IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Motea Alsayadi
04/01/2026

Outline

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

Executive Summary

Summary of Methodologies

- Data Acquisition: Web scraping (BeautifulSoup), SQL database integration, geospatial mapping (Folium)
- Machine Learning: Four classification models (Logistic Regression, SVM, Decision Trees, KNN) with GridSearchCV optimization and 10-fold cross-validation
- Interactive Dashboard: Plotly Dash application with real-time analytics, dual-input controls (site dropdown + payload slider)
- Geographic Analysis: Proximity calculations using Haversine formula, launch site infrastructure mapping

Summary of All Results

- Launch Performance: CCAFS SLC-40 has most launches (38), KSC LC-39A has highest success rate (94%)
- ML Predictive Accuracy: All four models achieved 83.33% test accuracy with 15/18 correct predictions
- Success Factors: FT booster version performs best; optimal payload range: 3,000-7,000 kg
- Geographic Insights: All sites coastal (<3km from shoreline), maintain safety buffers from populated areas
- Dashboard Analytics: Interactive visualization confirms payload optimization and booster evolution patterns

Introduction

Project Background and Context

SpaceX has revolutionized space launch economics through reusable rocket technology. With Falcon 9 launch costs at $62M versus competitors' $165M+, much of the savings comes from recovering and reusing the first stage. This project analyzes SpaceX's historical launch data to understand the factors influencing launch success and first-stage landing outcomes. The research provides critical insights for competitive bidding, mission planning, and space industry stakeholders seeking to understand SpaceX's operational advantages and reliability patterns.

Problems We Want to Find Answers

- Launch Success Determinants: What factors most influence whether a Falcon 9 launch will succeed or fail?
- Geographic Optimization: Why are launch sites located where they are, and what geographic patterns correlate with success?
- Payload Economics: How does payload mass affect mission outcomes, and what are the optimal payload ranges?
- Technology Evolution: Which booster versions perform best, and how has success improved over time?
- Predictive Capability: Can we accurately predict launch outcomes to support mission planning and risk assessment?
- Infrastructure Analysis: What transportation and logistical patterns support successful launch operations?

Section 1

Methodology

Data Collection

API Integration Strategy

- SpaceX REST API calls using Python requests library
- Multiple endpoint targeting (launches, rockets, launchpads, payloads, cores)
- Static JSON fallback for consistent data retrieval
- Cross-referenced data enrichment through ID-based API calls

Data Enrichment Functions

- `getBoosterVersion()`: Rocket name extraction from rocket ID
- `getLaunchSite()`: Geographic coordinates and site details
- `getPayloadData()`: Mass and orbit information from payload IDs
- `getCoreData()`: Landing outcomes and booster reuse metrics

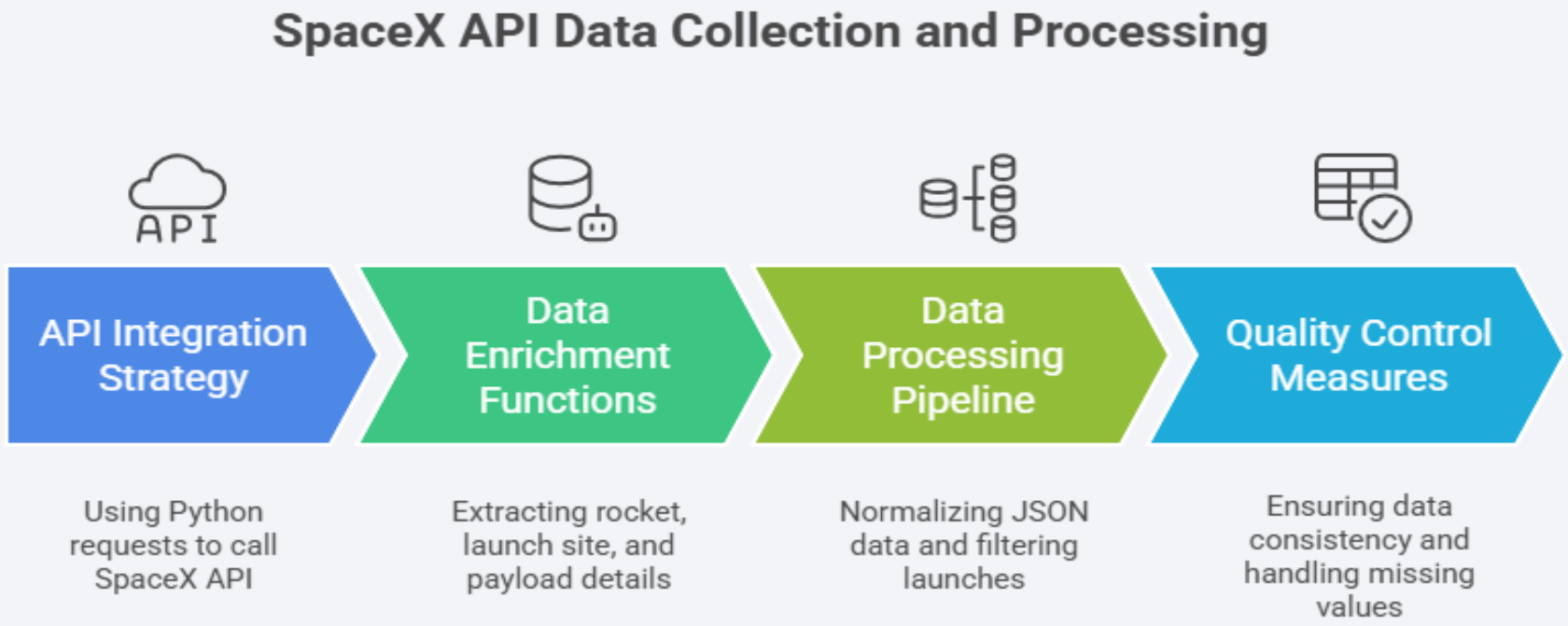
Data Processing Pipeline

- JSON normalization to structured DataFrame
- List value extraction and simplification
- Date/time parsing and filtering
- Falcon 1 launch exclusion for focused analysis

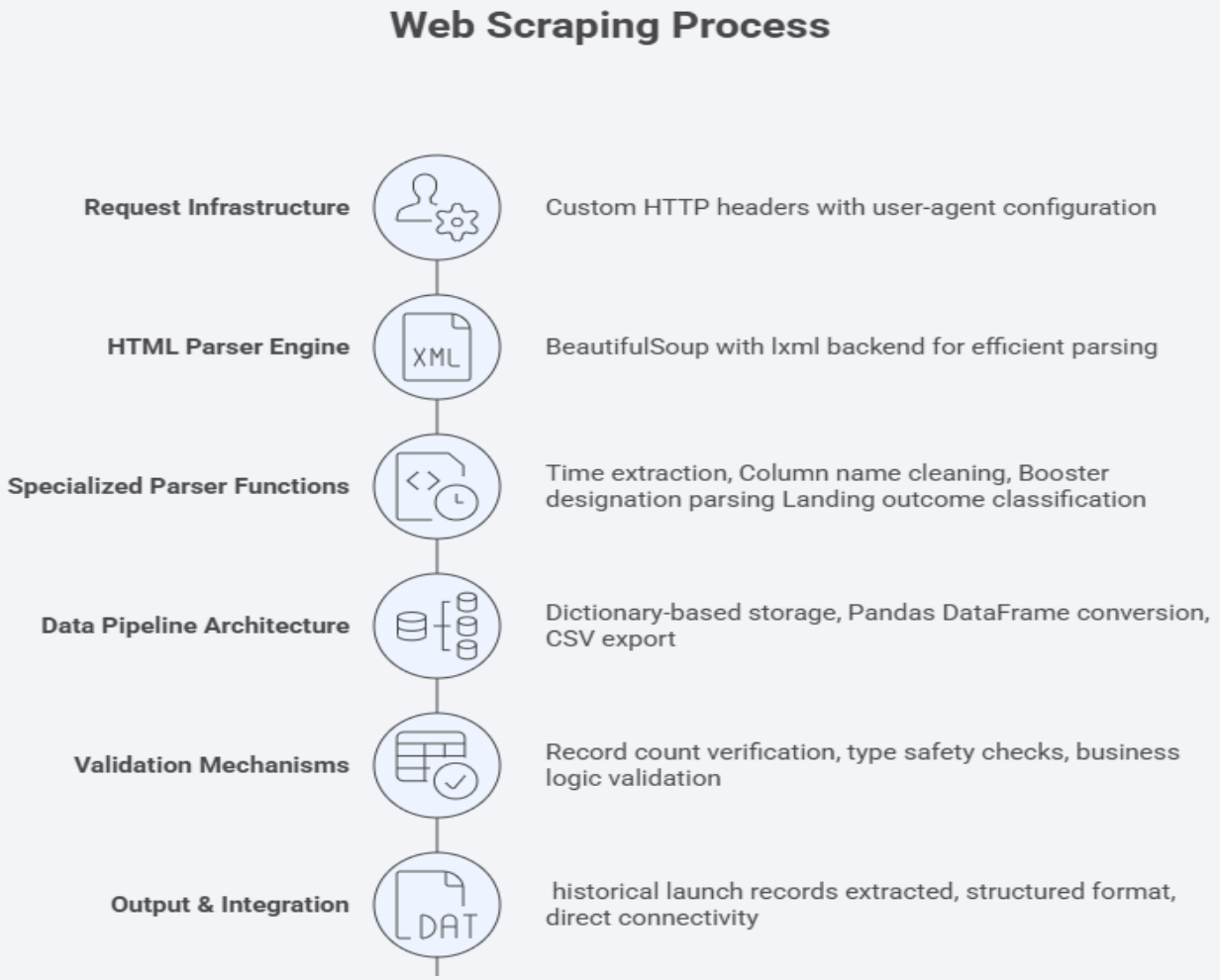
Quality Control Measures

- Null value detection and imputation (payload mass mean replacement)
- Data type standardization and consistency checks
- Flight number resetting for sequential analysis
- Missing value reporting and handling strategies

Data Collection – SpaceX API



Data Collection - Scraping



Data Wrangling

Data Quality Assessment

- Missing value identification and percentage calculation
- Data type classification (numerical vs categorical variables)
- LandingPad column analysis with 28.9% null values

Feature Engineering

- Landing outcome classification (success/failure binary encoding)
- Bad outcome set creation for failure pattern identification
- Class column generation for machine learning targets

