling & modelling.

Data wrangling

Converted outcomes into binary Class (success/failure).

Selected modelling features and normalised/encoded them where needed.

Handled categorical variables (Orbit, LaunchSite) via one-hot encoding later.

Result: model-ready dataset (X features + y label).

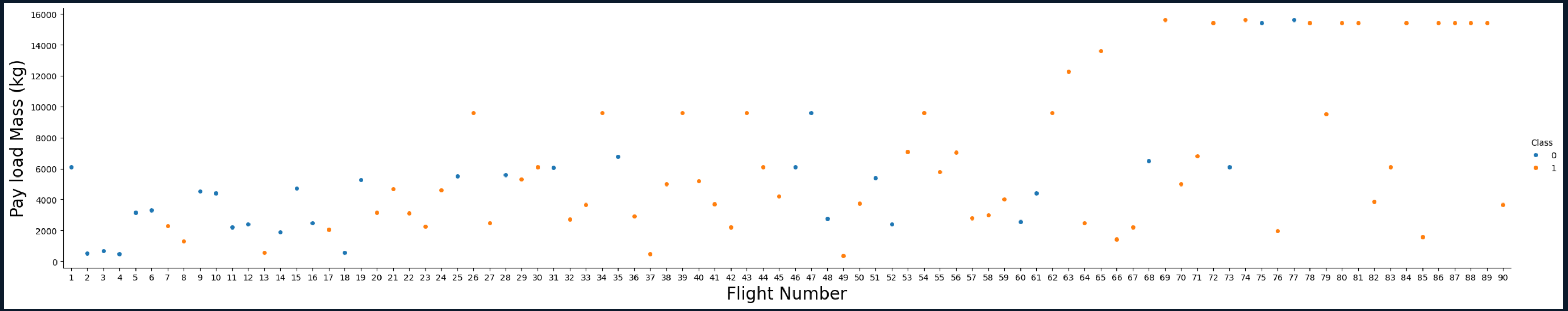
FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	nan	1.0	0	B0003	-80.577366	28.561857	0
2	2012-05-22	Falcon 9	525.0	LEO	CCAFS SLC 40	None None	1	False	False	False	nan	1.0	0	B0005	-80.577366	28.561857	0
3	2013-03-01	Falcon 9	677.0	ISS	CCAFS SLC 40	None None	1	False	False	False	nan	1.0	0	B0007	-80.577366	28.561857	0
4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	nan	1.0	0	B1003	-120.610829	34.632093	0
5	2013-12-03	Falcon 9	3170.0	GTO	CCAFS SLC 40	None None	1	False	False	False	nan	1.0	0	B1004	-80.577366	28.561857	0
6	2014-01-06	Falcon 9	3325.0	GTO	CCAFS SLC 40	None None	1	False	False	False	nan	1.0	0	B1005	-80.577366	28.561857	0
7	2014-04-18	Falcon 9	2296.0	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	nan	1.0	0	B1006	-80.577366	28.561857	1
8	2014-07-14	Falcon 9	1316.0	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	nan	1.0	0	B1007	-80.577366	28.561857	1
9	2014-08-05	Falcon 9	4535.0	GTO	CCAFS SLC 40	None None	1	False	False	False	nan	1.0	0	B1008	-80.577366	28.561857	0
10	2014-09-07	Falcon 9	4428.0	GTO	CCAFS SLC 40	None None	1	False	False	False	nan	1.0	0	B1011	-80.577366	28.561857	0

Exploratory Data Analysis (EDA)

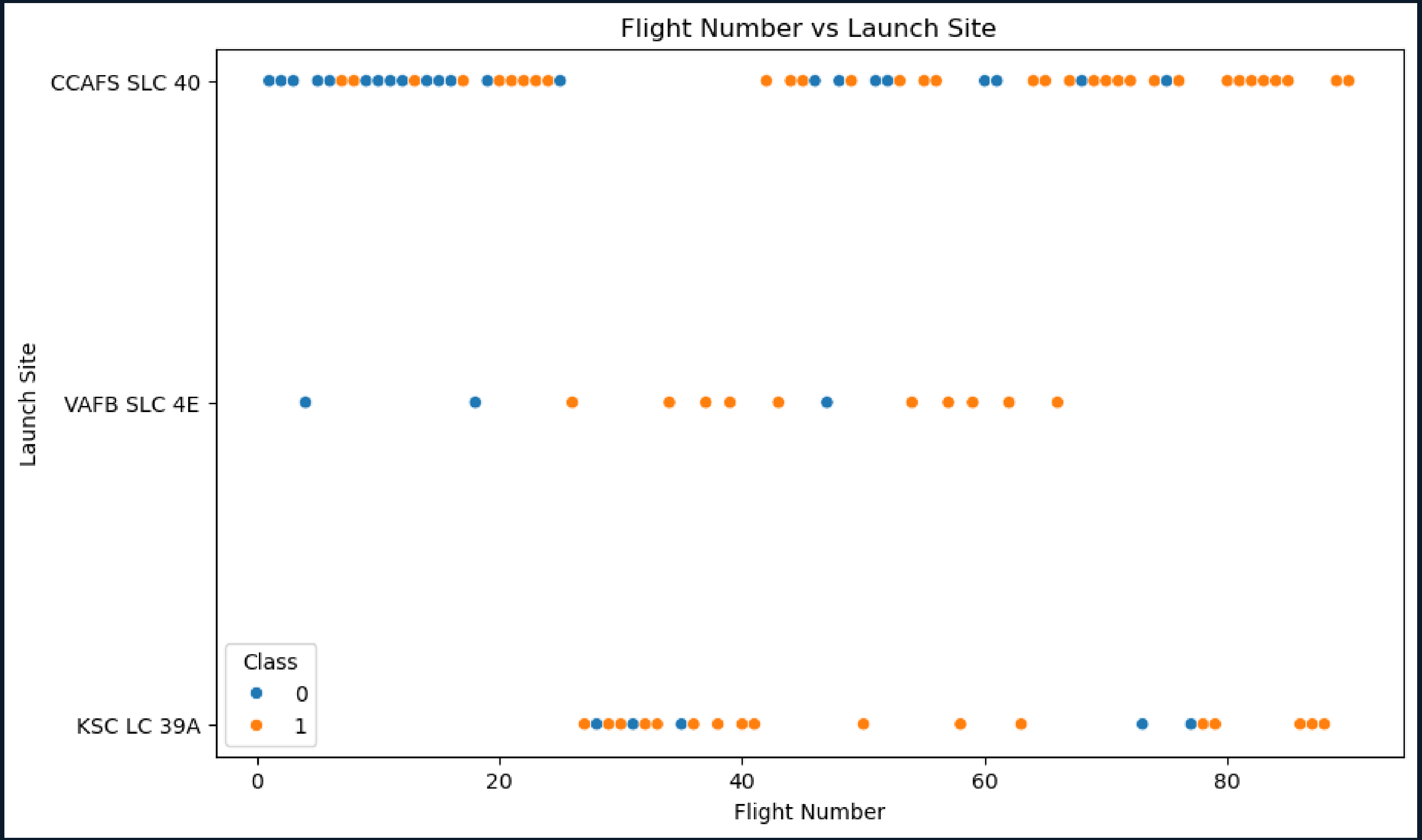
EDA is where the data stops being a spreadsheet and starts being a story.

