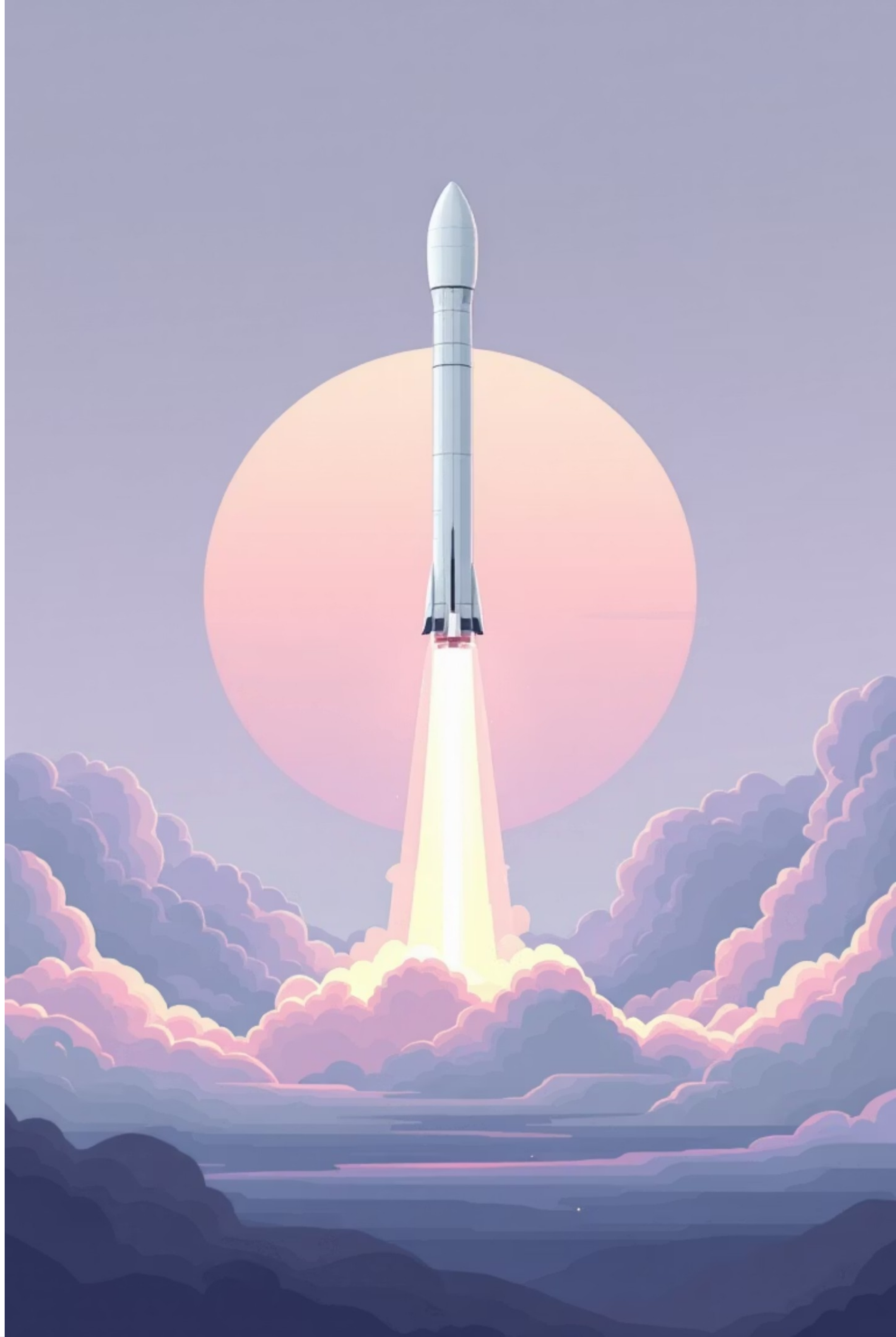
Predicting SpaceX Falcon 9 First-Stage Landing Success