Data Transformation

- Launch site distribution analysis using value_counts()
- Orbit type filtering (GTO exclusion as transfer orbit)
- Success rate calculation from binary classification

Export Preparation

- CSV export with index=False for clean file structure

EDA with Data Visualization

- **Flight Number vs. Payload Mass (Scatter Plot)**
  Used to analyze how launch experience (Flight Number) and payload weight influence launch success. The visualization shows that success rates improve with increased flight experience, even for heavier payloads.
- **Flight Number vs. Launch Site (Scatter Plot)**
  Helps compare launch success and failure patterns across different launch sites over time. Reveals that some sites show improved success as the number of launches increases.
- **Payload Mass vs. Launch Site (Scatter Plot)**
  Used to examine payload distribution across launch sites. Highlights that certain sites (e.g., VAFB-SLC) do not handle heavy payloads (>10,000 kg).
- **Success Rate by Orbit Type (Bar Chart)**
  Visualizes average launch success per orbit. Identifies orbits such as GEO, HEO, and ES-L1 with the highest success rates.
- **Flight Number vs. Orbit Type (Scatter Plot)**
  Used to assess whether launch success improves with experience for specific orbits. Shows a clear positive trend for LEO but not for GTO.
- **Payload Mass vs. Orbit Type (Scatter Plot)**
  Analyzes how payload weight affects success across orbits. Indicates higher success rates for heavy payloads in LEO, ISS, and Polar orbits.
- **Yearly Launch Success Trend (Line Chart)**
  Displays the evolution of average launch success over time. Shows a steady improvement in success rate from 2013 to 2020.
- **Purpose of Visualization:**
  These charts were selected to uncover relationships, trends, and patterns between launch outcomes and key variables, supporting informed feature selection for predictive modeling.
- https://github.com/profm9/IBM-Data-Science-Capstone-SpaceX/blob/main/notebooks/05_EDA_Visualization.ipynb

EDA with SQL

- **Identified Unique Launch Sites**
  Retrieved distinct launch site names to understand where SpaceX missions were conducted.
- **Filtered Launches by Site Prefix**
  Selected sample records where launch sites start with *"CCA"* to analyze missions from Cape Canaveral.
- **Calculated Total Payload for NASA (CRS) Missions**
  Aggregated payload mass to measure total cargo delivered for NASA-related CRS missions.
- **Computed Average Payload for F9 v1.1 Boosters**
  Used average aggregation to evaluate payload capability of a specific booster version.
- **Found First Successful Ground Pad Landing Date**
  Applied the MIN() function to determine when the first successful ground pad landing occurred.
- **Filtered Successful Drone Ship Landings by Payload Range**
  Identified booster versions that successfully landed on drone ships while carrying medium-heavy payloads (4000–6000 kg).
- **Counted Mission Outcomes (Success vs Failure)**
  Grouped mission outcomes to compare overall success and failure frequencies.
- **Identified Boosters with Maximum Payload Capacity**
  Used a subquery with MAX() to find booster versions that carried the heaviest payloads.
- **Analyzed Drone Ship Failures in 2015 by Month**
  Extracted month and filtered records to study drone ship landing failures during 2015.
- **Ranked Landing Outcomes by Frequency Over Time**
  Counted and ranked different landing outcomes between 2010 and 2017 to understand landing performance trends.
- https://github.com/profm9/IBM-Data-Science-Capstone-SpaceX/blob/main/notebooks/04_EDA_SQL.ipynb

Build an Interactive Map with Folium

- **Base Map (folium.Map)**
  Created an interactive map centered at NASA Johnson Space Center to provide geographical context for all launch sites.
- **Launch Site Circles (folium.Circle)**
  Added circular highlights at each launch site location to visually emphasize and distinguish SpaceX launch facilities on the map.
- **Launch Site Labels (folium.Marker + DivIcon)**
  Used text-based markers to clearly display the names of each launch site directly on the map for easy identification.
- **Marker Clusters (MarkerCluster)**
  Grouped multiple launch records occurring at the same coordinates to reduce map clutter and improve readability.
- **Success & Failure Markers (Color-coded Markers)**
  Added green markers for successful launches and red markers for failed launches to visually compare success rates across sites.
- **Mouse Position Tool (MousePosition)**
  Enabled real-time latitude and longitude display to help identify coordinates of nearby features such as coastlines, cities, railways, and highways.
- **Distance Markers (DivIcon)**
  Displayed calculated distances (in km) between launch sites and nearby geographic features directly on the map.
- **Connecting Lines (folium.PolyLine)**
  Drew lines between launch sites and nearby points (coastline, city) to visually represent proximity relationships.
- **Purpose:**
  These map objects were added to enable interactive geographic exploration, visualize launch success patterns, and analyze how proximity to coastlines, cities, and infrastructure may influence launch site selection and operations.
- https://github.com/profm9/IBM-Data-Science-Capstone-SpaceX/blob/main/notebooks/06_Interactive_Map_Folium.ipynb

Build a Dashboard with Plotly Dash

- **Dashboard Components**
  - **Launch Site Dropdown:** Allows selection of *All Sites* or a specific launch site.
  - **Pie Chart (Success Analysis):**
    - Shows **total successful launches by site** when *All Sites* is selected.
    - Shows **Success vs. Failure distribution** when a specific site is selected.
  - **Payload Range Slider:** Filters launches based on payload mass (kg).
  - **Scatter Plot (Correlation Analysis):**
    - Displays **Payload Mass vs. Launch Success (class)**.
    - Colored by **Booster Version Category** for comparison.
- **Why These Plots & Interactions**
  - The **dropdown** enables focused analysis per launch site.
  - The **pie chart** provides a quick, intuitive comparison of launch success.
  - The **slider** allows dynamic filtering to study payload impact.
  - The **scatter plot** reveals correlations between payload mass, success rate, and booster versions.
  - Overall, the dashboard supports **interactive exploration** of SpaceX launch performance and decision-making insights.
- https://github.com/profm9/IBM-Data-Science-Capstone-SpaceX/blob/main/app/spacex-dash-app.py

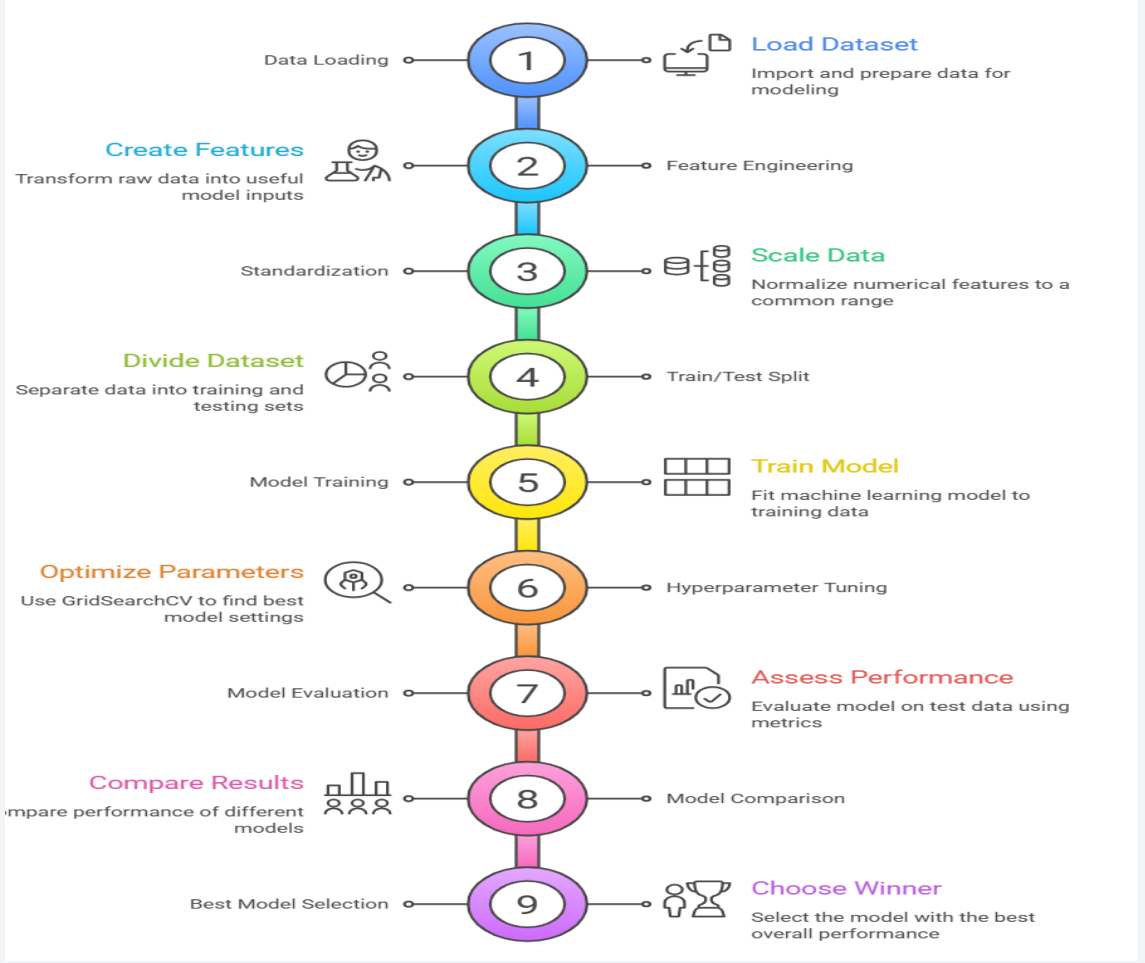
Predictive Analysis (Classification)

Summary of the Process

- **Data Preparation**
  - Loaded SpaceX datasets and selected **Class** as the target variable.
  - Converted target labels to NumPy array.
  - Standardized features using **StandardScaler** to improve model performance.
- **Data Splitting**
  - Split data into **80% training** and **20% testing** using train_test_split.
- **Model Building**
  - Trained four classification models:
    - Logistic Regression
    - Support Vector Machine (SVM)
    - Decision Tree
    - K-Nearest Neighbors (KNN)
- **Model Improvement**
  - Applied **GridSearchCV (cv = 10)** for hyperparameter tuning.
  - Optimized key parameters (e.g., C, kernel, max_depth, n_neighbors).
- **Model Evaluation**
  - Evaluated models using:
    - **Cross-validation accuracy**
    - **Test accuracy**
    - **Confusion matrix** (to analyze false positives/negatives)
- **Results & Best Model**
  - All models achieved similar test accuracy (~83%).
  - **Decision Tree** achieved the **highest validation accuracy (≈ 87.5%)**.
  - Selected **Decision Tree** as the **best performing model** based on validation performance.