We explored success rate patterns across:

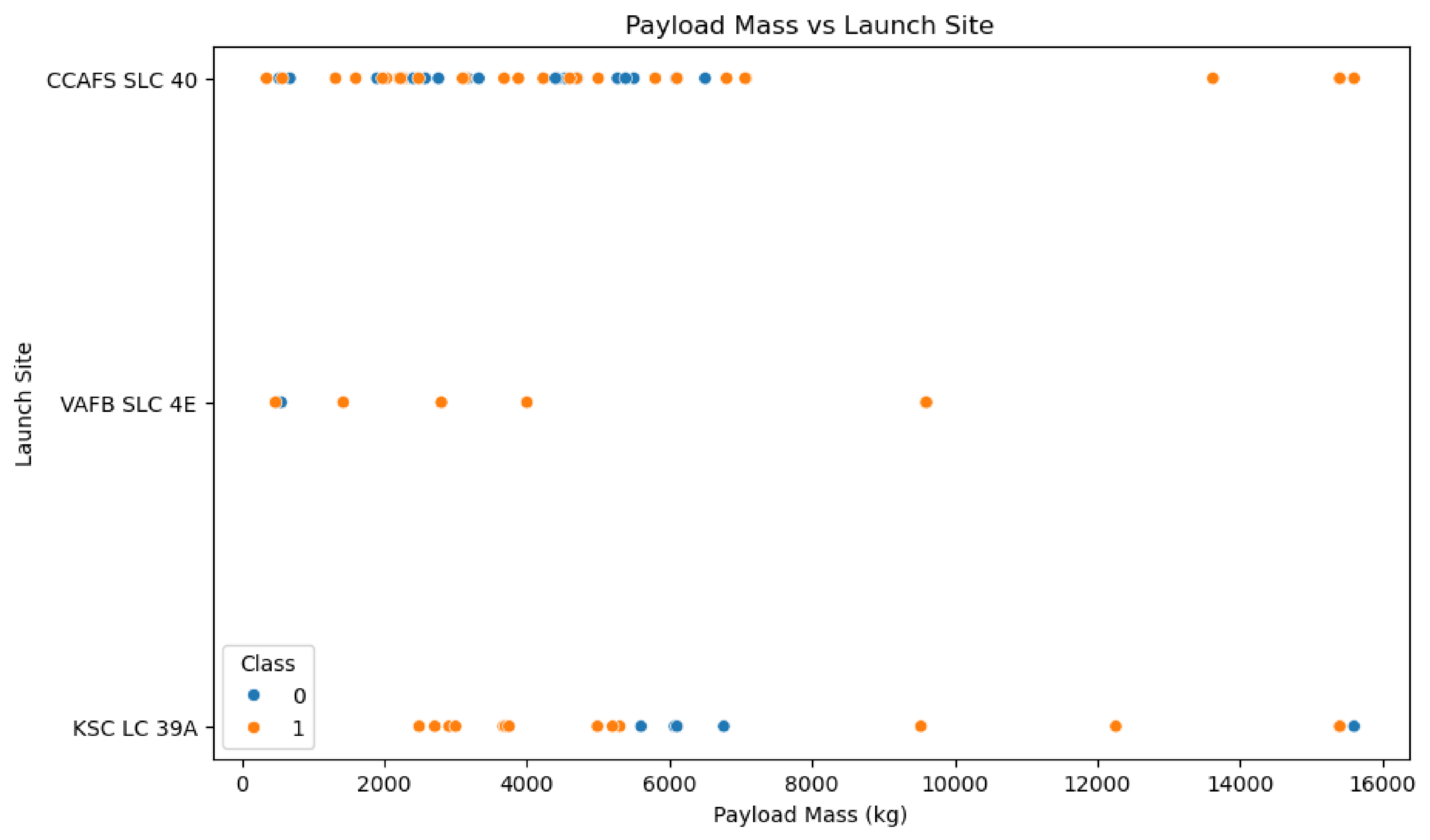
- launch sites
- orbits
- payload mass
- time (yearly trend)



