Applied Data Science Capstone Project

Krethe Vandhana M

30th July 2025

Github link:https://github.com/VM7199/Applied_data_science_lab

Outline

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

Executive Summary

Summary of Methodologies

- Collected launch data using SpaceX API and Wikipedia scraping
- Cleaned data, created Class label, and merged datasets
- Performed EDA using SQL, Seaborn, and Matplotlib
- Built interactive maps (Folium) and dashboards (Plotly Dash)
- Standardized features and split data into training and test sets
- Tuned models using GridSearchCV with 10-fold cross-validation
- Evaluated performance using accuracy scores and confusion matrices
- Trained Logistic Regression, SVM, Decision Tree, and KNN models

Summary of Results

- Logistic Regression: CV accuracy ~83%, Test accuracy ~67%, moderate performance with some false positives
- SVM: CV accuracy ~86%, Test accuracy ~72%, effective but parameter-sensitive
- Decision Tree: CV accuracy 87.5%, Test accuracy ~77%, best performer with entropy & depth tuning
- KNN: CV accuracy ~82%, Test accuracy ~72%, simple but slightly less accurate
- Best Model: Decision Tree – balanced accuracy and interpretability
- Impact: Enables prediction of first stage landing success and aids in launch cost estimation

Executive Summary

Summary of Methodologies

- Collected launch data using SpaceX API and Wikipedia scraping
- Cleaned data, created Class label, and merged datasets
- Performed EDA using SQL, Seaborn, and Matplotlib
- Built interactive maps (Folium) and dashboards (Plotly Dash)
- Standardized features and split data into training and test sets
- Tuned models using GridSearchCV with 10-fold cross-validation
- Evaluated performance using accuracy scores and confusion matrices
- Trained Logistic Regression, SVM, Decision Tree, and KNN models

Introduction

Project Background & Context

- SpaceX aims to reduce launch costs by reusing Falcon 9’s first stage
- Successful landings allow rockets to be reused, saving millions per launch
- Predicting landing success helps estimate cost and improve reliability

Problems to Address

- Can we accurately predict whether the first stage will land successfully?
- Which launch features most influence landing outcomes?
- What model performs best in making such predictions?

Methodology

Executive Summary

- ***Data Collection:***

  Acquired launch data using SpaceX API and web scraping from Wikipedia

- ***Data Wrangling & Processing:***

  Cleaned missing values, created target labels (Class), and standardized features

- ***Exploratory Data Analysis (EDA):***

  Used SQL, Matplotlib, and Seaborn to uncover trends in payload, orbit, and launch outcomes

•***Interactive Visual Analytics:***

Created interactive maps using Folium and dashboards using Ploty Dash

•***Predictive Analysis:***

Trained Logistic Regression, SVM, Decision Tree, and KNN models to predict landing success

•***Model Tuning & Evaluation:***

Applied GridSearchCV for hyperparameter tuning; evaluated models using accuracy and confusion matrices

Data Collection

- ***SpaceX REST API***:

  Retrieved structured launch data directly from the SpaceX API using Python requests

- ***Web Scraping (Wikipedia)***:

  Used BeautifulSoup to scrape Falcon 9 launch history, including landing outcomes and booster details

- **Dataset Creation**:

  Merged API and scraped data to form a comprehensive dataset (dataset_part_2.csv and dataset_part_3.csv)

  - Included features like payload mass, orbit, launch site, customer, booster version, landing outcome, etc.

•***Output***:

Final datasets were used as inputs for EDA, visualization, and machine learning prediction

```python
#Define helper functions
# Global variables to store extracted data
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []

# Get booster version from rocket ID
def getBoosterVersion(data):
    for x in data['rocket']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
            BoosterVersion.append(response['name'])

# Get launch site details
def getLaunchSite(data):
    for x in data['launchpad']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
            Longitude.append(response['longitude'])
            Latitude.append(response['latitude'])
            LaunchSite.append(response['name'])
```

```python
# Get payload mass and orbit
def getPayloadData(data):
    for load in data['payloads']:
        if load:
            response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
            PayloadMass.append(response['mass_kg'])
            Orbit.append(response['orbit'])

# Get core information
def getCoreData(data):
    for core in data['cores']:
        if core['core'] != None:
            response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
            Block.append(response['block'])
            ReusedCount.append(response['reuse_count'])
            Serial.append(response['serial'])
        else:
            Block.append(None)
            ReusedCount.append(None)
            Serial.append(None)
        Outcome.append(str(core['landing_success']) + ' ' + str(core['landing_type']))
        Flights.append(core['flight'])
        GridFins.append(core['gridfins'])
        Reused.append(core['reused'])
        Legs.append(core['legs'])
        LandingPad.append(core['landpad'])
```

Data Collection – SpaceX API

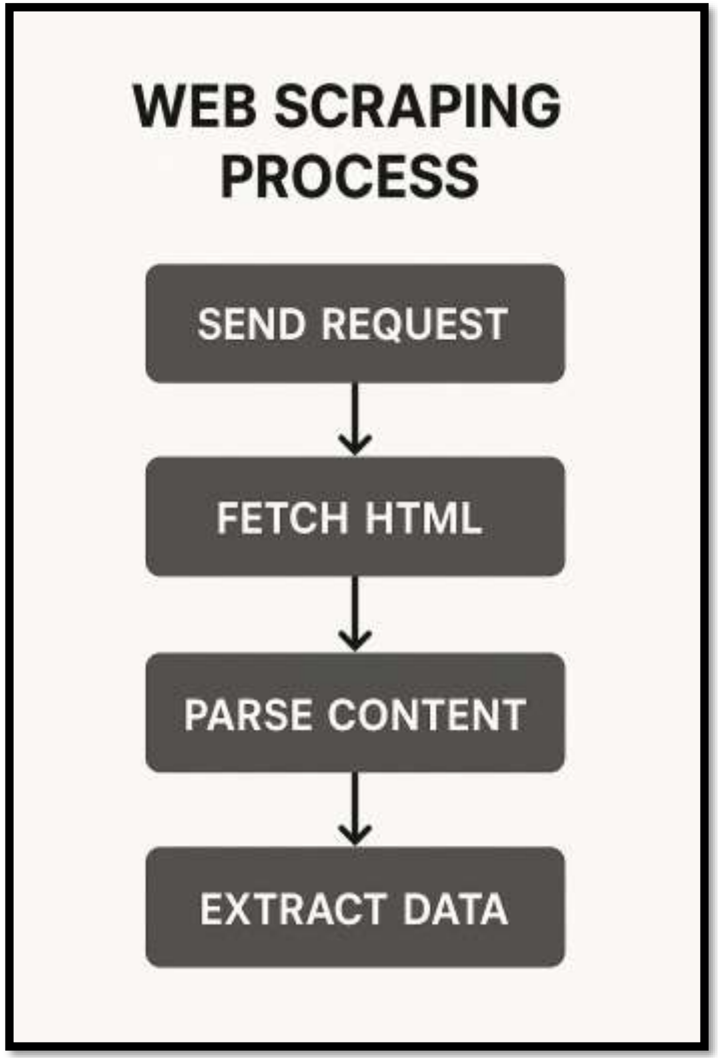

```
[15]: # Handle Missing Values
      # Check missing values
      print(data_falcon9.isnull().sum())

      # Replace missing PayloadMass with mean
      mean_payload = data_falcon9['PayloadMass'].mean()
      data_falcon9['PayloadMass'].replace(np.nan, mean_payload, inplace=True)

      # Check again
      print(data_falcon9.isnull().sum())
```

```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad        0
Block             0
ReusedCount       0
Serial            0
Longitude         0
Latitude          0
dtype: int64
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad        0
Block             0
ReusedCount       0
Serial            0
Longitude         0
Latitude          0
dtype: int64
```

Data Collection - Scraping




```
[4]: df.dtypes

[4]: Flight No.          int64
     Launch site        object
     Payload            object
     Payload mass       object
     Orbit              object
     Customer           object
     Launch outcome     object
     Version Booster    object
     Booster landing    object
     Date               object
     Time               object
     dtype: object

[7]: df['Launch site'].value_counts()

[7]: Launch site
     CCAFS            40
     KSC              33
     Cape Canaveral   20
     VAFB             16
     CCSFS            12
     Name: count, dtype: int64

[6]: df['Orbit'].value_counts()

[6]: Orbit
     LEO            67
     GTO            33
     SSO             7
     Polar           7
     MEO             3
     HEO             2
     Polar orbit     1
     Sub-orbital     1
     Name: count, dtype: int64
```

```python
# Load dataset from previous lab
df = pd.read_csv('dataset_part_1.csv')
df.head()
```

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.07B0003.18	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.07B0004.18	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.07B0005.18	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.07B0006.18	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.07B0007.18	No attempt\n	1 March 2013	15:10

```python
# Check for missing values
df.isnull().sum() / len(df) * 100
```

```
Flight No.         0.000000
Launch site        0.000000
Payload            0.000000
Payload mass       0.000000
Orbit              0.000000
Customer           0.826446
Launch outcome     0.000000
Version Booster    0.000000
Booster landing    0.000000
Date               0.000000
Time               0.000000
dtype: float64
```

```python
# Select outcomes that are considered "unsuccessful"
bad_outcomes = set(landing_outcomes.keys()[[1, 3, 5, 6, 7]])
bad_outcomes
```

```
{'Controlled', 'Failure ', 'No attempt', 'Precluded', 'Uncontrolled'}
```

```python
# 1 for success, 0 for fail
df['Class'] = [0 if outcome in bad_outcomes else 1 for outcome in df['Booster landing']]
df[['Booster landing', 'Class']].head()
```