Predictive Analysis (Classification)

Model Development workflow



https://github.com/profm9/IBM-Data-Science-Capstone-SpaceX/blob/main/notebooks/07_Machine_Learning_Prediction.ipynb

Results

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

GridSearchCV
estimator: LogisticRegression
LogisticRegression

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
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
```

TASK 5

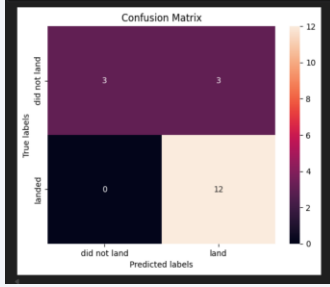
Calculate the accuracy on the test data using the method `score`:

```
print("Test accuracy:", logreg_cv.score(X_test, Y_test))
```

```
Test accuracy: 0.8333333333333334
```

Lets look at the confusion matrix:

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

```
svm_cv = GridSearchCV(svm, parameters, cv=10)
svm_cv.fit(X_train, Y_train)
```

GridSearchCV
estimator: SVC
SVC

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

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

Generate Code Markdown

TASK 7

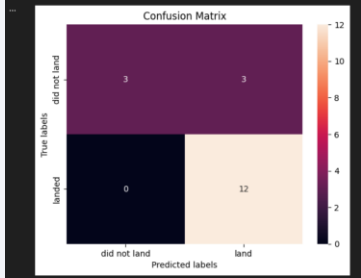
Calculate the accuracy on the test data using the method `score`:

```
print("Test accuracy:", svm_cv.score(X_test, Y_test))
```

```
Test accuracy: 0.8333333333333334
```

We can plot the confusion matrix

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

```
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv.fit(X_train, Y_train)
```

GridSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier

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

TASK 9

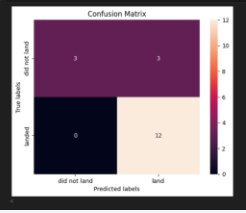
Calculate the accuracy of tree_cv on the test data using the method `score`:

```
print("Test accuracy:", tree_cv.score(X_test, Y_test))
```

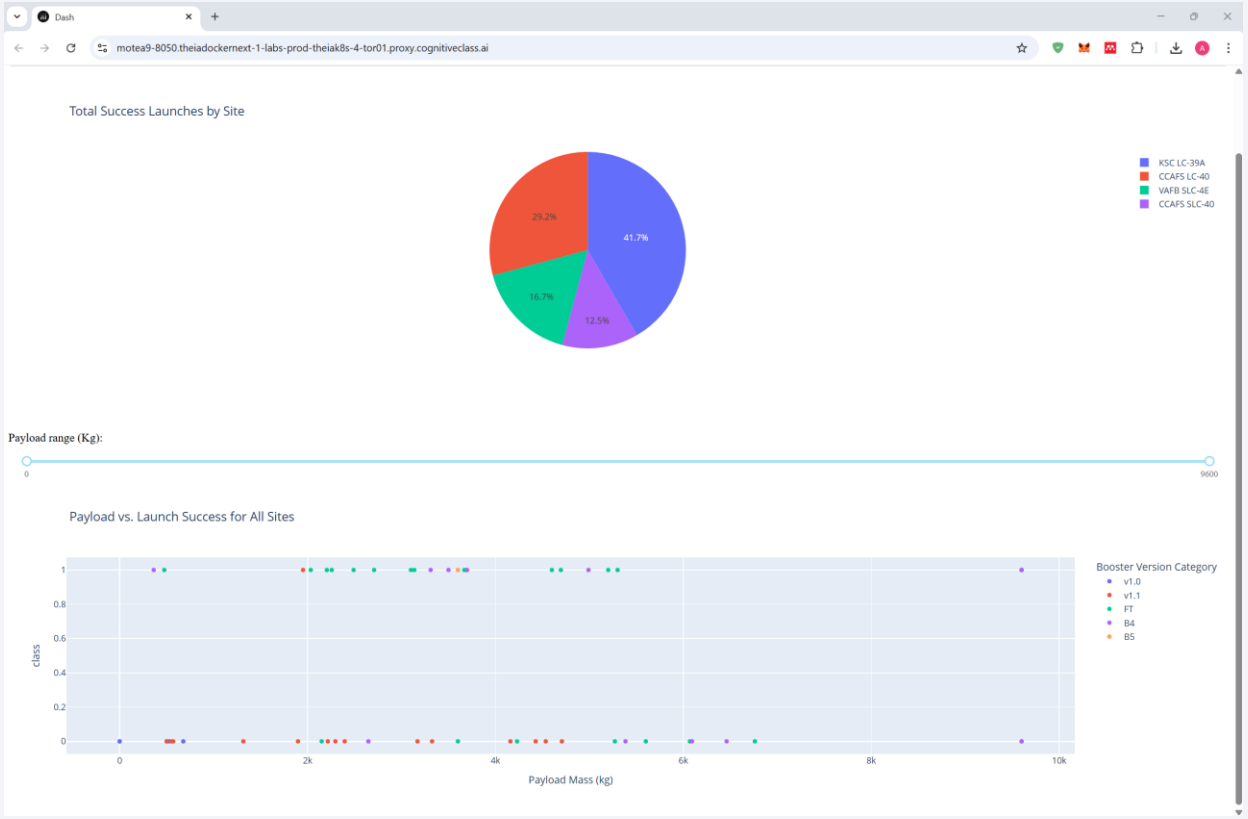
```
Test accuracy: 0.8333333333333334
```

We can plot the confusion matrix

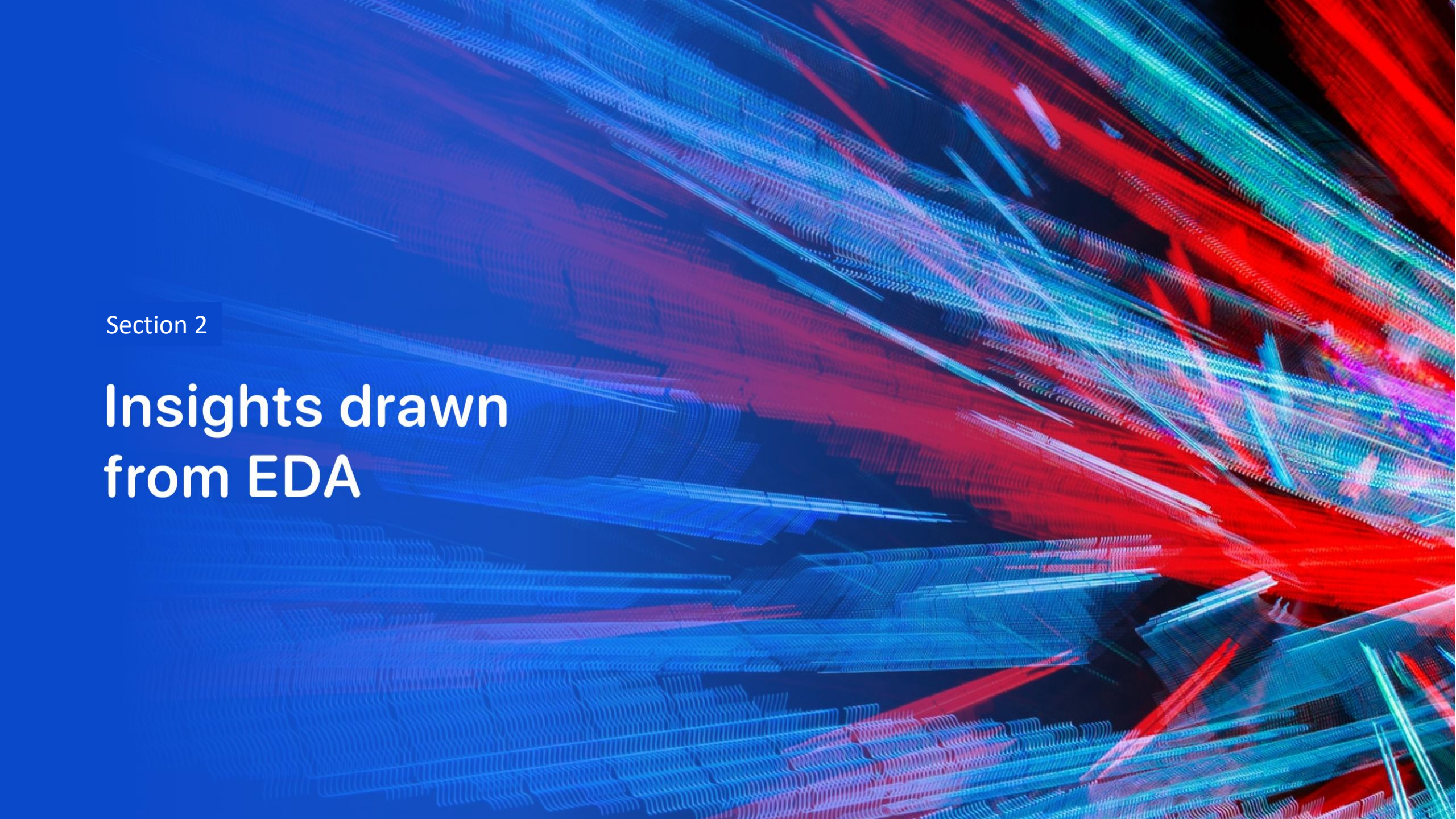
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Results

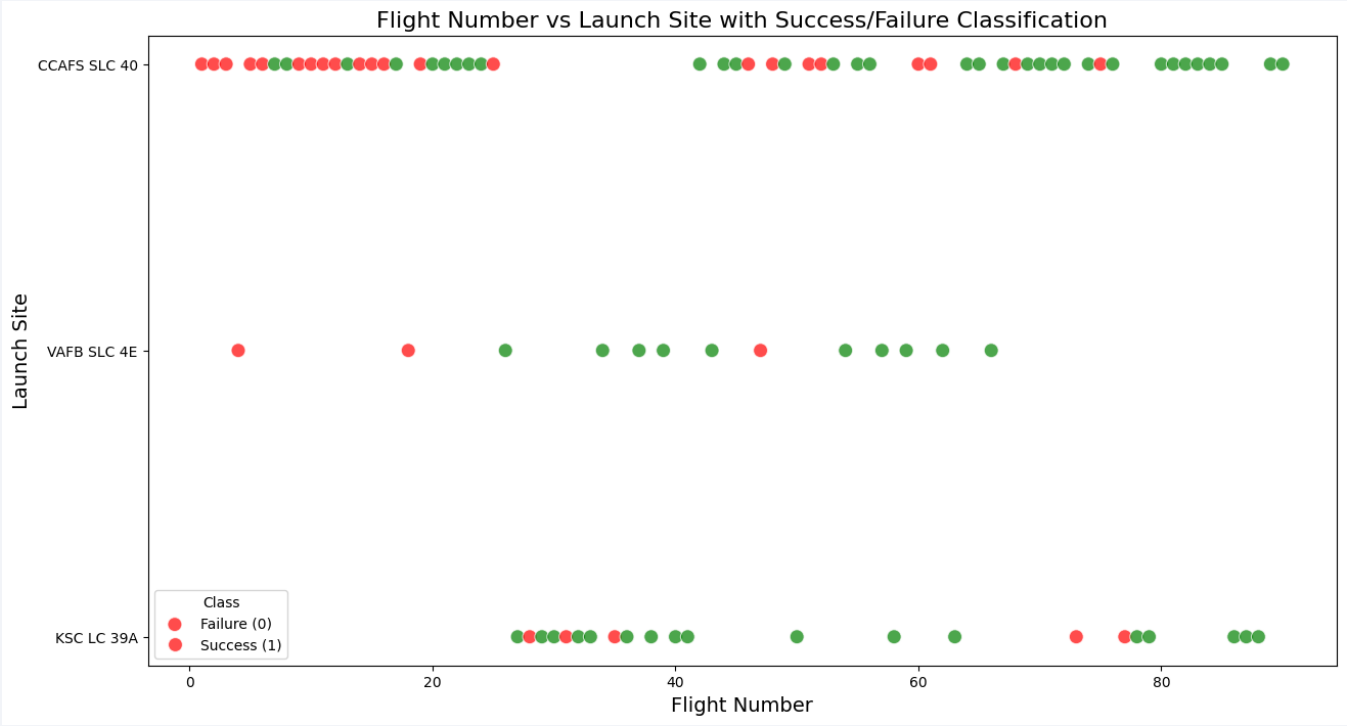

Dash
motea9-8050.theiadockernext-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai
Total Success Launches by Site
KSC LC-39A
CCAFS LC-40
VAFB SLC-4E
CCAFS SLC-40
41.7%
29.2%
16.7%
12.5%
Payload range (Kg):
0
9600
Payload vs. Launch Success for All Sites
Booster Version Category
v1.0
v1.1
FT
B4
B5
class
Payload Mass (kg)


TASK 10
Create a k nearest neighbors object then create a GridSearchCV object knn_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.
TASK 11
Calculate the accuracy of knn_cv on the test data using the method score:
We can plot the confusion matrix
Confusion Matrix
did not land
landed
True labels
Predicted labels
TASK 12
Find the method performs best:


Section 2
Insights drawn from EDA

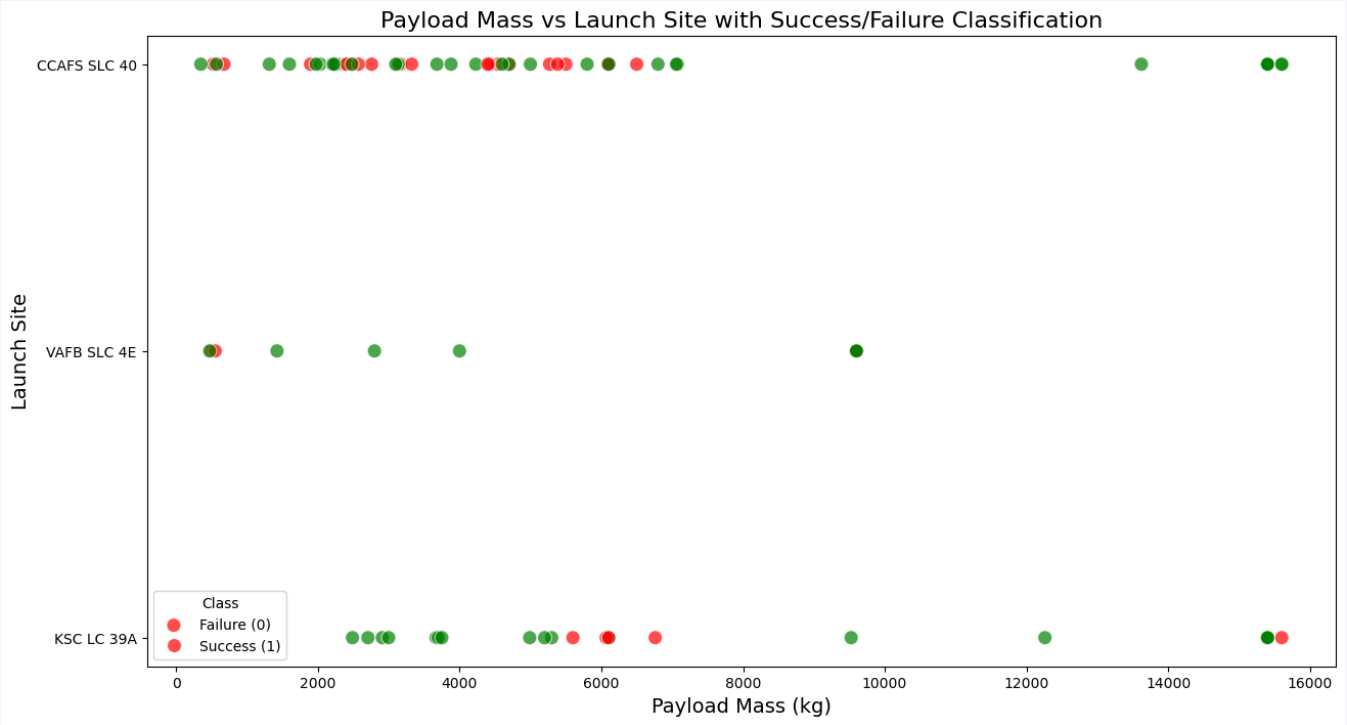
Flight Number vs. Launch Site

- Early flights show more **failures**, especially at **CCAFS SLC 40**, indicating the learning phase of launches.
- As flight number increases, **success rates improve across all sites**, reflecting operational maturity and reuse experience.
- **KSC LC 39A** and later missions show a higher concentration of **successful launches**.



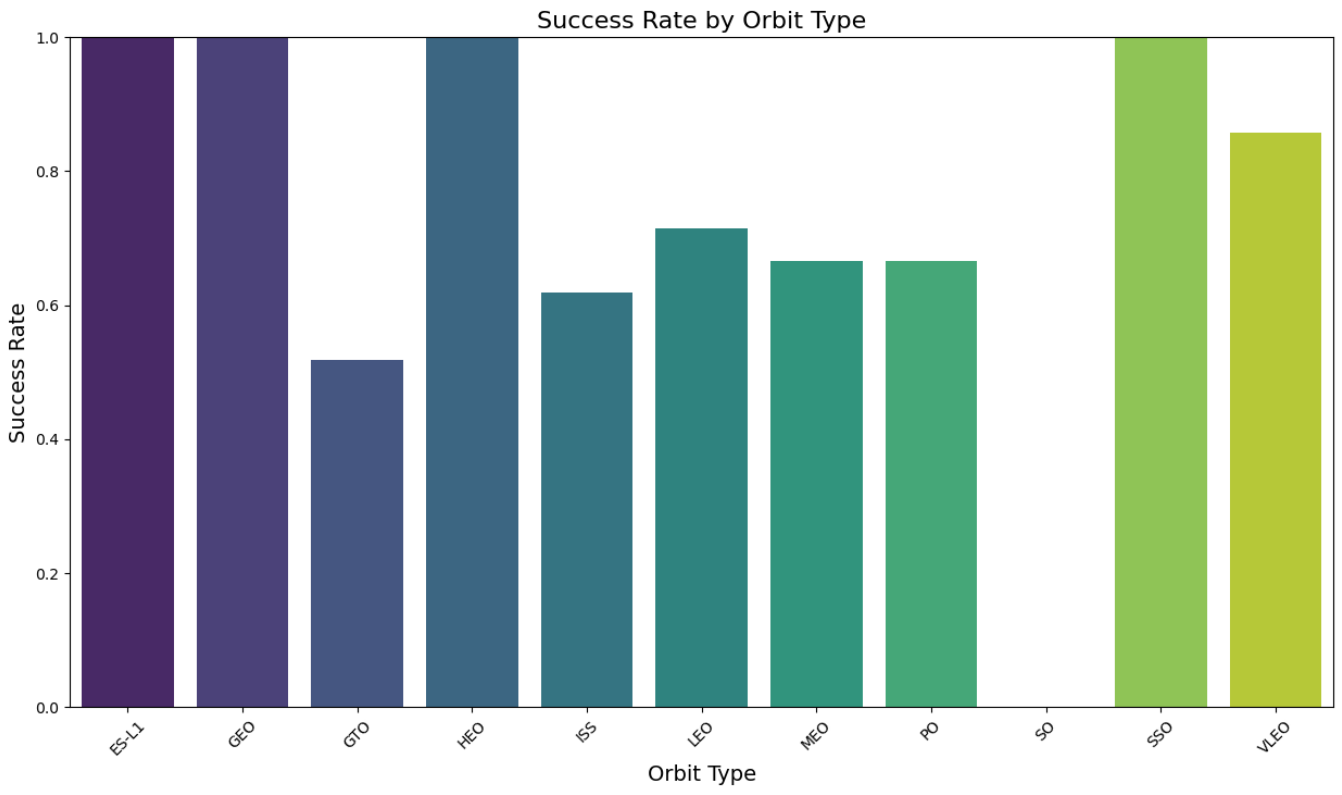
Payload vs. Launch Site

- **Higher payload masses** are generally associated with **successful launches**, especially at **KSC LC 39A**.
- **VAFB SLC 4E** handles fewer launches, mostly with **moderate payloads**, and shows a high success ratio.
- Failures are more frequent at **lower payload ranges**, suggesting early missions or less optimized configurations.



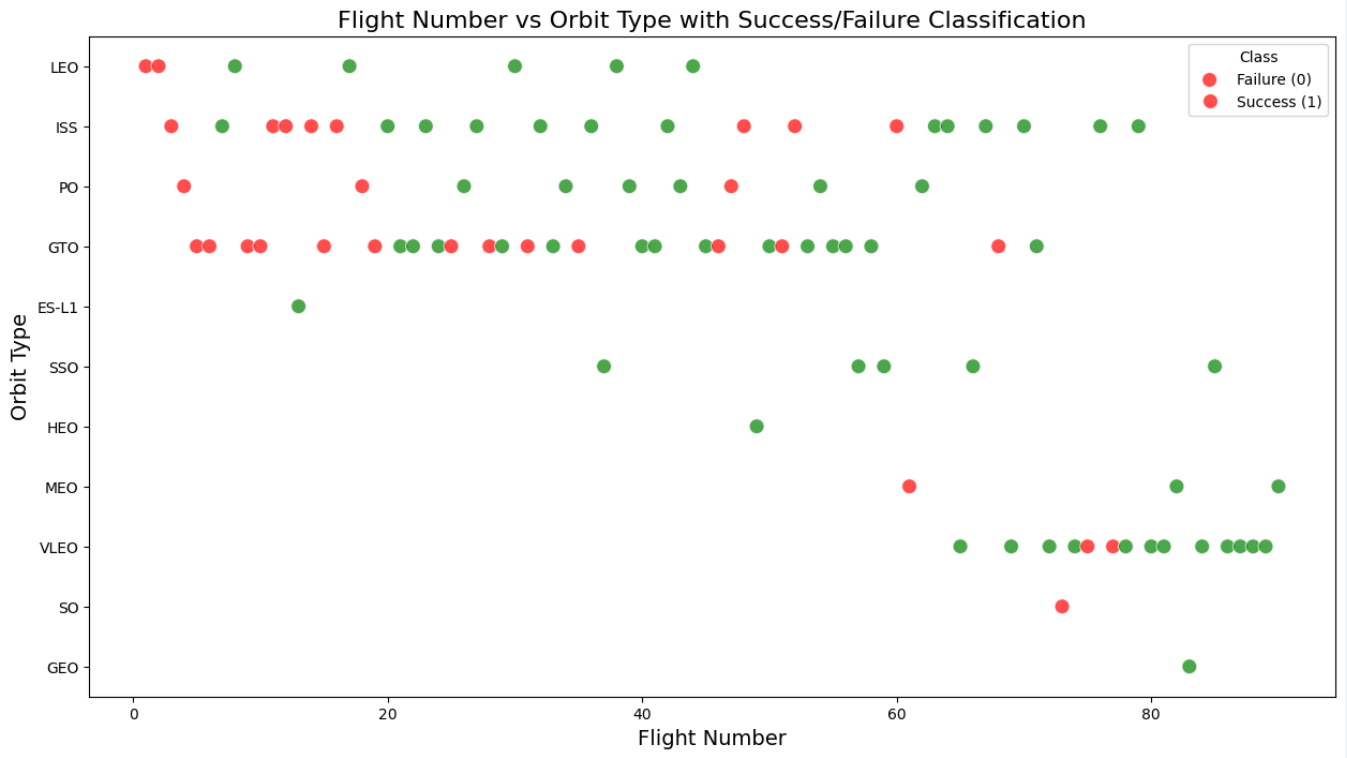
Success Rate vs. Orbit Type

- **ES-L1, GEO, HEO, SSO** orbits show **100% success**, indicating high reliability for these mission types.
- **VLEO, LEO, MEO, PO, ISS** orbits have **moderate to high success rates**, ranging from about 62% to 86%.
- **GTO** orbit shows a **lower success rate**.
- **SO** orbit has **0% success** based on available data, indicating challenges with this specific orbit type.