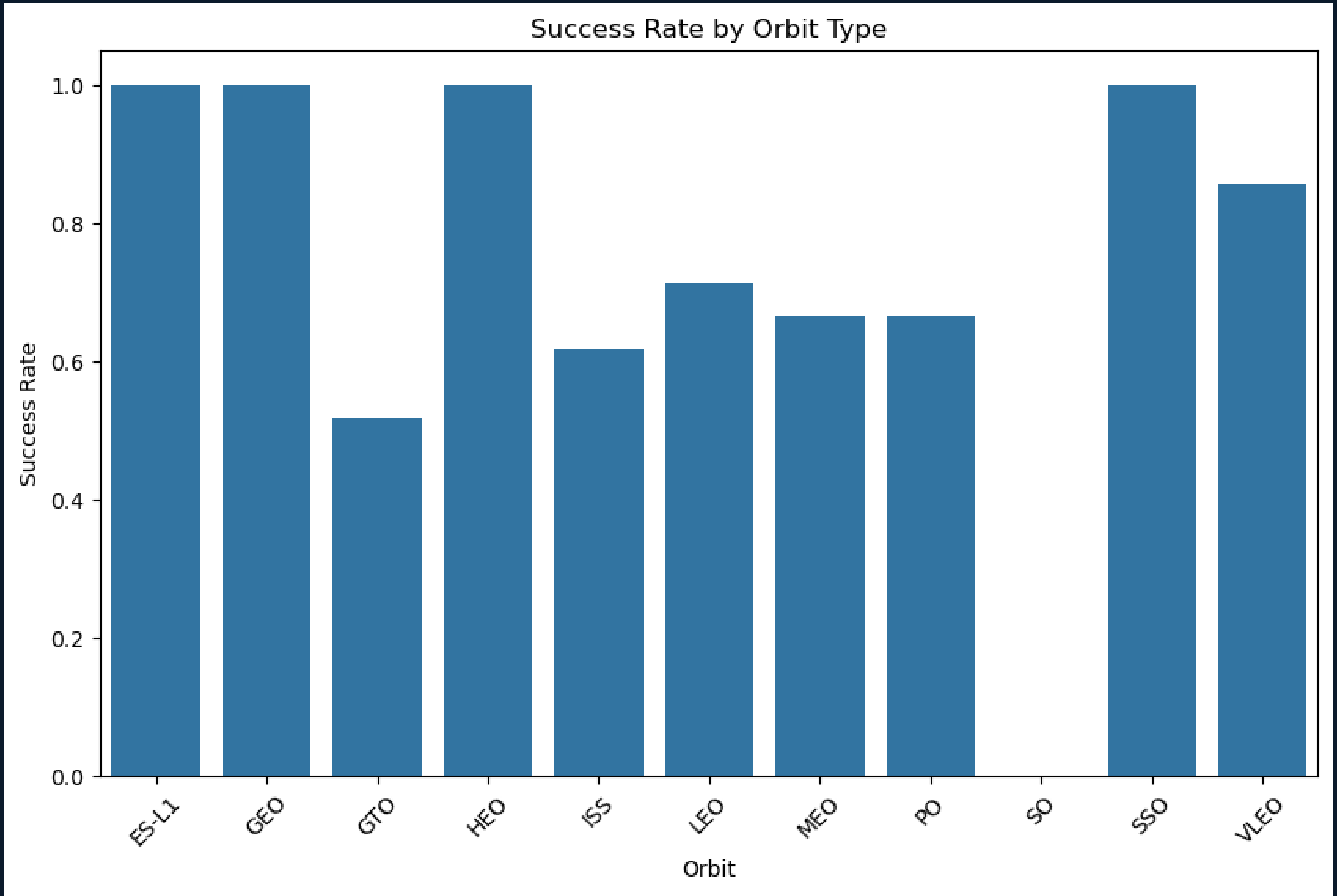
EDA: Flight number vs Launch Site



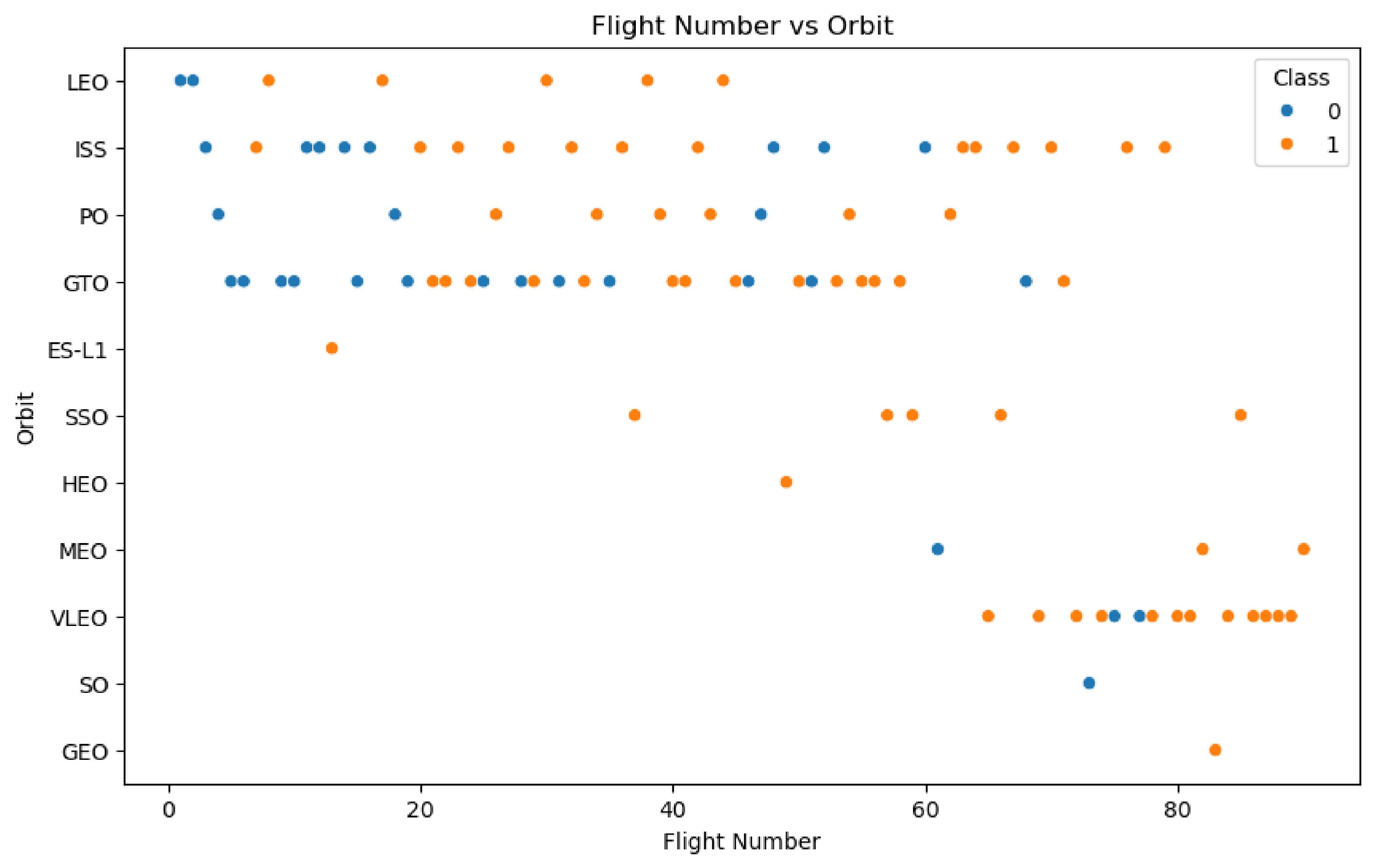
EDA: Payload mass vs Launch Site



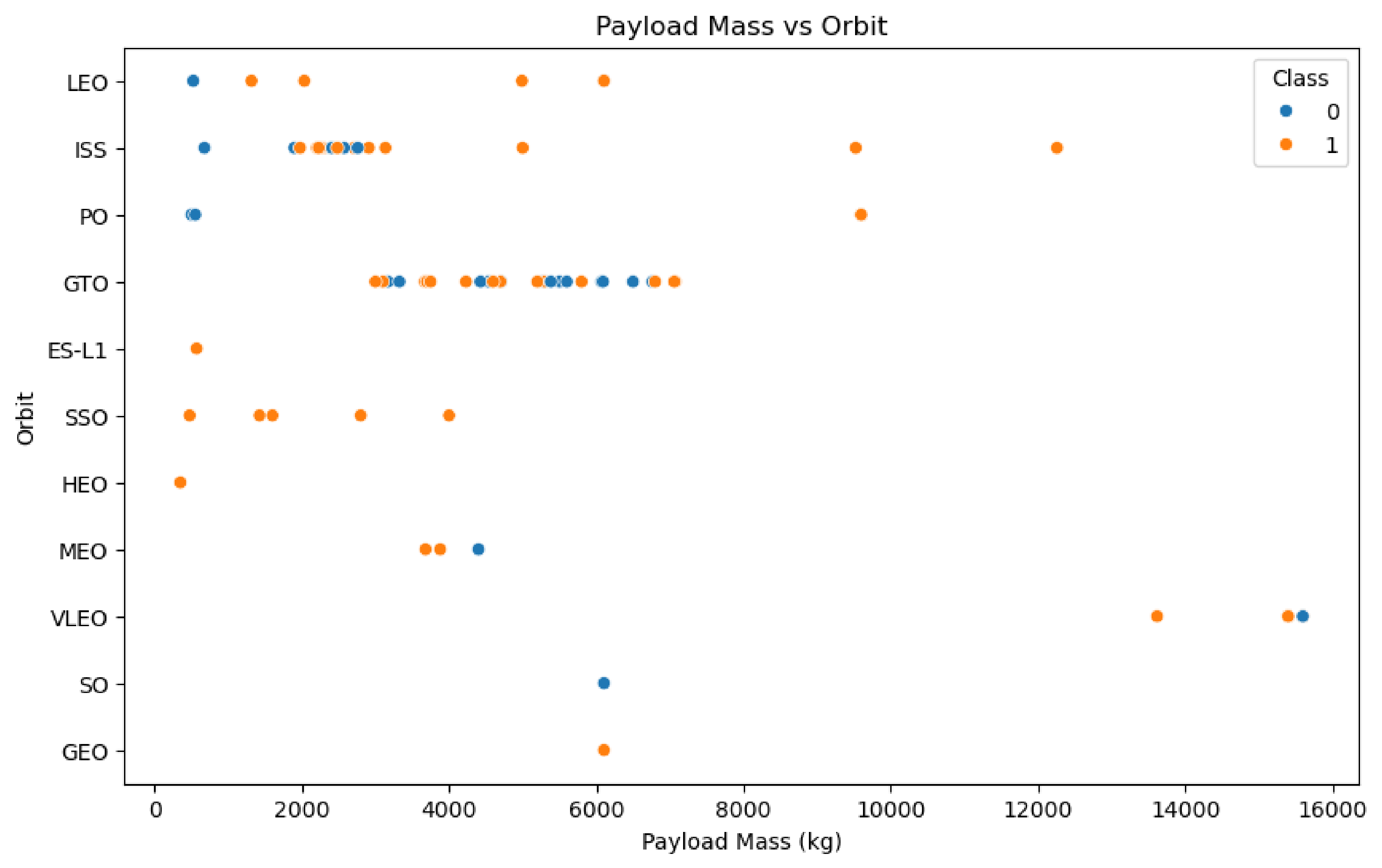
EDA: Success rate vs Orbit



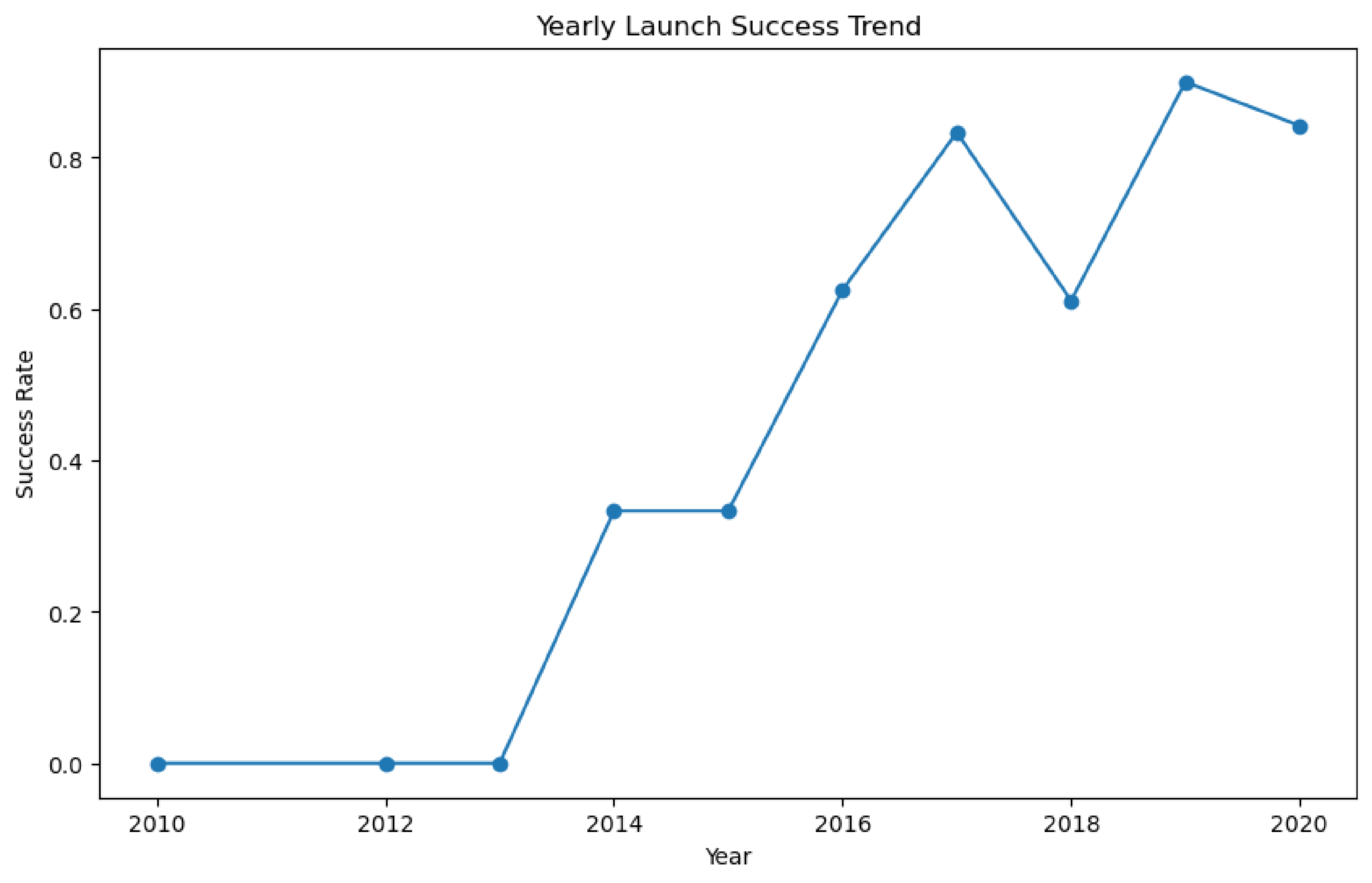
EDA: Flight number vs Orbit



EDA: Payload mass vs Orbit



EDA: Yearly success trend



SQL analysis (targeted questions)

Why SQL here? Because sometimes you don't need another plot: you need an exact answer!

We queried the launch dataset for:

- launch site lists and filters
- payload aggregates (sum/avg)
- first successful events
- counts by mission outcome

```
pd.read_sql_query("""
SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload_mass_kg
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)'
""", con)
```

Out[26]:

	total_payload_mass_kg
0	45596

```
pd.read_sql_query("""
SELECT AVG(PAYLOAD_MASS__KG_) AS total_payload_mass_kg
FROM SPACEXTBL
WHERE Booster_Version = 'F9 v1.1'
""", con)
```

Out[29]:

	total_payload_mass_kg
0	2928.4

SQL Task 1: Distinct launch sites

Launch Site:
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

SQL Task 2: Launch sites starting with 'CCA'

Launch Site:
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

SQL Task 3: Total payload mass for NASA (CRS)

Total payload mass:
45 596 kg

SQL Task 4: Average payload mass for booster 'F9 v1.1'

Total payload mass
2 928.4 kg

SQL Task 5: First successful landing on Ground Pad

First successful ground pad
2015-12-22

SQL Task 6: Booster versions for payload range 4,000–6,000 kg

Booster Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

SQL Task 7: Mission outcome counts

Mission Outcome	total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

SQL Task 8: Booster versions with maximum payload

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

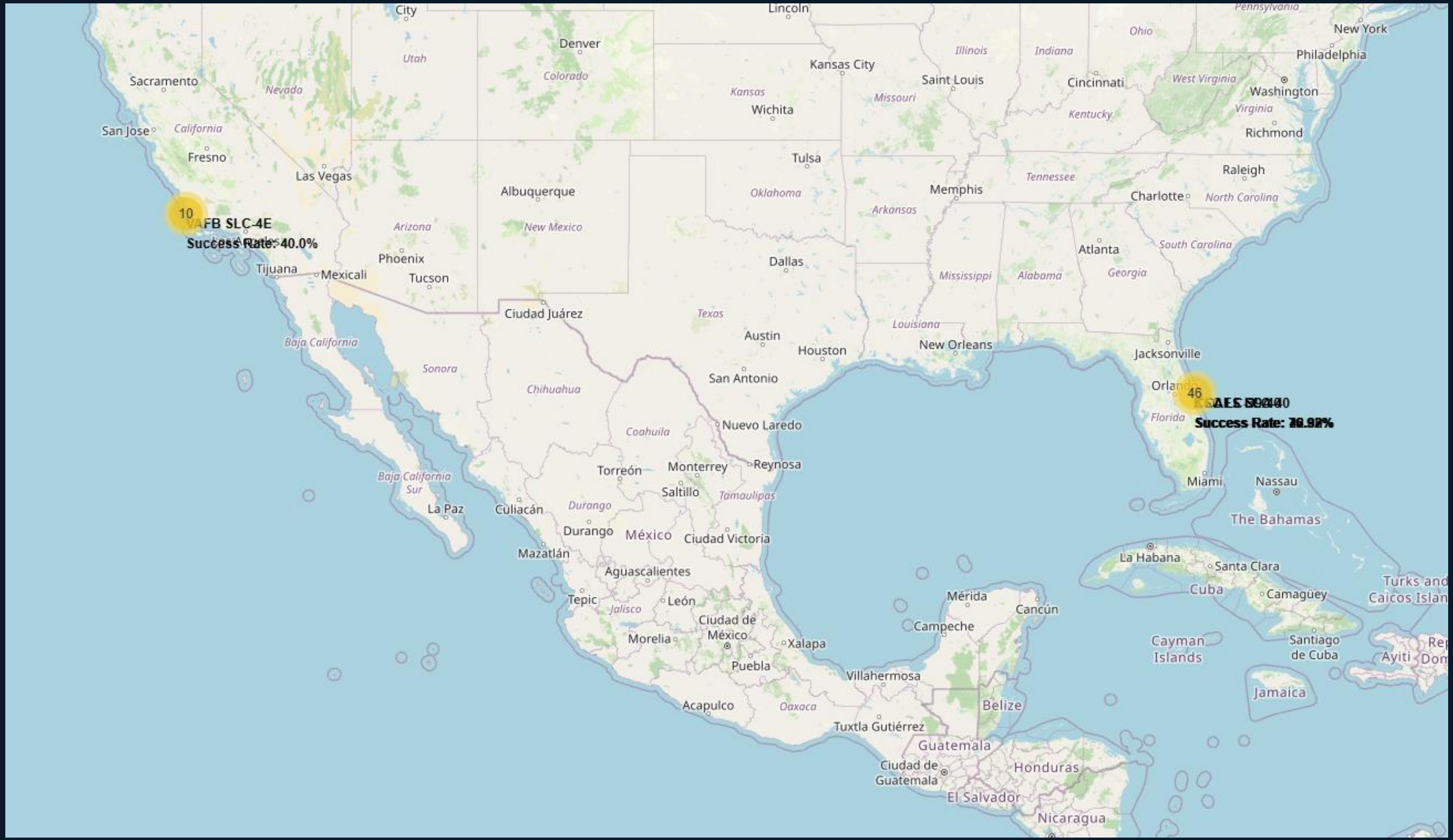
SQL Task 9: Failure (drone ship) month + booster + site

month	Landing Outcome	Booster Version	Launch Site
1	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
4	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

SQL Task 10: Landing outcome totals (2010–2020)