A Data-Driven Approach

MALOO BELL Loïck MichelOctober 9, 2025



Presentation Roadmap

01

Executive Summary

Key findings and model performance overview

02

Introduction

Project background and problem statement

03

Methodology

Data collection through predictive modeling

04

Results

EDA insights and model comparison

05

Conclusion

Impact and future directions

Executive Summary

Comprehensive Methodology

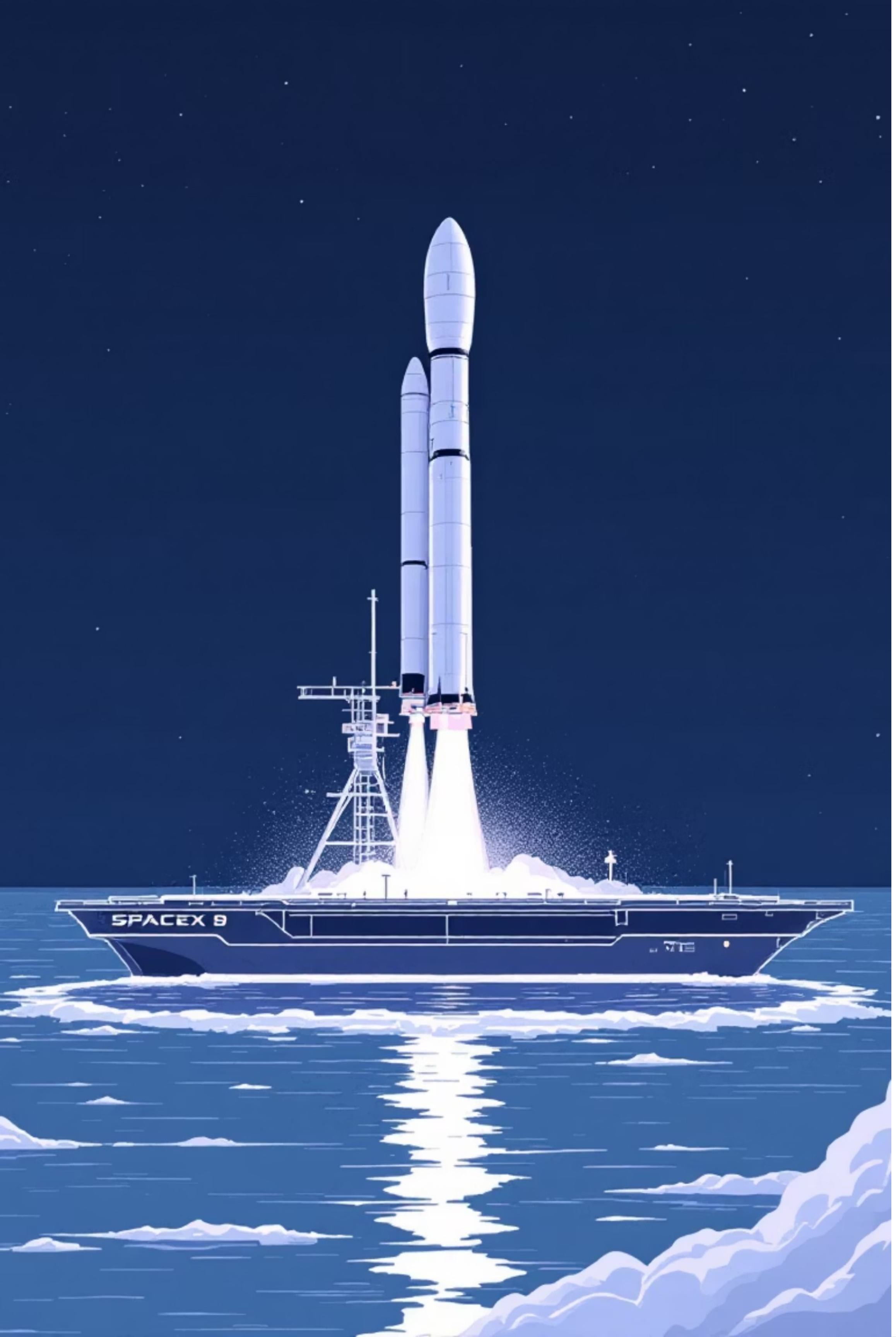
This project executed an end-to-end data science workflow to analyze and predict SpaceX landing outcomes. The process began with robust data collection, combining web scraping of public records with direct API calls for technical specifications.

Subsequent stages involved rigorous data wrangling, in-depth exploratory analysis using both SQL queries and Python-based visualizations, and the development of interactive tools, including a Plotly Dash dashboard.

Key Results

The analysis uncovered several key insights, most notably a strong positive correlation between flight number and landing success, confirming a significant operational learning curve.

The **KSC LC-39A launch site** was identified as having the highest success rate. A tuned K-Nearest Neighbors (KNN) model achieved ~86% accuracy in predicting landing outcomes.



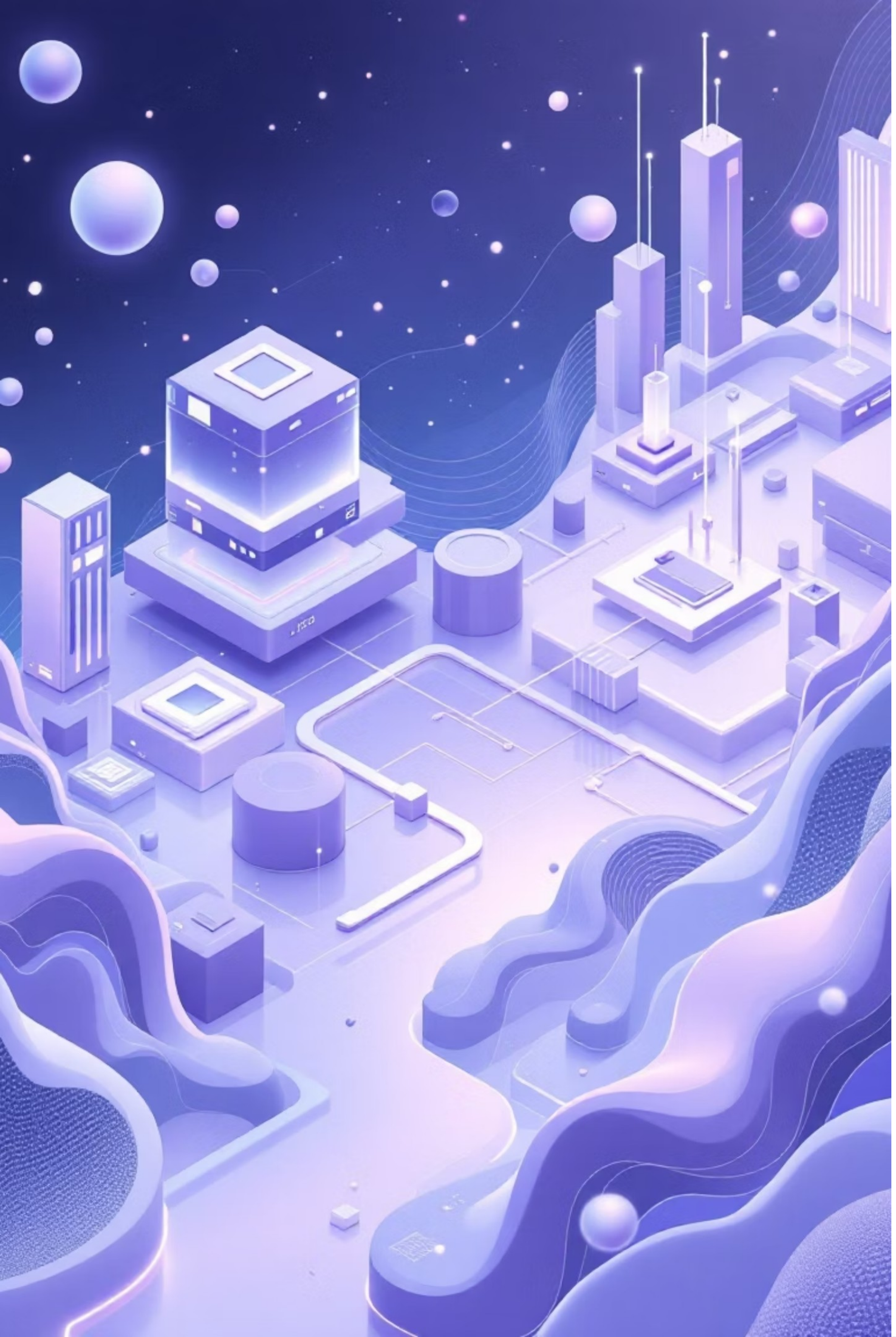
Introduction: The SpaceX Revolution

Project Background

SpaceX has fundamentally disrupted the aerospace industry by mastering the art of landing and reusing the first stage of its Falcon 9 rockets. This capability drastically reduces the cost of access to space, enabling ambitious projects like the Starlink satellite constellation.