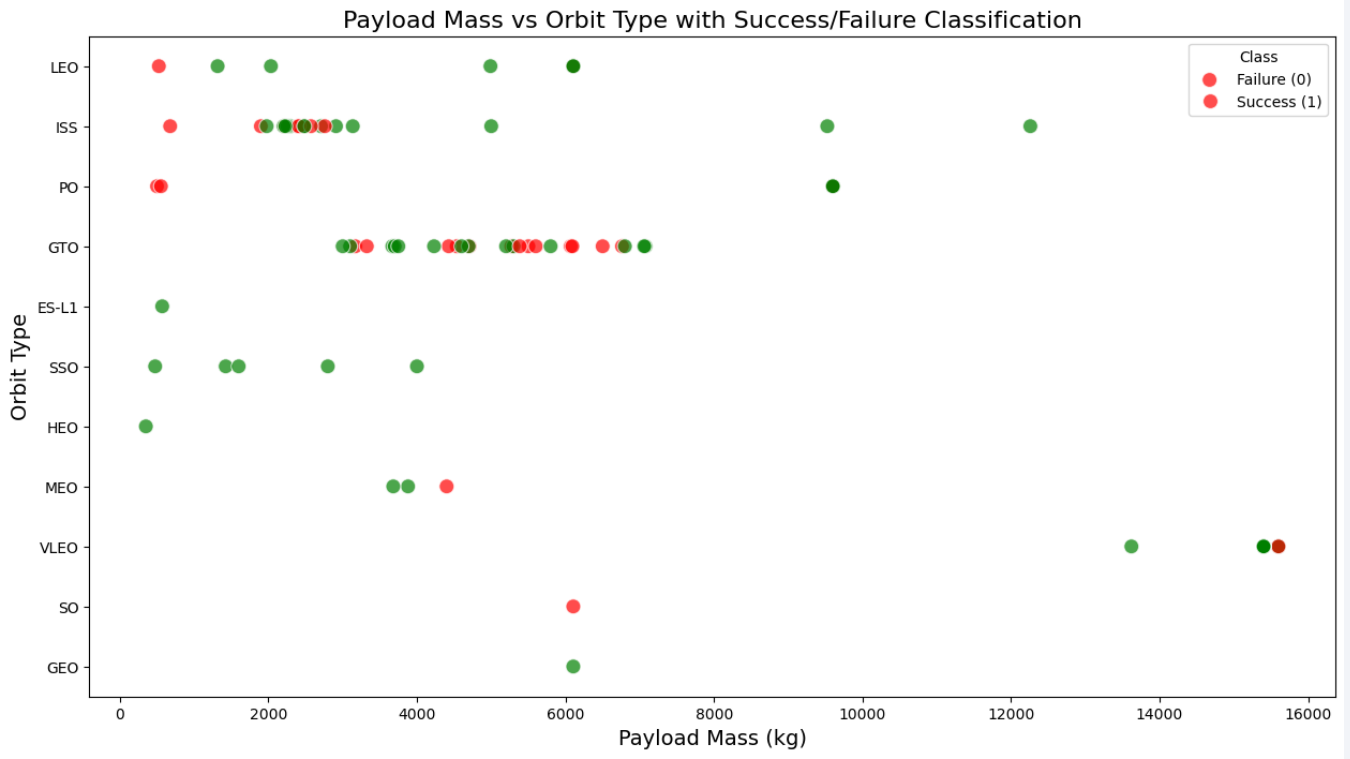
Flight Number vs. Orbit Type

- **Early flight numbers** show a mix of orbit types with **more failures**, especially in complex orbits like **GTO**, reflecting initial challenges.
- As flight numbers increase, **success rates improve**, indicating growing experience and better mission planning.
- **LEO and ISS** dominate across many flights, showing their popularity for routine missions.
- Specialized orbits like **ES-L1, SSO, and GEO** appear in later flights with **consistent success**, highlighting technological maturity.



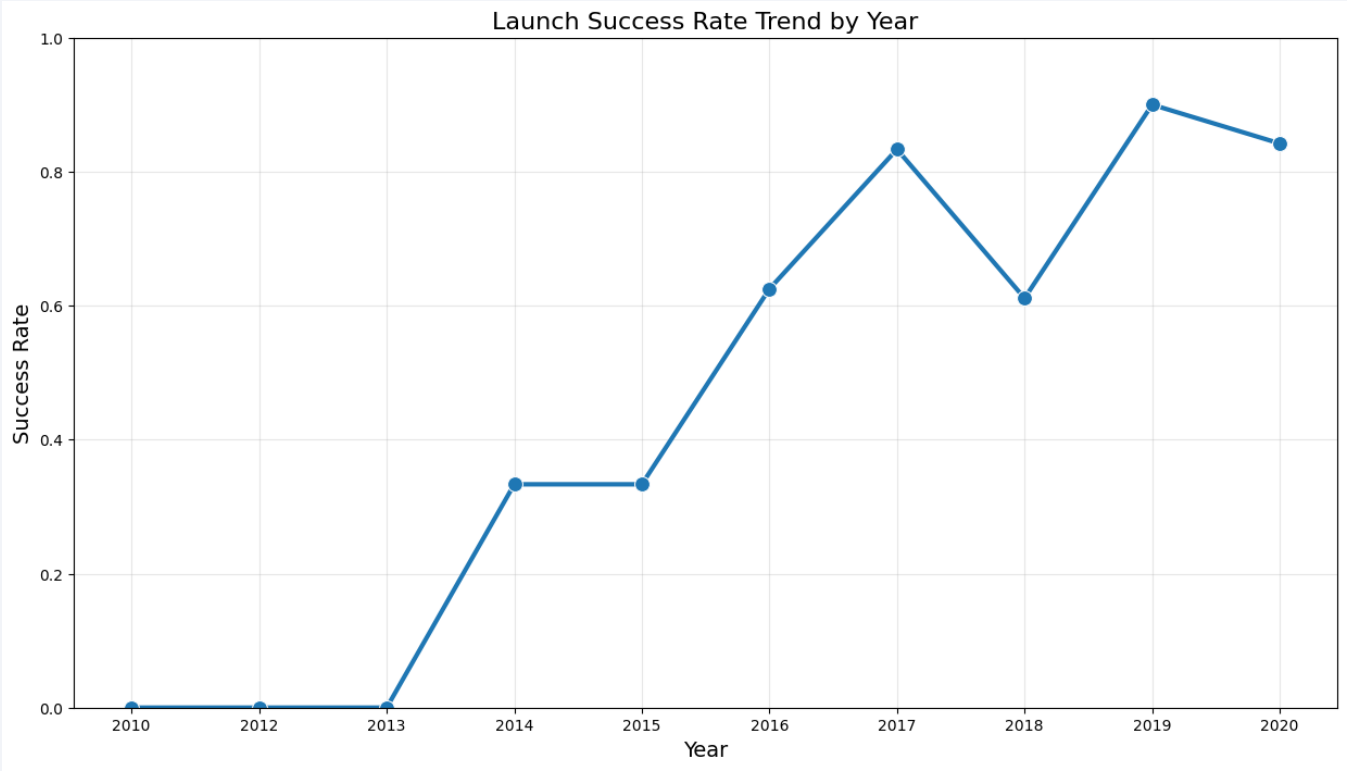
Payload vs. Orbit Type

- **Higher payload masses** are mostly associated with **LEO, ISS, and GTO** missions, reflecting their capacity for larger cargo.
- **GTO** shows a mix of successes and failures, especially at higher payloads, indicating its technical challenges.
- **Specialized orbits** like **ES-L1, SSO, and GEO** tend to carry **lighter payloads** but show **high success rates**, suggesting precision-focused missions.
- **Failures** are more frequent at **lower payload masses**, likely tied to early or experimental missions.



Launch Success Yearly Trend

- From **2010 to 2013**, the success rate remained at **0.0**, marking the early development phase.
- A gradual rise began in **2014–2015** (~33%), followed by a sharp improvement in **2016–2017**, peaking near **85%**.
- A dip occurred in **2018**, but recovery followed with **2019** reaching around **90%**.
- **2020** saw a slight decline, yet maintained a high success rate (~85%), reflecting overall operational maturity.



All Launch Site Names

Task 1

Display the names of the unique launch sites in the space mission

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

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

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

Launch Site Names Begin with 'CCA'

Task 2

Display 5 records where launch sites begin with the string 'CCA'

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

Python

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

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 total payload carried by boosters from NASA

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

