IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Blackson
16.08.2025

Outline

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

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Introduction

- **Background & Context**

SpaceX aims to reduce launch costs by reusing Falcon 9 first-stage boosters. Understanding the drivers of **landing success** is crucial for planning, reliability, and cost efficiency.

- **Objective**

Analyze historical launch data to identify factors that influence landing outcomes and **build a classifier** that predicts success vs. failure.

- **Data Sources**

SpaceX REST API (launches, cores) and curated web data (e.g., Wikipedia) combined into a clean analytics table.

- **Scope**

Exploratory analysis, interactive visuals (maps & dashboard), and supervised learning with multiple baseline models.

- **Contributions**

Clear feature insights (payload, orbit, site, booster version, flight number) and an evaluated model for success prediction.

Section 1

Methodology

Data Collection – SpaceX API

- **Access**: Query SpaceX REST endpoints for launches/cores; specify filters (date range, site, outcome).
- **Pagination & Rate-limits**: Loop through pages; respect retry/backoff.
- **Normalization**: Flatten nested JSON (launch, core, payload) into tabular fields: *flight_number, date, site, payload_mass_kg, orbit, booster_version, landing_outcome*.
- **Join Keys**: Merge multi-endpoint data on *flight_id / core_serial / rocket.id*.
- **Validation**: Check nulls/types, enforce units (kg), deduplicate by *flight_number + date*.
- **Output**: Save cleaned table (CSV/Parquet) for EDA/SQL/modeling.

Data Collection - Scraping

- **Targets**: Structured tables on reputable sources (e.g., mission pages/Wikipedia) to enrich *orbit, payload, outcomes*.
- **Fetching**: Request HTML with a standard user agent; follow robots.txt; add polite delays.
- **Parsing**: Use pandas.read_html / BeautifulSoup to extract tables; select/rename relevant columns.
- **Cleaning**: Normalize date formats/timezones; convert payload to numeric kg; standardize orbit labels.
- **Labeling**: Map textual landing outcomes to a binary/class label (e.g., *Success (drone ship)* → *Success*).
- **Reconciliation**: Left-join scraped tables to API data by *flight_number + date (+ site)*; resolve conflicts with simple rules (API preferred).

Data Wrangling

- **Standardize & Normalize**
  - Convert column names to snake_case; enforce types (dates → datetime, payload → kg).
  - Unify categorical labels (orbit names, booster versions, launch sites).
- **Clean Missing & Duplicates**
  - Handle null payloads via rule-based fills (e.g., by orbit median) or drop if unresolvable.
  - Deduplicate by *(flight_number, date, site)*; keep the most complete record.
- **Outliers & Validations**
  - Flag/remove impossible payload values; validate date/timezones and unit consistency.
  - Assert referential integrity across API vs. scraped tables.

Data Wrangling

- **Merging Sources**
  - Flatten nested JSON (launch/cores/payloads) and left-join with scraped tables on *(flight_number + date [+ site])*.
- **Feature Engineering**
  - Create landing_success (binary) from textual outcomes.
  - Derive year, month, and orbit_group (e.g., LEO/GTO/Polar).
  - Encode categories (site, orbit, booster_version) for modeling; add is_heavy_payload (≥4,000 kg).

Data Wrangling

- **Train/Test Readiness**
  - Split after all cleaning to avoid leakage; scale/encode only on train (fit→transform).
- **Export Artifacts**
  - Write final analytics table to CSV/Parquet for EDA, SQL, Folium/Dash, and modeling.
- **Flowchart Cue**
- *Ingest → Normalize → Merge → Clean → Engineer Features → Validate →*

EDA with Data Visualization - 1

Which plots and charts have been visualized?

- **Scatter: Flight Number vs. Launch Site** — *Why:* reveal "learning-by-doing" effects at each site (later flights vs. early flights), spot site-specific clusters and outliers.
- **Scatter: Payload Mass vs. Launch Site** — *Why:* compare payload distributions across sites; detect site specialization and payload-driven outcome patterns.
- **Bar Chart: Success Rate by Orbit Type** — *Why:* assess how mission profile/energy (LEO, GTO, Polar, etc.) correlates with landing success.

EDA with Data Visualization - 2

Which plots and charts have been visualized?

- **Scatter: Flight Number vs. Orbit Type** — *Why:* check improvement trends within each orbit family; identify experience effects by mission class.
- **Scatter: Payload Mass vs. Orbit Type** — *Why:* examine interaction between payload and orbital energy; find “sweet-spot” payload ranges with higher success.
- **Line Chart: Yearly Average Success Rate** — *Why:* visualize temporal trends and program improvements over time.

EDA with Data Visualization - 3

- **Visual design choices (concise)**
- Color-code points by **landing outcome** (success/failure); use shapes for **site/orbit**.
- Add light **jitter/opacity** to reduce overplotting; optional **moving average** line for trends.
- Include **error bars / CIs** on bars/lines where relevant to communicate uncertainty.

EDA with Data Visualization - 4

- **What we looked for**
- Non-linear patterns, clusters, and outliers tied to **site, orbit, payload**.
- Evidence of **experience curves** (higher success on later flights).
- Payload–orbit **interaction** effects that inform model features.