	Booster landing	Class
0	Failure	1
1	Failure	1
2	No attempt\n	1
3	No attempt	0
4	No attempt\n	1

```python
df['Class'].mean()
```

```
np.float64(0.7768595041322314)
```

Data Wrangling

- Loaded CSV data using **pandas.read_csv()**
- Inspected data with **.head(), .info(), .describe()**
- Handled missing values:
  - Removed critical missing rows (**dropna()**)
  - Filled minor gaps using **mean/placeholder values**
- Converted columns to correct types (**to_datetime, astype()**)
- Created new features (e.g., **launch year, day of week**)
- Encoded categorical data using:
  - **get_dummies()** for one-hot encoding
  - **LabelEncoder** for ordinal features

- Filtered dataset for **relevant launches**
- Removed **duplicates and unused columns**
- Saved clean data for **visualization and modeling**

EDA with Data Visualization

- ***Bar Charts***

  → Compared success/failure counts by **launch site**

  → Identified sites with the highest success rates

- ***Pie Chart***

  → Visualized the **distribution of successful launches** per site

  → Helped spot site-wise contribution to overall success

- ***Scatter Plot***

  → Plotted **payload mass vs. launch success**

  → Analyzed correlation between payload size and mission outcome

- ***Histogram***

  → Showed **distribution of payload mass**

  → Helped understand payload ranges used most frequently

- ***Line Plot***

  → Displayed **success trends over time**

  → Identified patterns or improvements across launch years

- ***Box Plot***

  → Compared **payload mass across different orbit types**

  → Highlighted spread and outliers in different categories

EDA with SQL

SQL Queries Performed (EDA with SQL)

•***Counted total number of launches***

→ Used COUNT(*) to find total entries in the dataset

•***Calculated average payload mass***

→ Used AVG(payload_mass) for understanding general launch capacity

•***Summed payload mass for specific customers***

→ Example: SUM(payload_mass) where customer = 'NASA (CRS)'

•***Filtered launches by orbit type***

→ Used WHERE orbit = 'LEO' to focus on Low Earth Orbit missions

•***Grouped data by launch site***

→ Used GROUP BY launch_site to compare site-level statistics

•***Identified number of successful launches***

→ Used WHERE class = 1 to count successful missions

•***Sorted launches by payload mass***

→ Used ORDER BY payload_mass DESC to find heaviest launches

•***Used pattern matching with LIKE***

→ Handled inconsistent customer names with LIKE '%NASA%'

Build an Interactive Map with Folium

- ***Markers***

  → Plotted each SpaceX **launch site** location

  → Helped visually identify launch site coordinates on the map

- ***Popups***

  → Added **launch site names** to markers

  → Provided context when clicking on a marker

- ***Circles***

  → Drew **circle zones** around launch sites

  → Indicated a visual boundary or area of influence around each site o ***Circle Markers***

→ Used for **highlighting success/failure locations**

→ Color-coded by mission outcome for quick interpretation

o ***Lines (Polylines)***

→ Connected **launch site to satellite path locations** (if available)

→ Showed hypothetical or example orbital paths

o ***Tile Layers***

→ Added **different base maps** (e.g., OpenStreetMap, Stamen Terrain)

→ Enhanced map readability based on viewing preference

Build a Dashboard with Plotly Dash

•***Pie Chart (Success Count by Launch Site)***

→ Visualized the **proportion of successful launches per site**

→ Helps compare launch site performance at a glance

•***Payload Mass vs. Launch Outcome Scatter Plot***

→ Shows **relationship between payload mass and launch success**

→ Helps identify patterns or limits that affect mission outcomes

•***Dropdown Menu (Launch Site Selector)***

→ Allows users to **filter data** by specific launch site

→ Enables focused analysis per location

•***Range Slider (Payload Mass Filter)***

→ Lets users adjust the **payload mass range** interactively

→ Useful for identifying trends within specific mass thresholds

•***Dynamic Updates (Callbacks)***

→ Charts update **automatically** based on dropdown or slider input

→ Makes the dashboard interactive and user-friendly

Predictive Analysis (Classification)

•***Data Preprocessing***

→ Cleaned and normalized features

→ Applied **one-hot encoding** for categorical variables

•***Feature Selection***

→ Used correlation analysis and **feature importance**

→ Selected features most relevant to predicting success (Class)

•***Model Selection***

→ Trained multiple classifiers:

- **Logistic Regression**
- **Support Vector Machine (SVM)**
- **Decision Tree Classifier**
- **K-Nearest Neighbors (KNN)**

- ***Model Evaluation***

→ Evaluated using metrics:

  - **Accuracy**
  - **Precision / Recall**
  - **F1-score**
  - **Confusion Matrix**

- ***Hyperparameter Tuning***

→ Used **GridSearchCV** to find best parameters for each model

→ Optimized model performance via cross-validation

- ***Best Model Identified***

→ Chose model with **highest accuracy and F1-score**

→ Final model used for predicting launch outcome

Results

Exploratory data analysis results

- Found that **CCAFS** and **KSC LC-39A** had the highest number of successful launches
- Payload mass was most commonly between **2000–6000 kg**
- **LEO (Low Earth Orbit)** was the most frequent orbit used
- Success rates varied significantly by **launch site and customer**
- No strong correlation found between **payload mass and launch success**, but extreme weights had higher failure rates

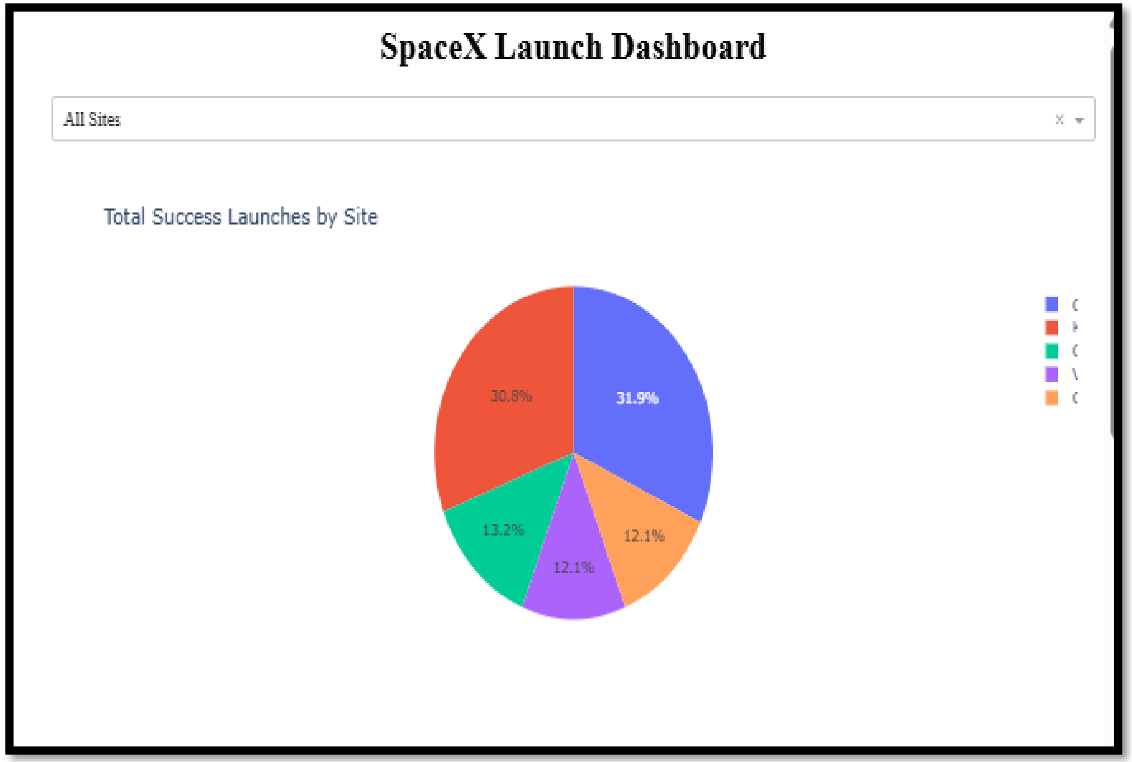
```
%sql SELECT * FROM SPACEXTABLE WHERE launch_site LIKE 'CCA%' LIMIT 5;
```

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

flight_no	launch_site	payload	payload_mass	orbit	customer	launch_outcome	version_booster	booster_landing	date	time	class
1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	F9 v1.07B0003.18	Failure	4 June 2010	18:45	1
2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.07B0004.18	Failure	8 December 2010	15:43	1
3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.07B0005.18	No attempt	22 May 2012	07:44	1
4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success	F9 v1.07B0006.18	No attempt	8 October 2012	00:35	0
5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success	F9 v1.07B0007.18	No attempt	1 March 2013	15:10	1