Problem Statement

Can we leverage historical launch data to build a robust predictive model? This involves investigating the influence of various factors—launch site, payload mass, and orbital destination—to create a tool that aids in operational decision-making.



Methodology

Methodology Overview

Data Collection

Acquired data by scraping Wikipedia and querying the SpaceX REST API to build a comprehensive historical record

Data Wrangling

Cleaned and merged datasets, handled missing values, and engineered binary target variable

Exploratory Analysis

Employed SQL queries and Python visualization libraries to identify key trends and correlations

Interactive Analytics

Developed Plotly Dash dashboard and Folium maps for intuitive data exploration

Predictive Modeling

Built, tuned, and evaluated multiple classification models (Logistic Regression, SVM, Decision Tree, KNN)

Data Collection Strategy

1

Wikipedia Scraping

Foundational historical data was collected by scraping the "List of Falcon 9 and Falcon Heavy launches" Wikipedia page using BeautifulSoup. This provided a high-level mission manifest with launch dates, outcomes, and basic mission parameters.

2

SpaceX API Enrichment

The Python requests library sent GET requests to the /launches/past endpoint of the SpaceX v4 REST API. Custom parsing functions extracted granular details including payload mass, core serial numbers, flight numbers, and landing specifics—enriching the scraped data with official technical specifications.

```python
# Takes the dataset and uses the cores column to call the API and append the data to the lists
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'])
```

[6] Python

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```python
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

[7] Python

```python
response = requests.get(spacex_url)
```

`Heavy launches` Wikipage updated on `9th June 2021`

```python
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686

headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
                  "AppleWebKit/537.36 (KHTML, like Gecko) "
                  "Chrome/91.0.4472.124 Safari/537.36"
}
```

[4] Python

Next, request the HTML page from the above URL and get a `response` object

Générer + Code + Marquage

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```python
# use requests.get() method with the provided static_url and headers
# assign the response to a object
response = requests.get(static_url, headers=headers)
```

[5] Python

Create a `BeautifulSoup` object from the HTML `response`

Data Wrangling & Processing

Data Integration

The two datasets (scraped and API) were carefully merged into a single, unified DataFrame. Missing PayloadMass values were imputed using the column's mean value to preserve dataset size while maintaining statistical integrity.

Feature Engineering

A critical step was engineering the binary **Class** target variable from the categorical Outcome column. A value of 1 was assigned to successful landings, while 0 was assigned to predefined failures (e.g., Failure (parachute), False Ocean).

Data Preparation

Categorical features like LaunchSite and Orbit were converted using one-hot encoding. All numerical features were standardized using StandardScaler, crucial for distance-based algorithms like SVM and KNN.

Exploratory Data Analysis

```
%sql sqlite:///my_data1.db
```

[6] Python

```
import pandas as pd
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/l
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="multi")
```

[7] Python

```
101
```

Note:This below code is added to remove blank rows from table

```
#DROP THE TABLE IF EXISTS

%sql DROP TABLE IF EXISTS SPACEXTABLE;
```

[8] Python

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