Landing Outcome	total
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

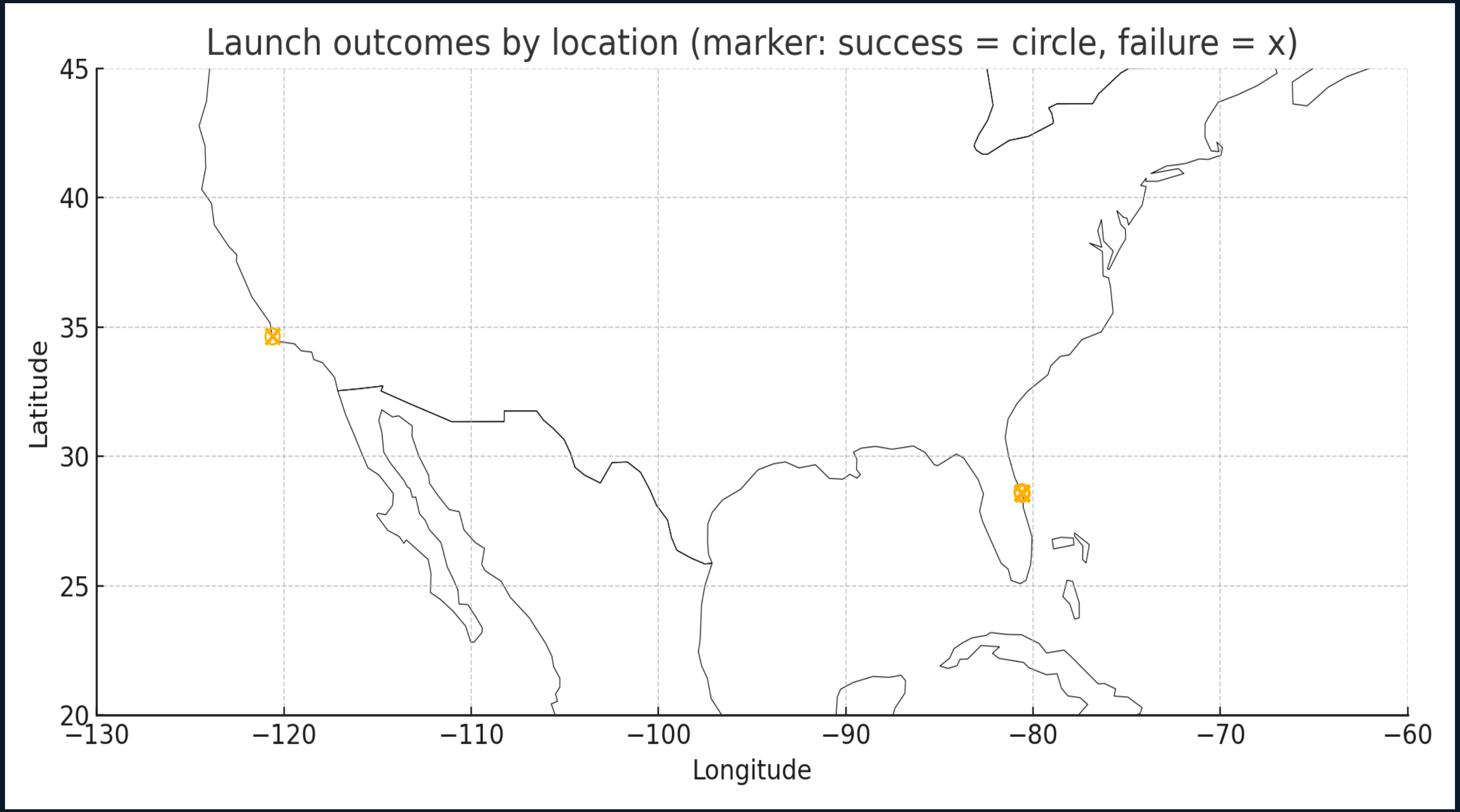
Geospatial analysis: launch sites



Four main sites in the dataset (Florida + California).

Location matters: sea access, infrastructure, and mission orbit constraints.

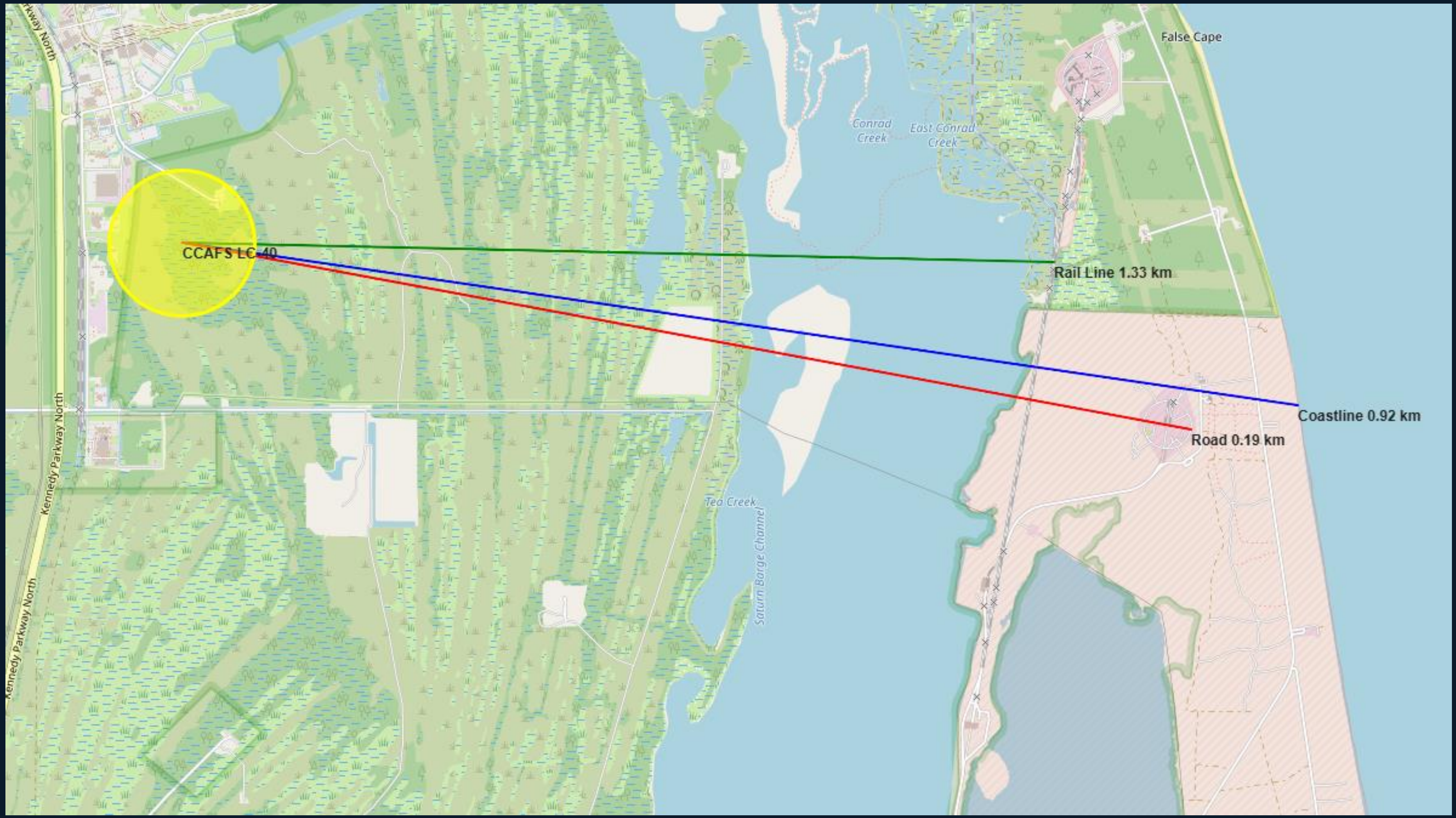
Geospatial analysis: outcomes



Marker legend: circle = success, x = failure (sample).

Even without fancy GIS, the spatial clustering is visible.

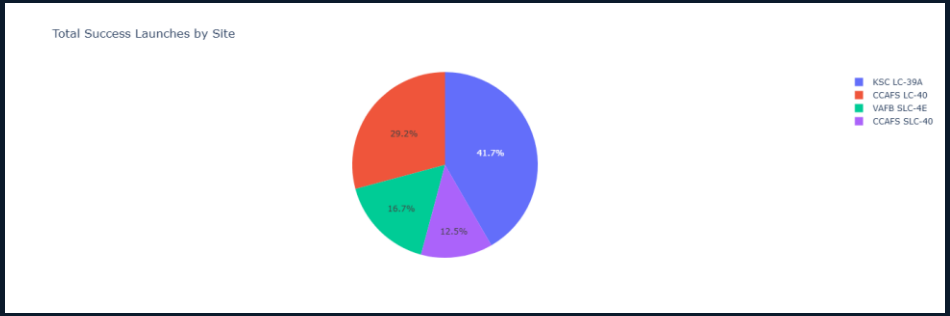
Geospatial analysis: proximity example

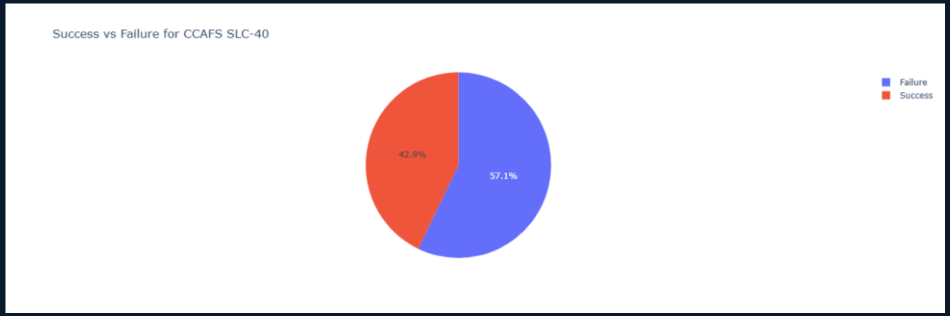


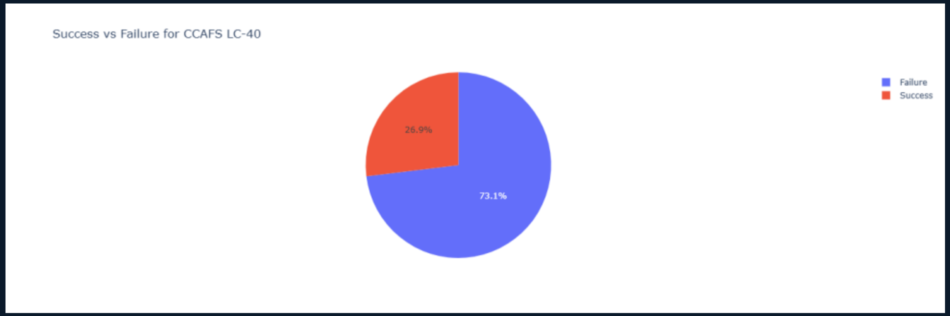
Example distances (approx.): Coast 0.92 km, Rail 1.33 km, Highway 0.19 km.

