IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Angel Luyo Martinez
01 February 2025

Outline

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

Executive Summary

- Summary of methodologies
  - SpaceX Data Collection API
  - SpaceX Web Scraping
  - SpaceX Data Wrangling
  - SpaceX Exploratory Data Analysis using SQLite
  - SpaceX Data Visualization using Pandas and Matplotlib
  - SpaceX Launch Site Analysis
  - SpaceX Dashboard Plotly Dash
  - SpaceX Machine Learning Prediction
- Summary of all results
  - Data collection and preprocessing.
  - Exploratory analysis and visualization.
  - Machine learning and dashboard.

Introduction

- Project background and context

  SpaceX advertises Falcon 9 rocket launches on its website at a cost of $62 million, whereas other providers charge upwards of $165 million per launch. Much of SpaceX's cost savings come from its ability to reuse the Falcon 9 first stage, making spaceflight significantly more economical. Understanding the conditions that lead to successful landings is crucial for cost estimation and optimization. By analyzing historical launch data, I aim to uncover patterns that influence landing success and explore how these insights can impact the future of commercial spaceflight.

- Problems you want to find answer

  In this capstone, I will predict if the Falcon 9 first stage will land successfully using launch data while also identifying key factors that influence landing outcomes. This analysis will help assess cost efficiency and competitive advantages in the aerospace industry.

Section 1

Methodology

Methodology

Executive Summary

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

Data Collection

- Overview of SpaceX Falcon 9 Data Collection

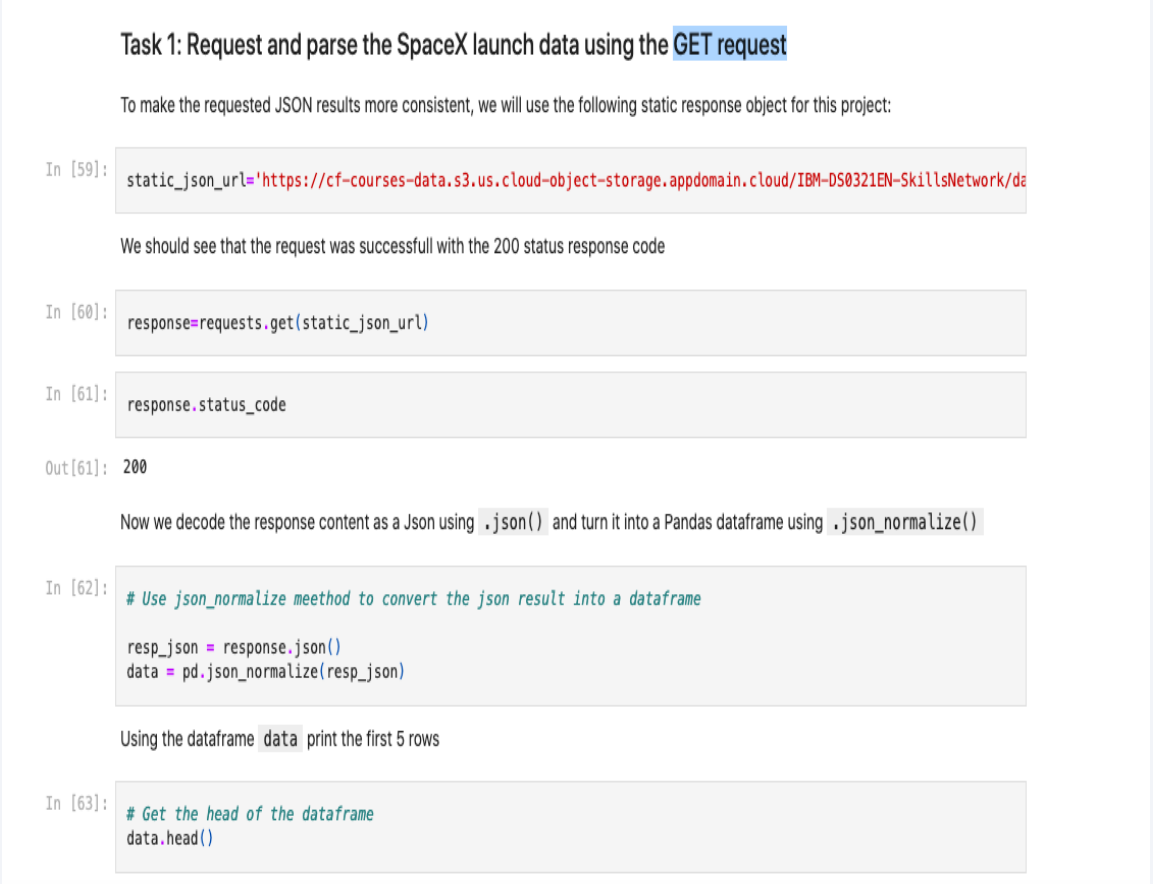
The SpaceX Falcon 9 data was collected using the SpaceX API, a RESTful API, by making a GET request to retrieve rocket launch data. To accomplish this, a series of helper functions were defined to extract relevant information using identification numbers from the launch data. The requested JSON response was then parsed, decoded, and converted into a Pandas DataFrame to ensure consistency and ease of analysis.