```
[]
```

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

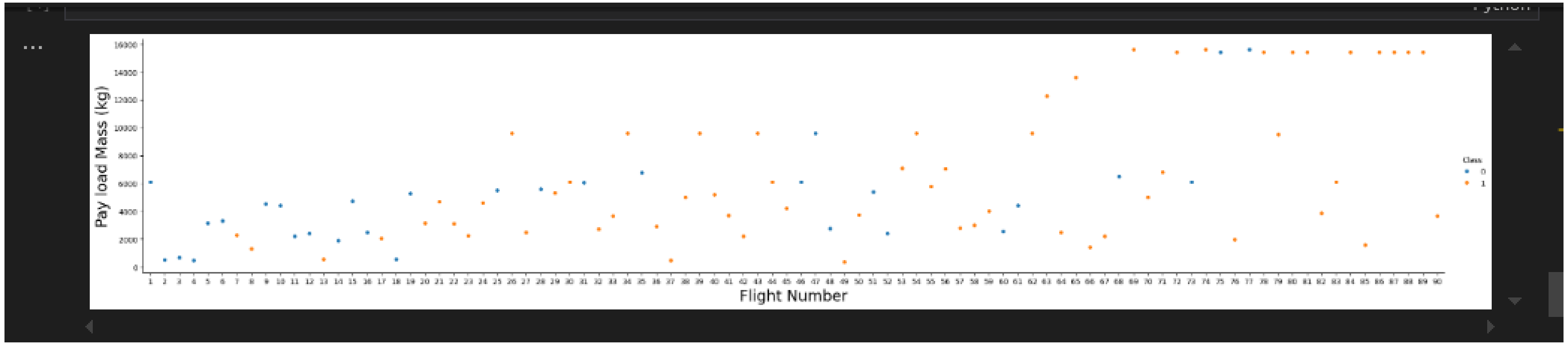
```
# Display 5 records where launch sites begin with the string 'CCA'
%sql select * from SPACEXTABLE where "Launch_Site" like 'CCA%' limit 5
```

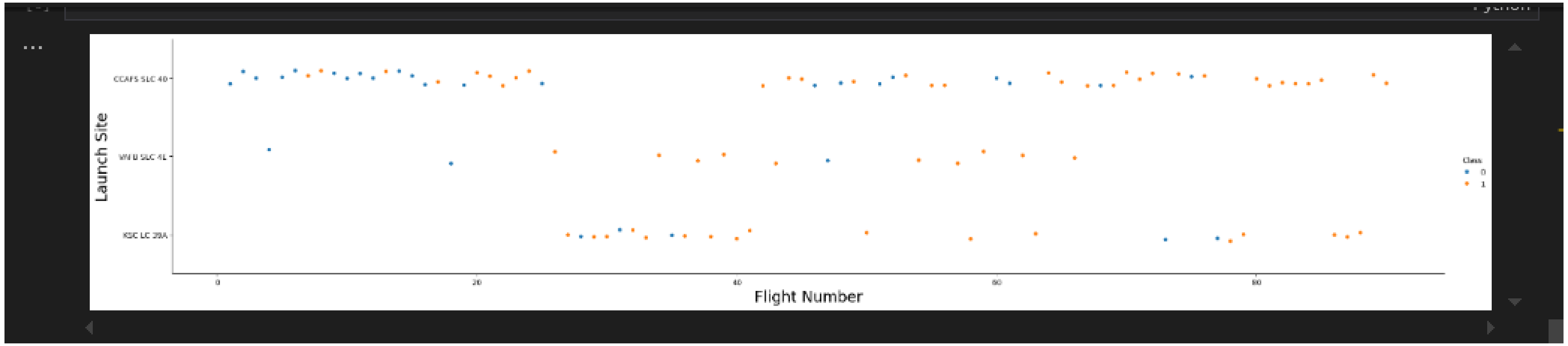
[16]

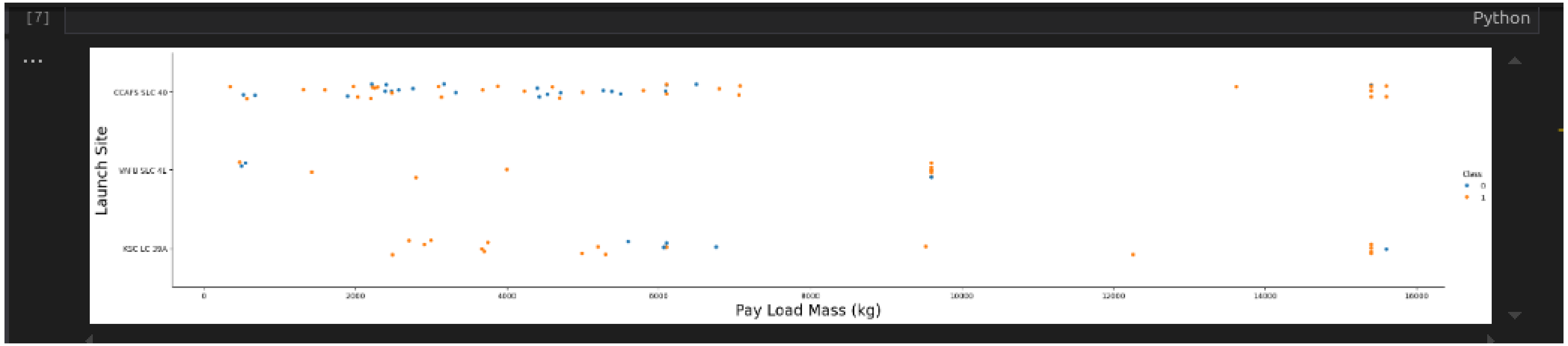
Python

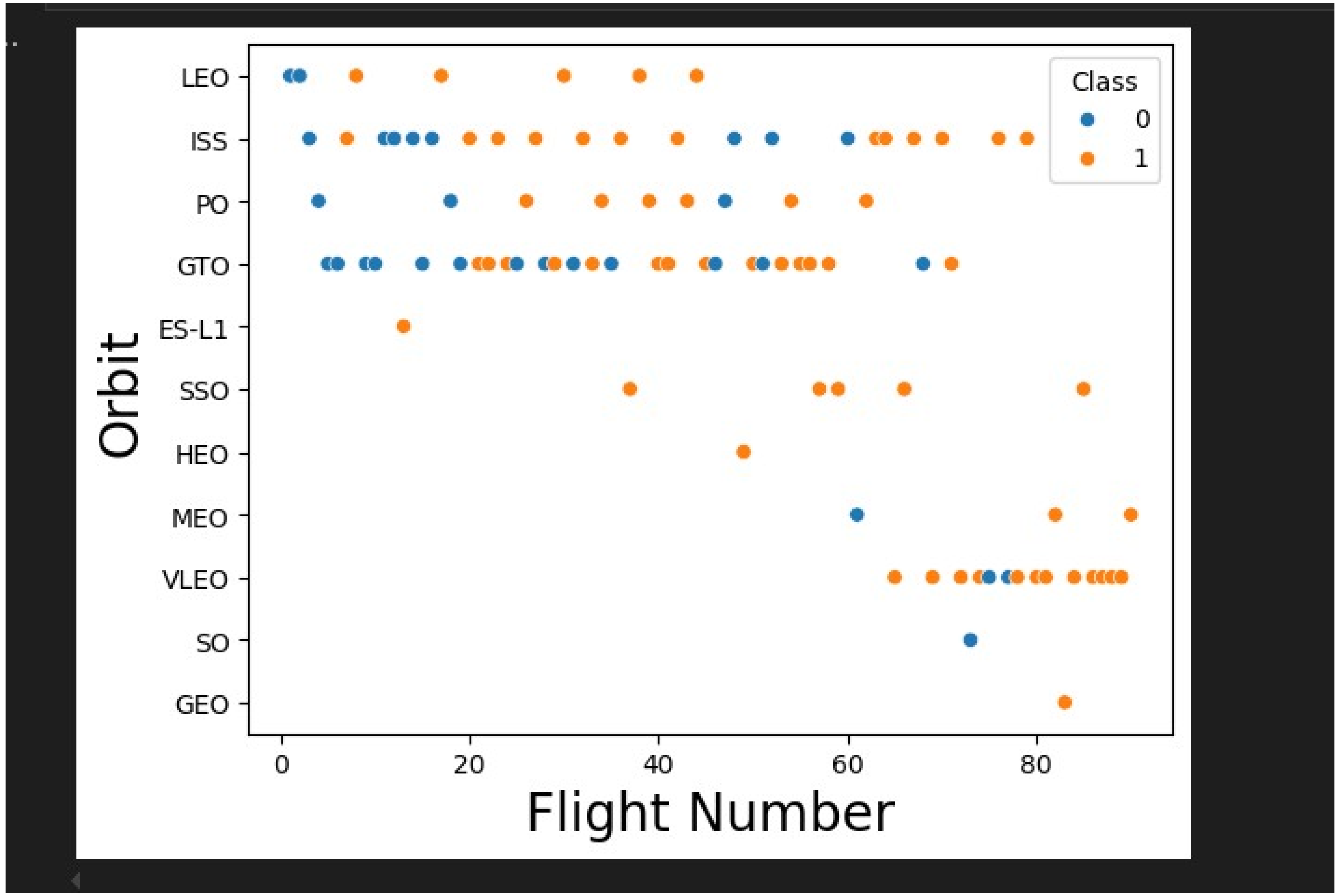
```
 * sqlite:///my_data1.db
Done.
```

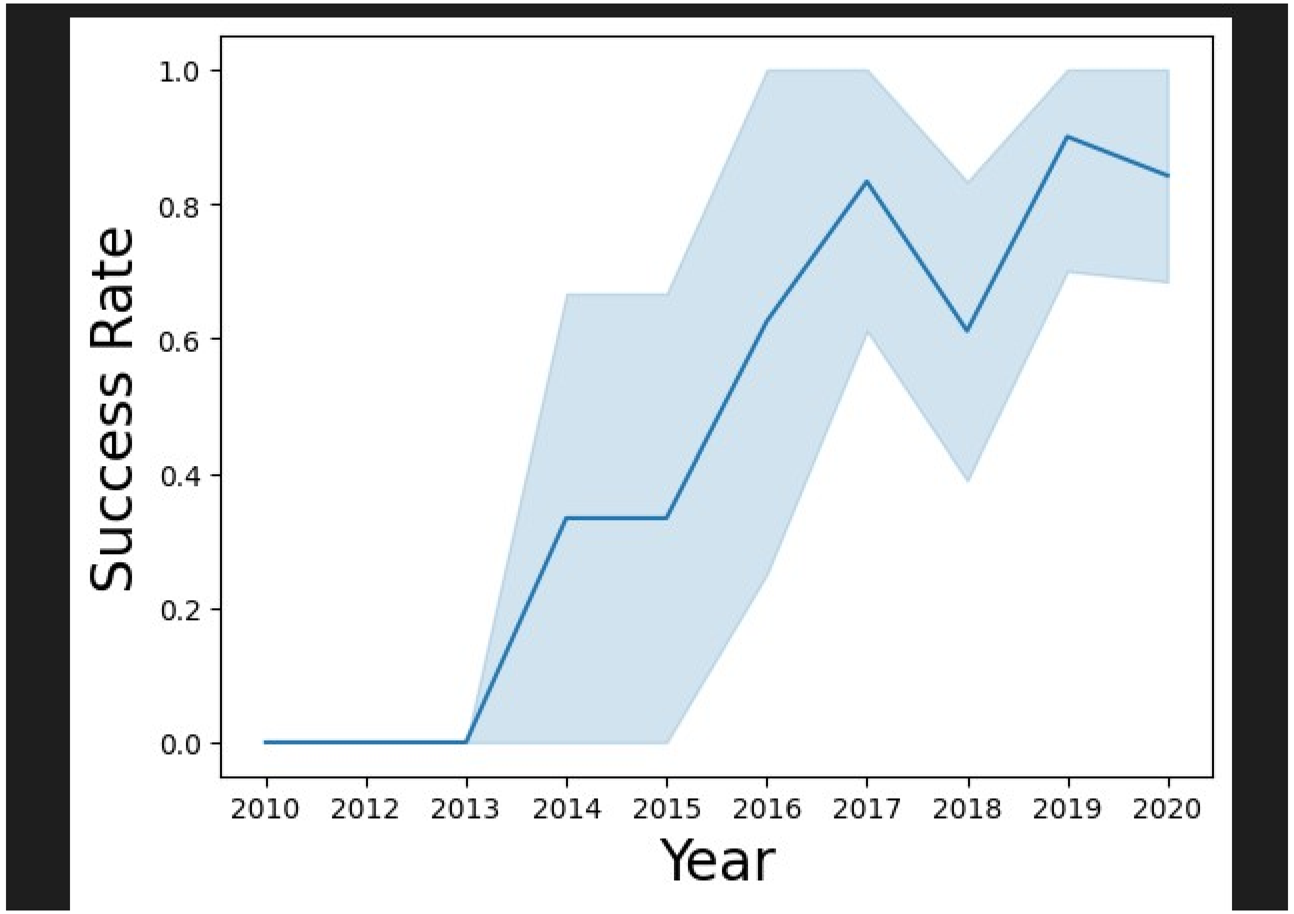
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outco
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachu
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight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