Results

Interactive analytics demo in screenshots



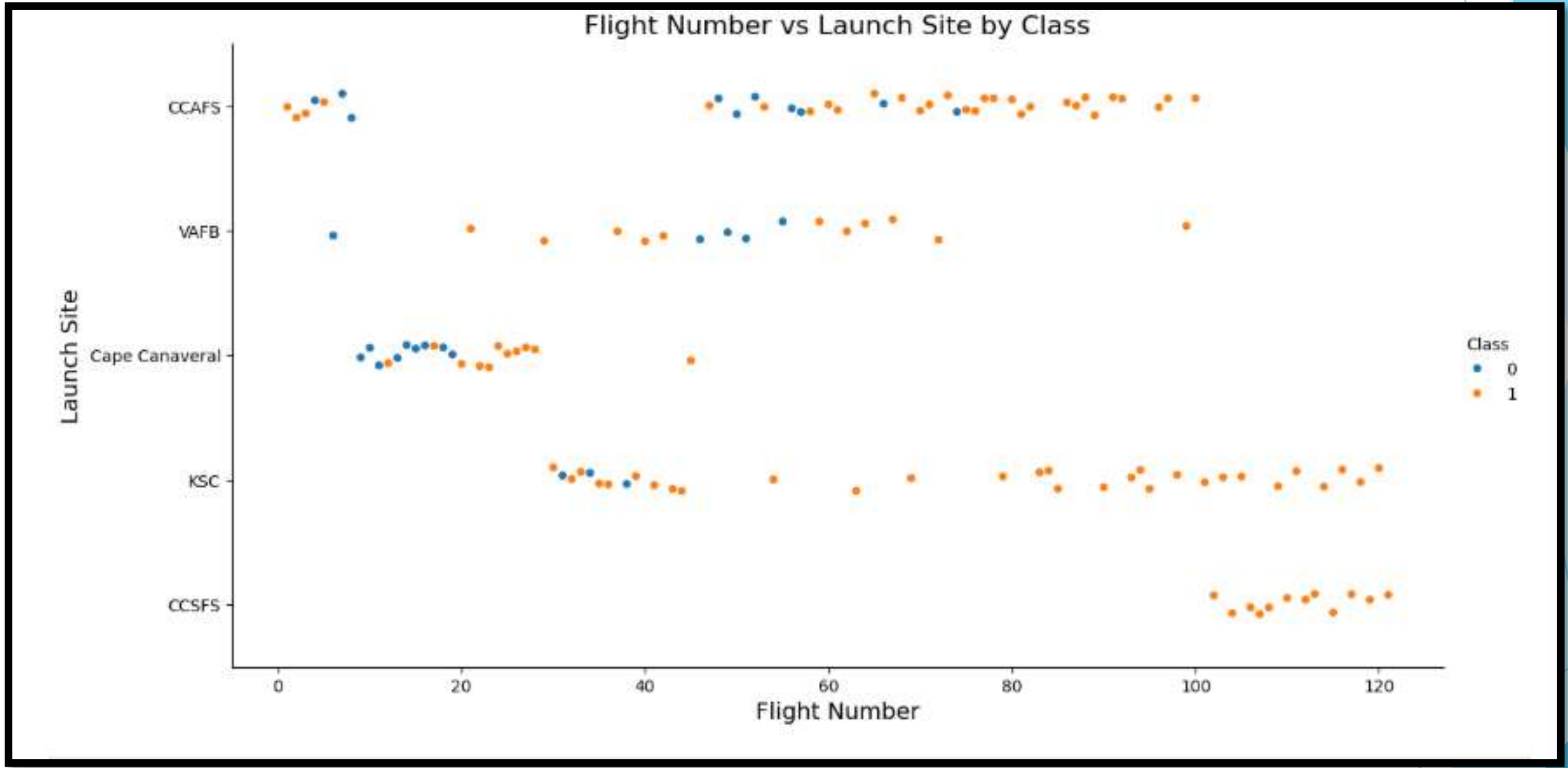
Results

Predictive analysis results

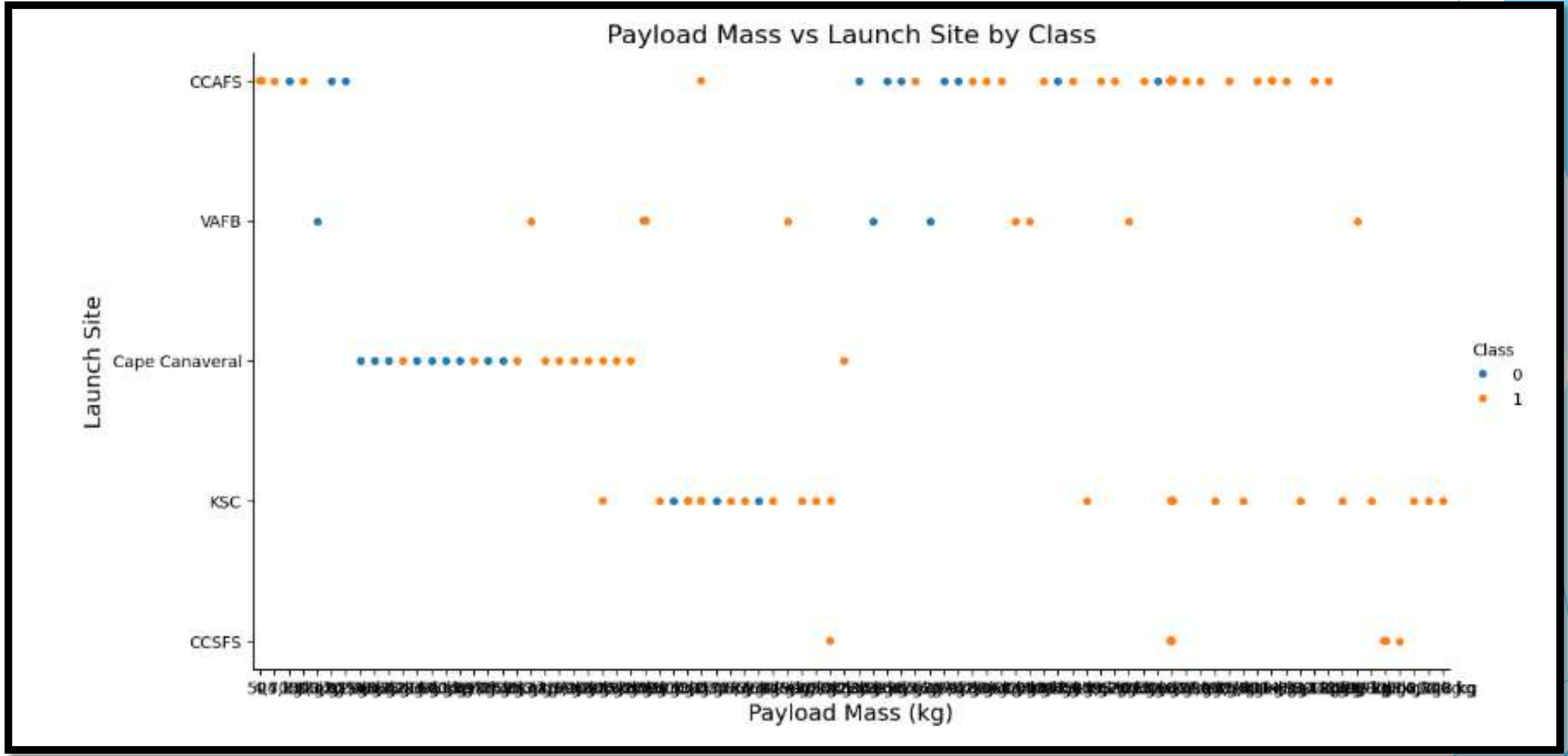
```
# TASK 12: Compare All Models
print("Logistic Regression Test Accuracy:", logreg_cv.score(X_test, Y_test))
print("SVM Test Accuracy:", svm_cv.score(X_test, Y_test))
print("Decision Tree Test Accuracy:", tree_cv.score(X_test, Y_test))
print("KNN Test Accuracy:", knn_cv.score(X_test, Y_test))
```

```
Logistic Regression Test Accuracy: 0.8333333333333334
SVM Test Accuracy: 0.8333333333333334
Decision Tree Test Accuracy: 0.6666666666666666
KNN Test Accuracy: 0.8333333333333334
```