```
%sql SELECT SUM("Payload_Mass__kg_") AS Total_Payload_Mass FROM SPACEXTABLE WHERE "Customer" LIKE '%NASA%' OR "Customer" LIKE '%CRS%'
```

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

Total_Payload_Mass
107010

Average Payload Mass by F9 v1.1

- the average payload mass carried by booster version F9 v1.1

Task 4

Display average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG("Payload_Mass__kg_") AS Avg_Payload_Mass FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1'
```

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

Avg_Payload_Mass
2928.4

First Successful Ground Landing Date

- the first successful landing outcome on ground pad

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
%sql SELECT MIN("Date") AS First_Successful_Ground_Pad_Landing FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)'
```

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

First_Successful_Ground_Pad_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (drone ship)' AND "Payload_Mass__kg_" > 4000 AND "Payload_Mass__kg_" < 6000
```

Python

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

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

Total Number of Successful and Failure Mission Outcomes

- the total number of successful and failure mission outcomes

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT CASE WHEN "Mission_Outcome" LIKE '%Success%' THEN 'Success' ELSE 'Failure' END AS Outcome_Type, COUNT(*) AS Count FROM SPACEXTABLE GROUP BY Outcome_Type
```

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

Outcome_Type	Count
Failure	1
Success	100

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "Payload_Mass__kg_" = (SELECT MAX("Payload_Mass__kg_") FROM SPACEXTABLE)
```

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

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

2015 Launch Records

- the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%sql SELECT substr(Date,6,2) AS Month,"Landing_Outcome", Booster_Version,Launch_Site FROM SPACEXTABLE WHERE substr(Date,1,4) = '2015' AND "Landing_Outcome" LIKE 'Failure%drone%';
```

Python

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

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

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%sql SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count FROM SPACEXTABLE  WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY "Landing_Outcome" ORDER BY Outcome_Count DESC
```

Python

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

Landing_Outcome	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

Section 3

Launch Sites Proximities Analysis

Global Launch Site Locations

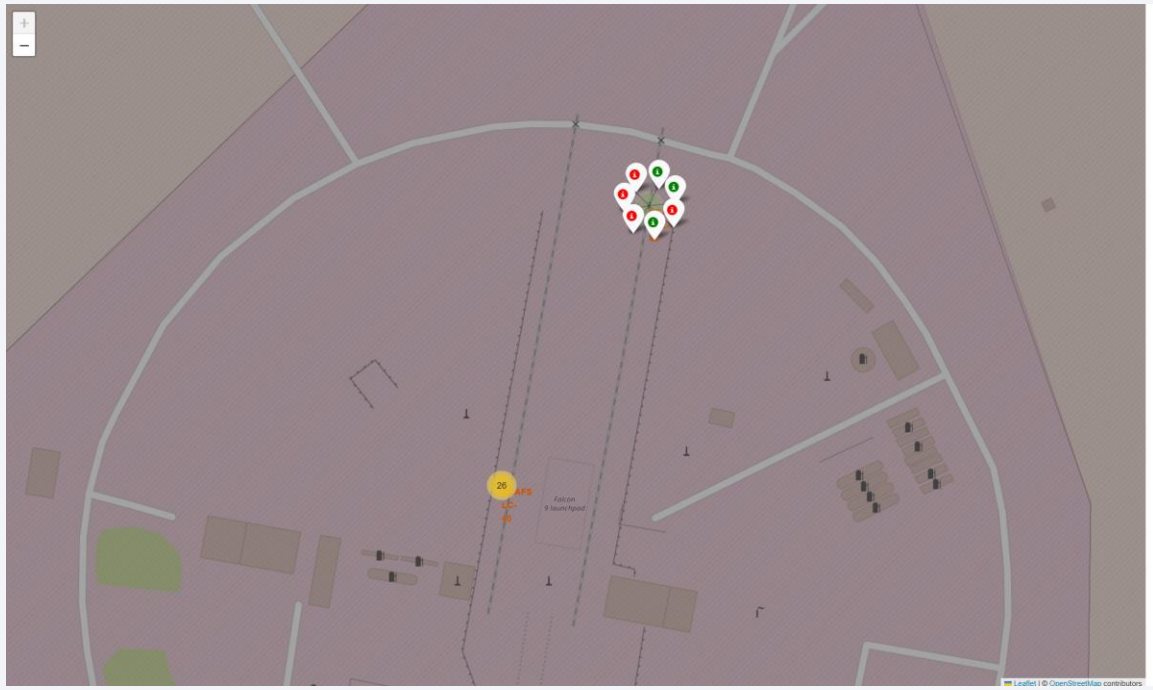
Important elements and findings:

- **Elements:** The map features custom markers and circles for each SpaceX launch site: **VAFB SLC-4E** on the West Coast, and **CCAFS LC-40, CCAFS SLC-40, and KSC LC-39A** on the East Coast. Each site is highlighted with a black circle of a 1km radius and a label.
- **Findings:**
  - **Coastal Proximity:** All launch sites are located very close to the ocean. This is a strategic choice to ensure that during launch, the rocket travels over open water, minimizing risk to populated areas in case of a malfunction.
  - **Equator Proximity:** The sites are situated in the southern part of the United States (Florida and Southern California). Being closer to the equator allows rockets to take better advantage of the Earth's rotational speed, which provides an additional velocity boost for reaching orbit.

Launch Success and Failure Clusters

Important elements and findings:

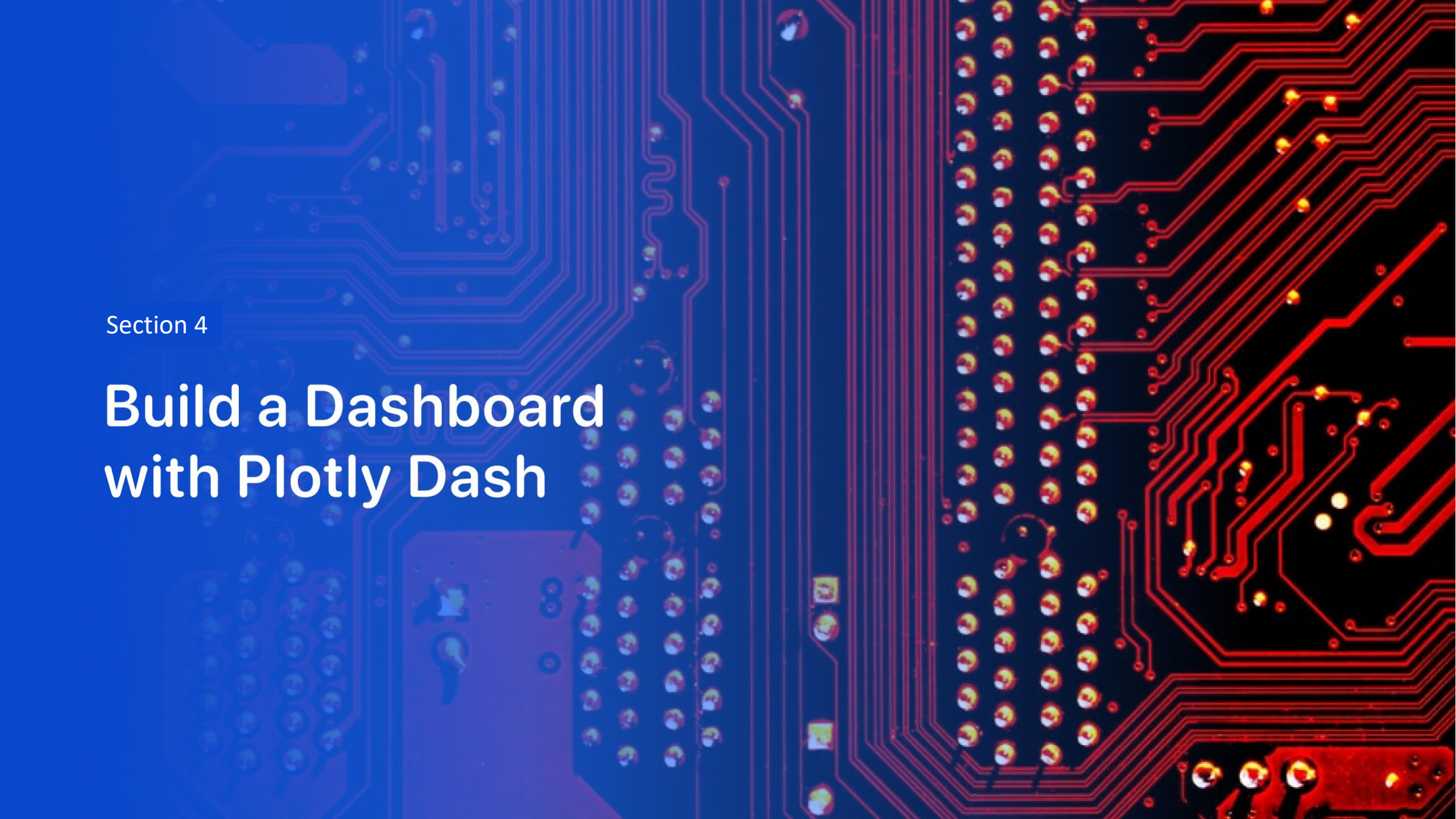
- **Elements:** This map utilizes the MarkerCluster plugin to group multiple launch records at the same coordinate. Individual markers are color-coded: **Green** indicates a successful launch (class=1), and **Red** indicates a failed launch (class=0).
- **Findings:**
  - **Visualizing Performance:** The cluster allows an analyst to quickly see the historical performance of a site. By clicking or zooming into the cluster, we can see the distribution of successes versus failures.
  - **Launch Density:** The yellow cluster icon (showing the number "26") indicates that there are 26 individual launch records centered at this specific launch pad (Falcon 9 launchpad at CCAFS SLC-40), making the map cleaner and more readable.