Additionally, web scraping was conducted to collect historical Falcon 9 launch records from a Wikipedia page titled *List of Falcon 9 and Falcon Heavy launches.* Using the BeautifulSoup and requests libraries, the HTML table containing the launch records was extracted, parsed, and converted into a Pandas DataFrame for further examination

Data Collection – SpaceX API

- The SpaceX Falcon 9 data was gathered through the SpaceX API, a RESTful API, by making a GET request to retrieve launch data. The response content was then parsed, decoded into a JSON format, and converted into a Pandas DataFrame for further analysis.

- Here is the GitHub URL for the completed notebook containing the SpaceX API calls: https://github.com/Gitangelito/Data-Science-Capstone-Coursera-Presentation/blob/main/1-jupyter-labs-spacex-data-collection-api.ipynb



Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

In [59]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
```

We should see that the request was successfull with the 200 status response code

In [60]:
```
response=requests.get(static_json_url)
```

In [61]:
```
response.status_code
```

Out[61]: 200

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

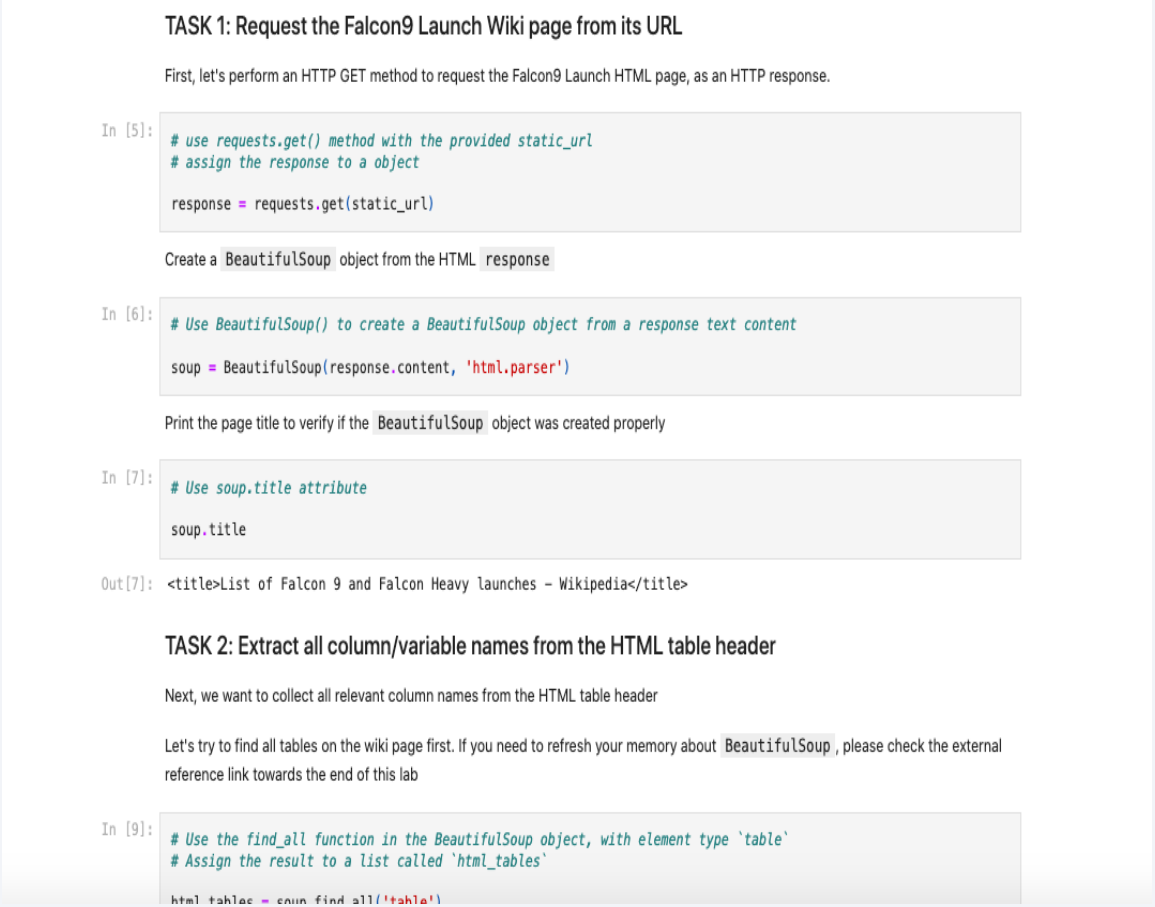
In [62]:
```
# Use json_normalize meethod to convert the json result into a dataframe