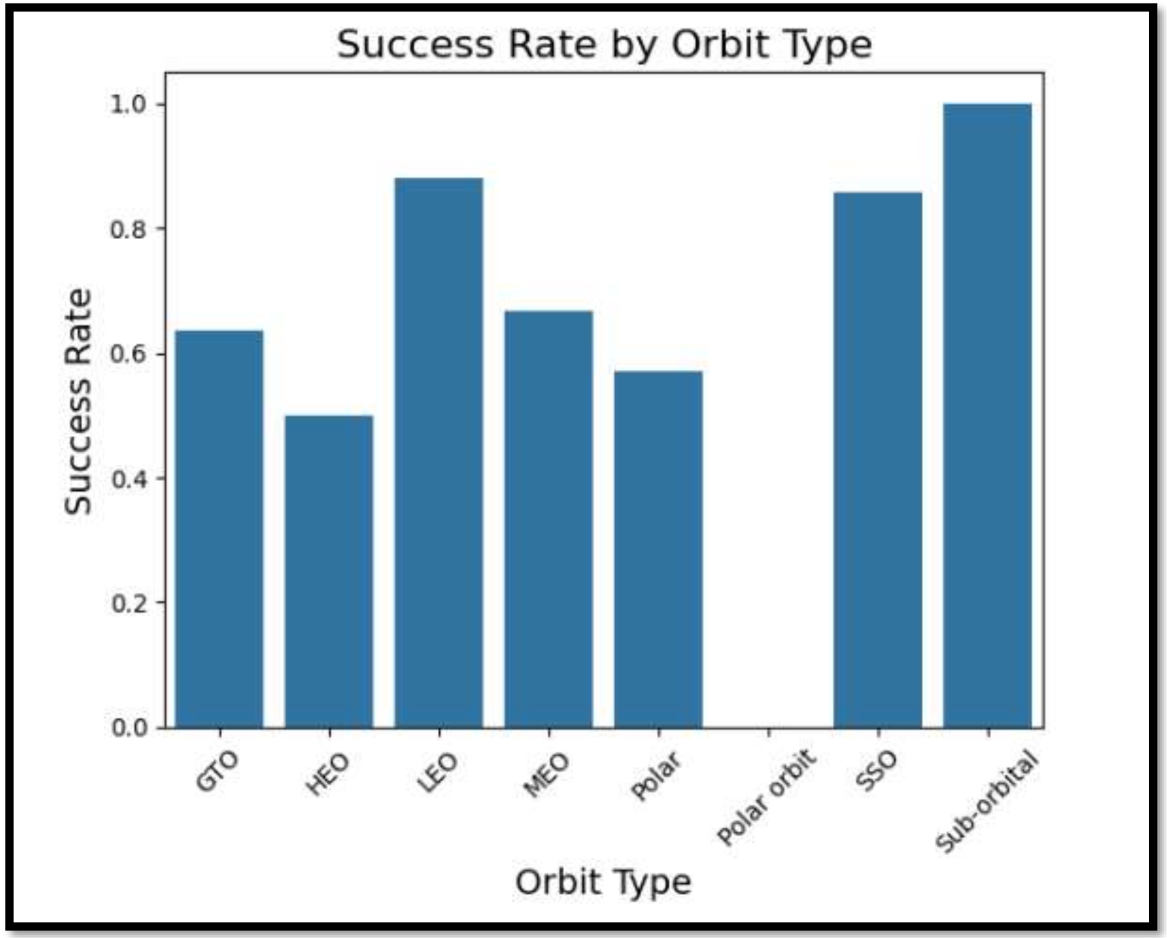
Flight Number vs. Launch Site



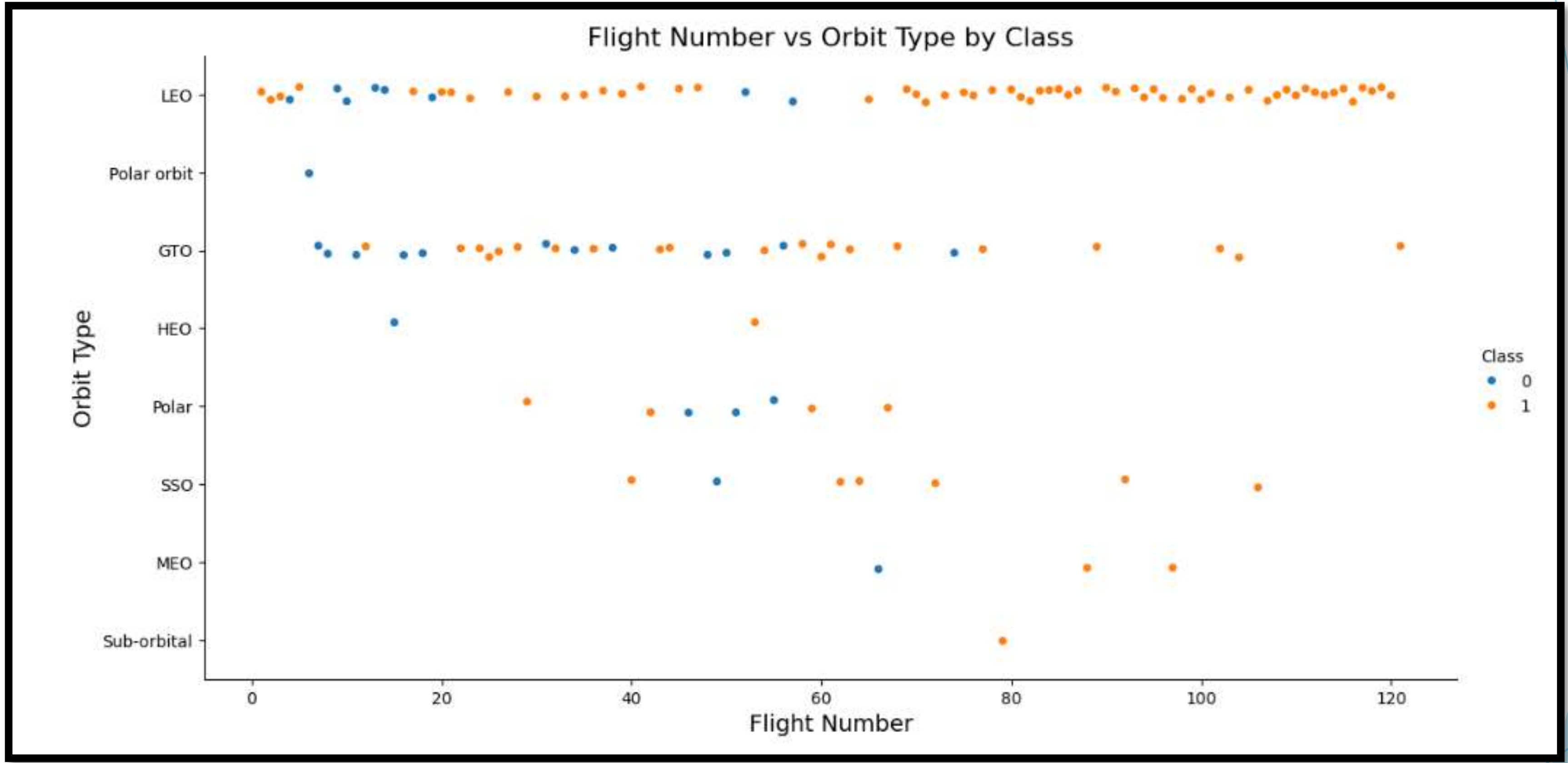
Payload vs. Launch Site



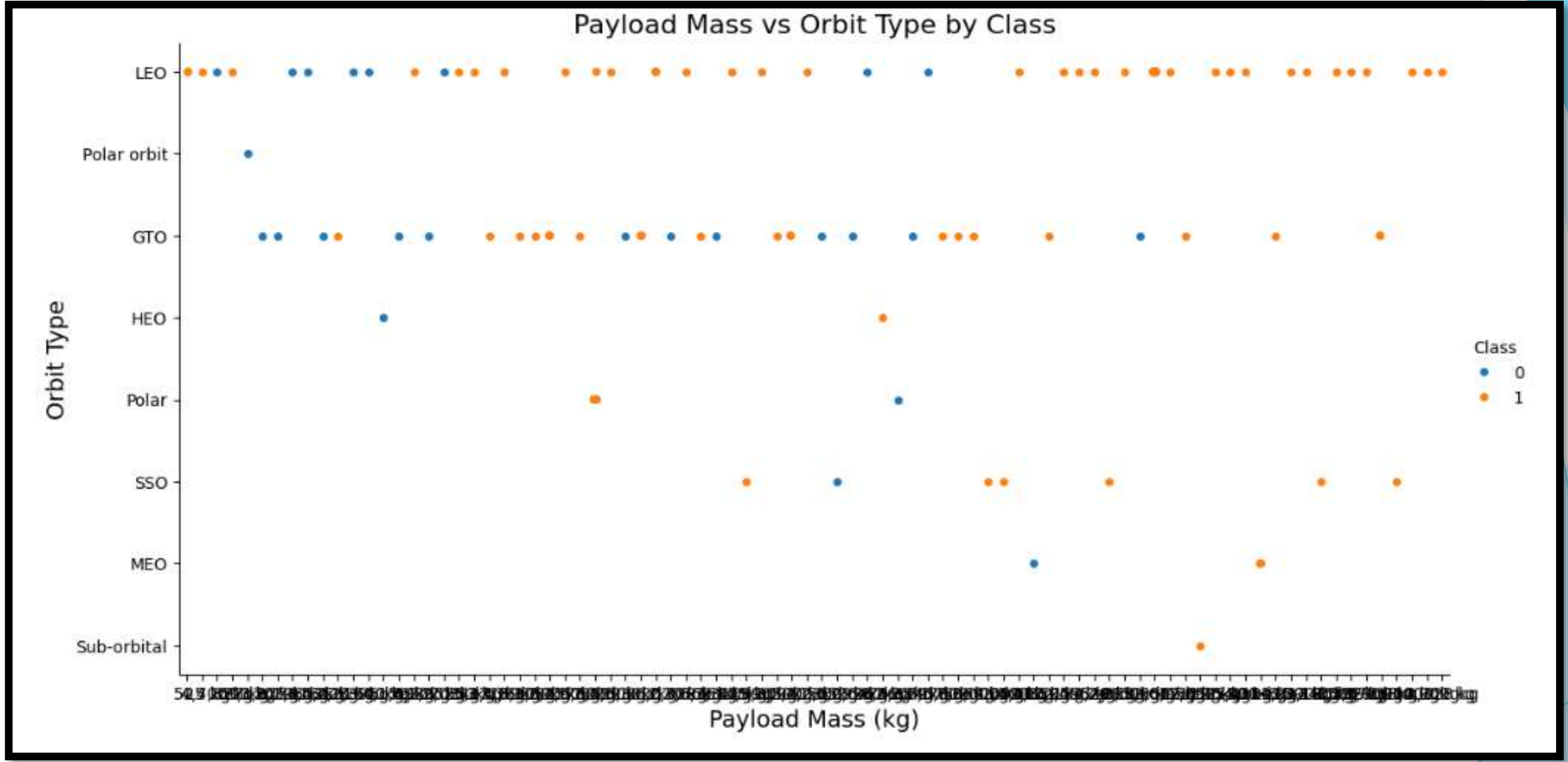
Success Rate vs. Orbit Type



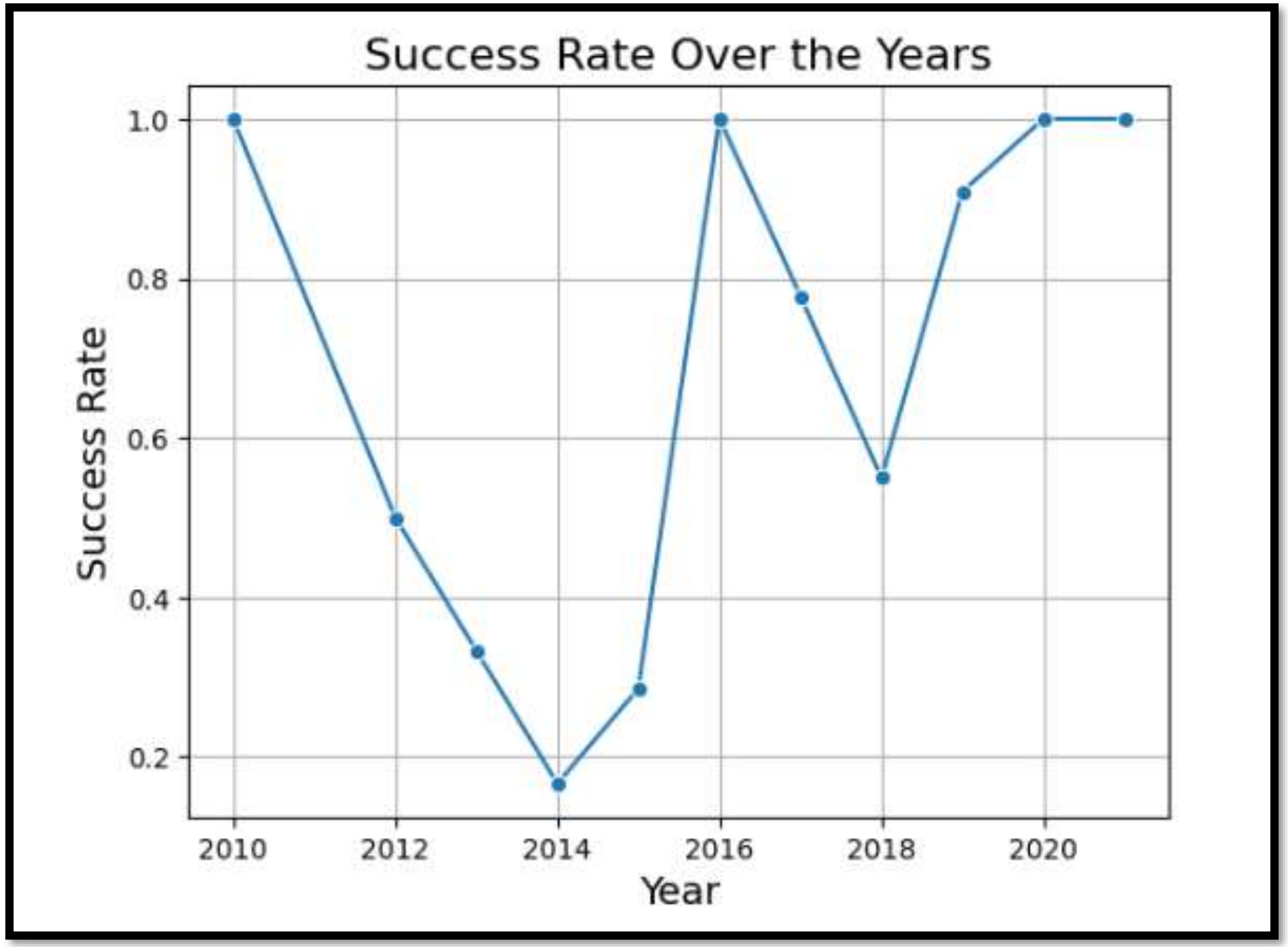
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

```
%sql SELECT DISTINCT launch_site FROM SPACEXTABLE;
```

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

launch_site
CCAFS
VAFB
Cape Canaveral
KSC
CCSFS

Launch Site Names Begin with 'CCA'

```
%sql SELECT * FROM SPACEXTABLE WHERE launch_site LIKE 'CCA%' LIMIT 5;
```

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

no	launch_site	payload	payload_mass	orbit	customer	launch_outcome	version_booster	booster_landing	date	time	class
1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	F9 v1.07B0003.18	Failure	4 June 2010	18:45	1
2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.07B0004.18	Failure	8 December 2010	15:43	1
3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.07B0005.18	No attempt	22 May 2012	07:44	1
4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success	F9 v1.07B0006.18	No attempt	8 October 2012	00:35	0
5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success	F9 v1.07B0007.18	No attempt	1 March 2013	15:10	1

Total Payload Mass

[13]:
```
%sql SELECT SUM(payload_mass) AS total_payload FROM SPACEXTABLE WHERE customer = 'NASA (CRS)';
```

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

[13]:

total_payload
None

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(payload_mass) AS avg_payload FROM SPACEXTABLE WHERE version_booster = 'F9 v1.1';
```

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

avg_payload
2.6

First Successful Ground Landing Date

```
%sql SELECT MIN(date) AS first_success_ground_pad FROM SPACEXTABLE WHERE booster_landing = 'Success (ground pad)';
```

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

first_success_ground_pad
None

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql
SELECT version_booster
FROM SPACEXTABLE
WHERE booster_landing = 'Success (drone ship)'
  AND CAST(payload_mass AS INTEGER) > 4000
  AND CAST(payload_mass AS INTEGER) < 6000;
```

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

version_booster

```
%%sql
SELECT launch_outcome, COUNT(*) AS outcome_count
FROM SPACEXTABLE
GROUP BY launch_outcome;
```

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

launch_outcome	outcome_count
Failure	1
Success	32
Success	88

Total Number of Successful and Failure Mission Outcomes

```
%%sql
SELECT version_booster
FROM SPACEXTABLE
WHERE booster_landing = 'Success (drone ship)'
  AND CAST(payload_mass AS INTEGER) > 4000
  AND CAST(payload_mass AS INTEGER) < 6000;
```

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

version_booster

```
%%sql
SELECT launch_outcome, COUNT(*) AS outcome_count
FROM SPACEXTABLE
GROUP BY launch_outcome;
```

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

launch_outcome	outcome_count
Failure	1
Success	32
Success	88

Boosters Carried Maximum Payload

```
%%sql
SELECT version_booster, payload_mass
FROM SPACEXTABLE
WHERE CAST(payload_mass AS INTEGER) = (
    SELECT MAX(CAST(payload_mass AS INTEGER)) FROM SPACEXTABLE
);
```

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

version_booster	payload_mass
F9 v1.1[	570 kg

2015 Launch Records

```
%%sql
SELECT
    SUBSTR(date, 6, 2) AS month,
    booster_landing,
    version_booster,
    launch_site
FROM SPACEXTABLE
WHERE booster_landing = 'Failure (drone ship)'
  AND SUBSTR(date, 1, 4) = '2015';
```

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

month	booster_landing	version_booster	launch_site

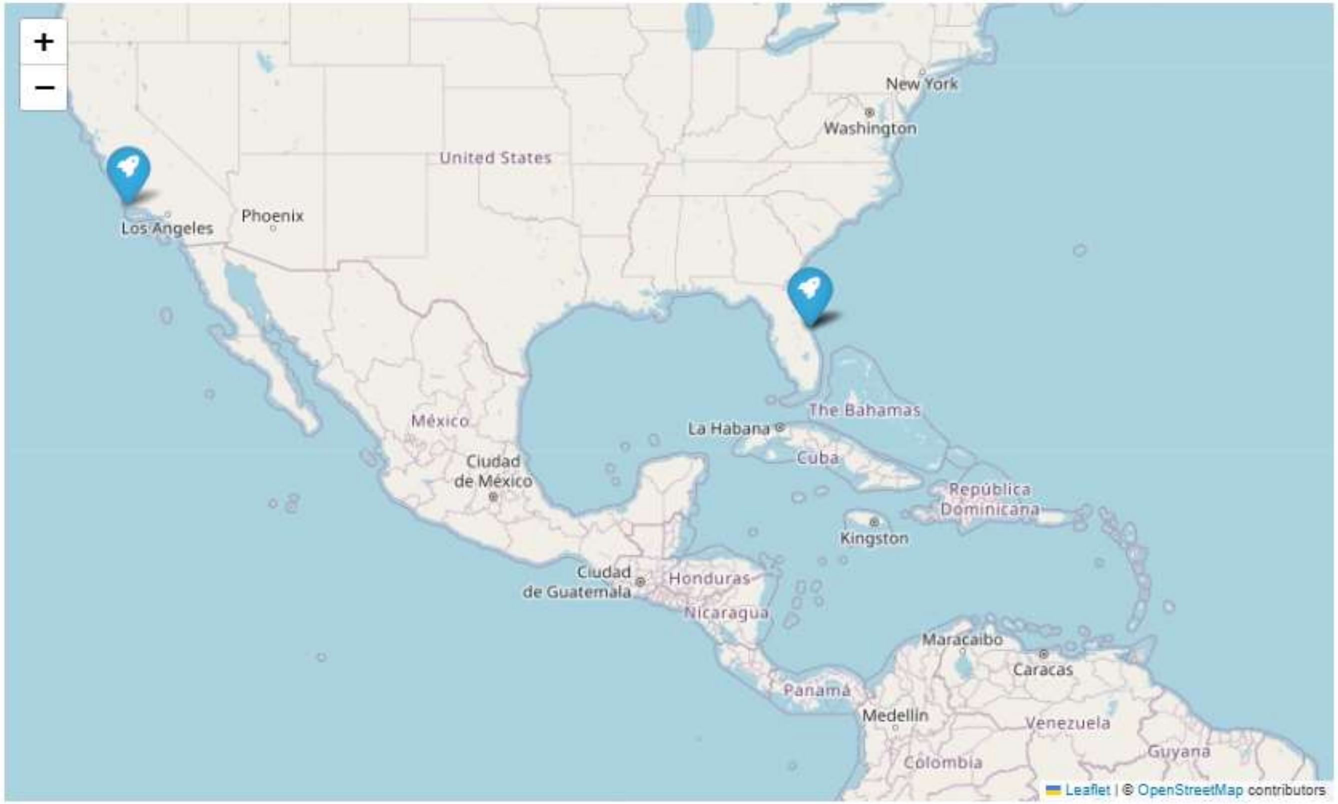
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

```
%%sql
SELECT booster_landing, COUNT(*) AS outcome_count
FROM SPACEXTABLE
WHERE date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY booster_landing
ORDER BY outcome_count DESC;
```

