IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Chiara Castelli
29 june 2025

Outline

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

Executive Summary

- Throughout the project, I applied fundamental data analytics and data science tools aimed at discovering meaningful insights.

- I started by enabling the possibility to work with good-quality data, demonstrated through clear visualization techniques, and finally built a predictive model using machine learning.

- I will present the results from:

- Exploratory Data Analysis
- Interactive Analytics Dashboard
- Predictive Classification Models

Introduction

- Space launch sites play a critical role in enabling access to space for scientific, commercial, and defense purposes. Selecting and managing these sites requires careful consideration of **geographic, logistical, and safety factors**.

- This project aims at visualizing **SpaceX launch sites** and their launch outcomes to better understand the geographic and operational context of space missions. By mapping launch locations, distinguishing successful and failed launches, and calculating distances to nearby infrastructures like railways, highways, coastlines, and cities, the project provides valuable insights into site accessibility, safety, and logistical advantages.
- The project also includes a **classification model** for predictive analysis.

Introduction

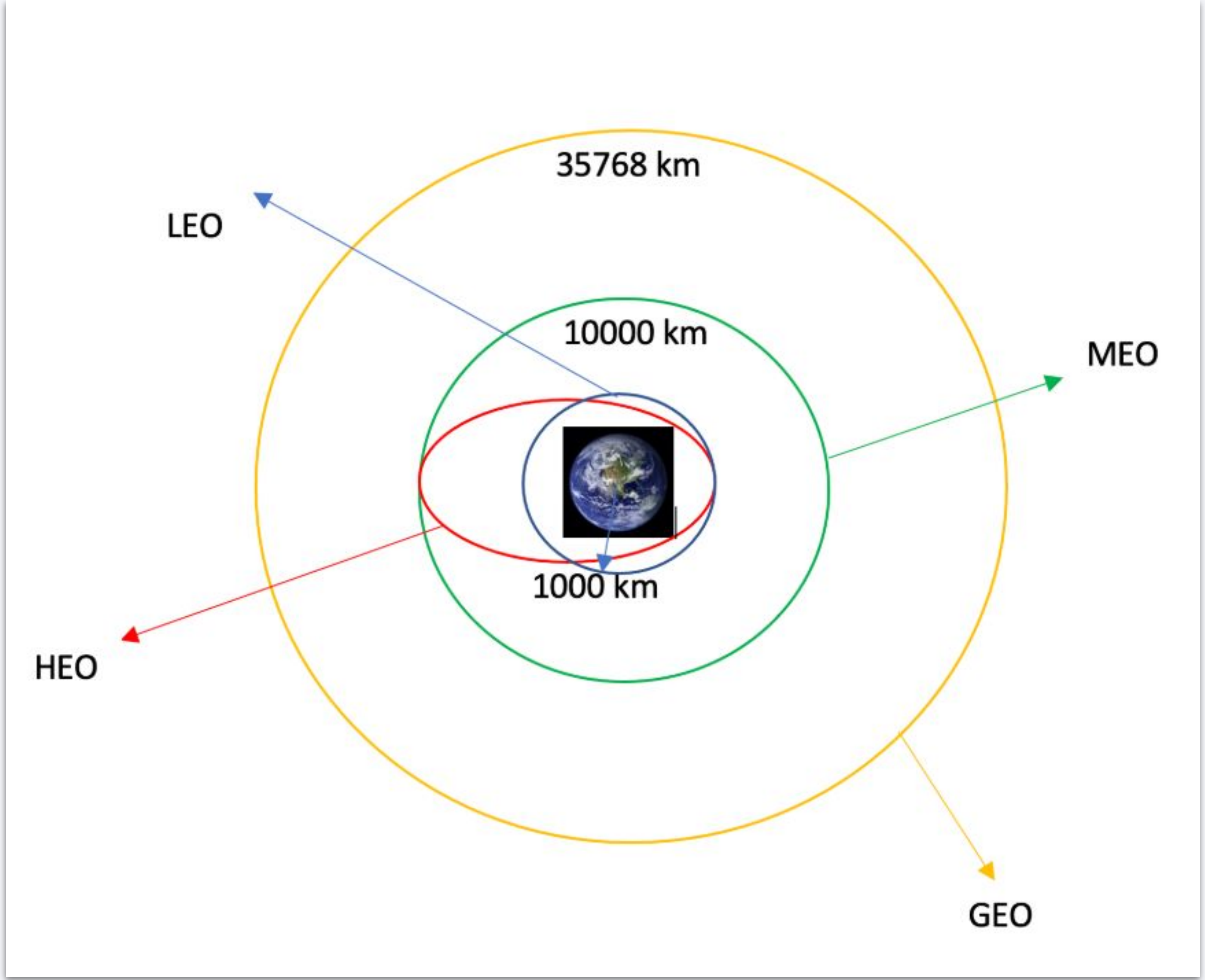
Each launch aims to a dedicated orbit.

Here are some common orbit types.

- **LEO**: Low Earth orbit (LEO)is an Earth-centred orbit with an altitude of 2,000 km (1,200 mi) or less (approximately one-third of the radius of Earth),[1] or with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity less than 0.25.[2] Most of the manmade objects in outer space are in LEO [1].
- **VLEO**: Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km. Operating in these orbits can provide a number of benefits to Earth observation spacecraft as the spacecraft operates closer to the observation[2].
- **GTO** A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. Located at 22,236 miles (35,786 kilometers) above Earth's equator, this position is a valuable spot for monitoring weather, communications and surveillance. Because the satellite orbits at the same speed that the Earth is turning, the satellite seems to stay in place over a single longitude, though it may drift north to south," NASA wrote on its Earth Observatory website [3] .
- **SSO (or SO)**: It is a Sun-synchronous orbit also called a heliosynchronous orbit is a nearly polar orbit around a planet, in which the satellite passes over any given point of the planet's surface at the same local mean solar time [4] .
- **ES-L1** :At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. L1 is one such point between the sun and the earth [5] .
- **HEO** A highly elliptical orbit, is an elliptic orbit with high eccentricity, usually referring to one around Earth [6].
- **ISS** A modular space station (habitable artificial satellite) in low Earth orbit. It is a multinational collaborative project between five participating space agencies: NASA (United States), Roscosmos (Russia), JAXA (Japan), ESA (Europe), and CSA (Canada) [7]
- **MEO** Geocentric orbits ranging in altitude from 2,000 km (1,200 mi) to just below geosynchronous orbit at 35,786 kilometers (22,236 mi). Also known as an intermediate circular orbit. These are "most commonly at 20,200 kilometers (12,600 mi), or 20,650 kilometers (12,830 mi), with an orbital period of 12 hours [8]
- **HEO** Geocentric orbits above the altitude of geosynchronous orbit (35,786 km or 22,236 mi) [9]
- **GEO** It is a circular geosynchronous orbit 35,786 kilometres (22,236 miles) above Earth's equator and following the direction of Earth's rotation [10]
- **PO** It is one type of satellites in which a satellite passes above or nearly above both poles of the body being orbited (usually a planet such as the Earth [11]

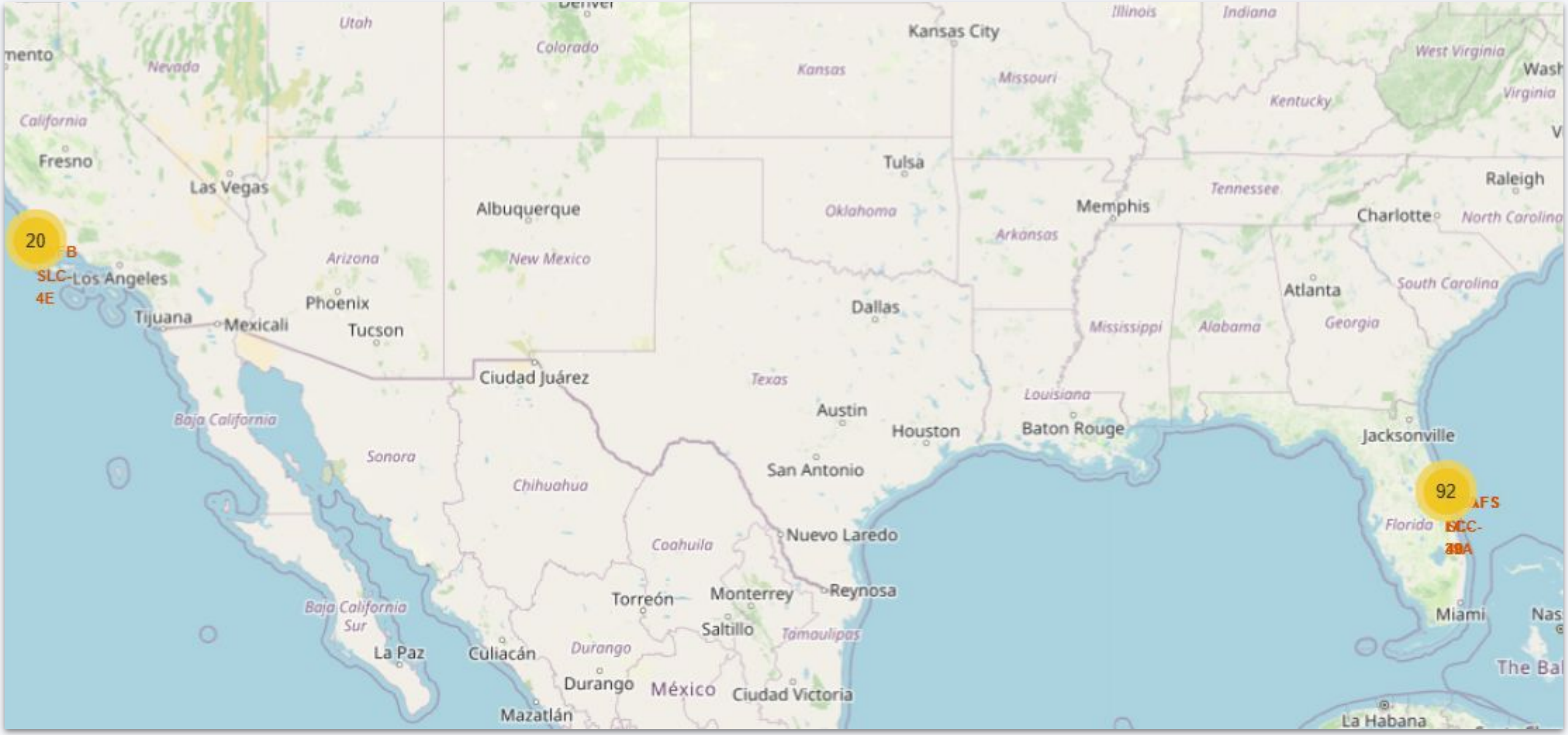
Introduction

Some of the orbits are shown here:



Introduction

Launch sites are primarily located in North America near the Equator Line.



Section 1

Methodology

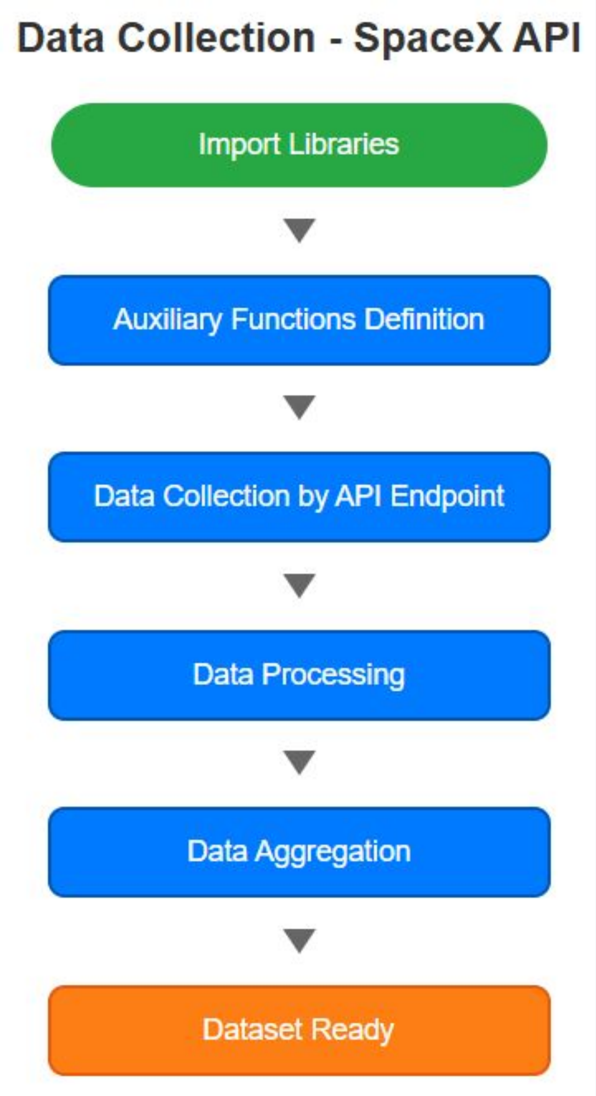
Methodology

Executive Summary

- Data collection methodology:
  - Data was collected from SpaceX API
- Perform data wrangling
  - Data was processed preparing it for analysis and modeling
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - I used Logistic Regression, Decision Tree, SVM and KNN to build different machine learning models and I evaluated them to find the best performer

Data Collection – SpaceX API

- I collected SpaceX Falcon 9 launch data by making **API requests** and loading it into a pandas DataFrame.
- I then defined auxiliary functions to extract booster version, launch site, payload, and landing outcome details.
- After filtering and cleaning, I enriched the DataFrame with these extracted features.
- Finally, I combined everything into a new DataFrame, previewed the data, and prepared it for further analysis.