resp_json = response.json()
data = pd.json_normalize(resp_json)
```

Using the dataframe `data` print the first 5 rows

In [63]:
```
# Get the head of the dataframe
data.head()
```

Data Collection – Web Scraping

- Web scraping was conducted to gather Falcon 9 historical launch records from Wikipedia using BeautifulSoup and Requests. The Falcon 9 launch records were extracted from an HTML table on the Wikipedia page, parsed, and converted into a structured DataFrame for analysis
- Here is the GitHub URL for the completed notebook containing the(https://github.com/Gitangelito/Data-Science-Capstone-Coursera-Presentation/blob/main/2-jupyter-labs-webscraping.ipynb)



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.

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object

response = requests.get(static_url)
```

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

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

soup = BeautifulSoup(response.content, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [7]:
```
# Use soup.title attribute

soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches – Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

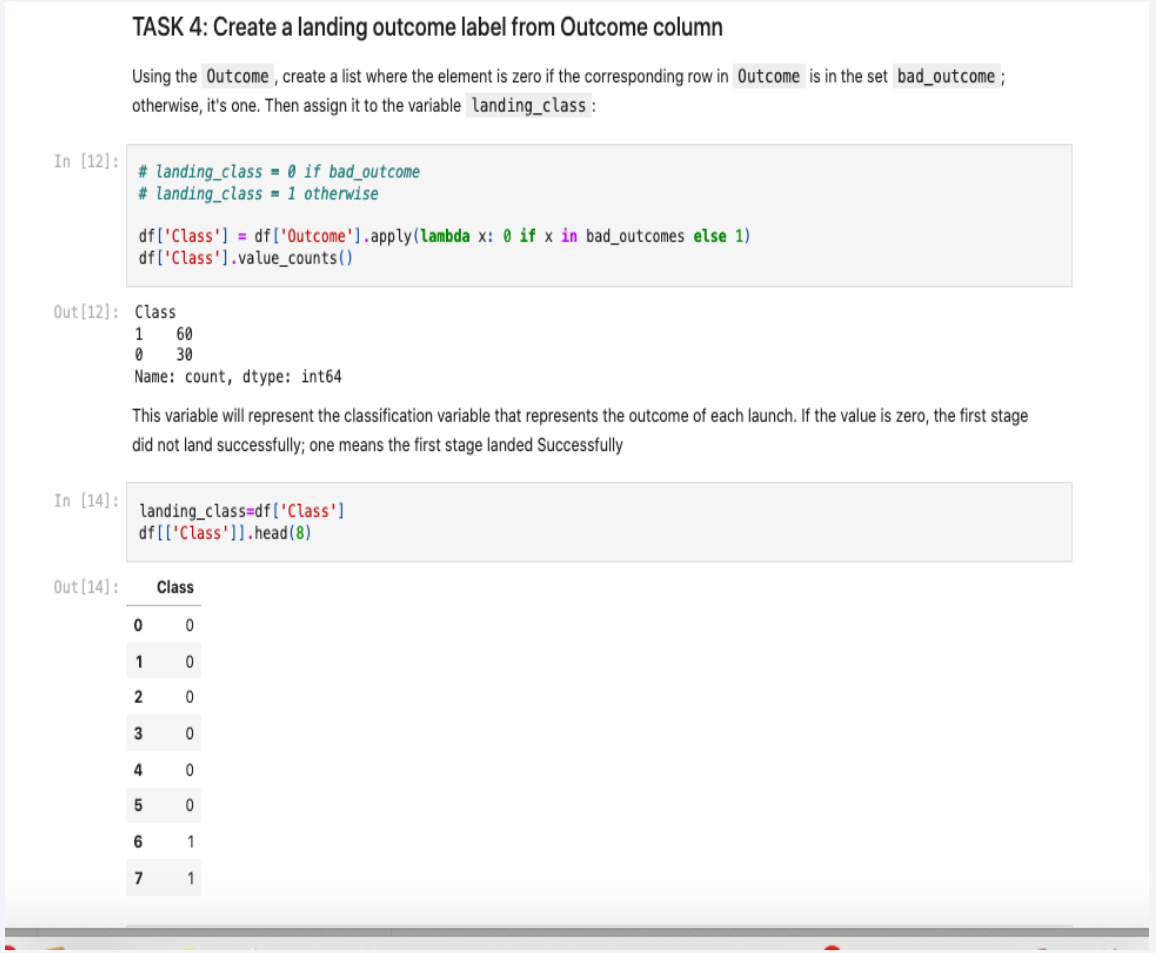
Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

In [9]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`

html_tables = soup.find_all('table')
```