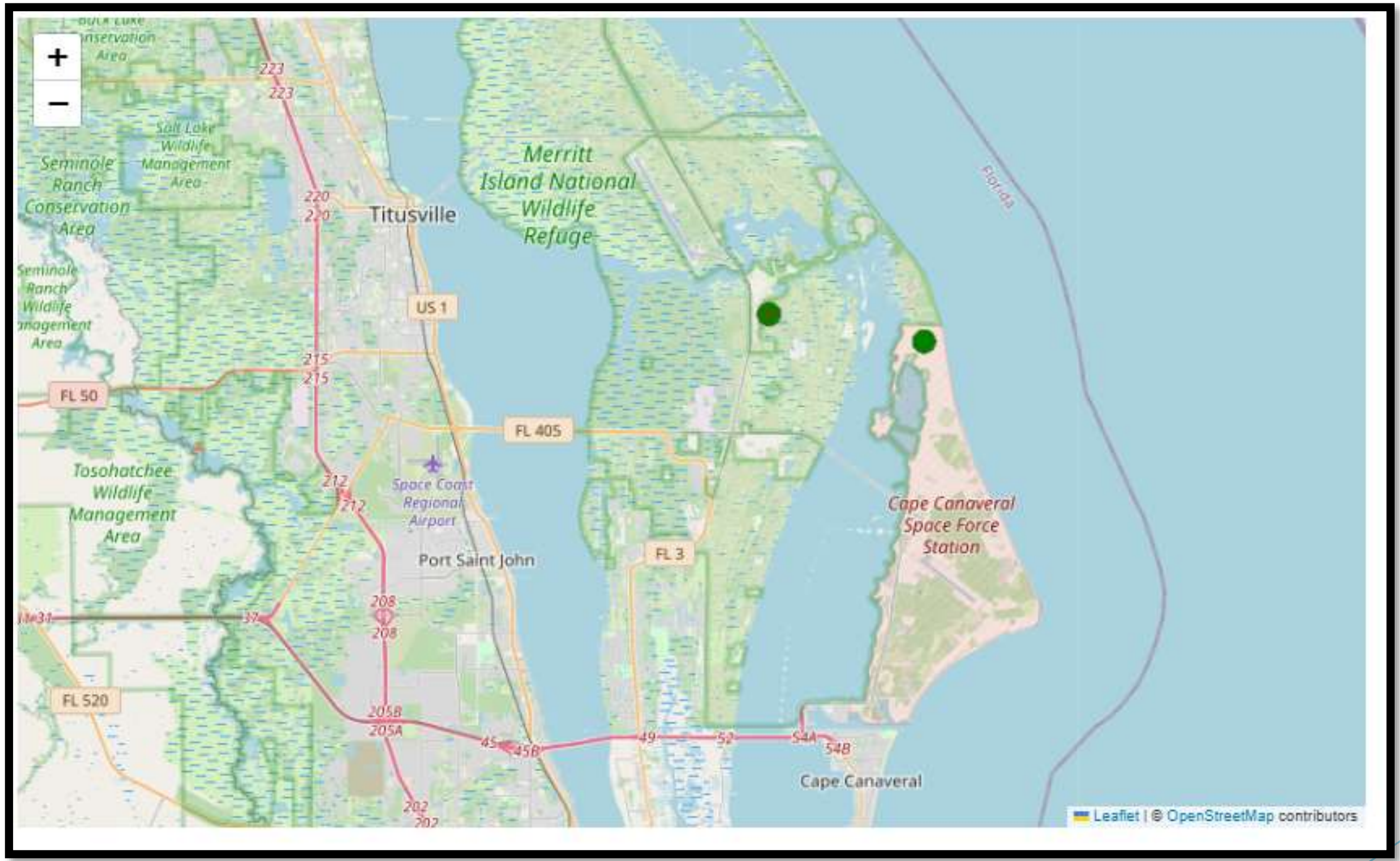
```
 * sqlite:///my_data1.db
Done.
```

booster_landing	outcome_count

<Folium Map Screenshot 1>



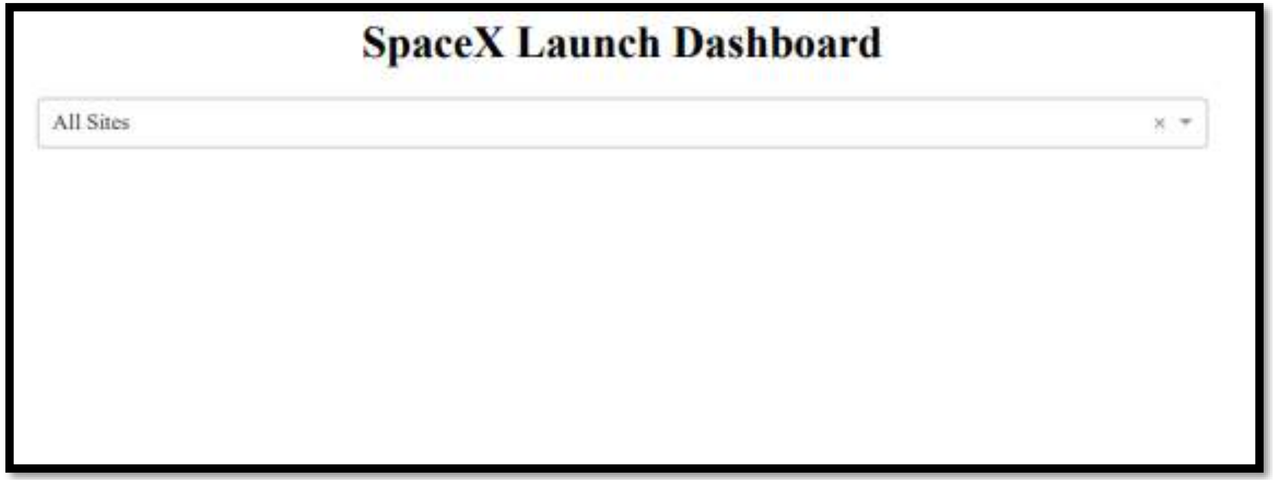
<Folium Map Screenshot 2>



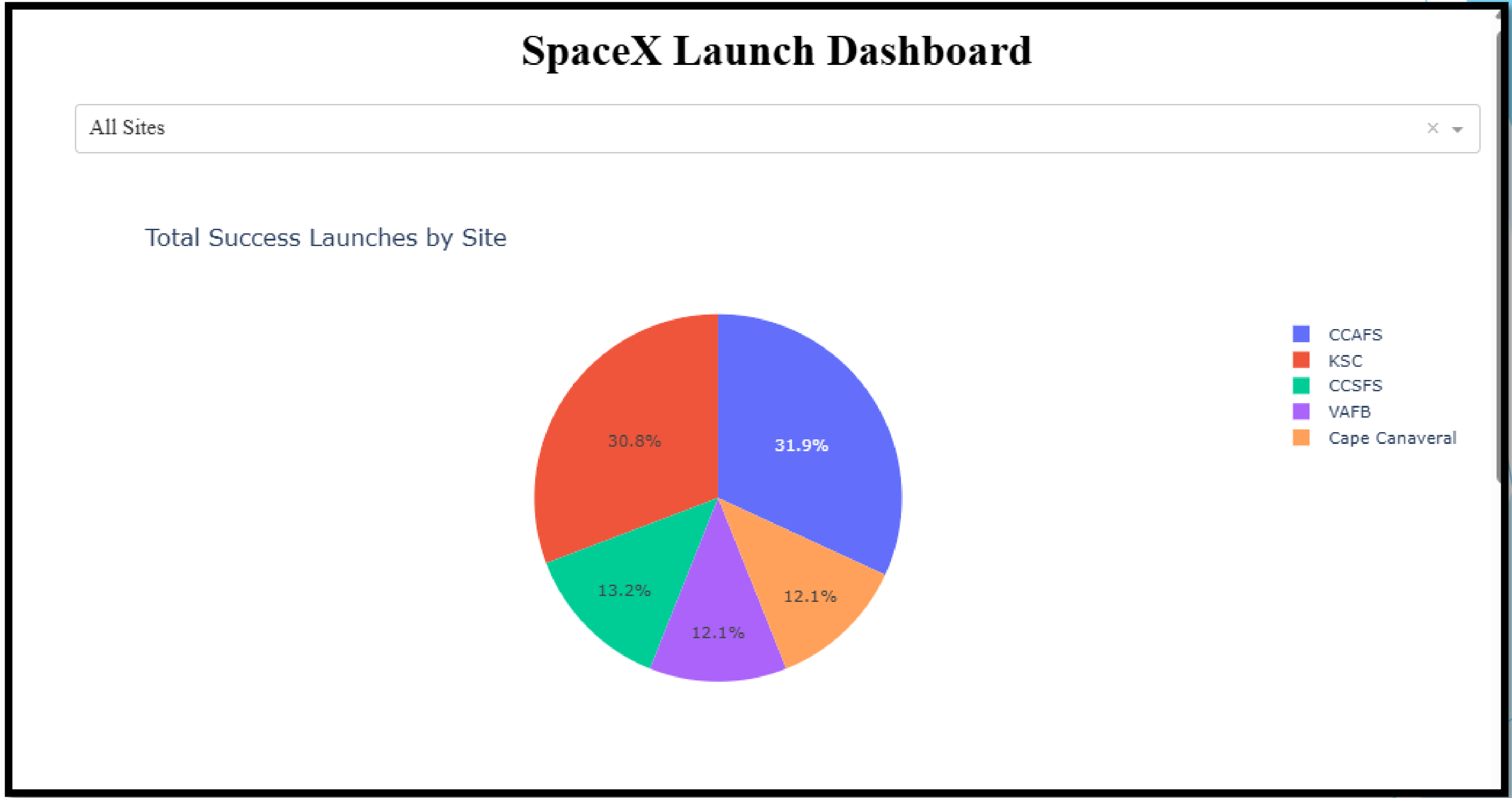
<Folium Map Screenshot 3>

	Launch Site	Feature	Distance (km)
0	CCAFS LC-40	Coastline	0.93
1	CCAFS LC-40	Railway	1.31
2	CCAFS LC-40	Highway	3832.99
3	CCAFS LC-40	City	78.51
4	VAFB SLC-4E	Coastline	3833.86
5	VAFB SLC-4E	Railway	3831.99
6	VAFB SLC-4E	Highway	0.36
7	VAFB SLC-4E	City	3760.48
8	KSC LC-39A	Coastline	7.79
9	KSC LC-39A	Railway	6.06
10	KSC LC-39A	Highway	3826.19
11	KSC LC-39A	City	71.77