GitHub [Data Collection SpaceX API]

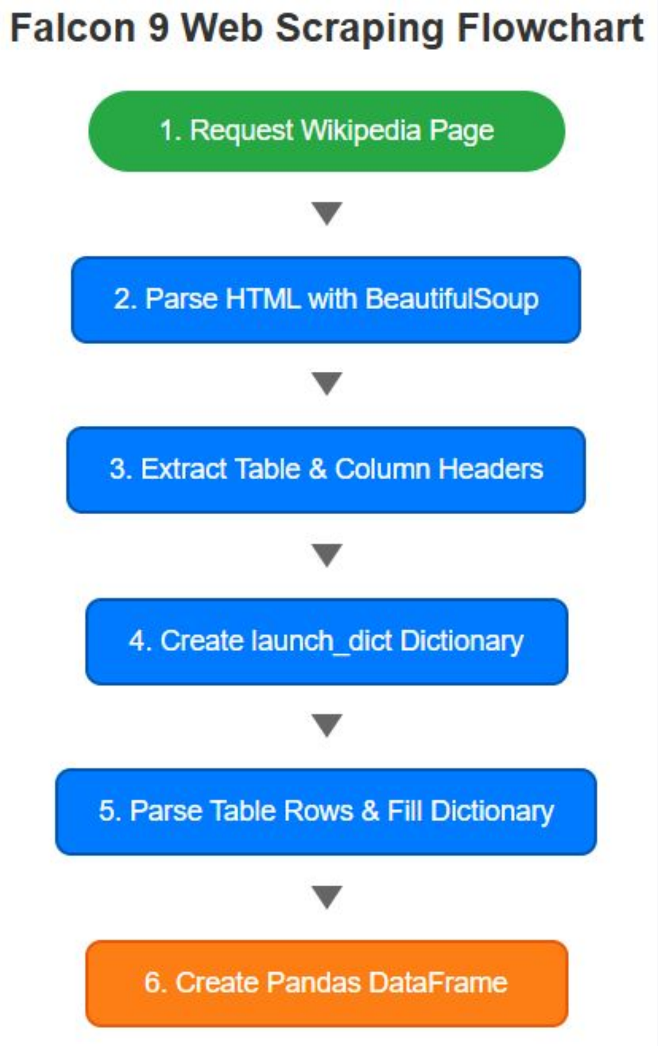
Data Collection - Scraping

- I processed the SpaceX Falcon 9 launch data by scraping, parsing, and loading it, using requests, BeautifulSoup4, pandas, and HTML parsing.

- Then, I extracted the launch records from a Wikipedia HTML table using BeautifulSoup, defined helper functions to process the web-scraped table, and converted the results into a pandas DataFrame.

- Then I set up the environment to use a static Wikipedia snapshot URL for consistency, and validated the successful retrieval of the page and the correct parsing of its title.

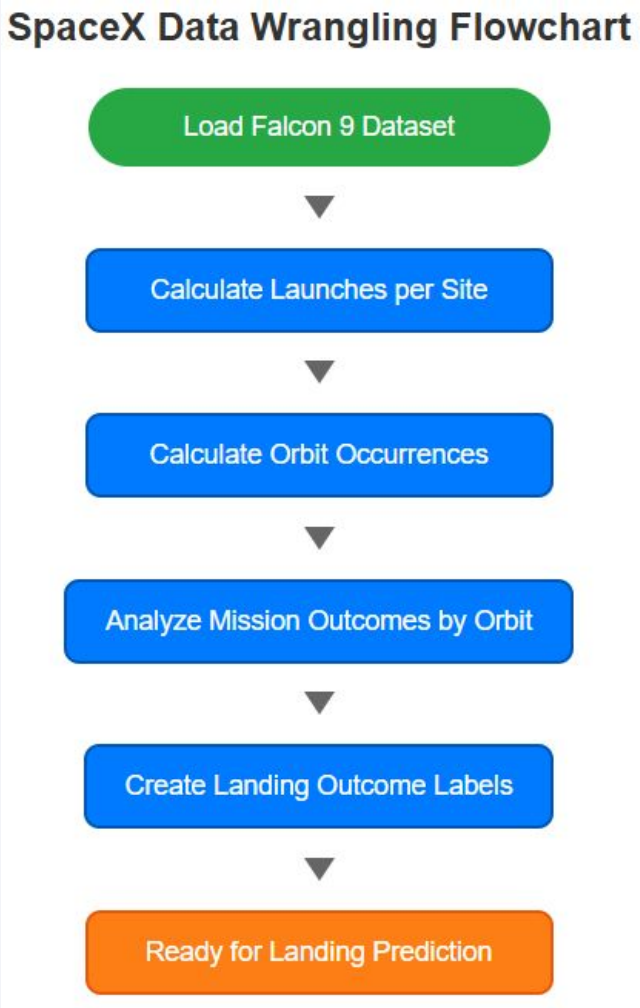
- Finally, I began extracting column names and prepared the table data for further analysis and machine learning tasks.



GitHub Webscraping

Data Wrangling

- I processed the SpaceX launch data by **loading, inspecting, and cleaning it**.
- Then, I analyzed launches by site and orbit, **categorized mission outcomes**.
- Then I **generated a binary classification label** for each row to indicate landing success, preparing the data for **machine learning classification**.



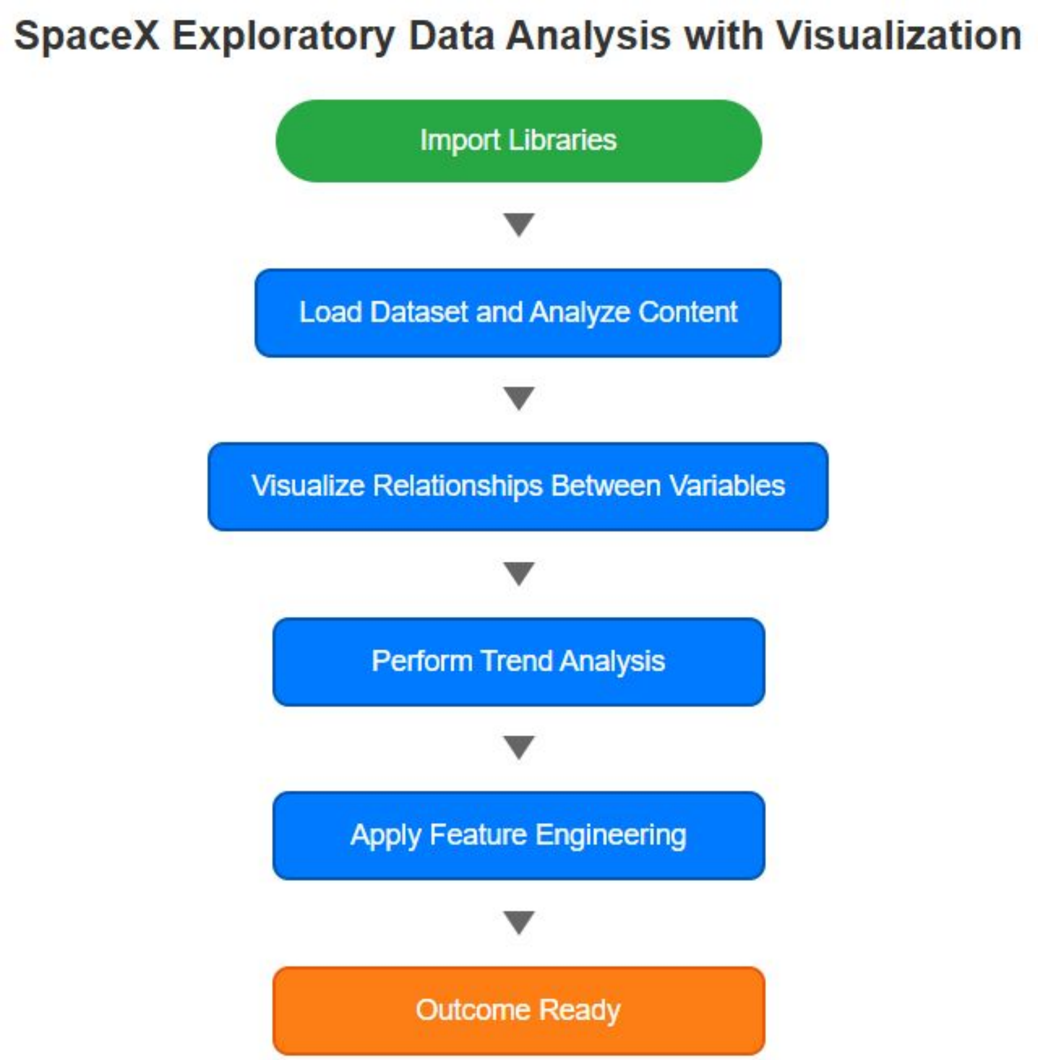
GitHub [Data Wrangling]

EDA with Data Visualization

- I created multiple **plots** to explore the data:
- Scatter plots for FlightNumber vs. LaunchSite.
- Relationships between PayloadMass, LaunchSite, and success rates.
- Bar charts to understand success rates for different orbit types.

- Then I performed trend analysis to capture **Launch Success Yearly Trend**.

- Later I applied **Feature Engineering** to extract meaningful variables from the dataset.

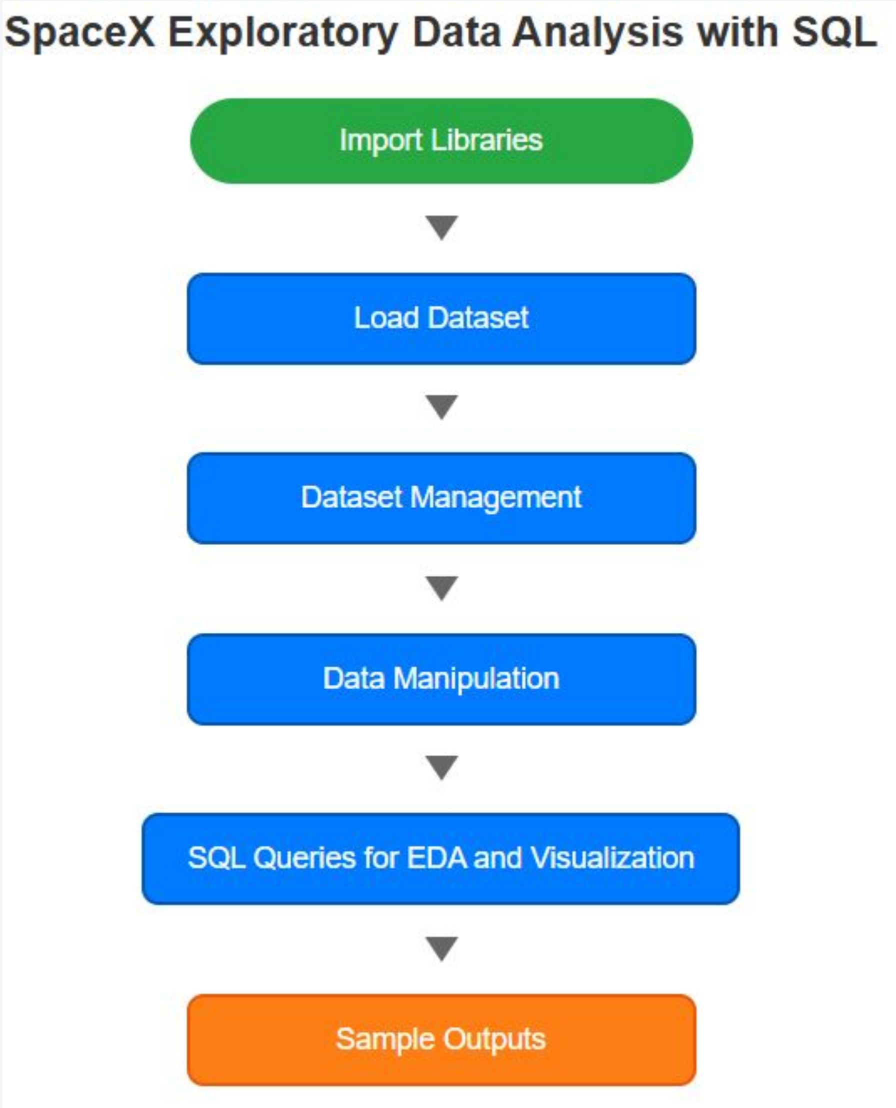


GitHub [EDA with Data Visualization]

EDA with SQL

For the Exploratory Data Analysis I used SQL to query the RDB and:

- Listed all unique launch sites in the dataset.
- Displayed records for launch sites starting with “CCA”.
- Calculated the total payload mass for missions launched by NASA (CRS).
- Found the average payload mass for rockets with booster version “F9 v1.1”.
- Identified the date of the first successful landing on a ground pad.
- Listed booster versions that had successful drone ship landings and carried payloads between 4000 and 6000 kg.
- Counted the total number of successful and failed missions.
- Used a subquery to find the booster versions that carried the maximum payload mass.