Data Wrangling

- Once the data was collected and converted into a pandas DataFrame, it was filtered by the BoosterVersion column to retain only Falcon 9 launches. Next, missing values in the LandingPad and PayloadMass columns were addressed, with missing PayloadMass values filled in using the column's mean value.
- EDA was performed to identify patterns and select the label for supervised model training
- Here is the link of the completed wrangling notebook https://github.com/Gitangelito/Data-Science-Capstone-Coursera-Presentation/blob/main/3-labs-jupyter-



TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

In [12]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

df['Class'] = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
df['Class'].value_counts()
```

Out[12]:
```
Class
1    60
0    30
Name: count, dtype: int64
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

In [14]:
```
landing_class=df['Class']
df[['Class']].head(8)
```

Out[14]:

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

EDA with Data Visualization

- Performed data analysis and feature engineering using pandas and matplotlib
- Explored relationships between flight number, payload mass, and launch site
- Visualized the relationship between flight number and launch site using scatter plots
- Used scatter plots to analyze the relationship between payload mass and launch site
- Examined the success rate of each orbit type using bar charts
- Investigated the relationship between flight number and orbit type with scatter plots
- Analyzed the connection between payload mass and orbit type using scatter plots
- Visualized the yearly trend in launch success using line plots
- GitHub URL with completed EDA https://github.com/Gitangelito/Data-Science-Capstone-Coursera-Presentation/blob/main/5-edadataviz.ipynb

EDA with SQL

- The following SQL queries were executed for exploratory data analysis (EDA):
  - Display the names of the unique launch sites in the space mission

```
In [17]: %sql select Distinct Launch_Site from SPACEXTBL;
```

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

```
In [24]: %sql select * from SPACEXTBL where Launch_Site like "cca%" limit 5;
```

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

```
In [26]: %sql select sum (PAYLOAD_MASS__KG_) As "TotalMass_in_Kgs",Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)'
```

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

```
In [28]: __KG_) as "PayLoadMass_in_Kgs" , Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';
```

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

```
In [33]: %sql select MIN(Date) from 'SPACEXTBL' where "Landing_Outcome" = "Success (ground pad)";
```

- 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 DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;]

- Here is the Github URL of the completed EDA SQL notebook
- https://github.com/Gitangelito/Data-Science-Capstone-Coursera-Presentation/blob/main/4-jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Developed a Folium map to visualize all launch sites, incorporating markers, circles, and lines to indicate the success or failure of launches at each location.

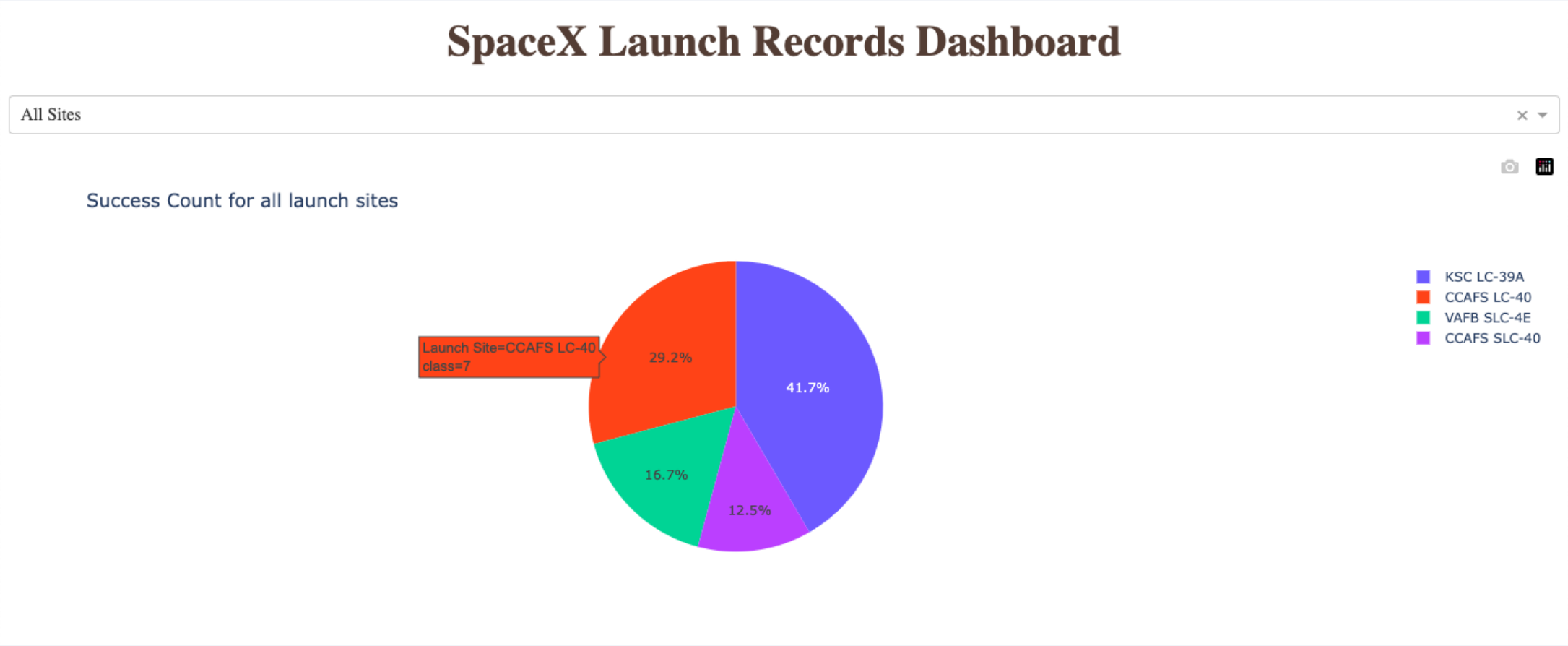
- Defined launch outcome categories, assigning 0 for failures and 1 for successes.