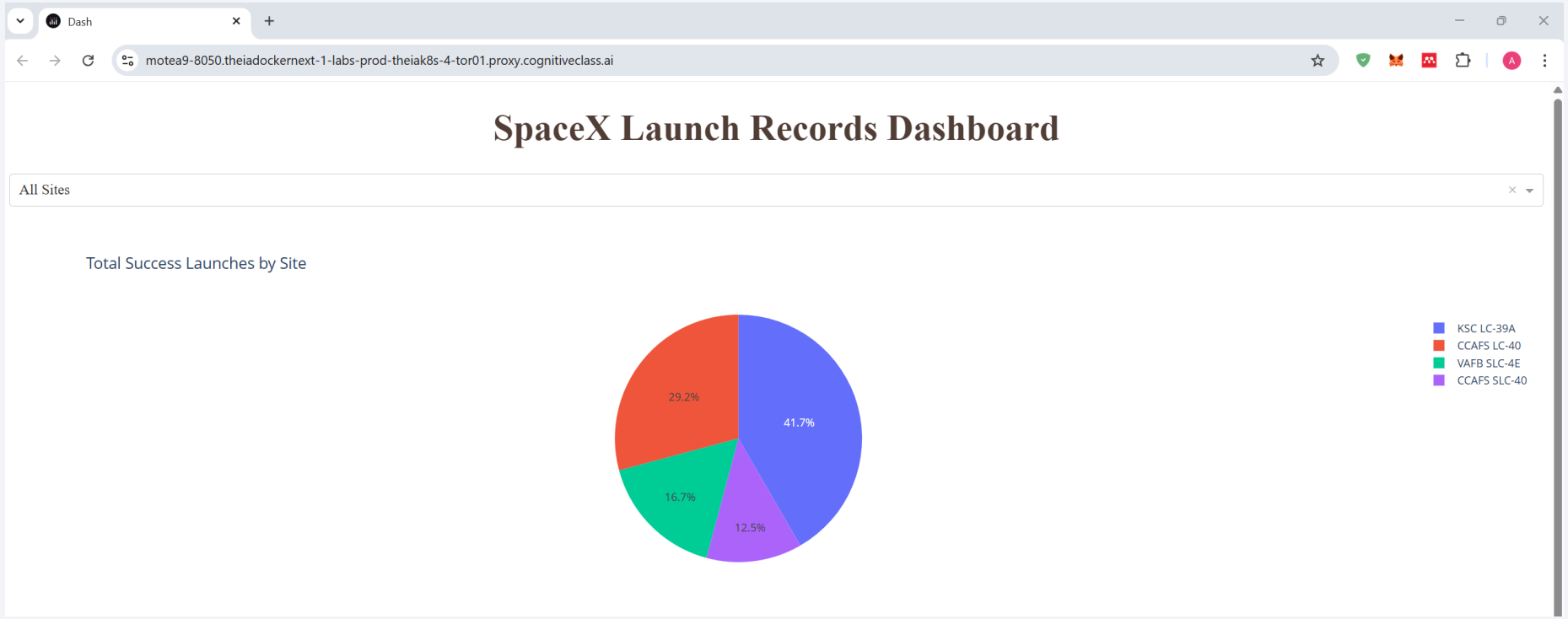
Proximity Analysis to Coastline and Infrastructure

Important elements and findings:

- **Elements:** This view includes a folium.PolyLine (the blue line) connecting the launch site to the nearest coastline. It also features a distance marker created using DivIcon, which displays the calculated distance (e.g., **0.51 KM**) between the two points.
- **Findings:**
  - **Safety Buffers:** The measurement confirms that the launch pad is roughly 0.5 km away from the Atlantic Ocean, providing a very narrow land buffer that prioritizes safety for inland areas.
  - **Logistical Connectivity:** The map shows the proximity of the launch site to major infrastructure like **Samuel C Phillips Parkway** (highway) and nearby railways. This proximity is vital for transporting large rocket components and propellant to the pad efficiently.

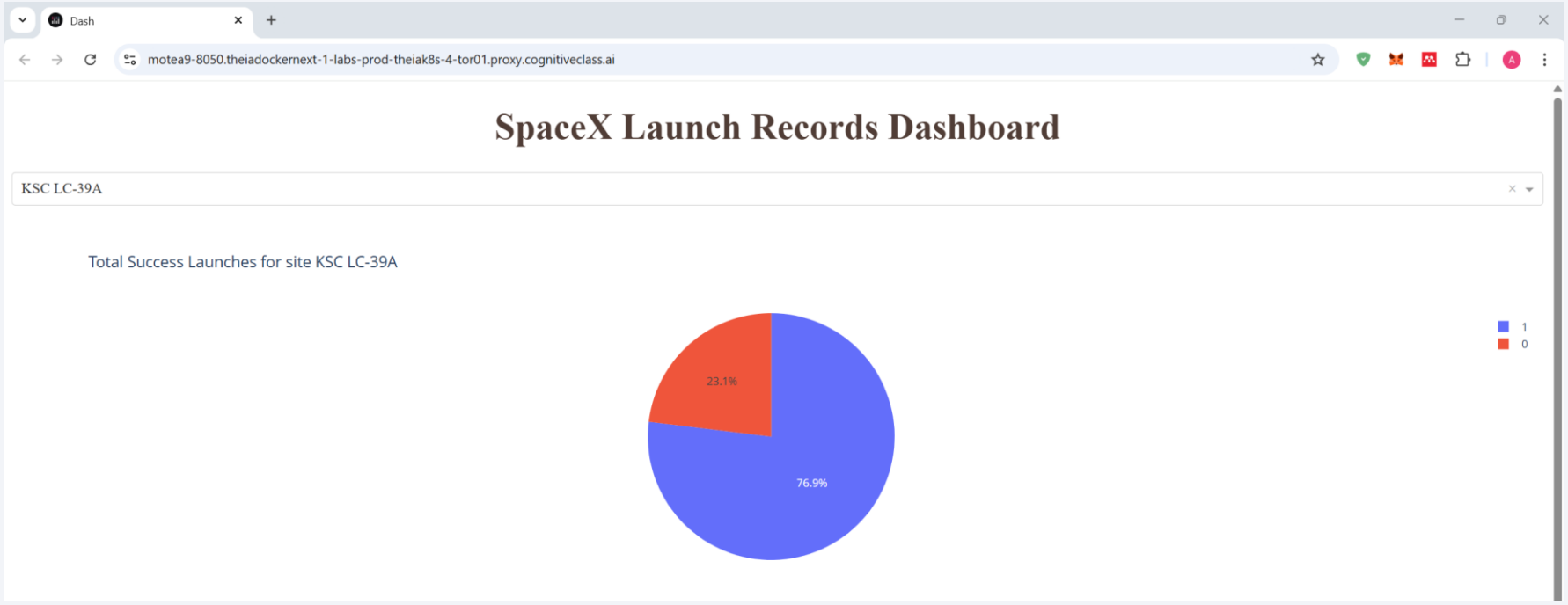

Section 4
Build a Dashboard with Plotly Dash

Total Successful Launches by Site



- **All Sites Comparison**: KSC LC-39A leads with **41.7%** of total successes
- **Site Ranking**: KSC > CCAFS LC-40 > VAFB > CCAFS SLC-40
- **Insight**: Identifies most productive launch facilities

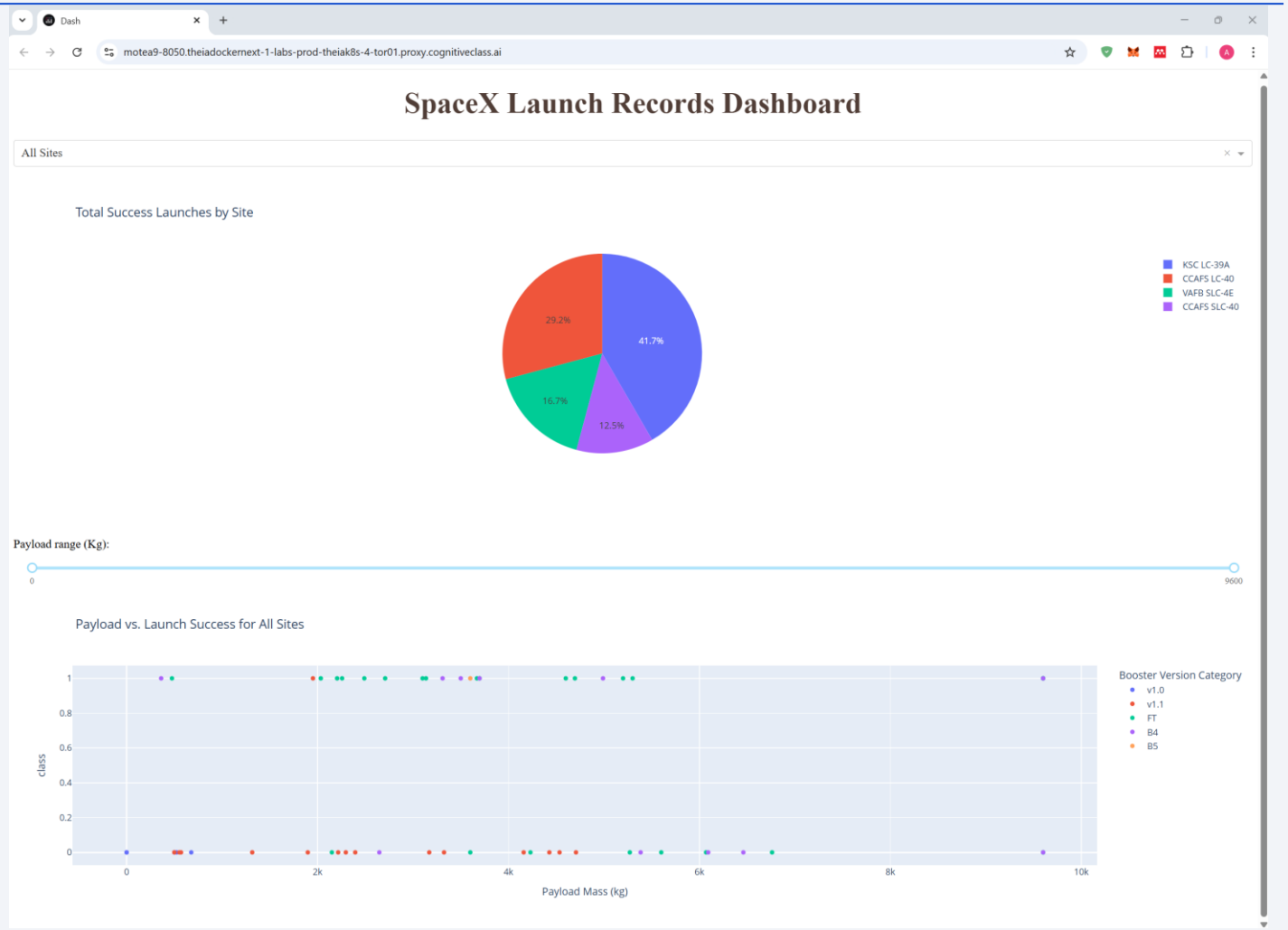
Success vs. Failure Rate for Site KSC LC-39A