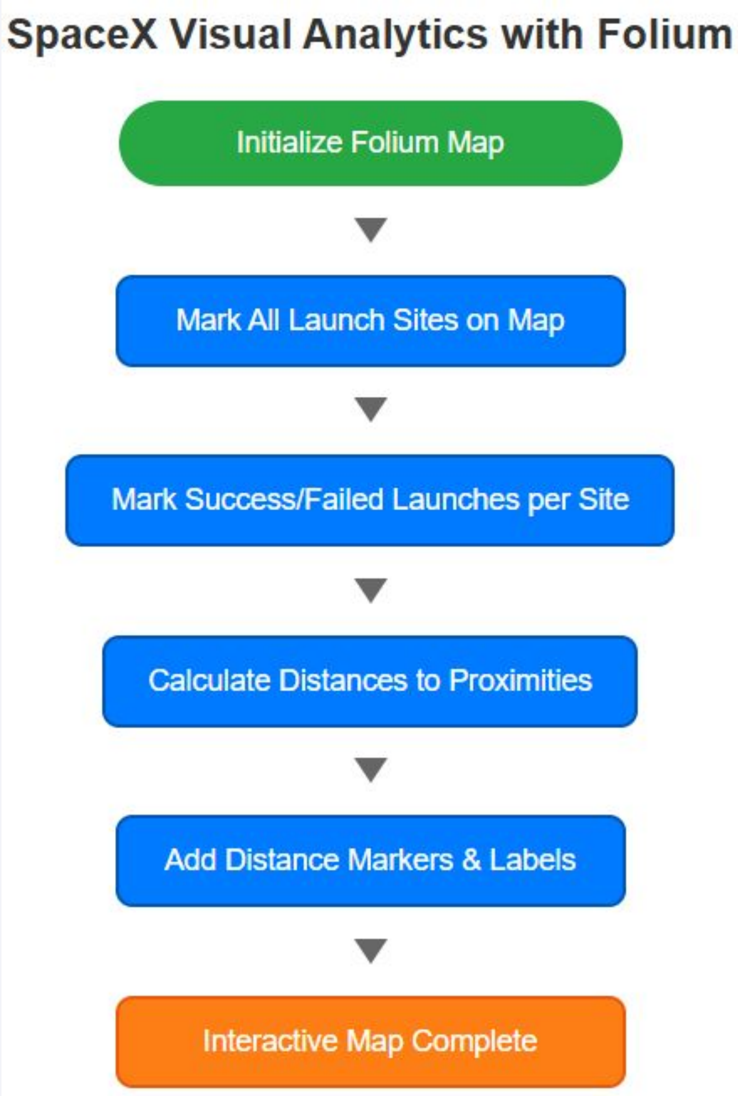


GitHub [EDA with SQL]

Build an Interactive Map with Folium

- For the analysis I built an interactive map to help in exploring SpaceX launch data.

- This **interactive map** easily allows to visualize mission success rates and launch sites correlations, while also allowing to explore the surrounding area and create map objects such as **markers**, **circles** and **lines** to highlight analyzed distances and get powerful insights.

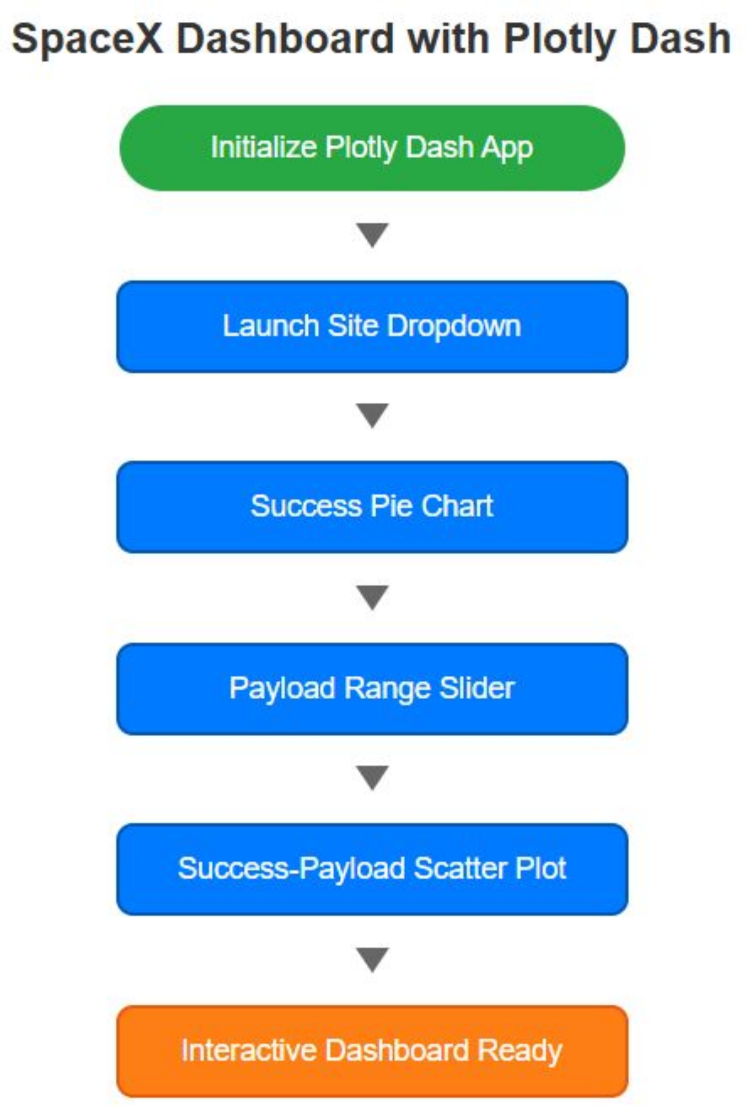


GitHub Folium

Build a Dashboard with Plotly Dash

- I built a dashboard that includes:
- **Launch Site Dropdown**: Allows users to select either all launch sites or a specific site for focused analysis.
- **Success Pie Chart**: Displays total successful launches for all sites or success vs failure breakdown for a selected site, providing a clear overview of mission outcomes.
- **Payload Range Slider**: Enables filtering launches based on payload mass to analyze its effect on success rates.
- **Success-Payload Scatter Plot**: Shows the relationship between payload mass and launch success, color-coded by booster version, helping to identify patterns and correlations.
- These visualizations and controls were added to enable interactive exploration of SpaceX launch data, allowing to filter and analyze key factors influencing mission success intuitively and effectively.

GitHub [SpaceX-Dashboard]()



Predictive Analysis (Classification)

I built different Classification Models for Predictive Analysis:

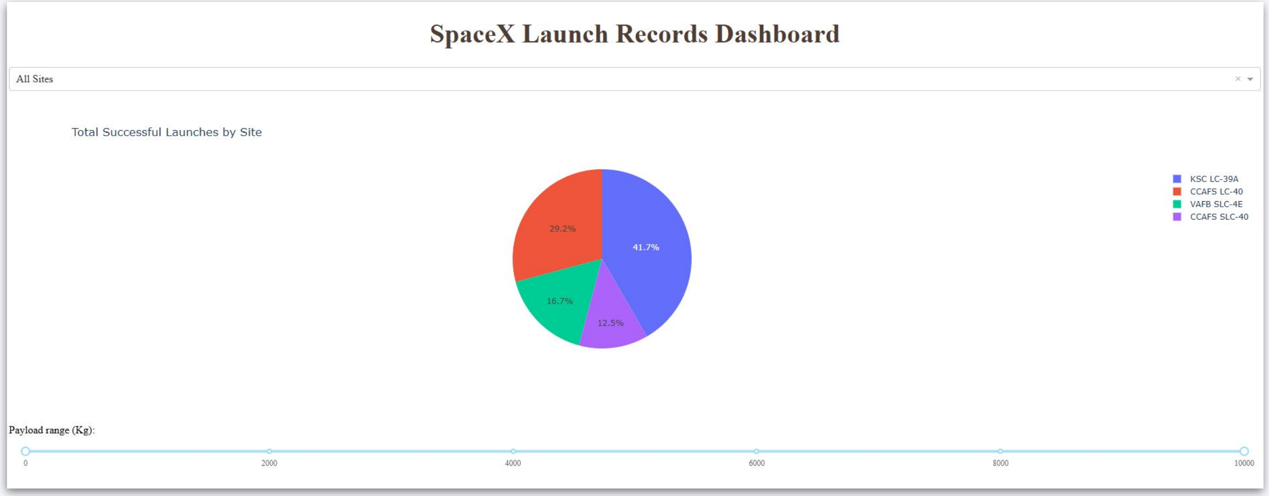
- Data Preparation: Cleaned and preprocessed the dataset for modeling
- Train-Test-Split for Data Splitting
- Model Building and Evaluation: Tested multiple classifiers (e.g., **Logistic Regression, Decision Tree, SVM, KNN**)
- Selection: Identified the best performing model based on evaluation results

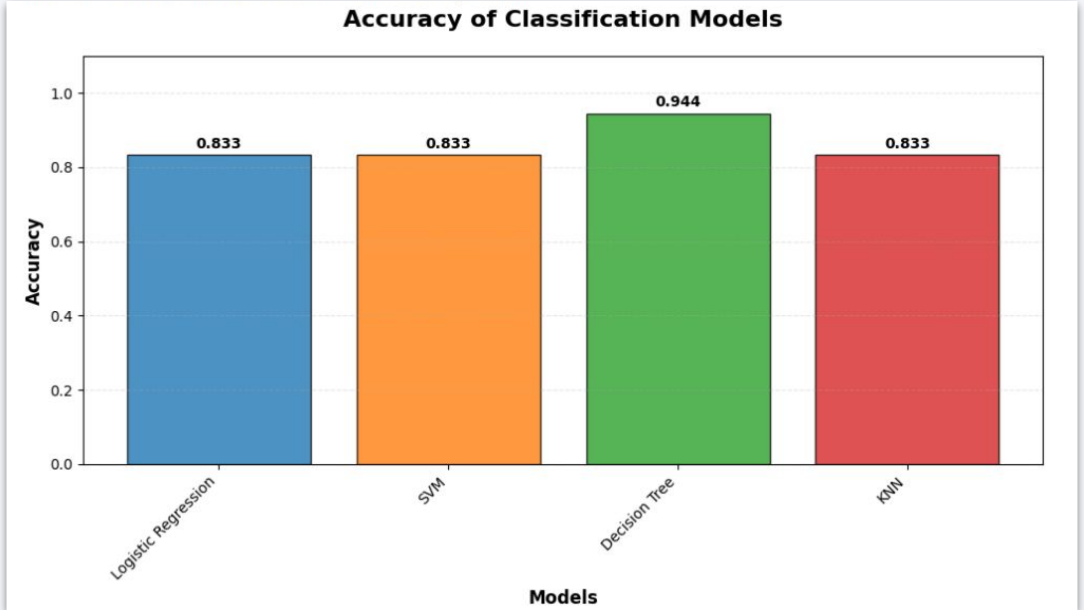
GitHub [Machine Learning Prediction]

Results

- I will present the results from:

- Exploratory Data Analysis
- Interactive Analytics Dashboard
- Predictive Classification Models





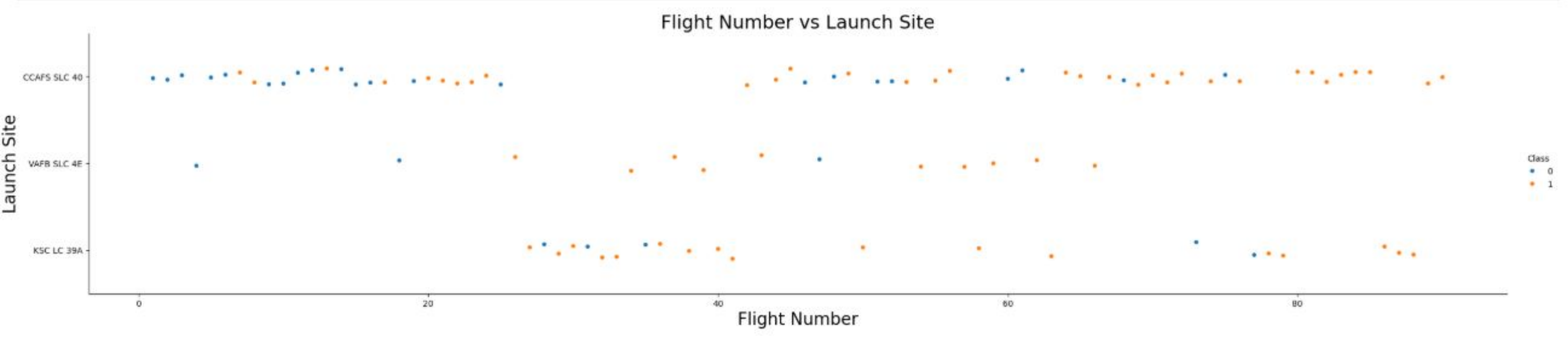
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

First, we want to observe the Flight Numbers and their corresponding Launch Site.