- Here is the GitHub URL for the completed interactive Folium map https://github.com/Gitangelito/Data-Science-Capstone-Coursera-Presentation/blob/main/6-lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Developed an interactive dashboard application using Plotly Dash to analyze launch outcomes by:
  - **Implementing a dropdown menu to select launch sites** – Allows users to filter data for specific launch locations.
  - **Creating a callback function to generate a success pie chart based on the selected site** – Provides a visual breakdown of success rates for each launch site.
  - **Adding a range slider for payload selection** – Enables users to explore how payload mass affects launch success.
  - **Implementing a callback function to render a success-payload scatter plot** – Helps identify trends between payload mass and launch success rates.
  - Here is the GitHub URL of your completed Plotly Dash lab https://github.com/Gitangelito/Data-Science-Capstone-Coursera-Presentation/blob/main/7-Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dashspacex_dash_app.py

Plotly Dashboard



Predictive Analysis (Classification)

- After importing the data into a Pandas DataFrame, I performed exploratory data analysis and identified training labels by:
  - Converting the **class** column into a NumPy array using .to_numpy() and assigning it to **Y** as the outcome variable.
  - Standardizing the feature dataset (**X**) using StandardScaler() from sklearn.preprocessing to ensure consistency in scale.
  - Splitting the data into training and testing sets with an **80-20 ratio** using train_test_split from sklearn.model_selection, setting test_size=0.2 and random_state=2 for reproducibility.

Predictive Analysis (Classification)

- To determine the best-performing ML model for the test data among SVM, classification trees, k-nearest neighbors, and logistic regression:
  - Created an object for each algorithm and initialized a GridSearchCV object with a defined set of parameters for each model.
  - Configured GridSearchCV with cv=10 and trained it on the dataset to identify the optimal hyperparameters.
  - Retrieved the best parameters using best_params_ and assessed model accuracy on validation data using best_score_.
  - Finally, evaluated test accuracy using the score method and visualized model performance by plotting a confusion matrix for each.