Pay load Mass (kg)
Flight Number
Class


CCAFS SLC 40
Launch Site
Flight Number
Class


[7]
Python
CCAFS SLC 40
Launch Site
Pay Load Mass (kg)
Class





Made with GAMMA

Key EDA Insights

Success Improves Over Time

Scatter plot analysis reveals a strong positive correlation between flight number and landing success—visual confirmation of SpaceX's iterative improvement process.

KSC LC-39A Leads

Bar chart comparison shows **KSC LC-39A** has the highest success rate among all launch sites, potentially due to newer infrastructure and mission profiles.

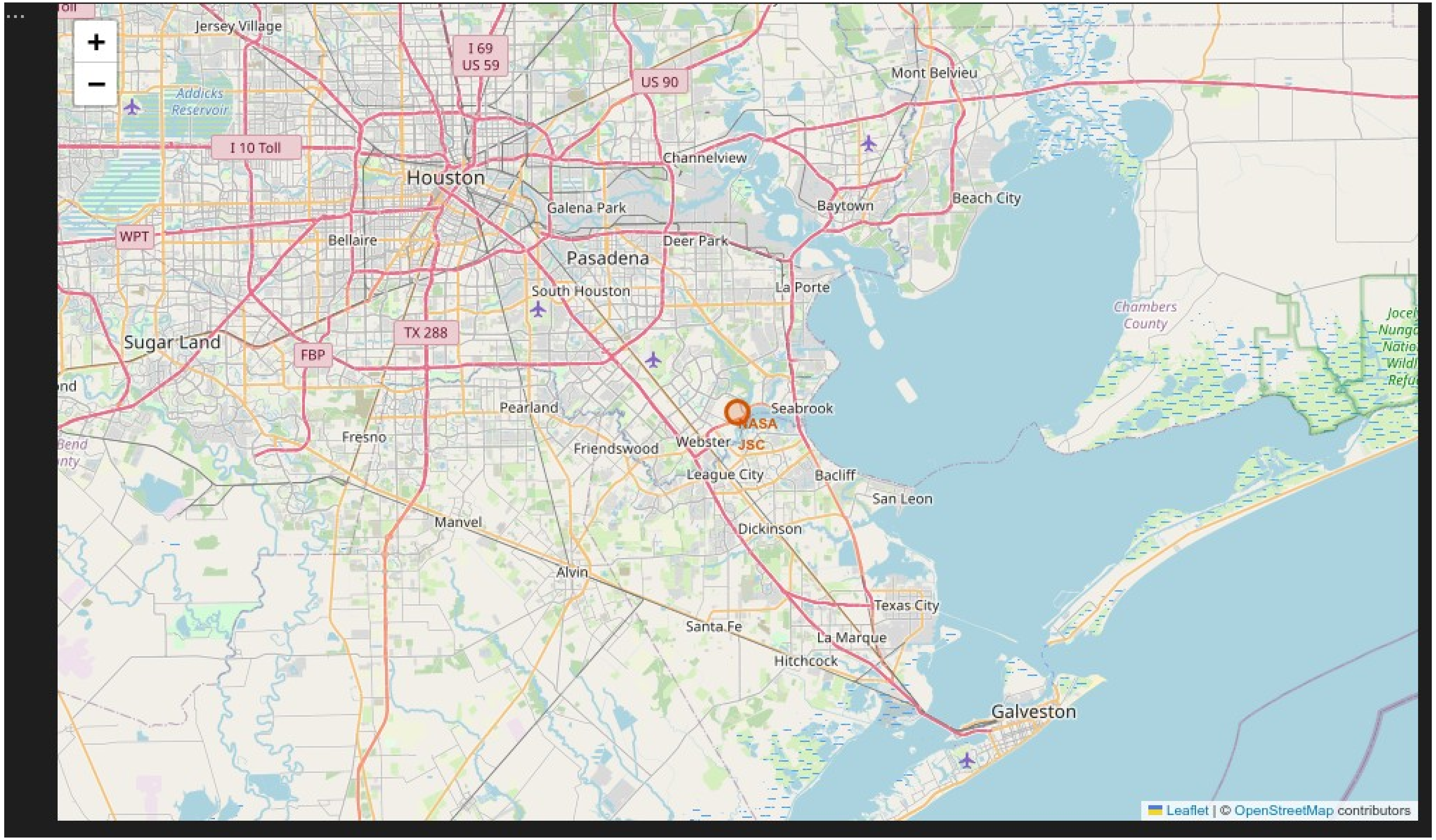
Payload Mass Trade-off