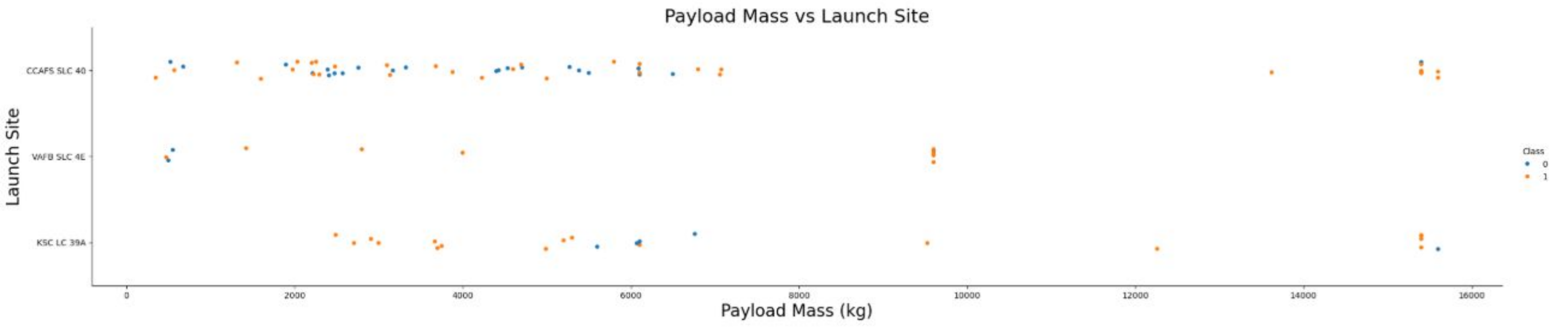
```
sns.catplot(x="FlightNumber", y="LaunchSite", hue="Class", data=df, aspect=5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.title("Flight Number vs Launch Site", fontsize=22)
plt.show()
```



Payload vs. Launch Site

We also want to observe if there is any relationship between Launch Site and their Payload Mass.

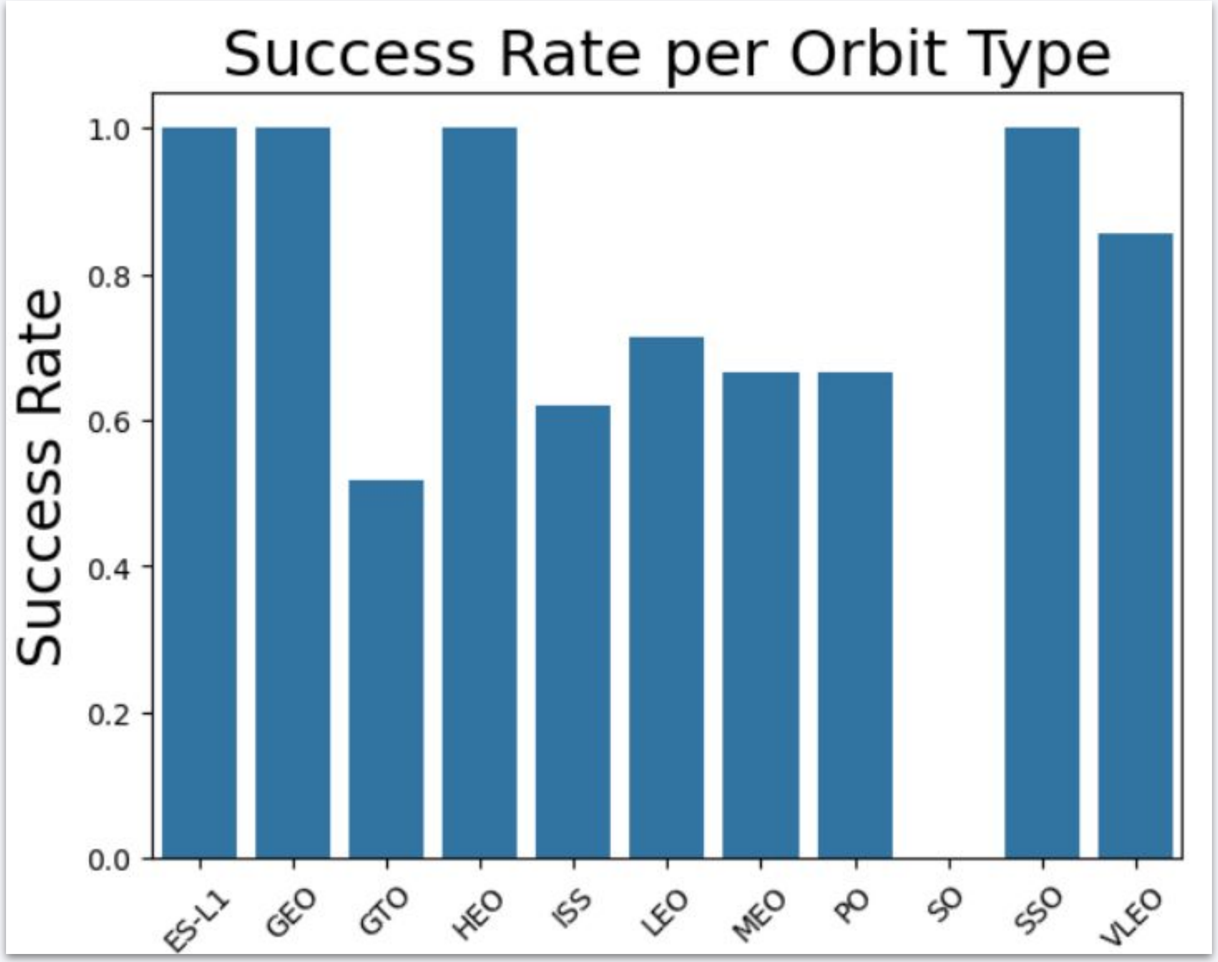
```
sns.catplot(x="PayloadMass", y="LaunchSite", hue="Class", data=df, aspect=5)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.title("Payload Mass vs Launch Site", fontsize=22)
plt.show()
```



Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

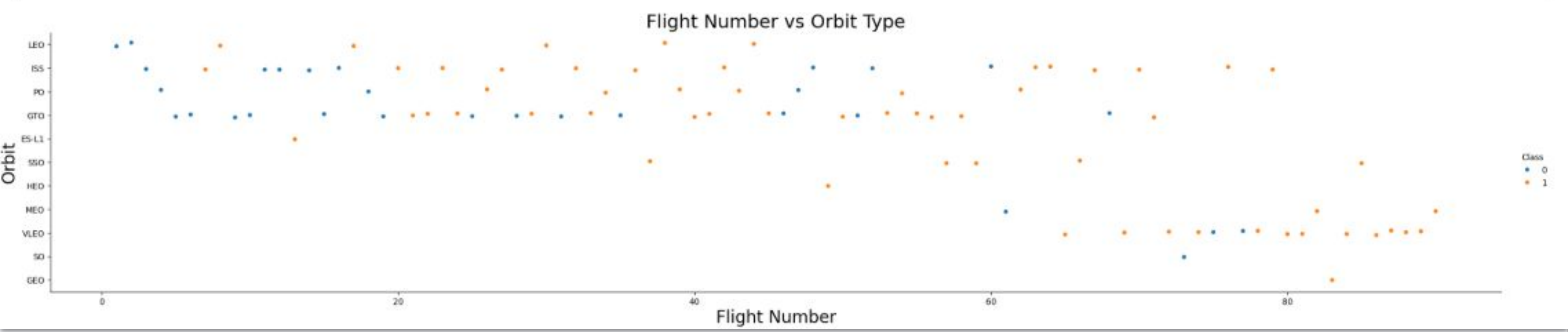
- We then plot a bar chart for the success rate of each Orbit type
- Here are presented the results:



Flight Number vs. Orbit Type

For each orbit, we want to see if there is any relationship between Flight Number and Orbit type.

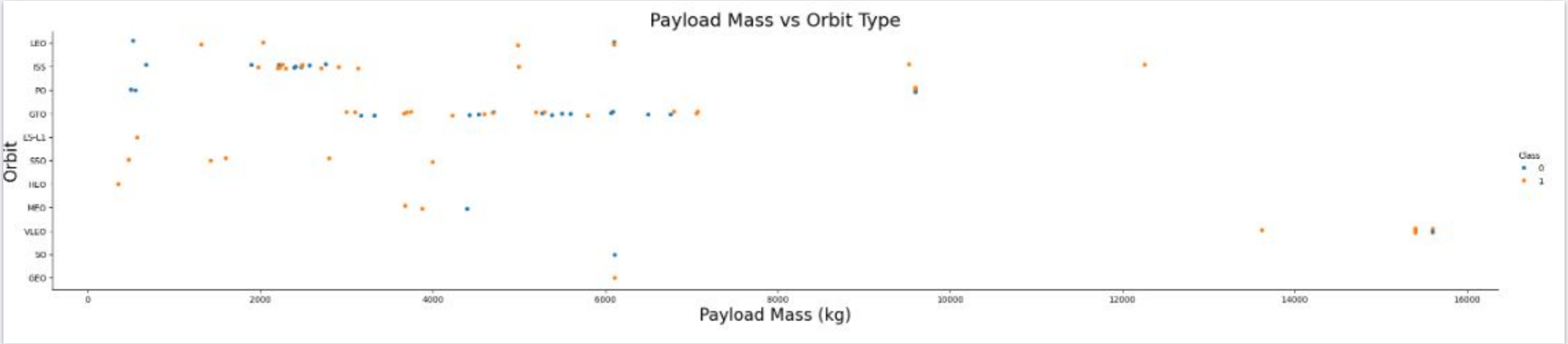
```
sns.catplot(x="FlightNumber", y="Orbit", hue="Class", data=df, aspect=5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Orbit", fontsize=20)
plt.title("Flight Number vs Orbit Type", fontsize=22)
plt.show()
```



Payload vs. Orbit Type

Similarly, we can plot the Payload Mass vs Orbit scatter point charts to reveal the relationship between Payload Mass and Orbit type.

```
sns.catplot(x="PayloadMass", y="Orbit", hue="Class", data=df, aspect=5)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Orbit", fontsize=20)
plt.title("Payload Mass vs Orbit Type", fontsize=22)
plt.show()
```



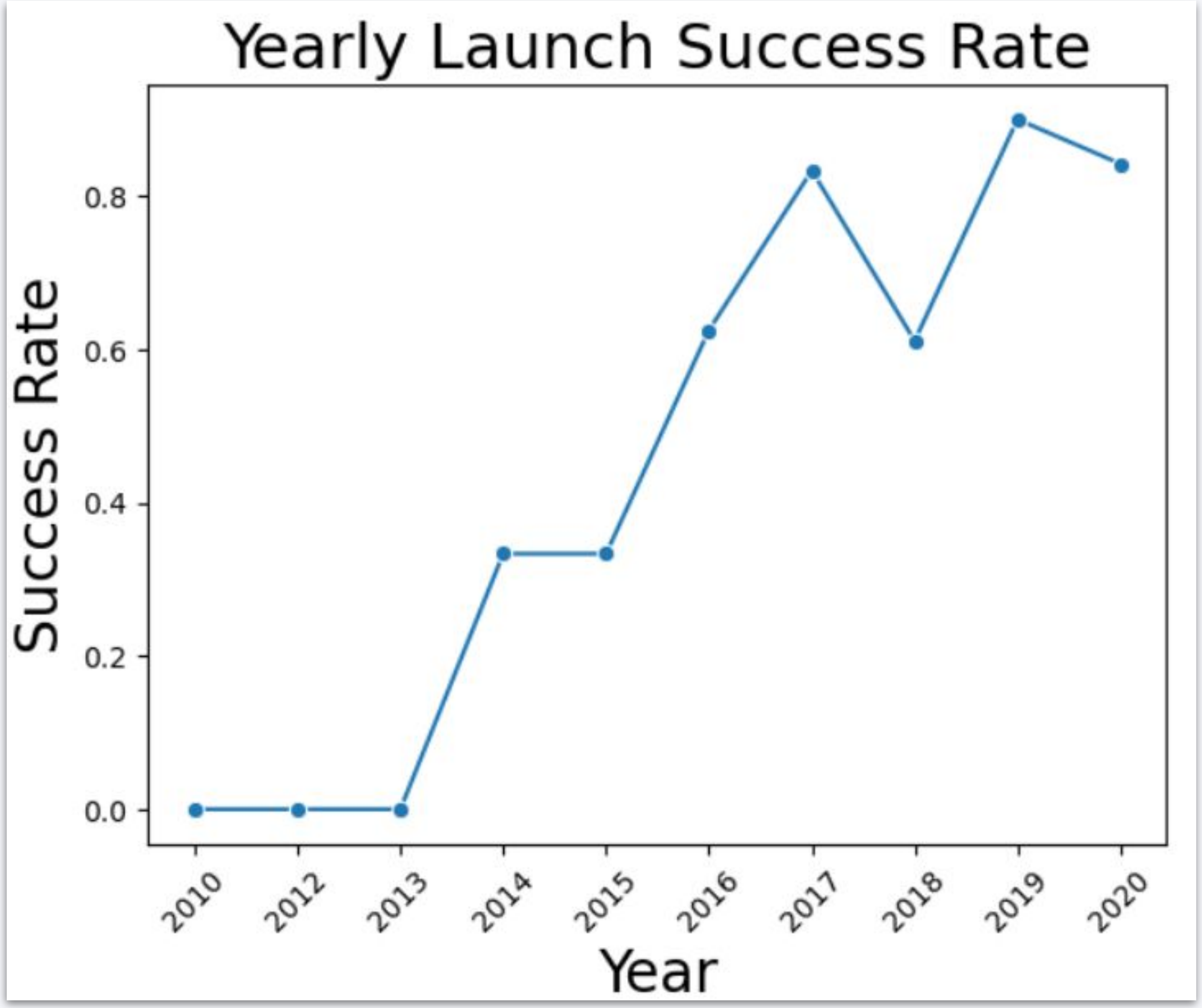
With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend

- We plot a line chart of yearly average success rate

- In the figure we can better visualize the launch success yearly trend and observe that the success rate kept increasing since 2013 until 2020.