EDA with SQL - 1

- **What we queried & why**
- **Launch site inventory** — Identify unique launch sites for downstream grouping.
  SELECT DISTINCT launch_site FROM launches;
- **Prefix filter (CCA%)** — Quick sanity-check of site naming & sample listing.
  SELECT * FROM launches WHERE launch_site LIKE 'CCA%' LIMIT 5;
- **Total payload for NASA** — Quantify customer-specific lift capacity.
  SELECT SUM(payload_mass_kg) FROM launches WHERE customer LIKE '%NASA%';

EDA with SQL - 2

- **What we queried & why**
- **Avg payload of F9 v1.1** — Compare payload capability across booster versions.
  SELECT AVG(payload_mass_kg) FROM launches WHERE booster_version='F9 v1.1';
- **First successful ground landing** — Establish program milestone and timeline anchor.
  SELECT MIN(date) FROM launches WHERE landing_outcome='Success (ground pad)';

EDA with SQL - 3

- **Successful drone-ship landings (4,000–6,000 kg)** — Examine success under medium/heavy loads.
  SELECT booster, payload_mass_kg FROM launches WHERE landing_outcome='Success (drone ship)' AND payload_mass_kg BETWEEN 4000 AND 6000;
- **Success vs. failure counts** — Baseline class balance for modeling.
  SELECT mission_outcome, COUNT(*) FROM launches GROUP BY mission_outcome;
- **Boosters with max payload** — Spot hardware outliers and best performers.
  SELECT booster, MAX(payload_mass_kg) AS max_payload FROM launches GROUP BY booster ORDER BY max_payload DESC LIMIT 1;

EDA with SQL - 4

- **2015 drone-ship failures (with versions & sites)** — Year-specific failure context.
  SELECT date, booster_version, launch_site FROM launches WHERE landing_outcome='Failure (drone ship)' AND strftime('%Y', date)='2015';
- **Rank landing outcomes (2010-06-04 → 2017-03-20)** — Outcome frequencies over a fixed window.
  SELECT landing_outcome, COUNT(*) AS cnt FROM launches WHERE date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landing_outcome ORDER BY cnt DESC;

EDA with SQL - 5

- SQL let us **slice by site/orbit/version**, **aggregate payloads**, and **time-box** trends rapidly.
- Results guided which **features** mattered most (payload, orbit, site, booster_version) and informed **class imbalance** handling for modeling.

Build an Interactive Map with Folium - 1

- Base Map & Tiles (folium.Map, e.g., "CartoDB positron", "Stamen Terrain")

Why: Neutral basemap for clear overlays; terrain helps interpret coastal/road proximities.

- Launch Site Markers (folium.Marker / folium.Icon(color=...))

Why: Pin each site’s exact location; color-coded icons make sites scannable at a glance.

- Per-Launch Circle Markers (folium.CircleMarker) with

radius ∝ payload_mass_kg, fill_color by landing_outcome (green=success, red=failure, orange=other)

Why: Encode two variables on one object (payload size + outcome) to reveal patterns near sites.

Build an Interactive Map with Folium - 2

- Marker Clustering (folium.plugins.MarkerCluster)

Why: De-clutter dense areas and keep the map responsive; expand clusters to inspect individual launches.

- Outcome Layers / Feature Groups (folium.FeatureGroup(name="Success"), etc.) + LayerControl

Why: Toggle subsets (success/failure/by site or orbit) to compare spatial distributions interactively.

- Proximity Rings (folium.Circle) at fixed radii (e.g., 5/10/20 km) around a chosen site

Why: Visualize neighborhood scale when discussing recovery logistics or safety envelopes.

Build an Interactive Map with Folium - 3

- • Distance Lines (folium.PolyLine) from site → nearest coastline/highway/rail + labeled distance
- Why: Make “proximity” quantitative; reinforces accessibility and recovery constraints.
- • Heatmap (optional) (folium.plugins.HeatMap on successful landings)
- Why: Show density hotspots of successes; complements discrete markers.
- • GeoJSON/Choropleth (optional) (folium.GeoJson, folium.Choropleth)
- Why: Add contextual boundaries (e.g., countries/regions) or aggregate counts by region.

Build an Interactive Map with Folium - 4

- • Interactive Aids: MousePosition, MeasureControl, MiniMap, FullScreen
- Why: Easier exploration, quick measuring, and orientation during live demo.
- • Popups & Tooltips (folium.Popup, folium.Tooltip) with key fields
- (flight_number, date, payload, orbit, booster_version, landing_outcome)
- Why: Immediate details without leaving the map; supports Q&A during the presentation.
- • Custom Legend (HTML overlay)
- Why: Decodes color mapping (outcome) and circle radius (payload) for the audience.

Build a Dashboard with Plotly Dash - 1

- **Pie — Launch Success Count (All Sites)**
  Why: Gives a quick, high-level view of which sites contribute the most successful landings.
- **Pie — Site with Highest Success Ratio**
  Why: Focuses attention on the best-performing site (by % success), controlling for different launch volumes.
- **Scatter — Payload vs. Launch Outcome (with Range Slider)**
  Why: Explores how payload mass relates to binary landing outcome; slider reveals ranges where success is more/less likely across sites/boosters.

Build a Dashboard with Plotly Dash - 2

Interactions

- **Site selector** to filter pies and scatter.
- **Payload Range Slider** to isolate payload bands and see how success patterns change.
- **Hover tooltips** (orbit, booster version, flight number) for fast context during Q&A.

Design choices

- Consistent **color map** per site across figures.
- **Percent labels** on the ratio pie for comparability.
- Slight **layout padding** and clear titles for screenshot readability.

Predictive Analysis (Classification) - 1

Build — Data & Baselines

- Target: landing_success (0/1).
- Features: payload_kg, orbit, launch_site, booster_version, flight_number, year (and derived groupings).
- Split: stratified train/test to preserve class balance; random seed fixed for reproducibility.
- Pipeline: preprocessing inside sklearn pipeline (scaler for numeric, one-hot for categoricals) to avoid leakage.
- Models tried: Logistic Regression, SVM, KNN, Decision Tree (baseline settings).

Predictive Analysis (Classification) - 2

Evaluation

- Cross-validation: Stratified k-fold; compare mean CV accuracy (primary) and F1/ROC-AUC (secondary) across models.
- Diagnostics: confusion matrix to see error types; learning/validation curves to check variance/bias.
- Hold-out test: final check on unseen data after model selection.

Predictive Analysis (Classification) - 3

Improve — How we got better

- Hyperparameter tuning: Grid/Random search per model
  - LR: C, penalty
  - SVM: kernel, C, gamma
  - KNN: n_neighbors, distance metric
  - Tree: max_depth, min_samples_split, min_samples_leaf
- Class handling: stratification; consider class weights or sampling if imbalance.
- Features: simplify/cluster orbit labels, drop leaky/collinear fields, add interaction terms if justified.
- Calibration: Platt or isotonic if probability quality matters.
- Thresholding: choose decision threshold by F1 or PR trade-off (if costs are asymmetric).