Heavier payloads correlate with slightly lower success rates—highlighting the engineering balance between mission objectives and booster reusability.

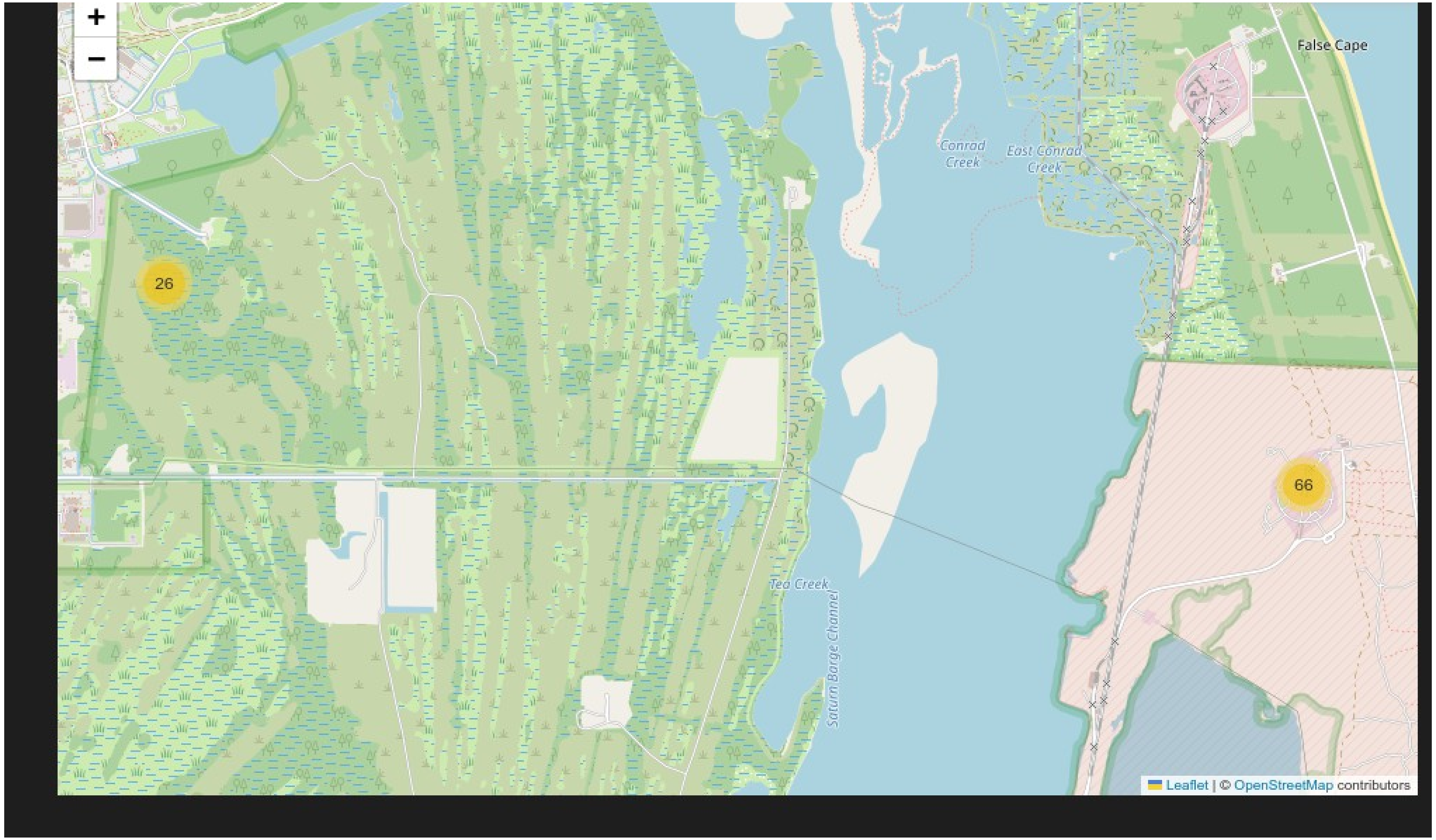
Orbit Matters

High-energy orbits like GTO (Geostationary Transfer Orbit) show lower success rates compared to less demanding LEO (Low Earth Orbit) missions.

SQL queries and Python visualization libraries (Seaborn, Matplotlib) were employed to systematically analyze the dataset, identifying patterns across **launch sites, booster versions, payload characteristics, and orbital destinations**.



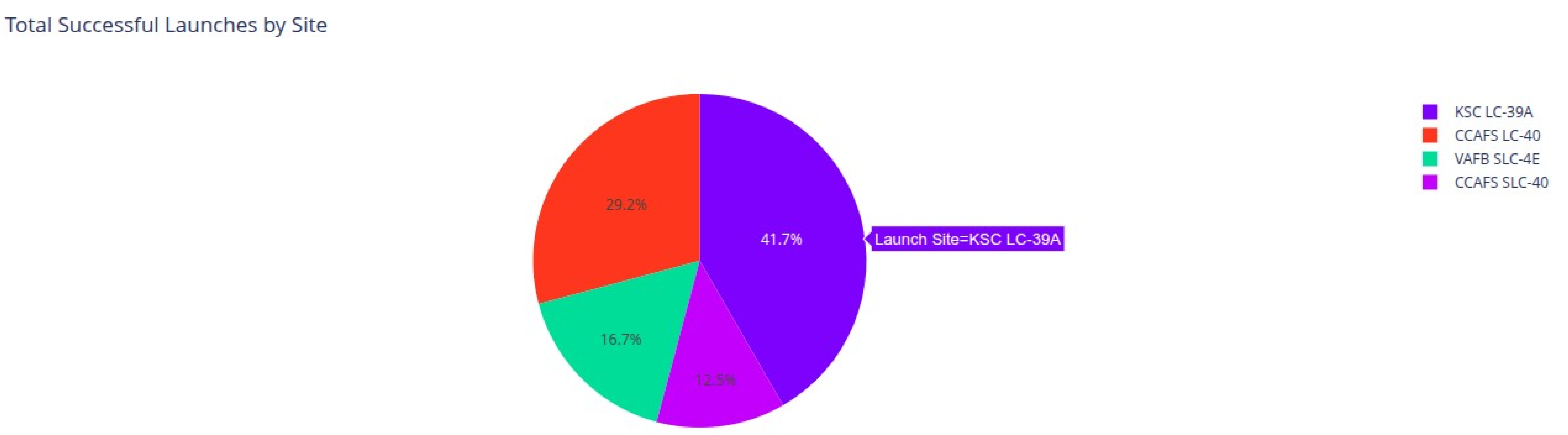
Made with GAMMA



+
−
False Cape
Conrad Creek
East Conrad Creek
26
66
Tea Creek