All Launch Site Names

- With this SQL query we can find the names of the unique launch sites

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

- As results we find these 4 unique launch site:

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

Launch Site Names Begin with 'CCA'

- With this SQL query we can find 5 records where launch sites begin with `CCA`.

- We find these 5 records:

```
%%sql
SELECT *
FROM SPACEXTABLE
WHERE LOWER("Launch_Site") LIKE 'cca%'
LIMIT 5;
```

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

- We are interested in calculating the total payload carried by boosters from NASA.
- We execute this query to find out the Total Payload Mass.

```
%%sql
SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass
FROM SPACEXTABLE
WHERE Customer = 'NASA (CRS)';
```

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

- We then want to calculate the average payload mass carried by booster version F9 v1.1

- So we execute this query to find out the Average Payload Mass by booster version F9 v1.1

```
%%sql
SELECT AVG("PAYLOAD_MASS__KG_") AS Avg_Payload
FROM SPACEXTABLE
WHERE "Booster_Version" = 'F9 v1.1';
```

Avg_Payload
2928.4

First Successful Ground Landing Date

- We are interested in finding the dates of the first successful landing outcome on ground pad.

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

- So we execute this query and find out that it was on the 22th December 2015.

First_Ground_Pad_Success
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Then it is interesting to take a look at the boosters which have successfully landed on drone ship and also had payload mass greater than 4000, but less than 6000.

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

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

Total Number of Successful and Failure Mission Outcomes

- Then we calculate the total number of successful and failure mission outcomes

- Results are presented here:

```
%%sql
SELECT MISSION_OUTCOME, COUNT(*) AS TOTAL
FROM SPACEXTABLE
GROUP BY MISSION_OUTCOME;
```

Mission_Outcome	TOTAL
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- We want to list the names of the booster which have carried the maximum payload mass
- We execute this query and obtain the following results:

```
%%sql
SELECT DISTINCT Booster_Version
FROM SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTABLE
);
```

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

- To conclude, we take an overview of the failed landing outcomes in drone ship, their booster versions, and launch site names for 2015.

- So, we query as follows and obtain the resulting list with 2 records:

```sql
%%sql
SELECT
    substr(Date, 6, 2) AS Month,
    Landing_Outcome,
    Booster_Version,
    Launch_Site
FROM SPACEXTABLE
WHERE Landing_Outcome = 'Failure (drone ship)'
  AND substr(Date, 0, 5) = '2015';
```

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

- Finally we 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

- Here is presented the ordered list:

```
%%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;
```

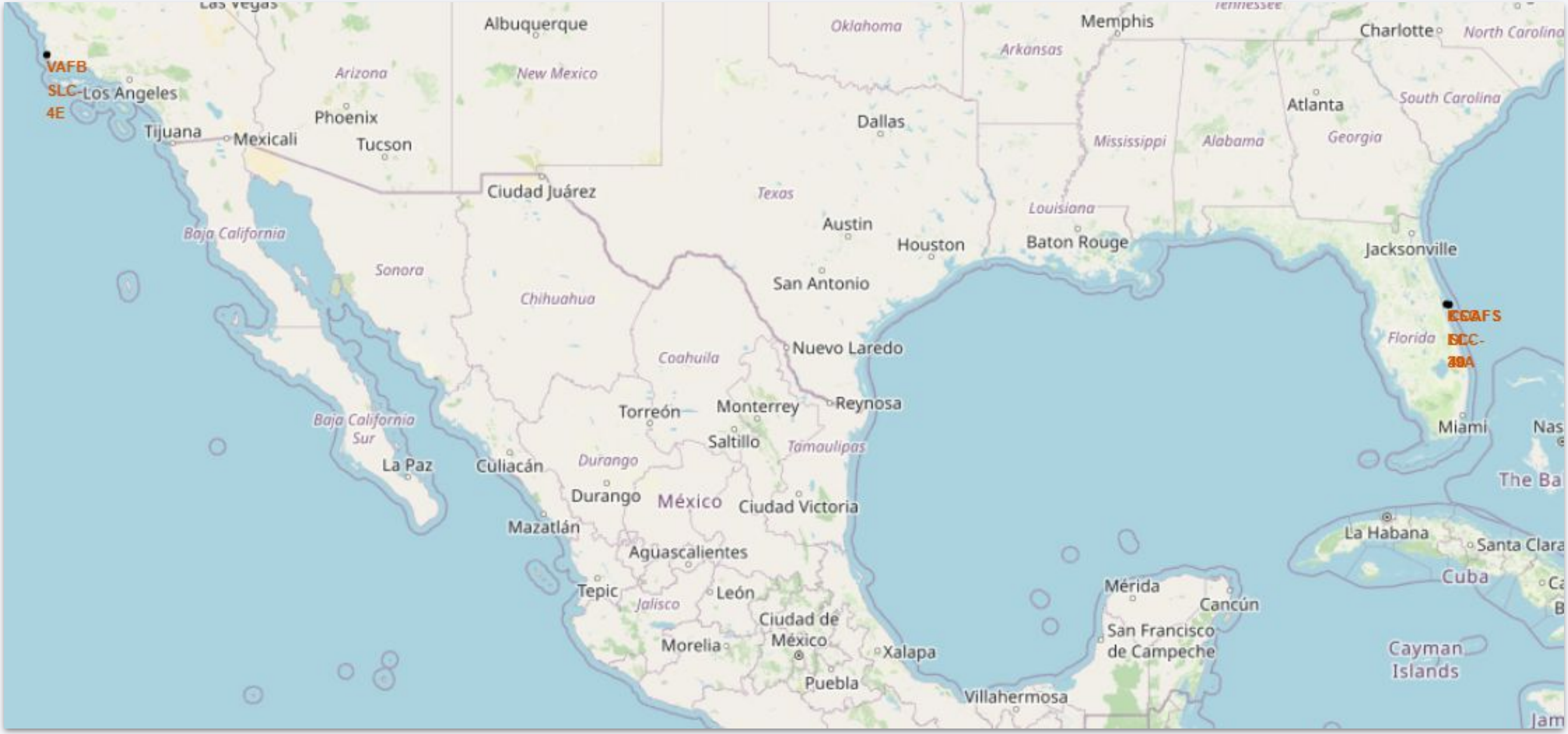
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