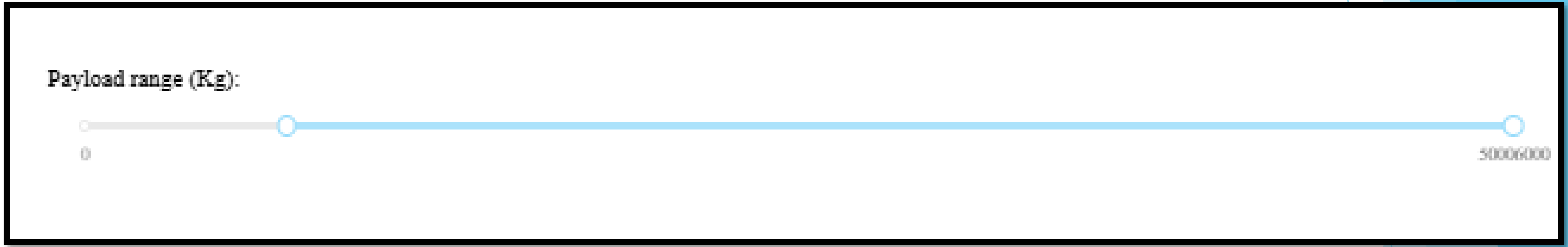
<Dashboard Screenshot 1>



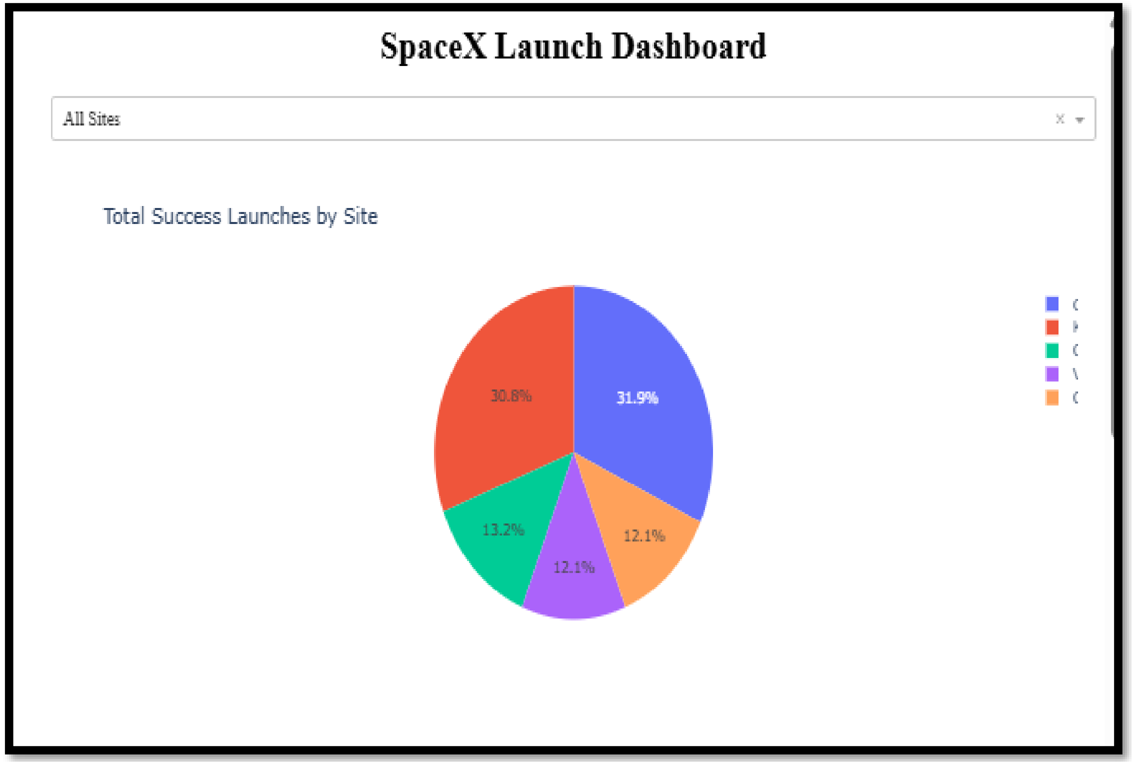
<Dashboard Screenshot 2>

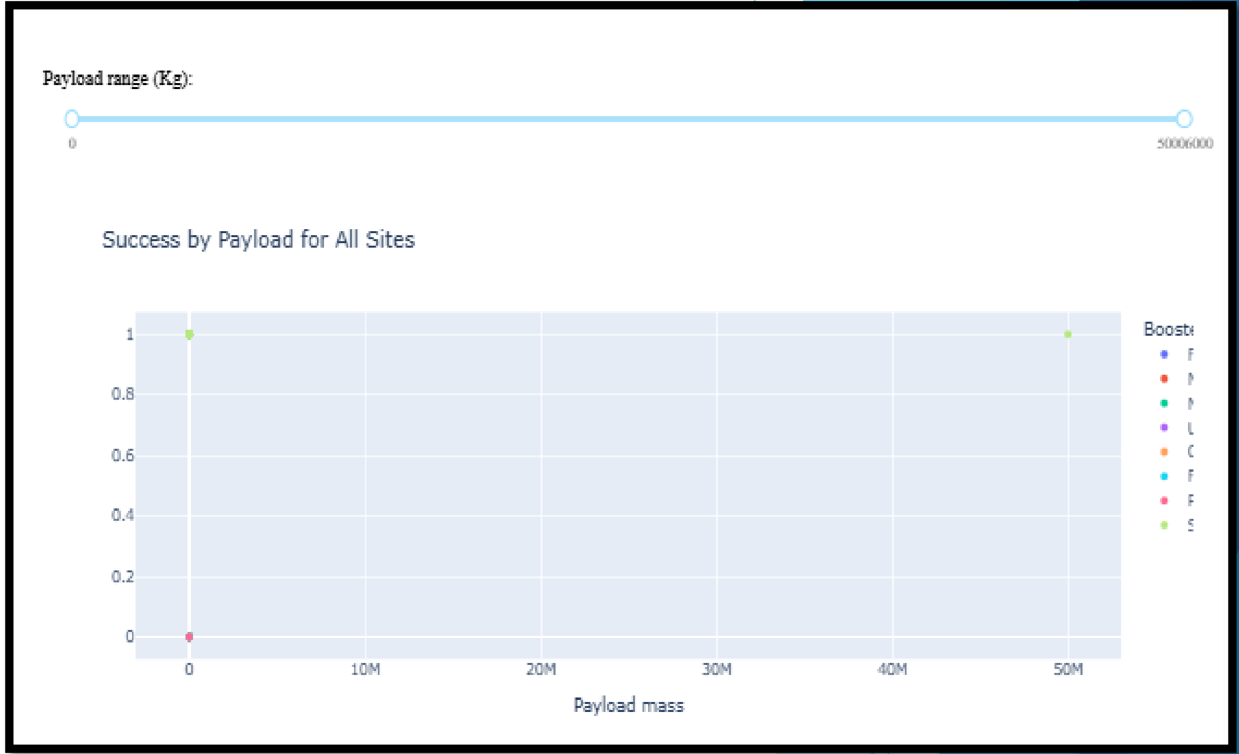


<Dashboard Screenshot 3>



<Dashboard Screenshot 4>



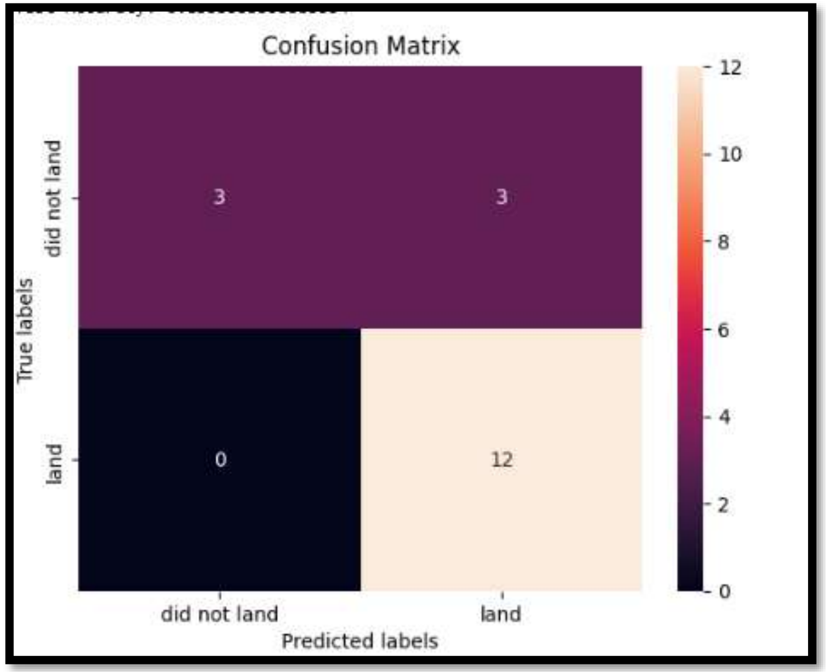


Classification Accuracy

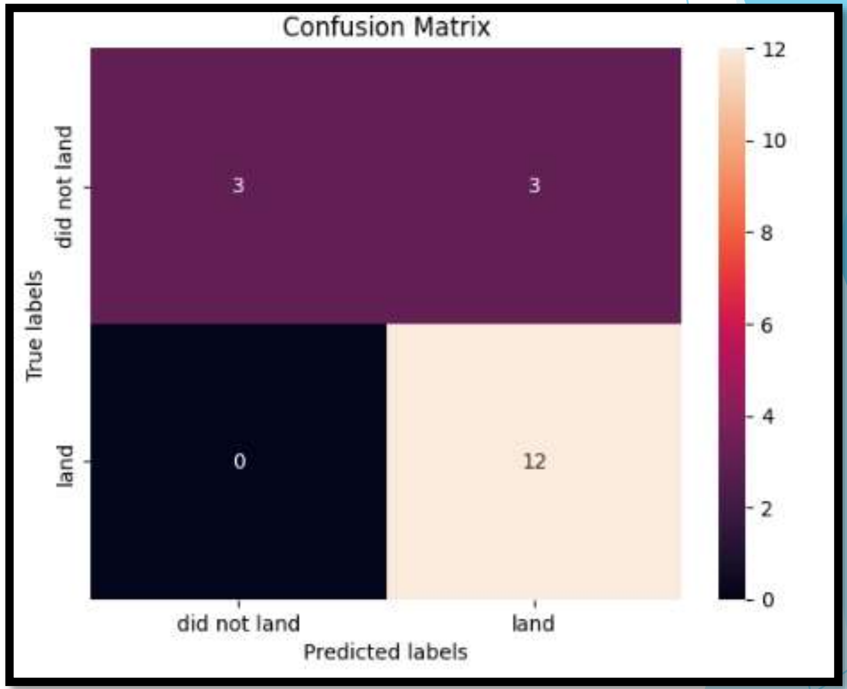
```
# TASK 12: Compare All Models
print("Logistic Regression Test Accuracy:", logreg_cv.score(X_test, Y_test))
print("SVM Test Accuracy:", svm_cv.score(X_test, Y_test))
print("Decision Tree Test Accuracy:", tree_cv.score(X_test, Y_test))
print("KNN Test Accuracy:", knn_cv.score(X_test, Y_test))
```

```
Logistic Regression Test Accuracy: 0.8333333333333334
SVM Test Accuracy: 0.8333333333333334
Decision Tree Test Accuracy: 0.6666666666666666