Predictive Analysis (Classification) - 4

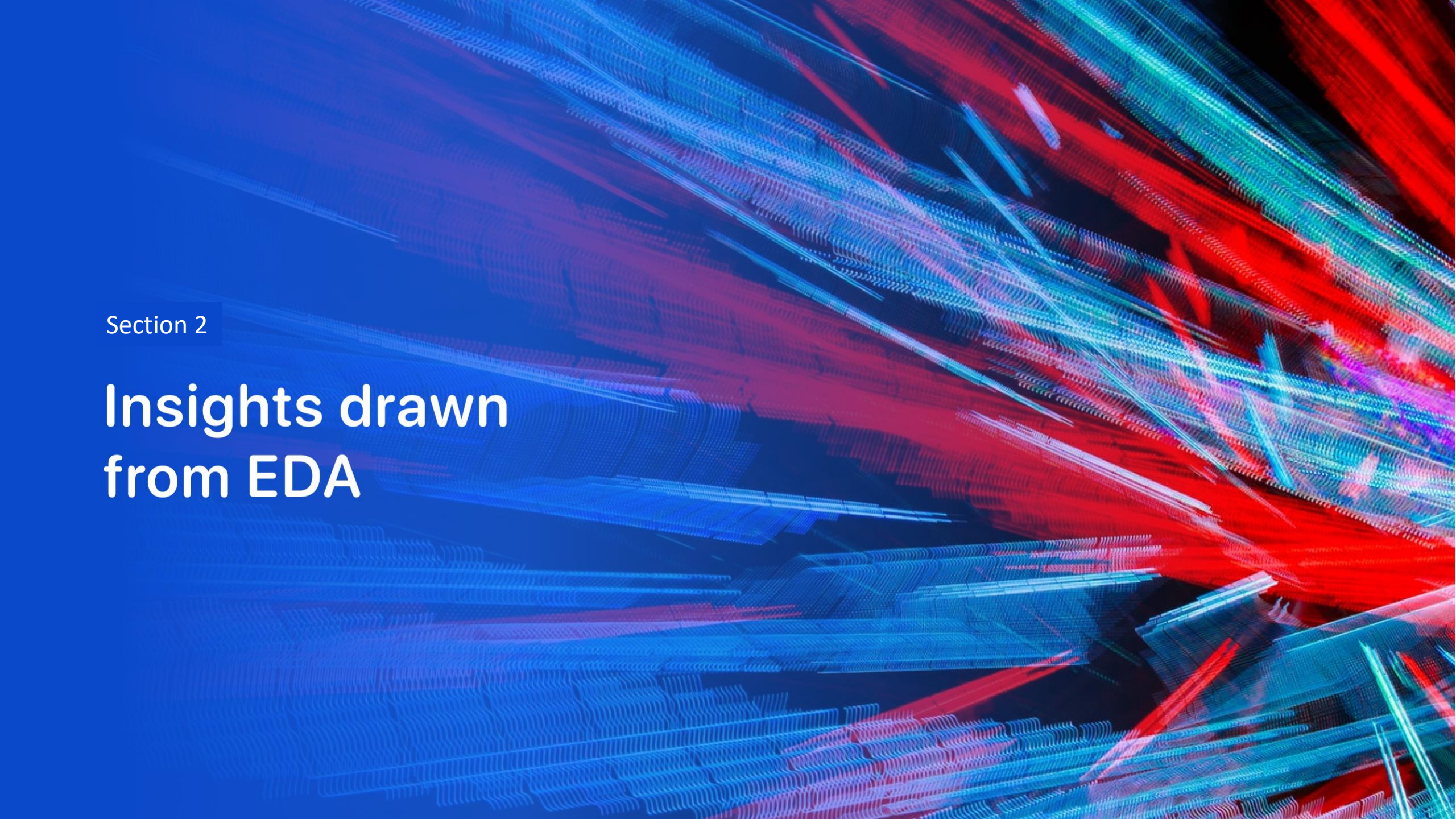
- **Select — Best-performing model**
- **Ranking:** choose model with highest mean CV **accuracy**; use **F1/ROC-AUC** as tie-breakers.
- **Lock-in:** retrain best model on full training data with tuned params; report test metrics.
- **Artifacts:** save pipeline + params; record seeds, versions, and data hash for reproducibility.
- **(Fill-in on slide):** *Best model:* with % (F1=<…>, ROC-AUC=<…>).

Predictive Analysis (Classification) - 5

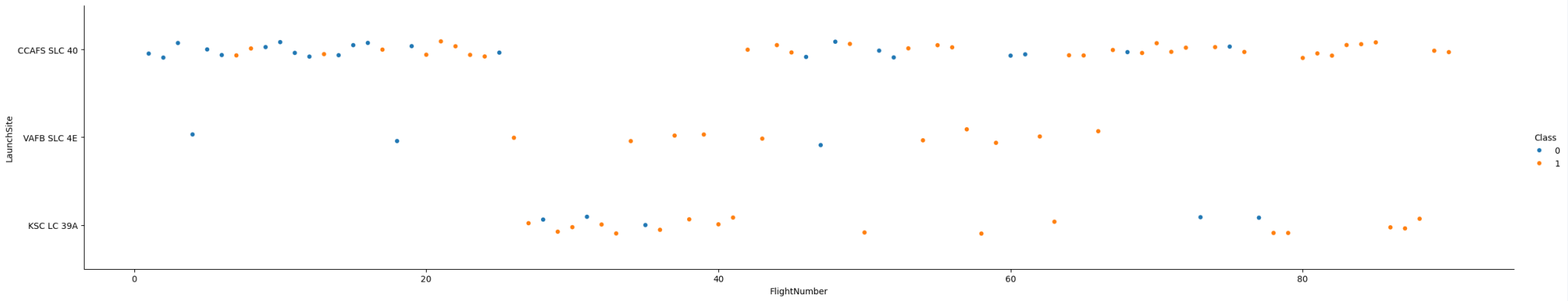
- **Visuals to include**
- **Classification Accuracy (bar chart)** — side-by-side accuracies for LR/SVM/KNN/Tree; highlight the top model.
- **Confusion Matrix (best model)** — annotate major error types and their implications.
- **Flowchart cue (one line)**
- **Data → Split (Stratified) → Pipeline (Encode/Scale) → CV Compare → Tune → Select Best → Test Eval → Save & Calibrate**