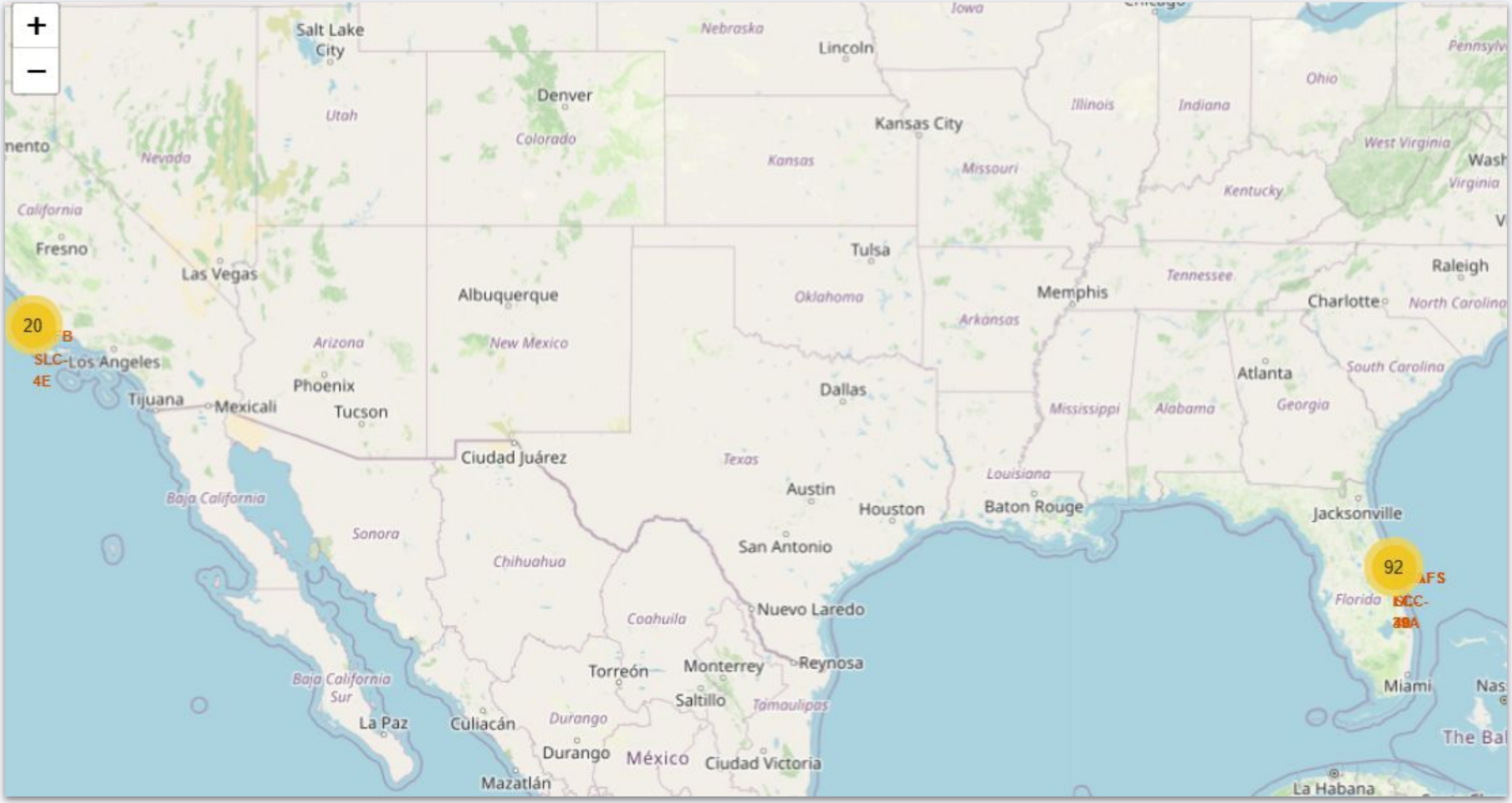
Launch Sites on Map with Folium

- **Here, I present all launch sites' location as markers on a global map.**



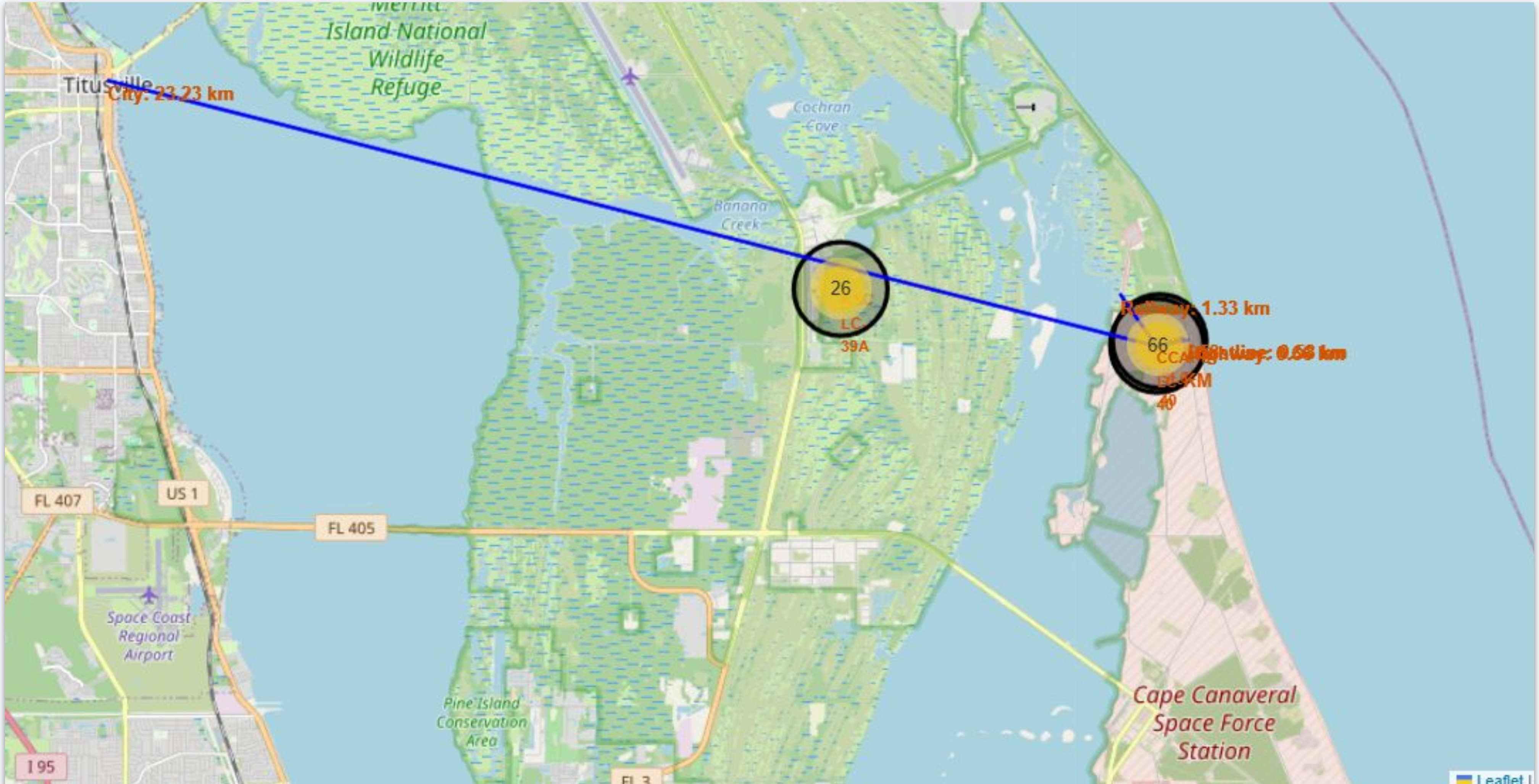
Launch Outcomes with Folium

- **I also marked the launch sites based on their successful or failed outcomes.**



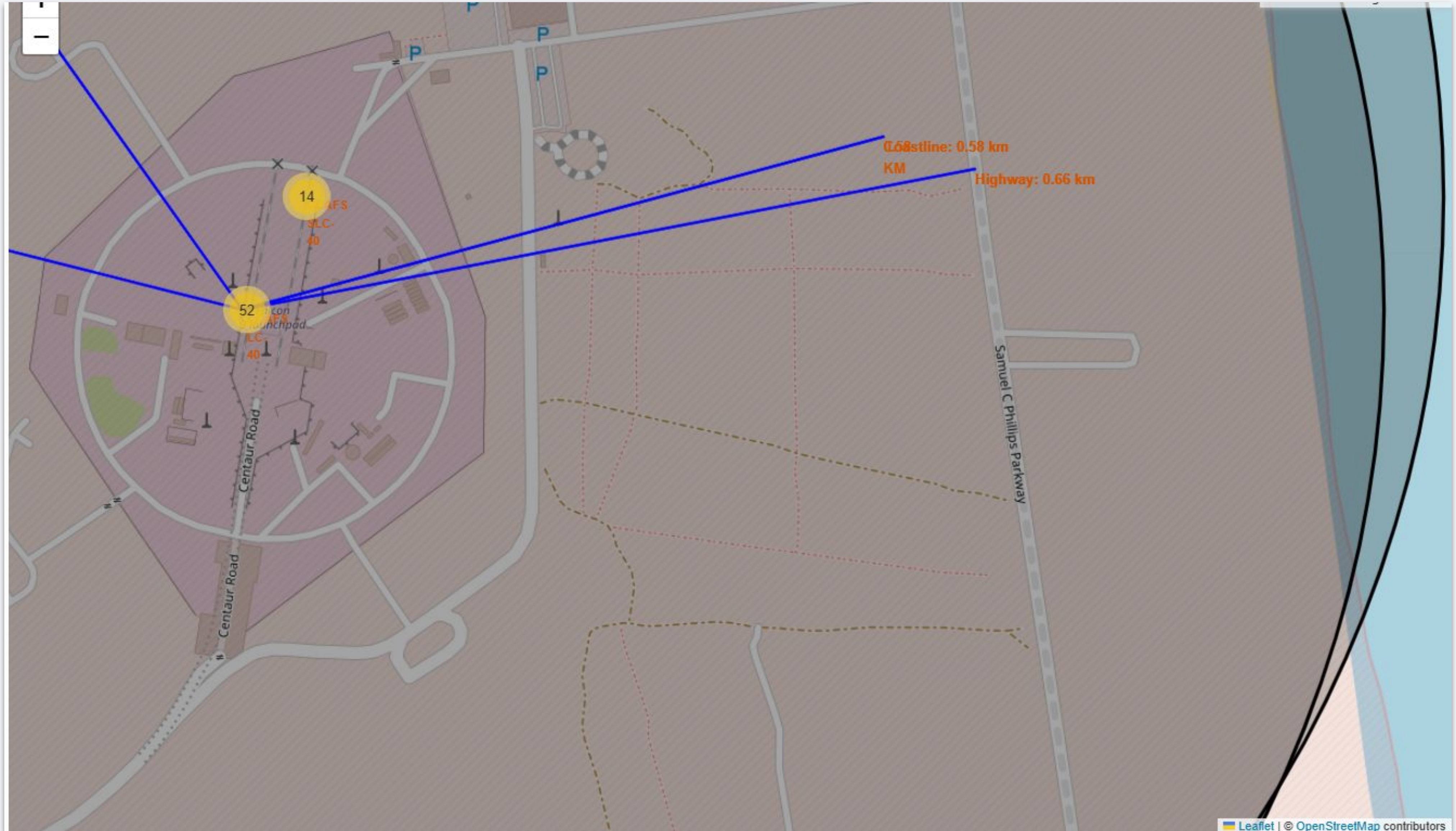
Launch Sites Proximities with Folium

- **And also checked the proximities of the launch sites by measuring their distance from railways and highways.**



Launch Sites Proximities with Folium

Launch Sites Proximities with Folium



Key Insights

- Are launch sites in close proximity to railways?

  The launch site is very close to a railway, which is useful for transporting heavy equipment and materials.

- Are launch sites in close proximity to highways?

  Close to a highway, providing high logistical accessibility for ground transportation.

- Are launch sites in close proximity to coastline?

  Very close to the coastline, which is ideal for safety reasons and allows launch trajectories over the ocean.

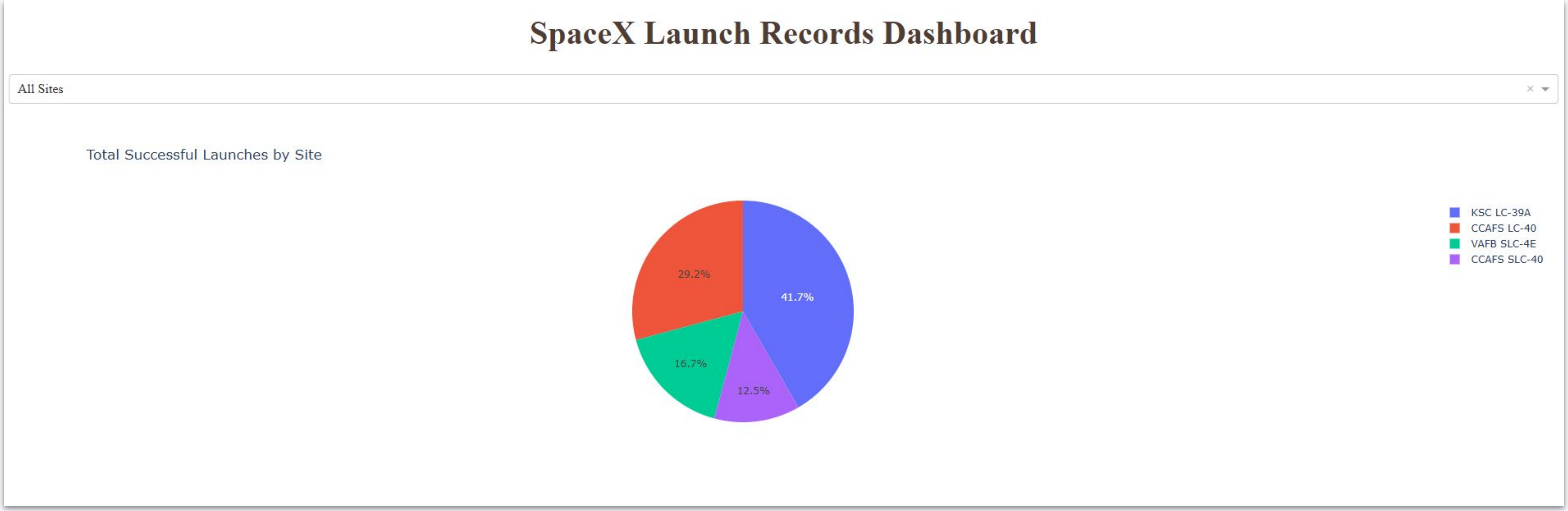
- Do launch sites keep certain distance away from cities?

  The site is far from urban areas, minimizing risks to the population in case of launch anomalies.

Section 4

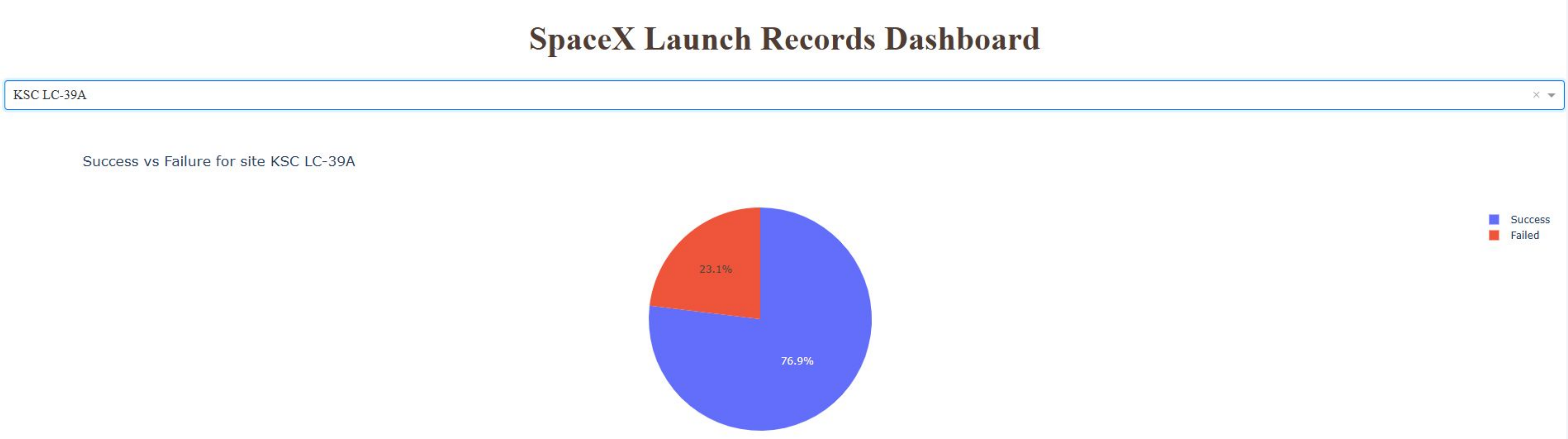
Build a Dashboard with Plotly Dash

Launch Success Distribution for All Sites



I created this dashboard to interactively observe Launch Success and corresponding Launch Sites.