KNN Test Accuracy: 0.8333333333333334
```

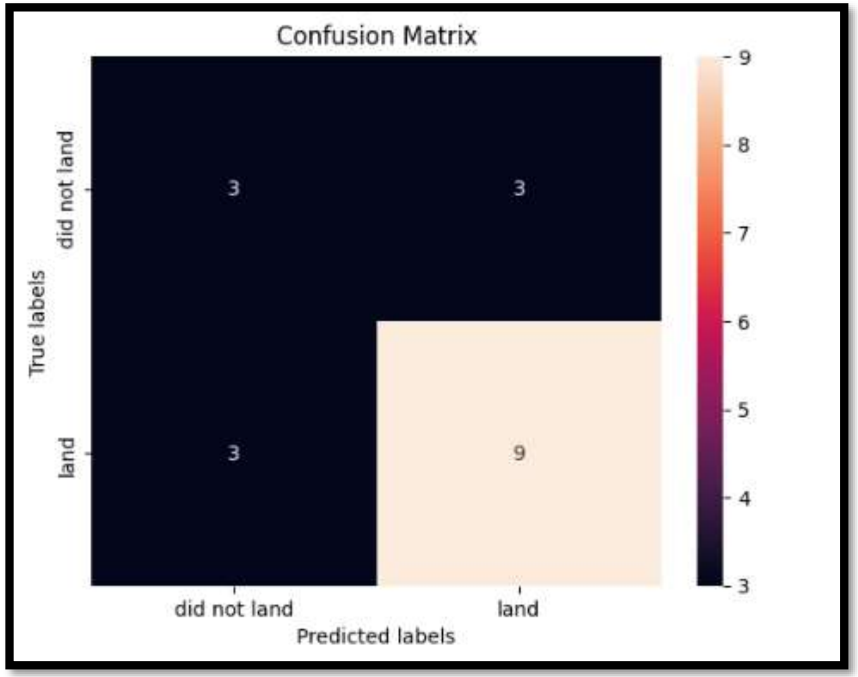
Confusion Matrix



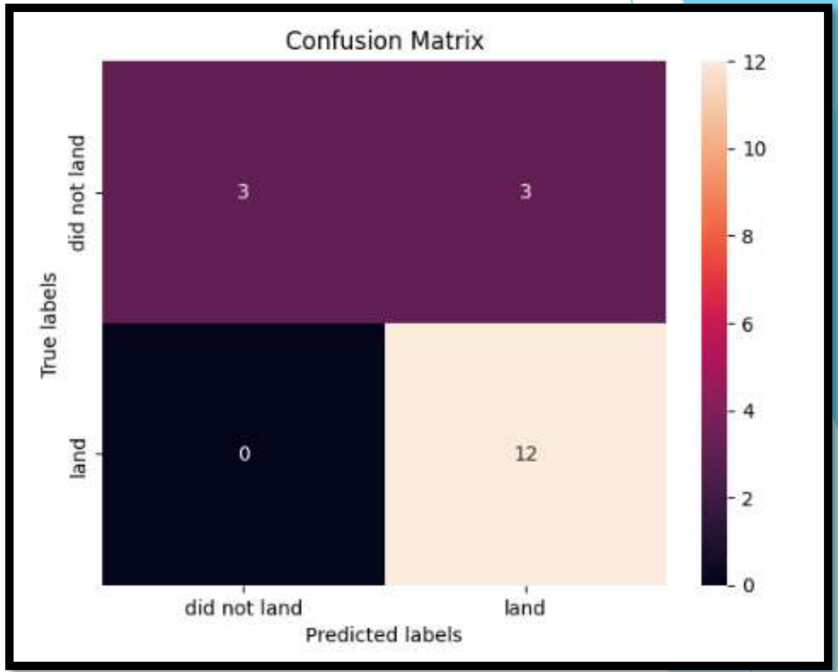
Logistic Regression



SVM Test Accuracy



Decision Tree



KNN

Conclusions

After querying the SpaceX dataset for payloads launched under the customer's name 'NASA (CRS)', the total payload mass returned was None. This indicates that:

1. **There may be no exact match** for 'NASA (CRS)' in the customer field due to formatting inconsistencies, extra spaces, or different naming conventions.
2. **Payload mass values for NASA (CRS) launches could be missing or null**, which causes the SUM() function to return None.
3. Alternatively, **the dataset may not contain any launches** explicitly labeled under 'NASA (CRS)' as the customer.


Thank you!