Results

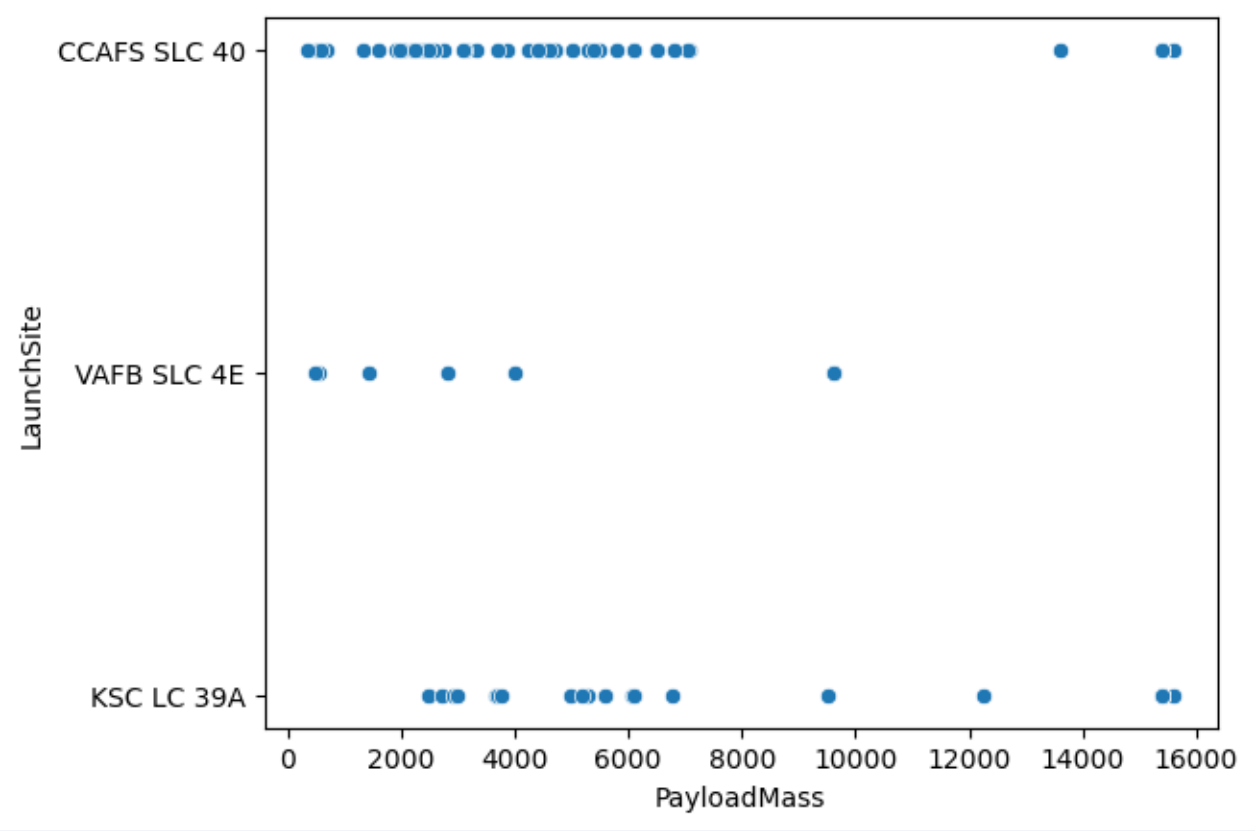
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