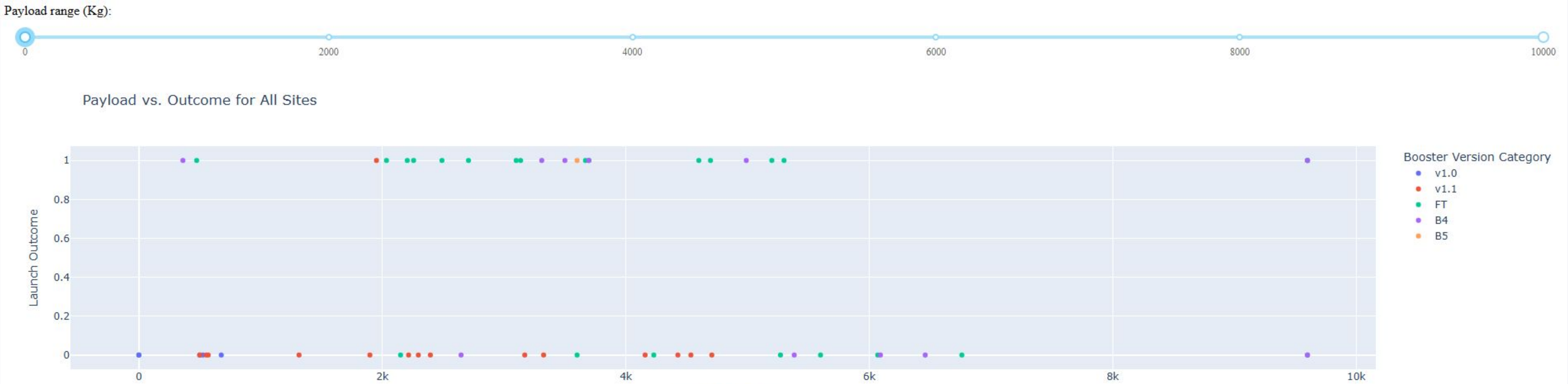
Success vs Failure Breakdown at Top Performing Launch Site



As we can see from the dashboard, KSC LC-39A was the best performing in terms of Launch Success.

Impact of Payload Mass on Launch Outcomes Across All Site

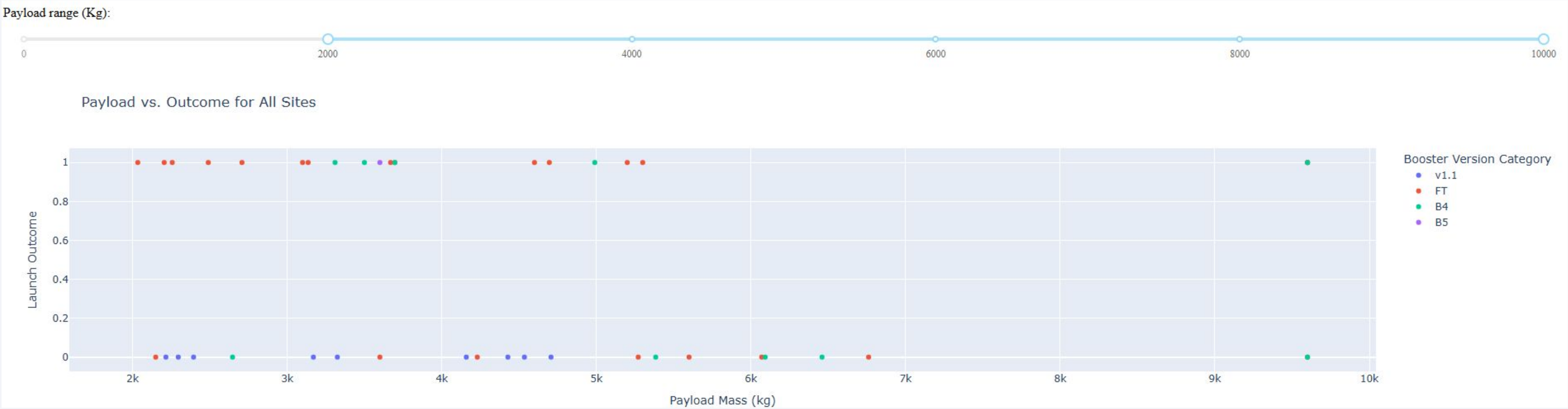
The dashboard can also be used to observe the impact of the Payload Mass on Launch Outcomes



Here we observe the scatter plot for Payload Mass from 0 Kg to 10000 Kg.

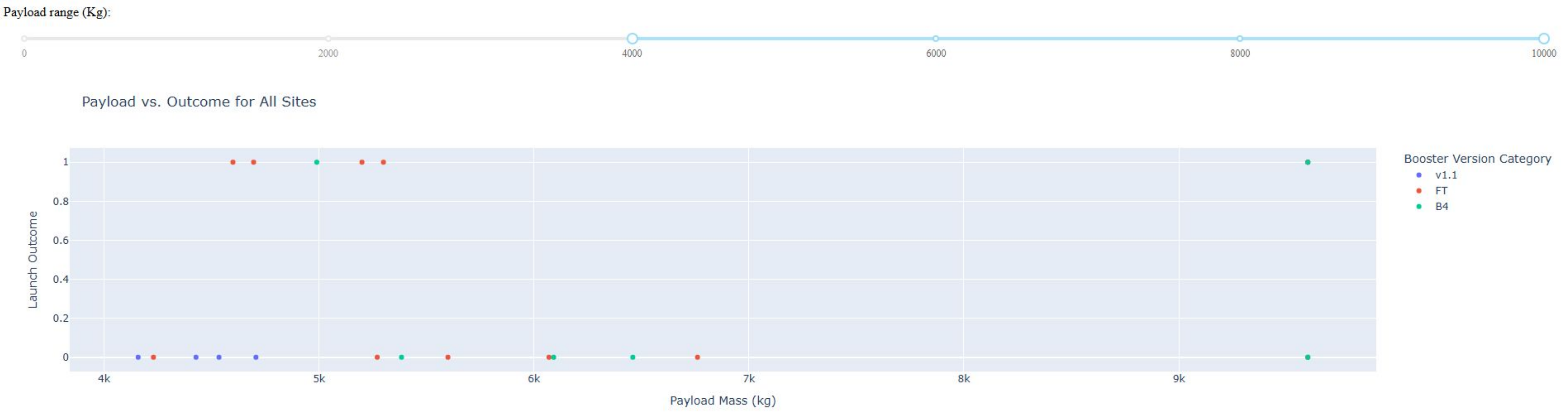
Impact of Payload Mass on Launch Outcomes Across All Site

In this dashboard it is possible to easily select various Payload Mass ranges using Range Slider feature I added.



Here in the scatter plot, we observe Payload vs. Outcome for all sites within the Payload Mass range of 2000 to 10000 Kg.

Impact of Payload Mass on Launch Outcomes Across All Site



In this scatter plot, we focus on the Payload vs. Outcome relationship for all sites within the Payload Mass range of 4000 to 10000 Kg.

Analyzing the data in this range highlights how higher payload masses influence the launch success rates.

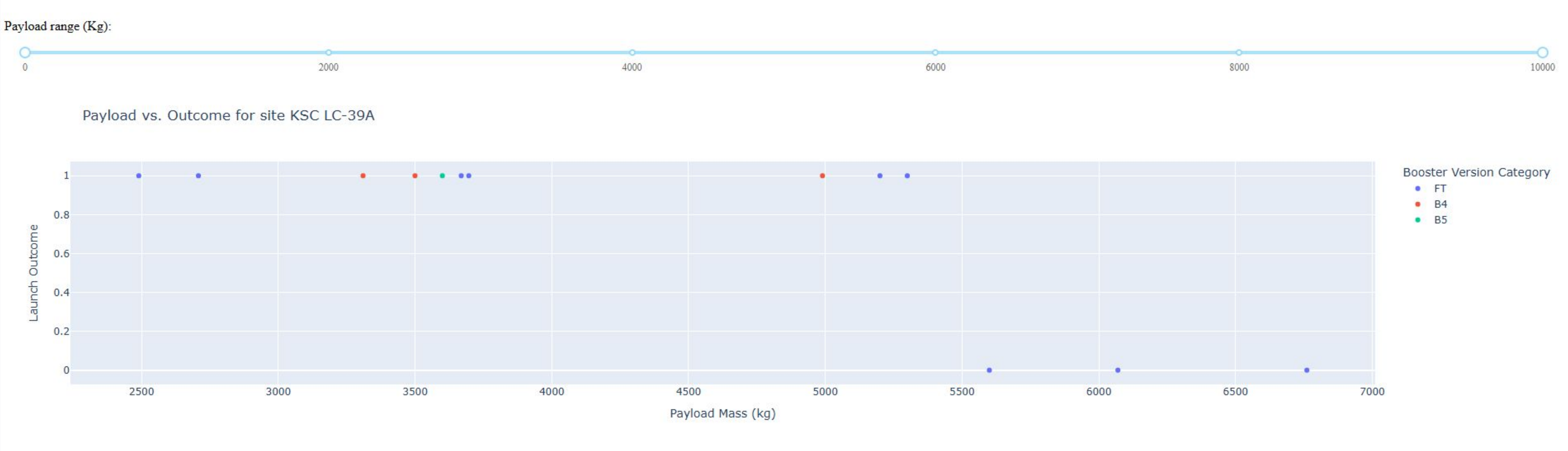
Impact of Payload Mass on Launch Outcomes Across All Site

As the payload mass increases beyond a certain point, we observe a noticeable drop in data points.

This indicates fewer launches with very heavy payloads in the dataset and none above 9000 Kg.

Impact of Payload Mass on Launch Outcomes for KSC LC-39A

Alternatively, the dashboard allows us to select a single launch site and observe the scatter plot for different payload mass ranges.



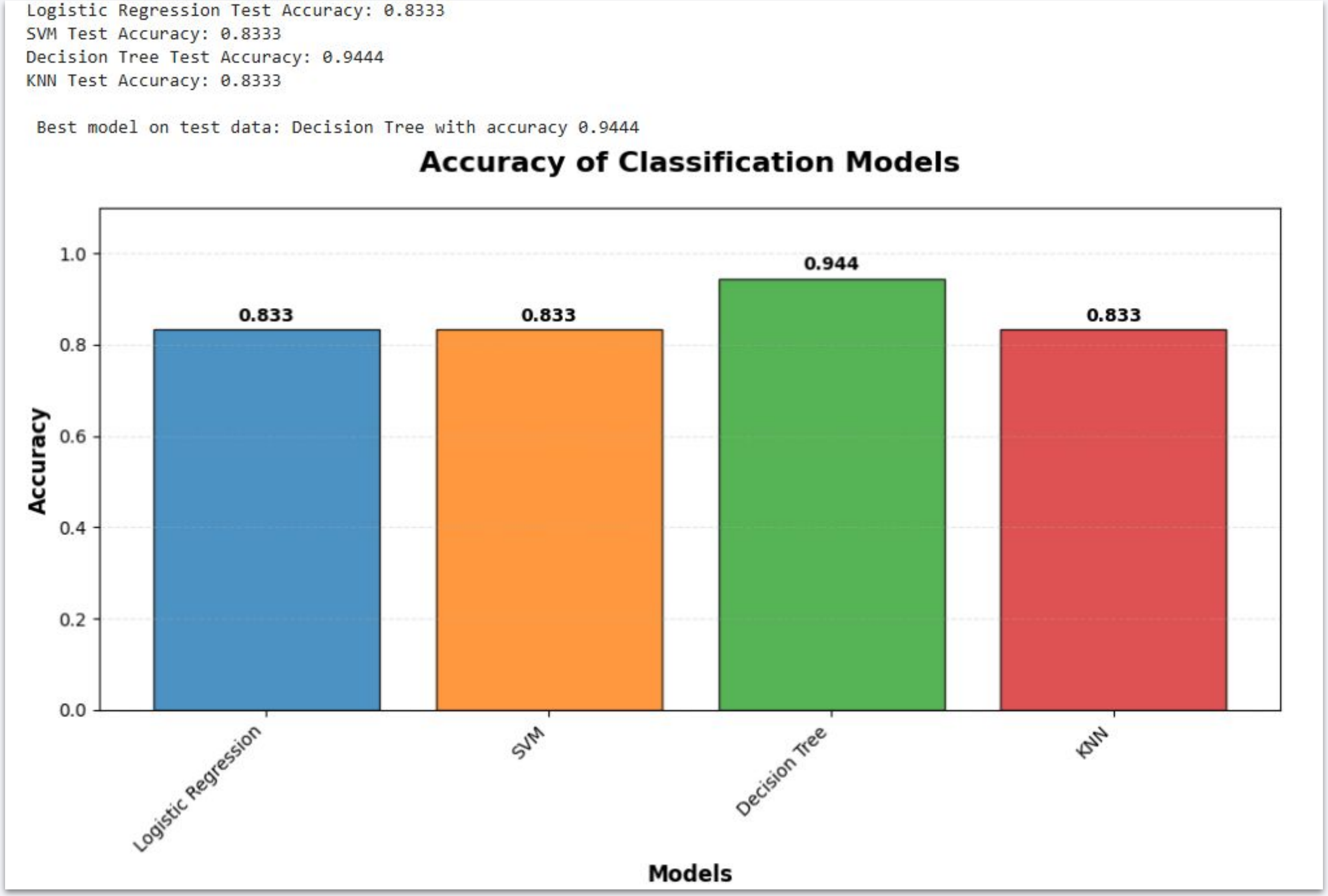
Here we see the scatter plot for KSC LC-39A, showing the relationship between payload mass and launch outcome across different Payload Mass ranges.