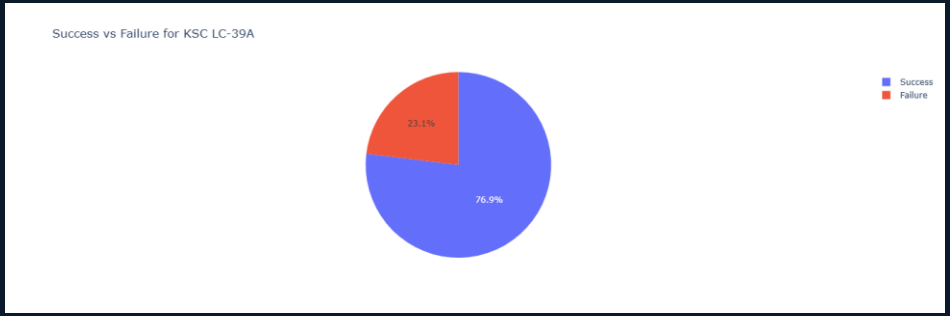
Interpretation: recovery logistics are realistic because infrastructure is close.

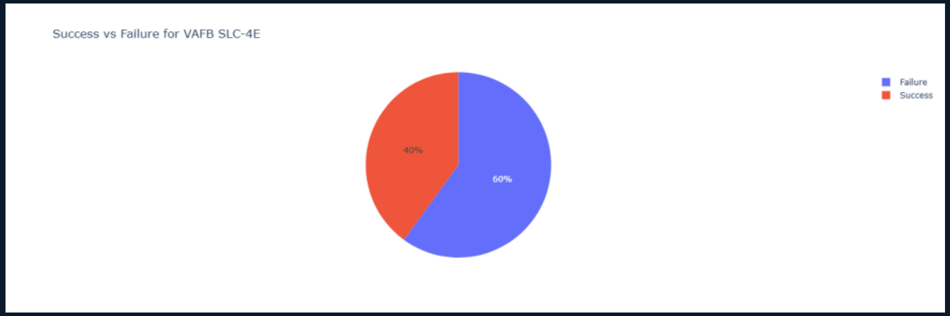
SpaceX Launch Records Dashboard





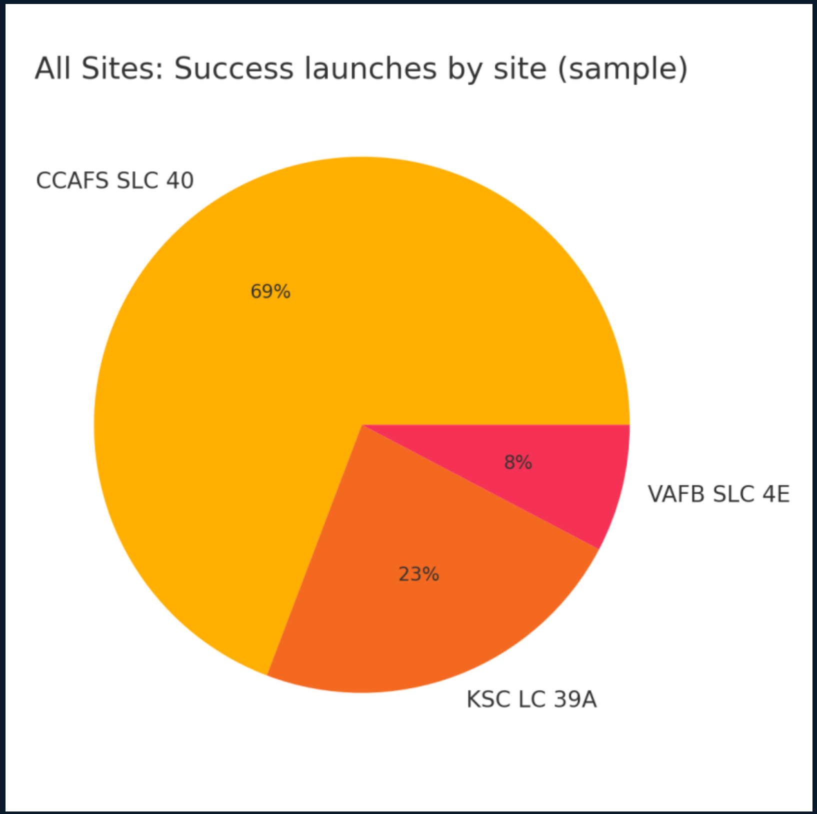


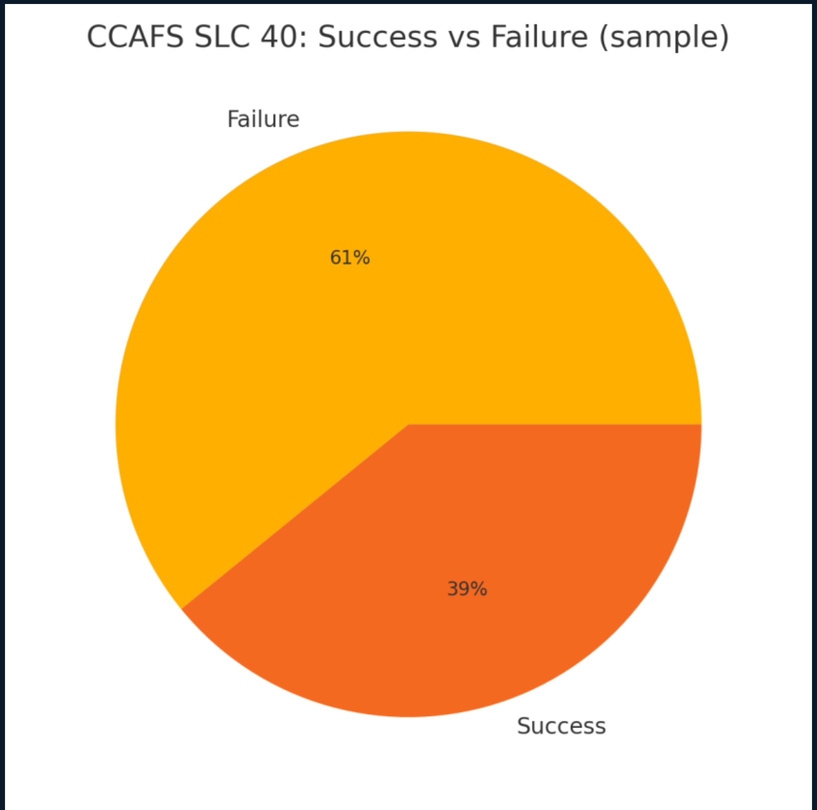




https://github.com/AlessioSantos/IBM-milestone-for-applied-data-science-done/blob/main/jupyter-labs-launch-site-location-v2-done_Alessio.ipynb

Dashboard: pie chart (Selected Site)