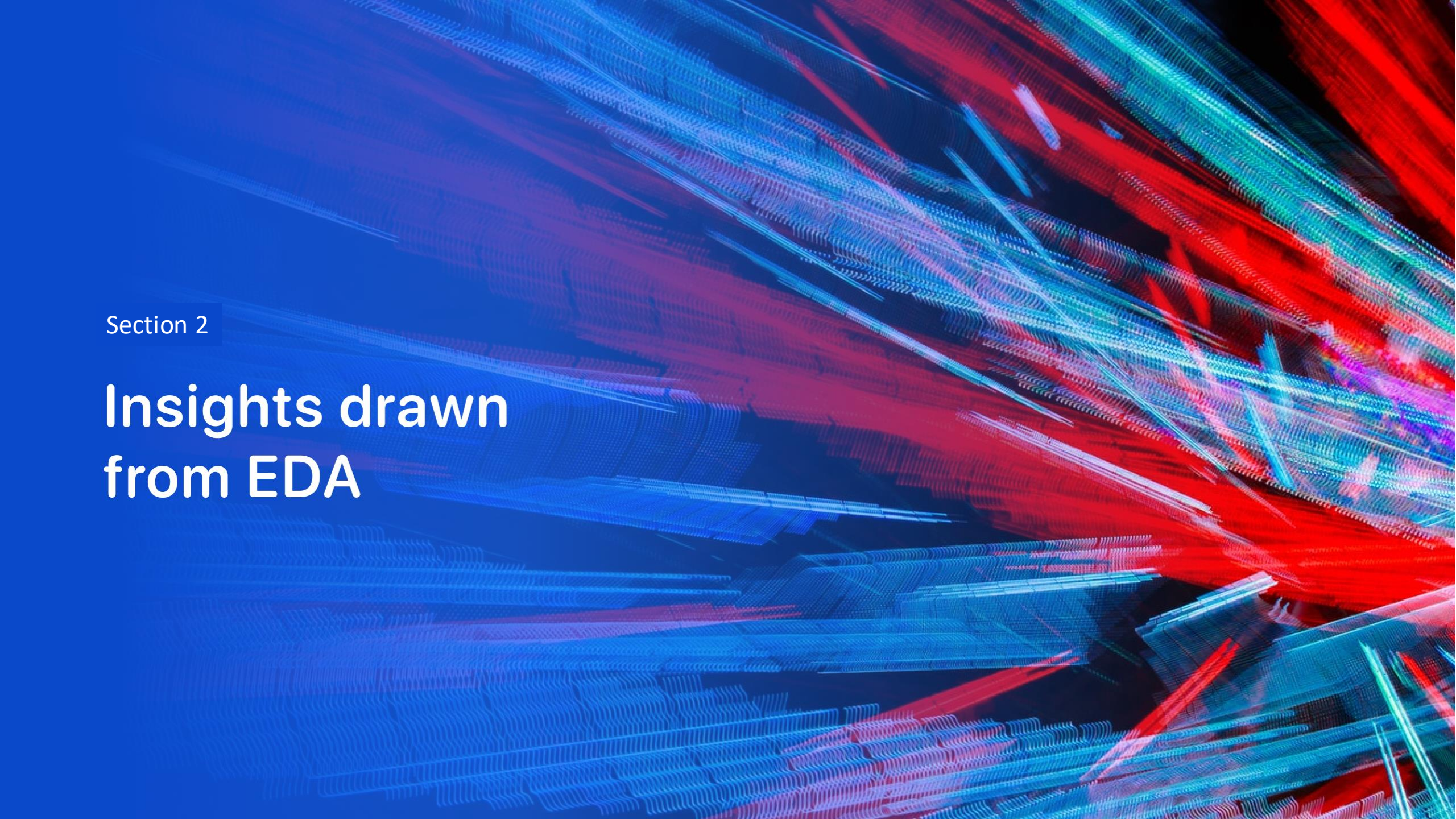
Predictive Analysis (Classification - Table)

- The table below presents the test data accuracy scores for each method, comparing SVM, classification trees, k-nearest neighbors, and logistic regression to determine the best-performing model. GitHub URL: https://github.com/Gitangelito/Data-Science-Capstone-Coursera-Presentation/blob/main/8-SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

	0	1	2	3
Method	Logistic Regression	SVM	Decision Tree	KNN
Test Data Accuracy	0.833333	0.833333	0.777778	0.833333