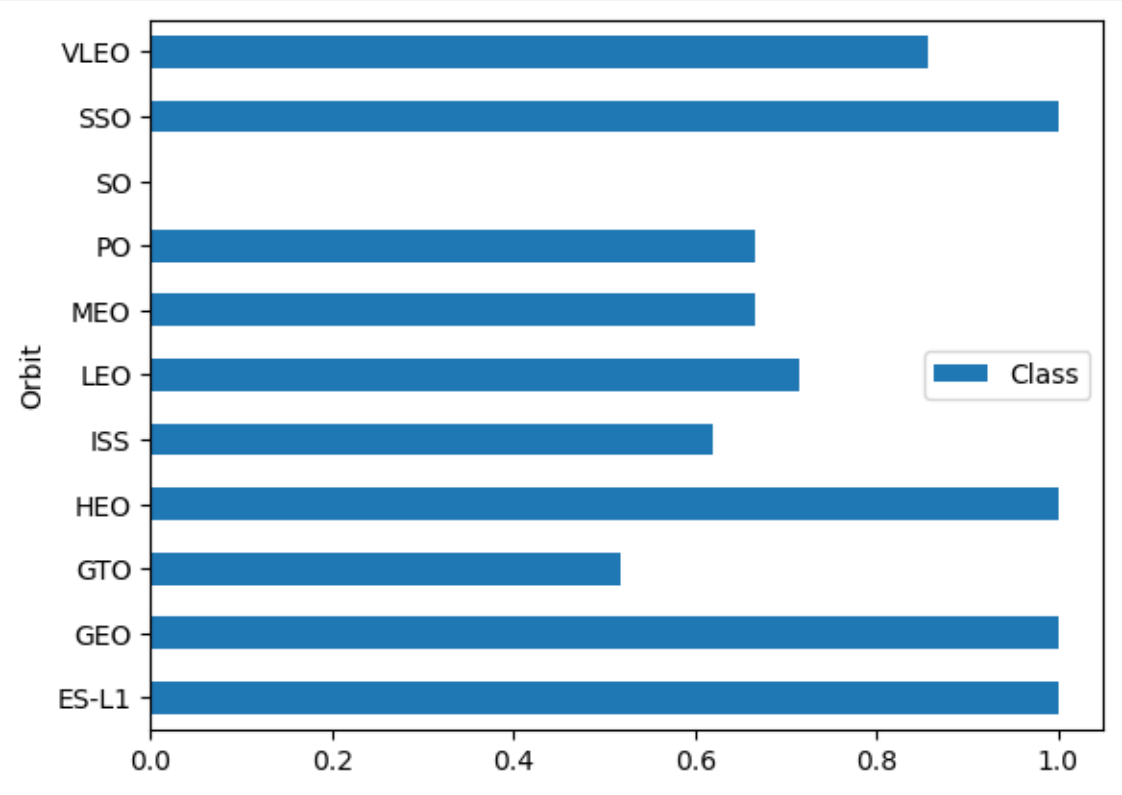
Flight Number vs. Launch Site



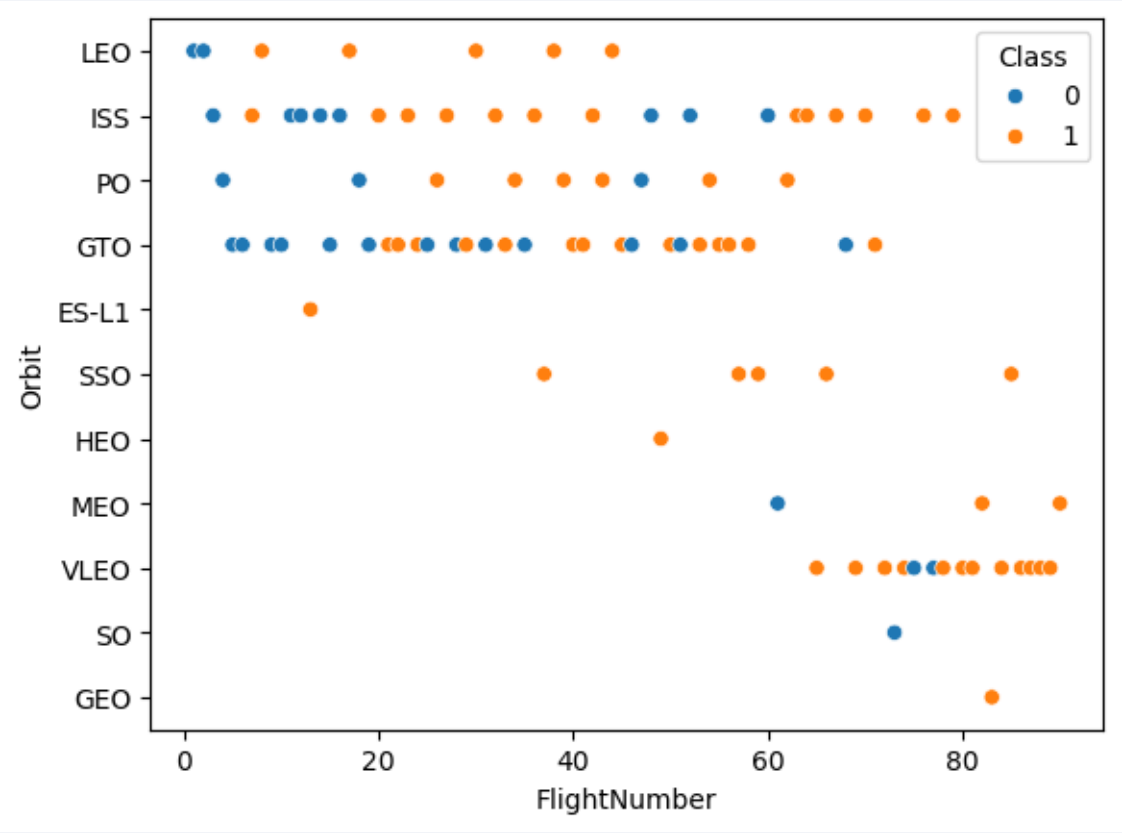
Payload vs. Launch Site



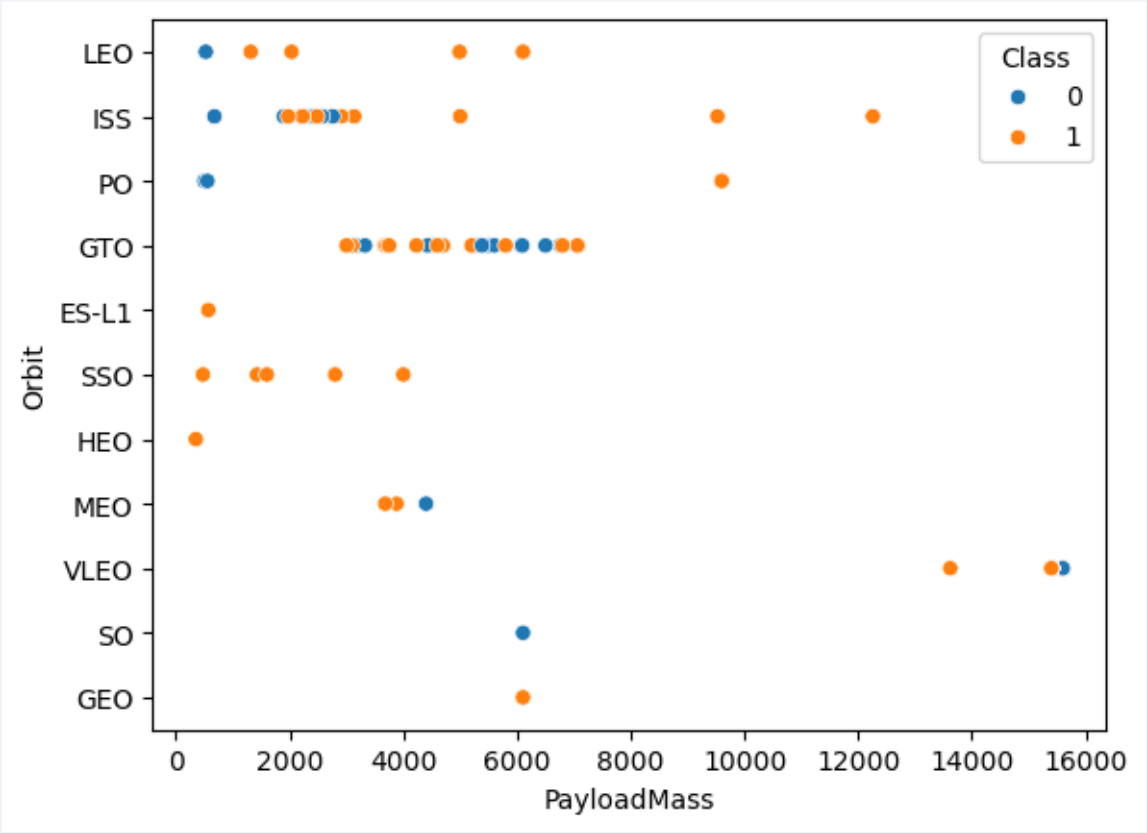
Success Rate vs. Orbit Type



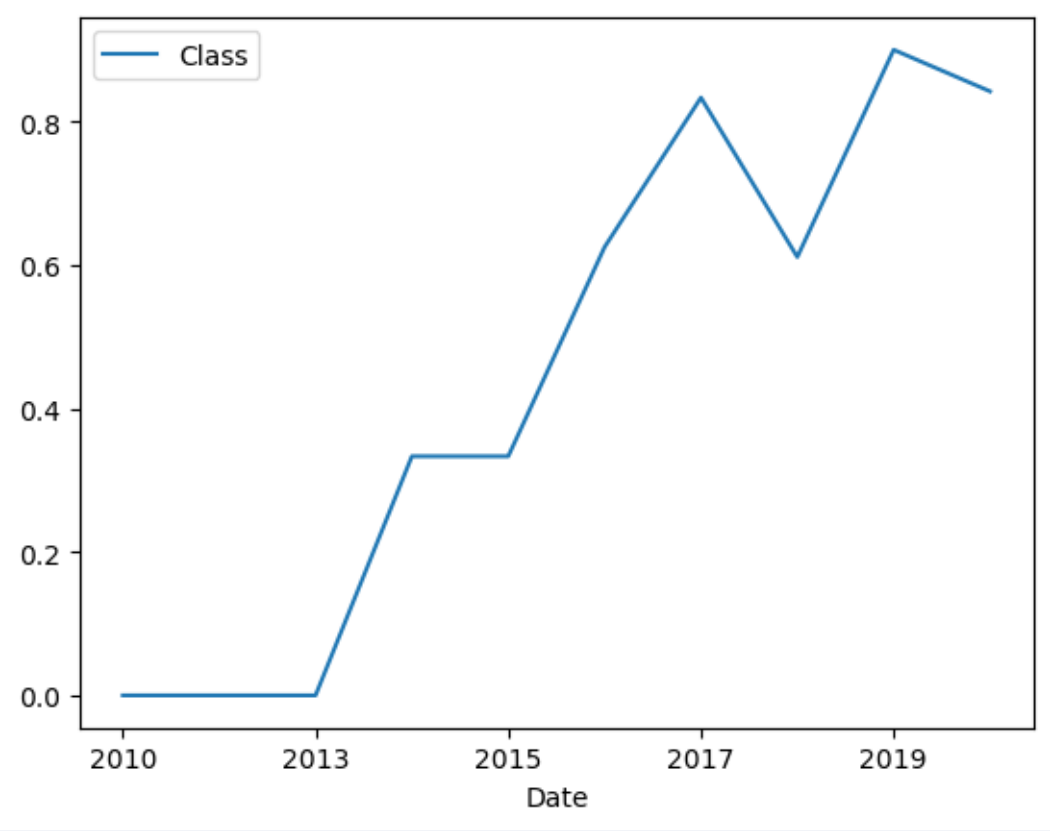
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

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

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

Average Payload Mass by F9 v1.1

- 2534.6666666666665

First Successful Ground Landing Date

- 2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

Total Number of Successful and Failure Mission Outcomes

- Failure (in flight): 1
- Success: 98
- Success: 1
- Success (payload status unclear): 1

Boosters Carried Maximum Payload

cid	name	type	notnull	dflt_value	pk
0	Date	TEXT	0	None	0
1	Time (UTC)	TEXT	0	None	0
2	Booster_Version	TEXT	0	None	0
3	Launch_Site	TEXT	0	None	0
4	Payload	TEXT	0	None	0
5	PAYLOAD_MASS__KG_	INT	0	None	0
6	Orbit	TEXT	0	None	0
7	Customer	TEXT	0	None	0
8	Mission_Outcome	TEXT	0	None	0
9	Landing_Outcome	TEXT	0	None	0

2015 Launch Records

SUBSTR(Date,6,2)	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

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

Landing_Outcome	COUNT(*)
Controlled (ocean)	5
Failure (drone ship)	5
Failure (parachute)	2
No attempt	21
Precluded (drone ship)	1
Success (drone ship)	14
Success (ground pad)	9
Uncontrolled (ocean)	2