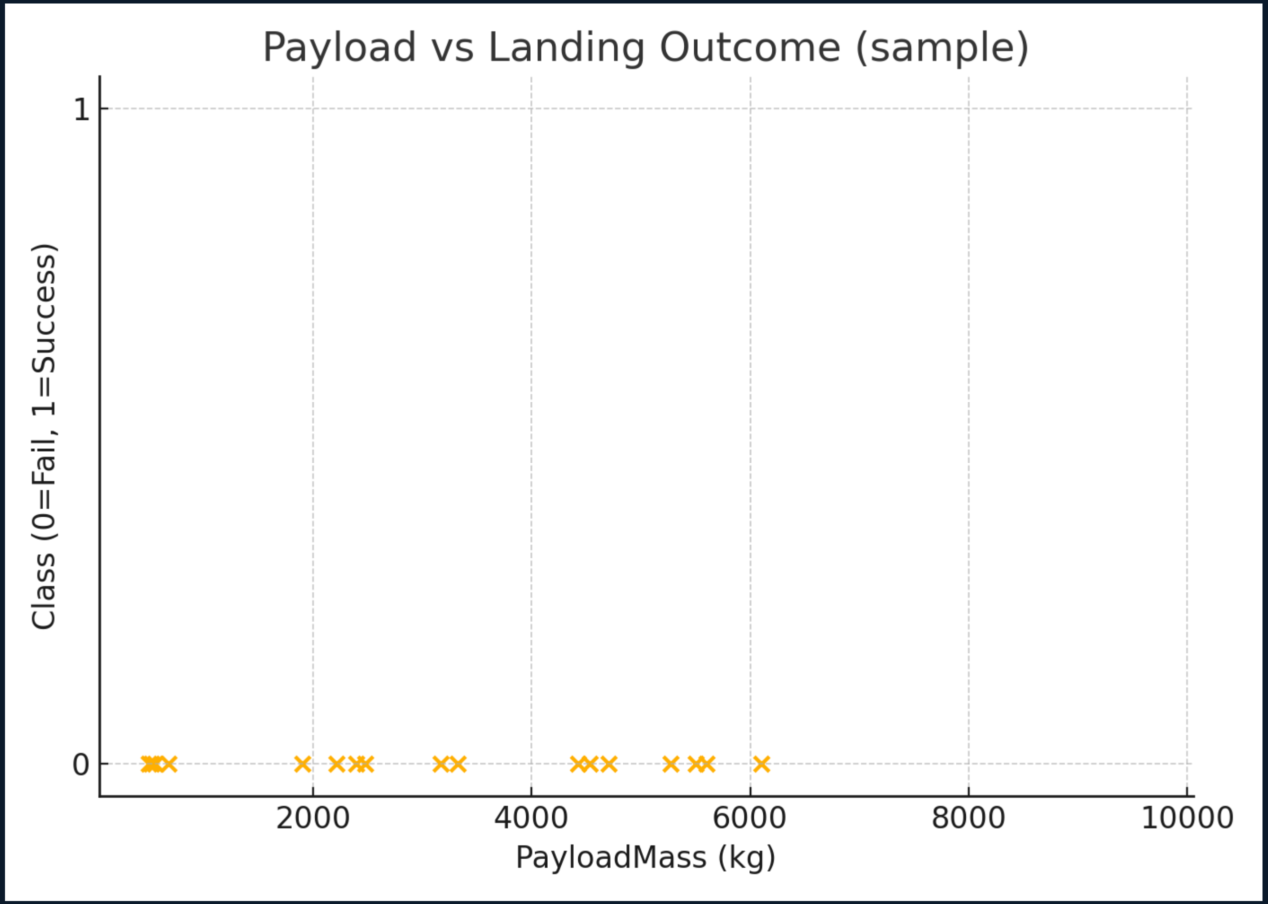




For a chosen site: success vs failure split (sample).

Useful for operational risk estimation per location.

Dashboard: scatter plot (Payload vs Outcome)



Payload mass vs landing outcome (sample).

A perfect separator? No. A useful signal? Absolutely.

Feature engineering

Categorical features (Orbit, LaunchSite) → one-hot encoded.

Numeric features scaled where needed (especially for SVM/KNN).

Train/test split used for fair evaluation.

Then we let four models compete (politely).

```
features = df[["Orbit", "LaunchSite", "LandingPad", "Serial"]]
features = features.dropna()
features = pd.get_dummies(features)
features.head()
```

Orbit_SSO	Orbit_VLEO	LaunchSite_CCAFS SLC 40	...	Serial_B1047	Serial_B1048	Serial_B1049	Serial_B1050	Seria
False	False	True	...	False	False	False	False	False
False	False	True	...	False	False	False	False	False
False	False	True	...	False	False	False	False	False
False	False	True	...	False	False	False	False	False
False	False	False	...	False	False	False	False	False

Machine learning model: Logistic Regression

Baseline linear classifier (fast, interpretable).

Best params: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}

CV Accuracy: 0.8464 | Test Accuracy: 0.8333

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

GridSearchCV

best_estimator_: LogisticRegression

LogisticRegression(C=0.01, penalty='l2')

LogisticRegression

Parameters

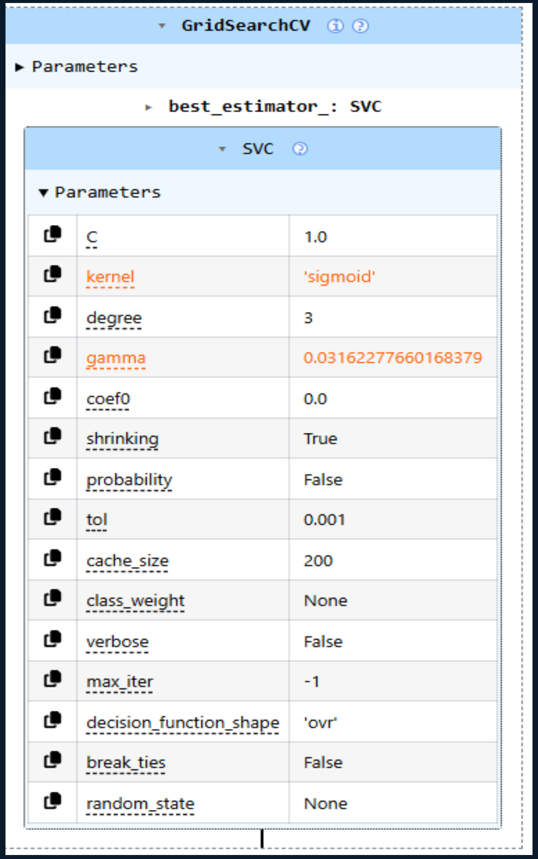
penalty	'l2'
C	0.01
l1_ratio	0.0
dual	False
tol	0.0001
fit_intercept	True
intercept_scaling	1
class_weight	None
random_state	None
solver	'lbfgs'
max_iter	100
verbose	0
warm_start	False
n_jobs	None

Machine learning model: SVM

Non-linear boundary via kernel (powerful but can be picky).

Best params: {'C': 1.0, 'gamma': 0.0316, 'kernel': 'sigmoid'}

CV Accuracy: 0.8482 | Test Accuracy: 0.8333



Machine learning model: Decision Tree

- Captures non-linear rules (“if orbit is X and payload is Y…”).
- Best params: {'criterion': 'entropy', 'max_depth': 8, 'splitter': 'random', ...}
- CV Accuracy: 0.8750 | Test Accuracy: 0.9444
- In this run, it wins. (Sometimes the simplest rulebook works.)

Machine learning model: KNN

Instance-based: predict by looking at nearest neighbours.

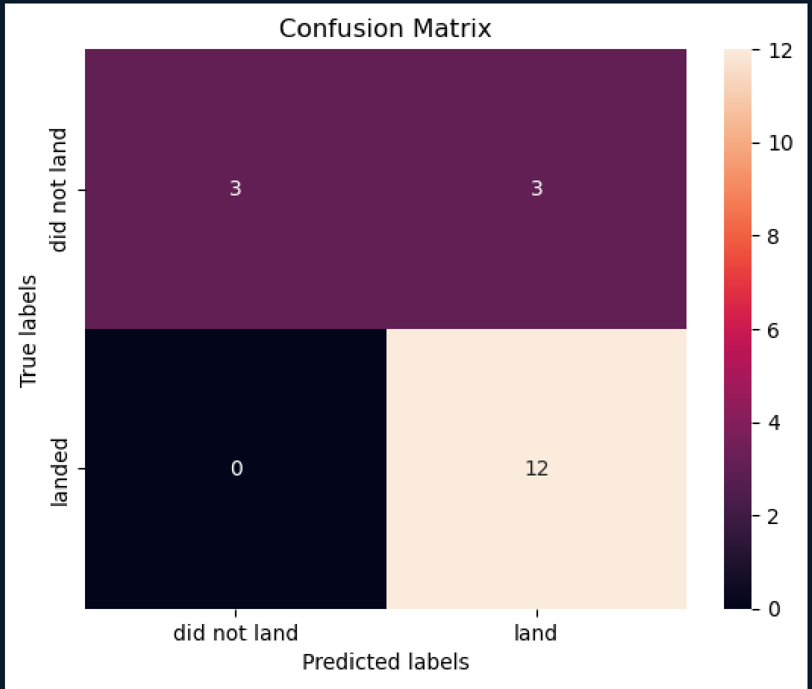
Best params: {'n_neighbors': 10, 'p': 1, 'weights': 'uniform'}