Saturn Barge Channel
Leaflet | © OpenStreetMap contributors

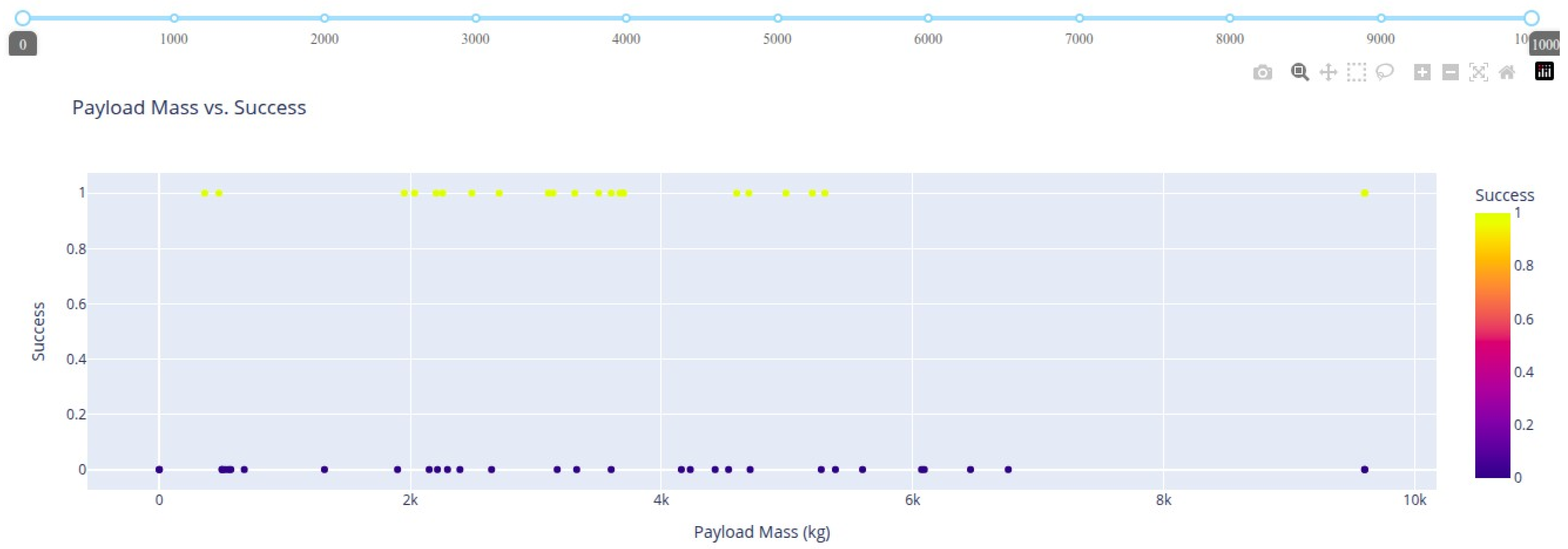
SpaceX Launch Records Dashboard

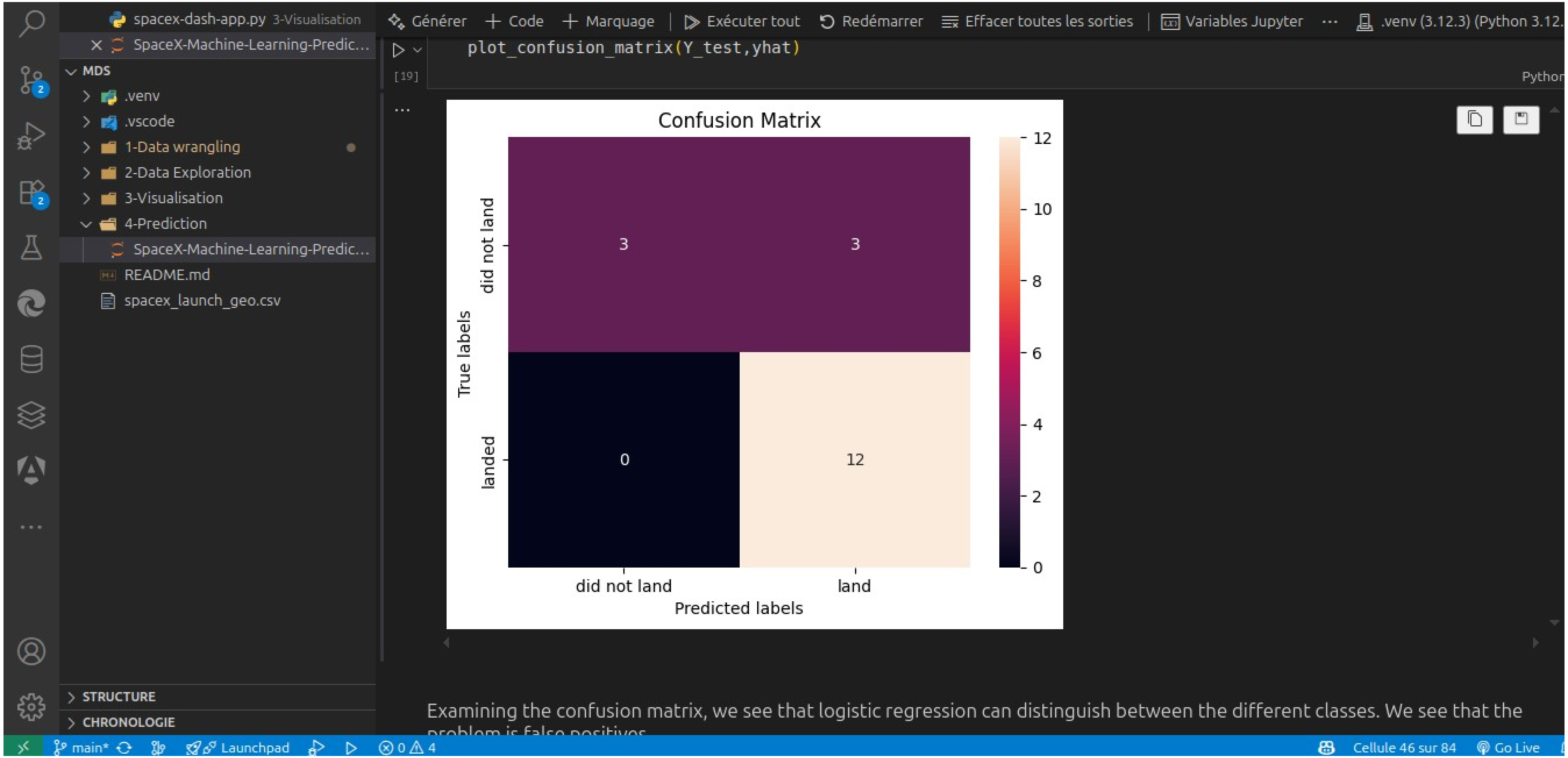
All Sites



Payload range (Kg):

Payload range (Kg):




spacex-dash-app.py 3-Visualisation
SpaceX-Machine-Learning-Predic...
MDS
.venv
.vscode
1-Data wrangling
2-Data Exploration
3-Visualisation
4-Prediction
SpaceX-Machine-Learning-Predic...
README.md
spacex_launch_geo.csv
STRUCTURE
CHRONOLOGIE
Générer
Code
Marquage
Exécuter tout
Redémarrer
Effacer toutes les sorties
Variables Jupyter
.venv (3.12.3) (Python 3.12.
plot_confusion_matrix(Y_test,yhat)
[19]
Python
Confusion Matrix
True labels
did not land
landed
3
3
0
12
did not land
land
Predicted labels
12
10
8
6
4
2
0
Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the
main*
Launchpad
0
4
Cellule 46 sur 84
Go Live

```python
# Find the method performs best
print("Logistic Regression Test Accuracy: ", logreg_acc)
print("SVM Test Accuracy: ", svm_acc)
print("Decision Tree Test Accuracy: ", tree_acc)
print("KNN Test Accuracy: ", knn_acc)
```

[37] Python

```
Logistic Regression Test Accuracy:  0.8333333333333334
SVM Test Accuracy:  0.8333333333333334
Decision Tree Test Accuracy:  0.7777777777777778
KNN Test Accuracy:  0.8333333333333334
```

Conclusion

This work was the pipeline of a data science project

Thank you