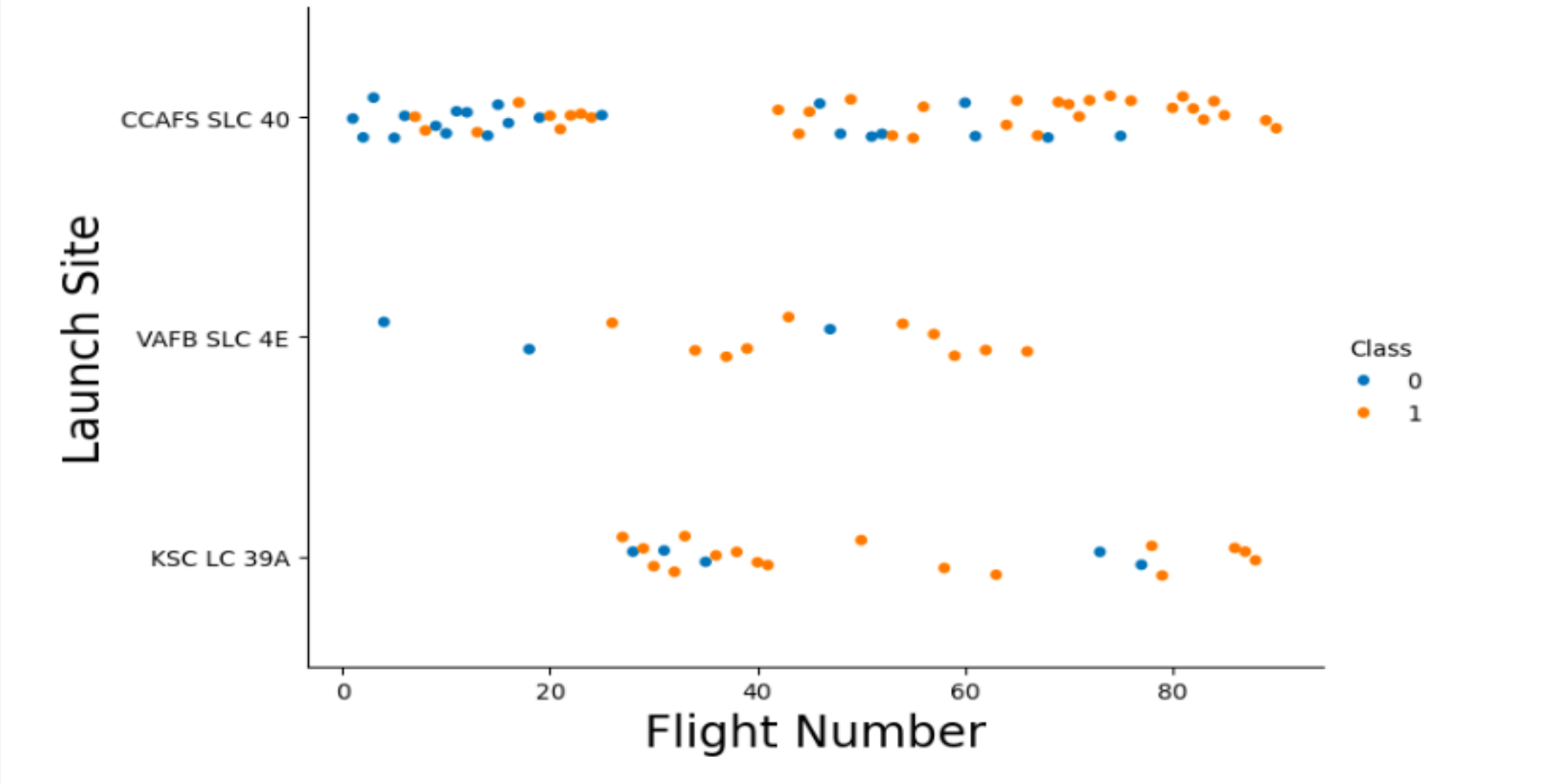
Results

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


Section 2
Insights drawn from EDA

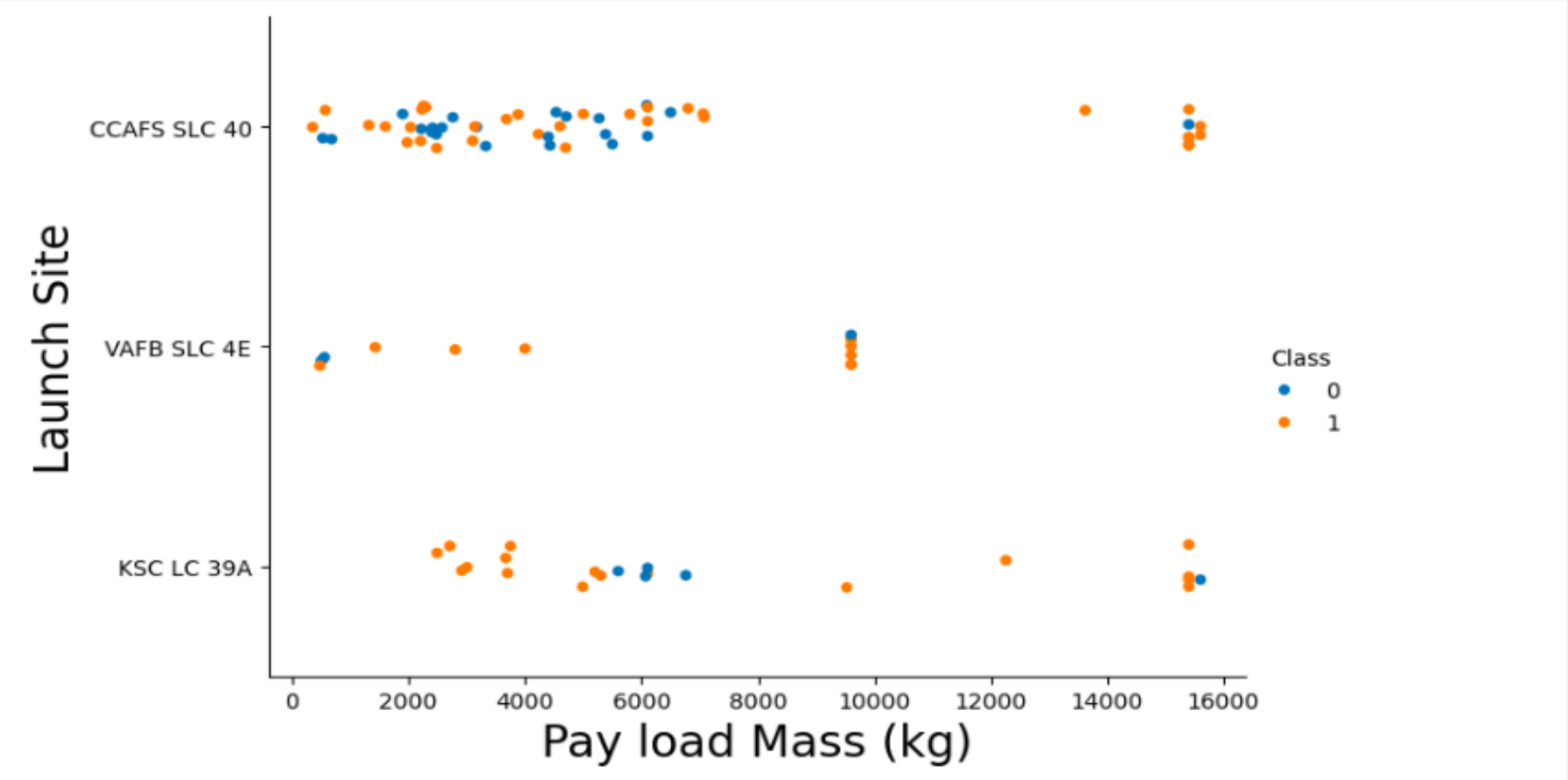