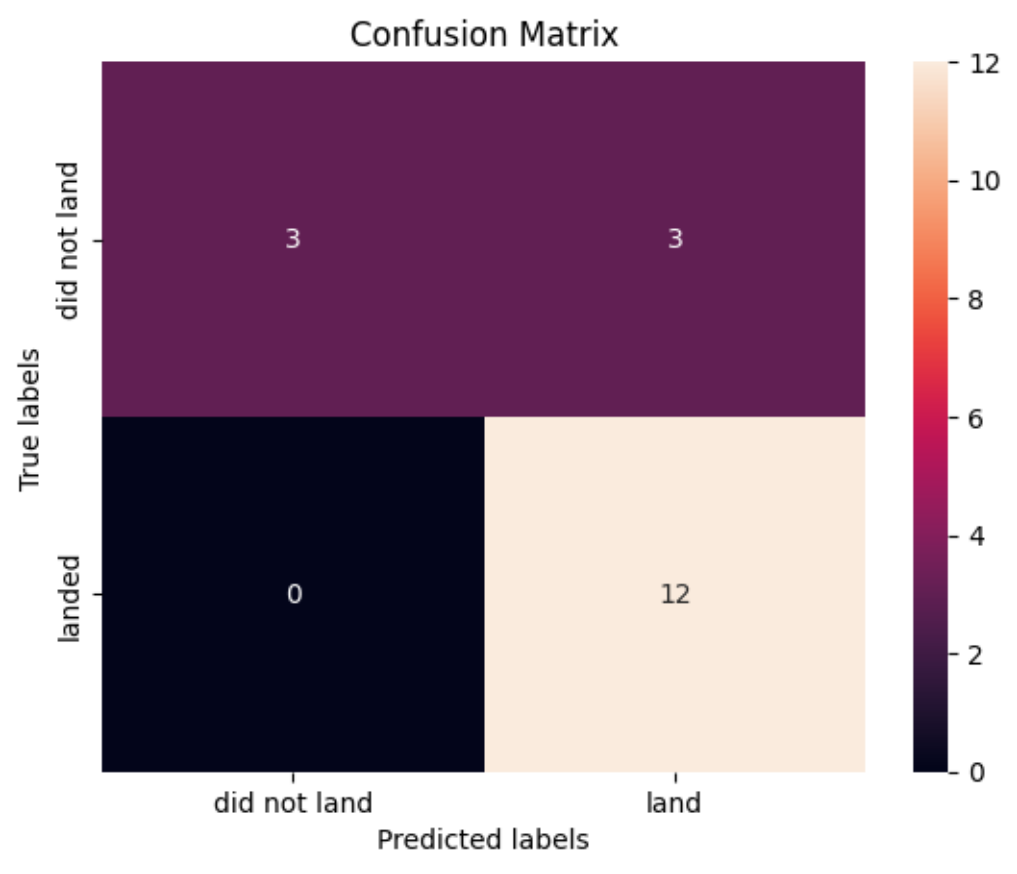
Section 3

Predictive Analysis (Classification)