Section 5

Predictive Analysis (Classification)

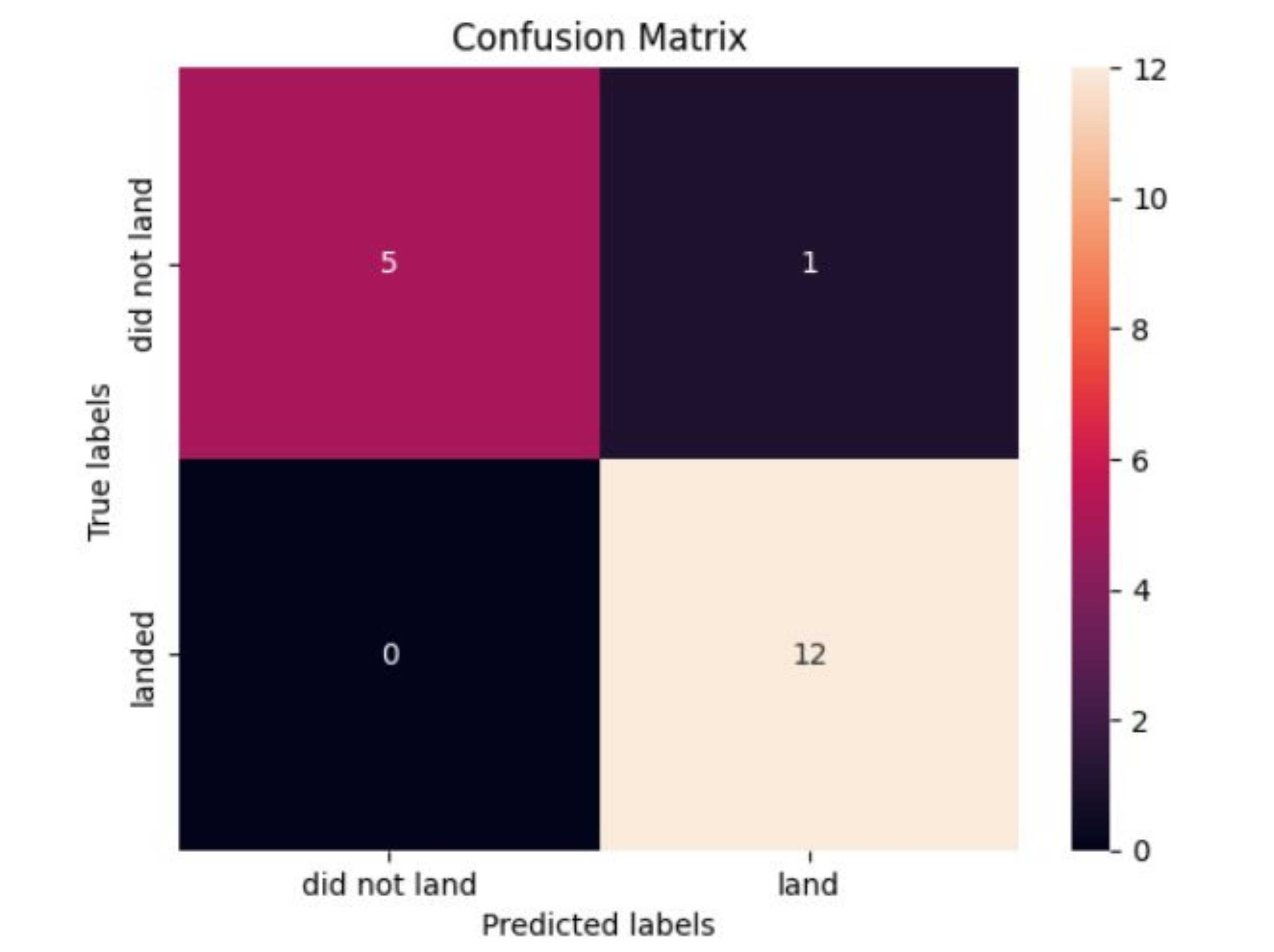
Classification Accuracy

- In the bar chart we can easily visualize the built model accuracy for all built classification models and see that the decision Tree model has the highest classification accuracy with 0.9444



Confusion Matrix

- Finally, here is presented the confusion matrix of the Decision Tree model
- With this I evaluated the model's performance beyond accuracy and identified different types of error.

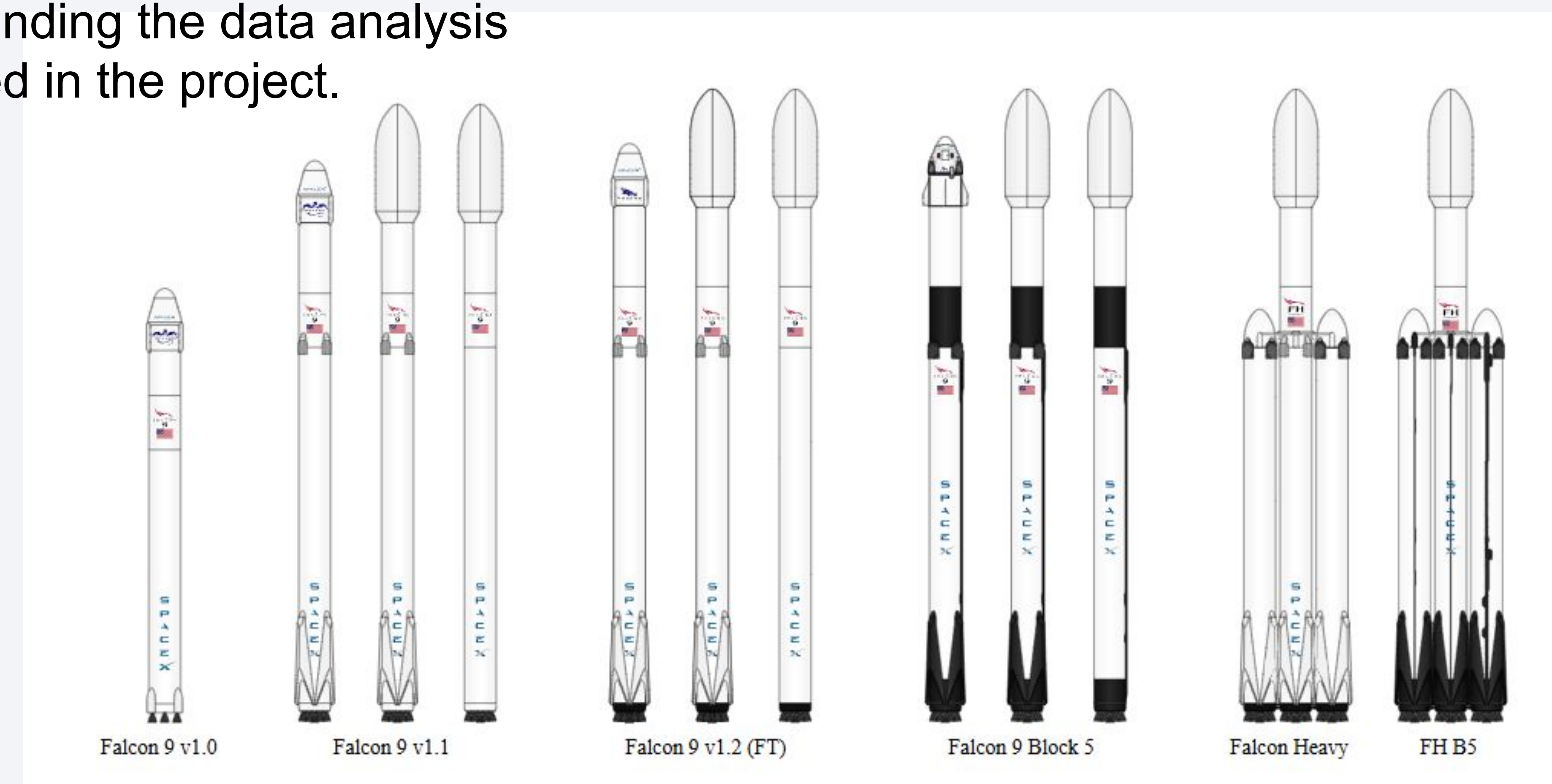


Conclusions

- I built 4 different models and evaluated their performance, then I compared their accuracy.
- I found that the Decision Tree Model built had the best accuracy with 0.9444 and I selected it as the best model for our data.
- I used the confusion matrix to evaluate overall performance of the Decision Tree model built.
- The model performs very well and I’m satisfied with the results.

Appendix

- In the appendix are included relevant assets like Python code snippets, SQL queries, charts, notebook outputs, and datasets that may help in understanding the data analysis presented in the project.



Appendix

I used the following libraries to collect the data and to store them in a dataset

```
import requests
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
# NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays
import numpy as np
# Datetime is a library that allows us to represent dates
 import datetime
```

```
# Get the head of the dataframe
print(data.head())
```

```
       static_fire_date_utc  static_fire_date_unix    tbd    net  window  \
0  2006-03-17T00:00:00.000Z           1.142554e+09  False  False     0.0
1                      None                    NaN  False  False     0.0
2                      None                    NaN  False  False     0.0
3  2008-09-20T00:00:00.000Z           1.221869e+09  False  False     0.0
4                      None                    NaN  False  False     0.0

                     rocket  success  \
0  5e9d0d95eda69955f709d1eb    False
1  5e9d0d95eda69955f709d1eb    False
2  5e9d0d95eda69955f709d1eb    False
3  5e9d0d95eda69955f709d1eb     True
4  5e9d0d95eda69955f709d1eb     True
```

Appendix

The data from these requests will be stored in lists and will be used to create a new dataframe.

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

```
# Create a data from launch_dict
df = pd.DataFrame(launch_dict)
```

Appendix

Here is what the dataset looked like when I first started working on it.

Show the summary of the dataframe

```
# Show the head of the dataframe
print(df.head())
```

```
  FlightNumber        Date BoosterVersion  PayloadMass Orbit  \
0            1  2006-03-24       Falcon 1         20.0   LEO
1            2  2007-03-21       Falcon 1          NaN   LEO
2            4  2008-09-28       Falcon 1        165.0   LEO
3            5  2009-07-13       Falcon 1        200.0   LEO
4            6  2010-06-04       Falcon 9          NaN   LEO

       LaunchSite    Outcome  Flights  GridFins  Reused   Legs LandingPad  \
0  Kwajalein Atoll  None None        1     False   False  False       None
1  Kwajalein Atoll  None None        1     False   False  False       None
2  Kwajalein Atoll  None None        1     False   False  False       None
3  Kwajalein Atoll  None None        1     False   False  False       None
4      CCSFS SLC 40  None None        1     False   False  False       None

  Block  ReusedCount    Serial   Longitude   Latitude
0   NaN            0  Merlin1A  167.743129   9.047721
1   NaN            0  Merlin2A  167.743129   9.047721
2   NaN            0  Merlin2C  167.743129   9.047721
3   NaN            0  Merlin3C  167.743129   9.047721
4   1.0            0     B0003  -80.577366  28.561857
```

Appendix

Data Analysis

Load Space X dataset, from last section.

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/data
df.head(10)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Bloc
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.(
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.(
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.(
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.(
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.(
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.(
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.(
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.(

Appendix

- Here, I included the Python code I used to find the best machine learning classification model for the dataset and to visualize it with a bar plot.

```
# Accuracy of Classification Models
models_test_accuracy = {
    "Logistic Regression": logreg_cv.score(X_test, Y_test),
    "SVM": svm_cv.score(X_test, Y_test),
    "Decision Tree": tree_cv.score(X_test, Y_test),
    "KNN": knn_cv.score(X_test, Y_test)
}

# Print all results
for model, acc in models_test_accuracy.items():
    print(f"{model} Test Accuracy: {acc:.4f}")

# Find the best one
best_model = max(models_test_accuracy, key=models_test_accuracy.get)
print(f"\n Best model on test data: {best_model} with accuracy {models_test_accuracy[best_model]:.4f}")

# Histogram
plt.figure(figsize=(10, 6))

model_names = list(models_test_accuracy.keys())
accuracy_values = list(models_test_accuracy.values())

bars = plt.bar(model_names, accuracy_values,
               color=['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728'],
               alpha=0.8, edgecolor='black', linewidth=1)
plt.title('Accuracy of Classification Models', fontsize=16, fontweight='bold', pad=20)
plt.xlabel('Models', fontsize=12, fontweight='bold')
plt.ylabel('Accuracy', fontsize=12, fontweight='bold')
# Rotazione delle etichette dell'asse x per una migliore leggibilità
plt.xticks(rotation=45, ha='right')
# y-axis limits
plt.ylim(0, 1.1)
# values on graph
for bar, value in zip(bars, accuracy_values):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.01,
             f'{value:.3f}', ha='center', va='bottom', fontweight='bold')
# Grid
plt.grid(axis='y', alpha=0.3, linestyle='--')
# automatic layout
plt.tight_layout()

# Show graph
plt.show()
```

Thank you!