- **KSC LC-39A Analysis**: **76.9% success rate** with only 23.1% failures
- **Confirmation**: Highest activity site also maintains exceptional reliability
- **Key Metric**: Demonstrated consistent performance excellence

Correlation Between Payload Mass and Launch Success

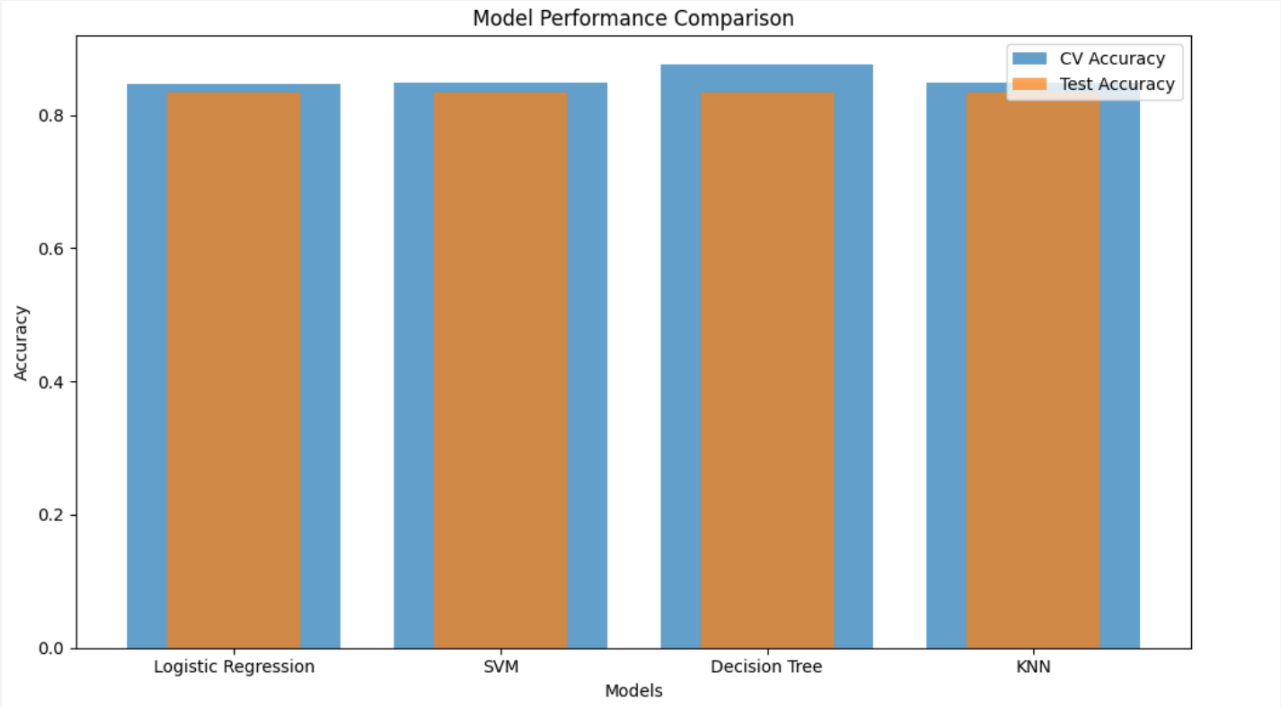
- **Success Patterns**: High success density in **2,000–4,000 kg** payload range
- **Booster Evolution**: FT and B5 versions show better reliability than older v1.0/v1.1
- **Heavy Payload Success**: Even near 10,000 kg, newer boosters achieve success
- **Visual Insight**: Clear progression in reliability with booster version advancement



Section 5

Predictive Analysis (Classification)

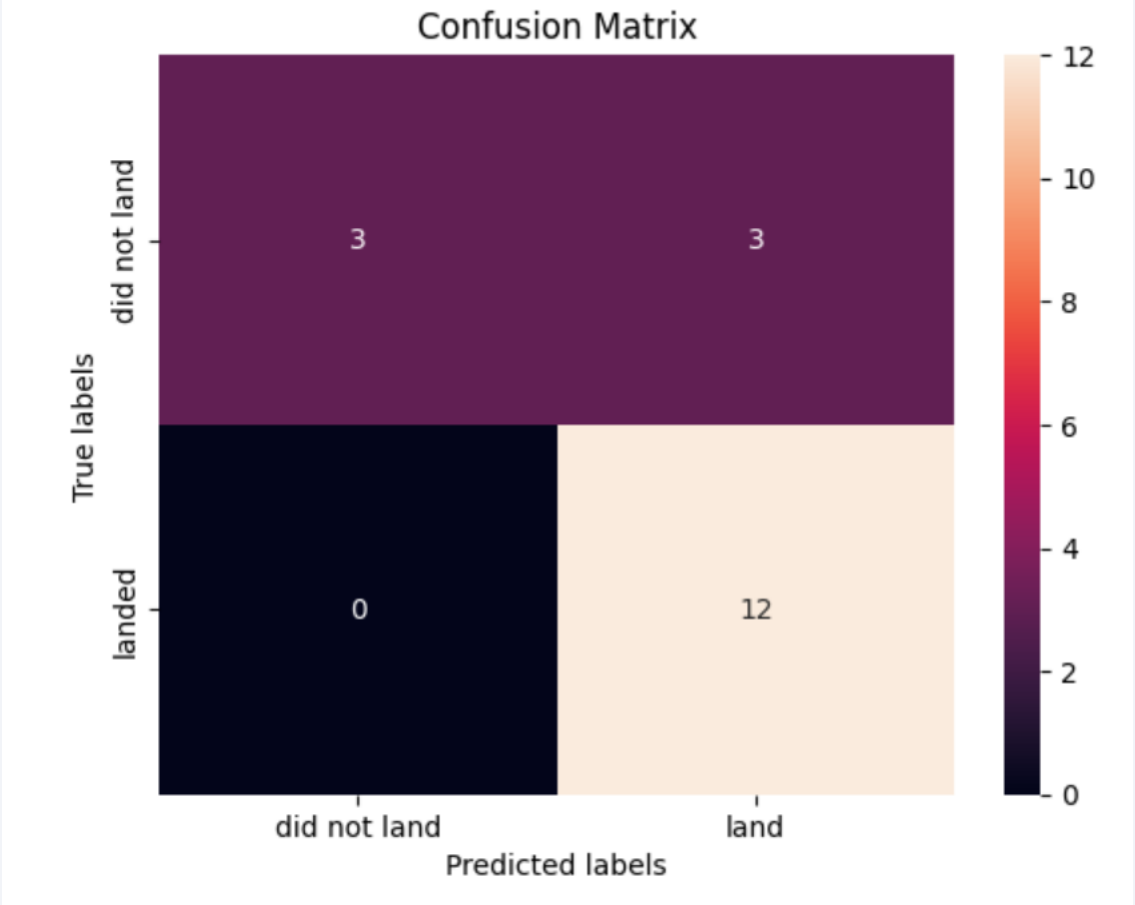
Classification Accuracy



- **Best Performing Model: Decision Tree**
- While all four models (Logistic Regression, SVM, Decision Tree, and KNN) achieved the **exact same Test Accuracy of 0.833**, the **Decision Tree** model is technically the best performer because it achieved the **highest Cross-Validation (CV) Accuracy of 0.875**.
- The other models had lower CV scores (Logistic Regression: 0.846; SVM: 0.848; KNN: 0.848), suggesting that the Decision Tree was slightly more effective at finding patterns within the training/validation data.

Confusion Matrix

- **True Positives (12):** The model correctly predicted that the rocket **landed** 12 times.
- **True Negatives (3):** The model correctly predicted that the rocket **did not land** 3 times.
- **False Negatives (0):** The model never failed to predict a successful landing; it was very reliable in identifying "landed" cases.
- **False Positives (3):** The model's primary weakness is false positives. In 3 cases, it predicted the rocket would "land," but it actually "did not land."



Conclusions

- **Top Performance: KSC LC-39A** is the most productive site, contributing **41.7%** of all successes with a high individual reliability rate of **76.9%**.
- **Operational Maturity:** Success rates improved drastically over time, rising from **0% (2010)** to nearly **90% (2019)** as SpaceX moved past the early learning phase.
- **Booster Reliability:** The **FT (Full Thrust) and B5** booster versions are the most reliable, significantly outperforming the older **v1.0 and v1.1** iterations.
- **Payload "Sweet Spot":** The highest density of successful launches occurs in the **2,000–4,000 kg** payload range.
- **Strategic Geography:** All sites are **coastal** to ensure safety over open water and located in **southern latitudes** to maximize the Earth's rotational speed boost.
- **Predictive Success:** Machine learning models accurately predict outcomes **83.3%** of the time, with the **Decision Tree** being the most effective model (87.5% CV accuracy).

Thank you!