CV Accuracy: 0.8482 | Test Accuracy: 0.8333

```
# 6. Optional: Evaluate on test set
if 'X_test' in locals() and 'y_test' in locals():
    print("Test accuracy: {:.4f}".format(knn_cv.score(X_test, y_test)))
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
Fitting 10 folds for each of 116 candidates, totalling 1160 fits
Tuned hyperparameters (best parameters):  {'n_neighbors': 10, 'p': 1, 'weights': 'uniform'}
Best cross-validation accuracy: 0.8482
tuned hpyerparameters :(best parameters)  {'n_neighbors': 10, 'p': 1, 'weights': 'uniform'}
accuracy : 0.8482142857142858
```

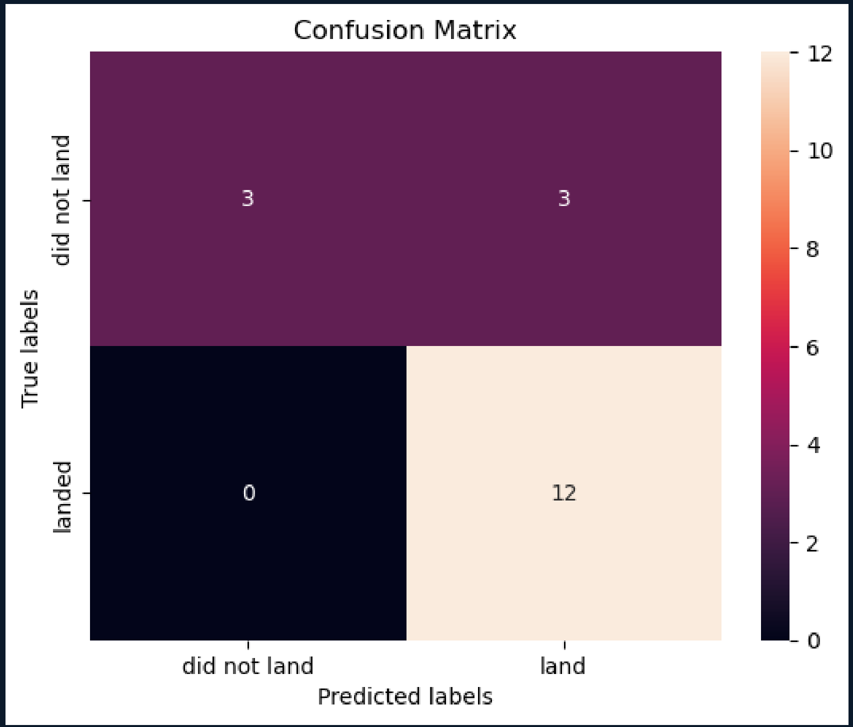
Logistic Regression: confusion matrix



Reads as: how often we predicted success/failure correctly vs incorrectly.

Goal: maximise correct predictions, especially for “success” decisions.

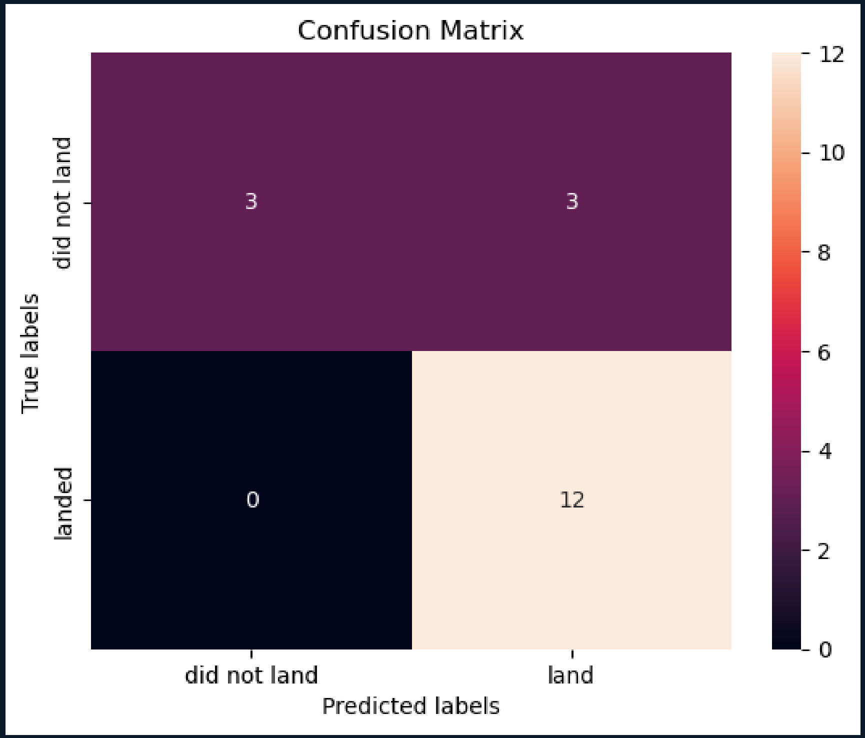
SVM: confusion matrix



Reads as: how often we predicted success/failure correctly vs incorrectly.

Goal: maximise correct predictions, especially for “success” decisions.

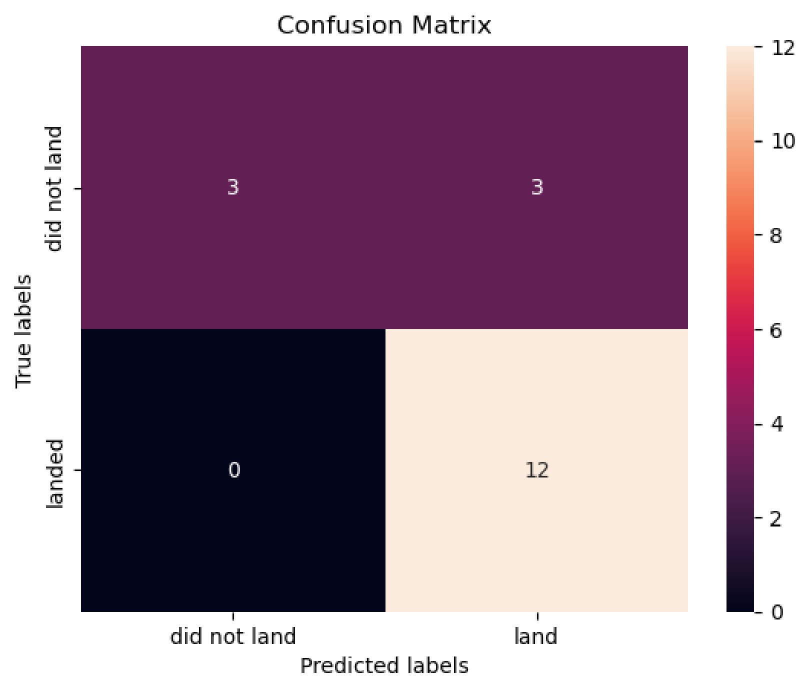
Decision Tree: confusion matrix



Reads as: how often we predicted success/failure correctly vs incorrectly.

Goal: maximise correct predictions, especially for “success” decisions.

KNN: confusion matrix



Reads as: how often we predicted success/failure correctly vs incorrectly.

Goal: maximise correct predictions, especially for "success" decisions.

Model comparison (CV vs Test)

Model	CV Accuracy	Test Accuracy
Logistic Regression	0.8464	0.8333
SVM	0.8482	0.8333
Decision Tree	0.875	0.9444
KNN	0.8482	0.8333

Winner here: Decision Tree (0.9444 test accuracy).

Next step (if we want to be extra careful): cross-validate more folds, test calibration, and watch for overfitting.

Conclusions & next steps

If you remember one thing: orbit + site + payload tell a lot.

We built an end-to-end pipeline from raw data to prediction.

Decision Tree performed best on this run; other models were competitive but slightly behind.

Future work:

use the full dataset and time-aware validation

try ensemble models (Random Forest / Gradient Boosting)

add weather/sea state features if available

And yes, next time we’ll keep GridSearch on a shorter leash ☺!