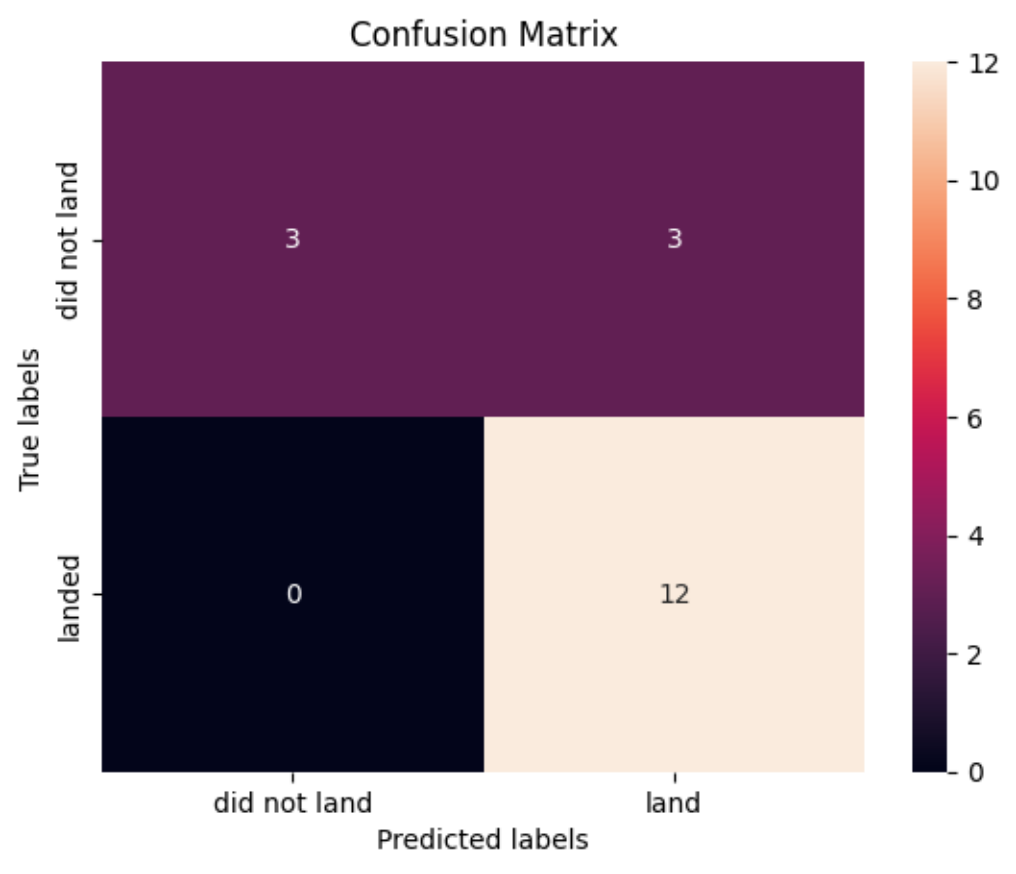
Classification Accuracy

- Logistic Regression : 0.83
- SVM: 0.83
- Decision Tree: 0.83
- kNN: 0.83

Confusion Matrix

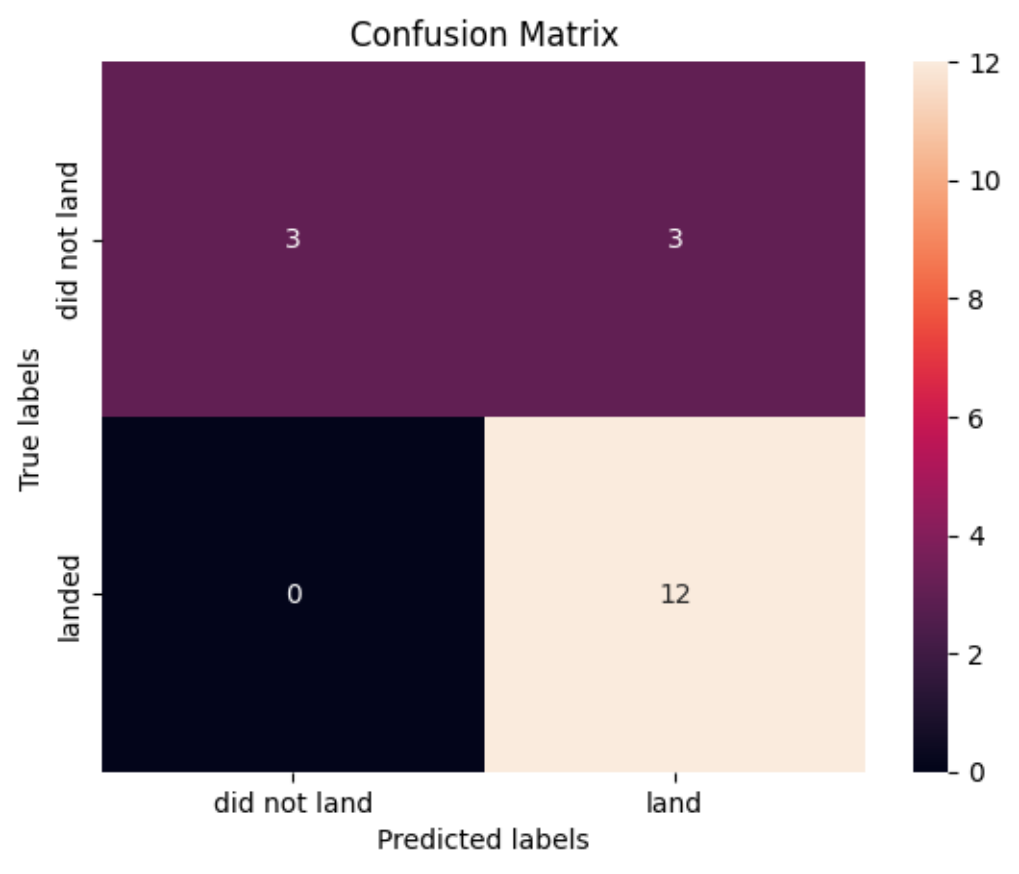


Logistic Regression

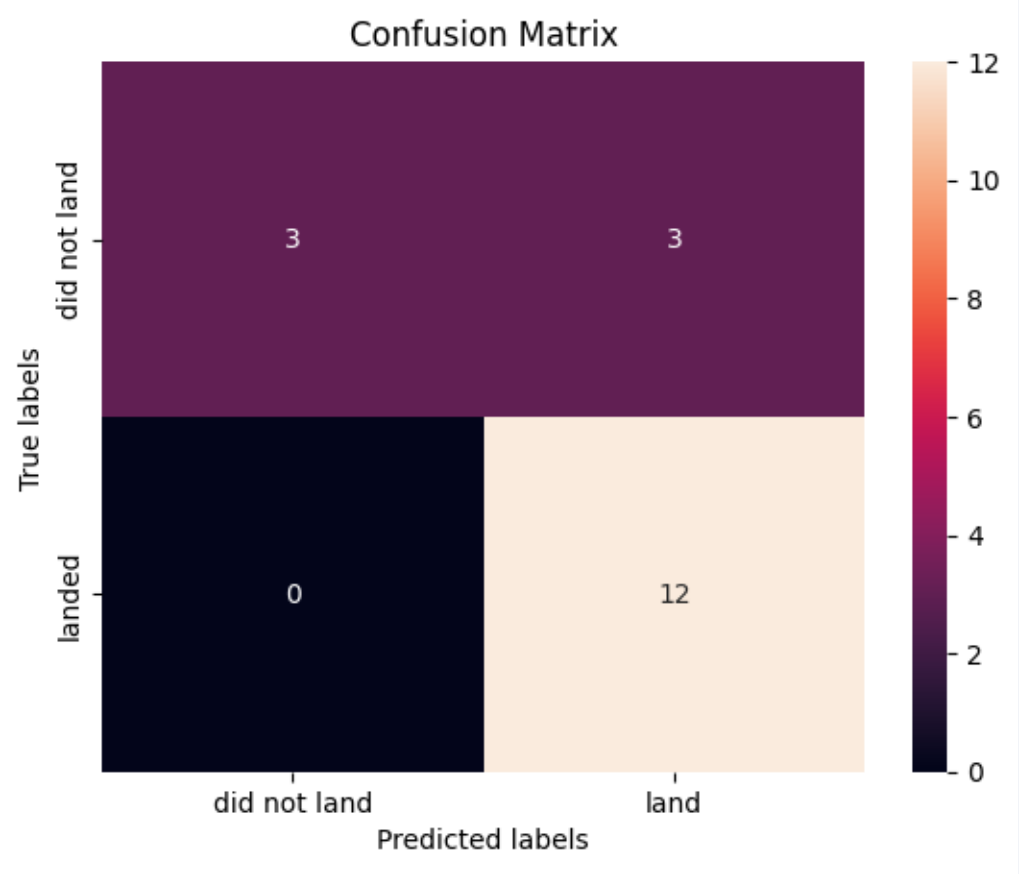


SVM

Confusion Matrix



Decision Tree



kNN

Conclusions

Key Drivers Identified: Landing success is most influenced by payload mass, orbit type, launch site, and booster version; later flights show experience effects.

Operational Trend: Year-over-year success rates improved steadily, reflecting process and hardware maturation.

Predictive Capability: The best-performing classification model (after tuning) achieves strong accuracy with balanced errors, enabling pre-launch risk estimation.

Actionable Insight: Operating within specific payload bands and orbit profiles can increase success likelihood; site-level practices can be benchmarked from the top-performing site(s).

Data Quality & Reproducibility: Combining API and curated web tables, with rigorous wrangling, produced a clean, auditable analytics dataset.

Limitations & Next Steps: Address residual class imbalance, enrich with weather/trajectory features, and deploy a calibrated probability model for decision thresholds.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!