Flight Number vs. Launch Site

- Scatter plot of Flight Numbers vs. Launch Site



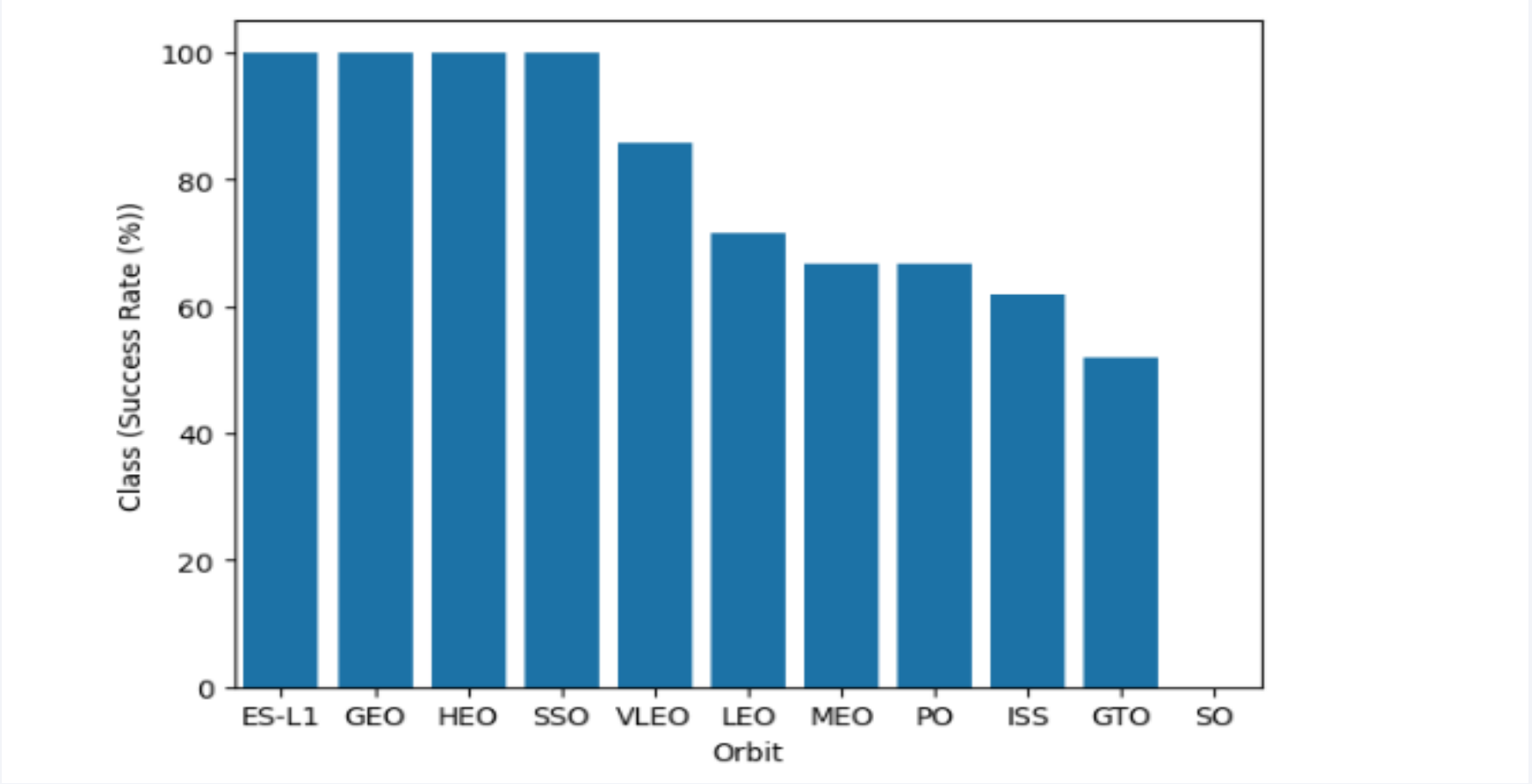
Payload vs. Launch Site

- A scatter plot of Payload vs. Launch Site



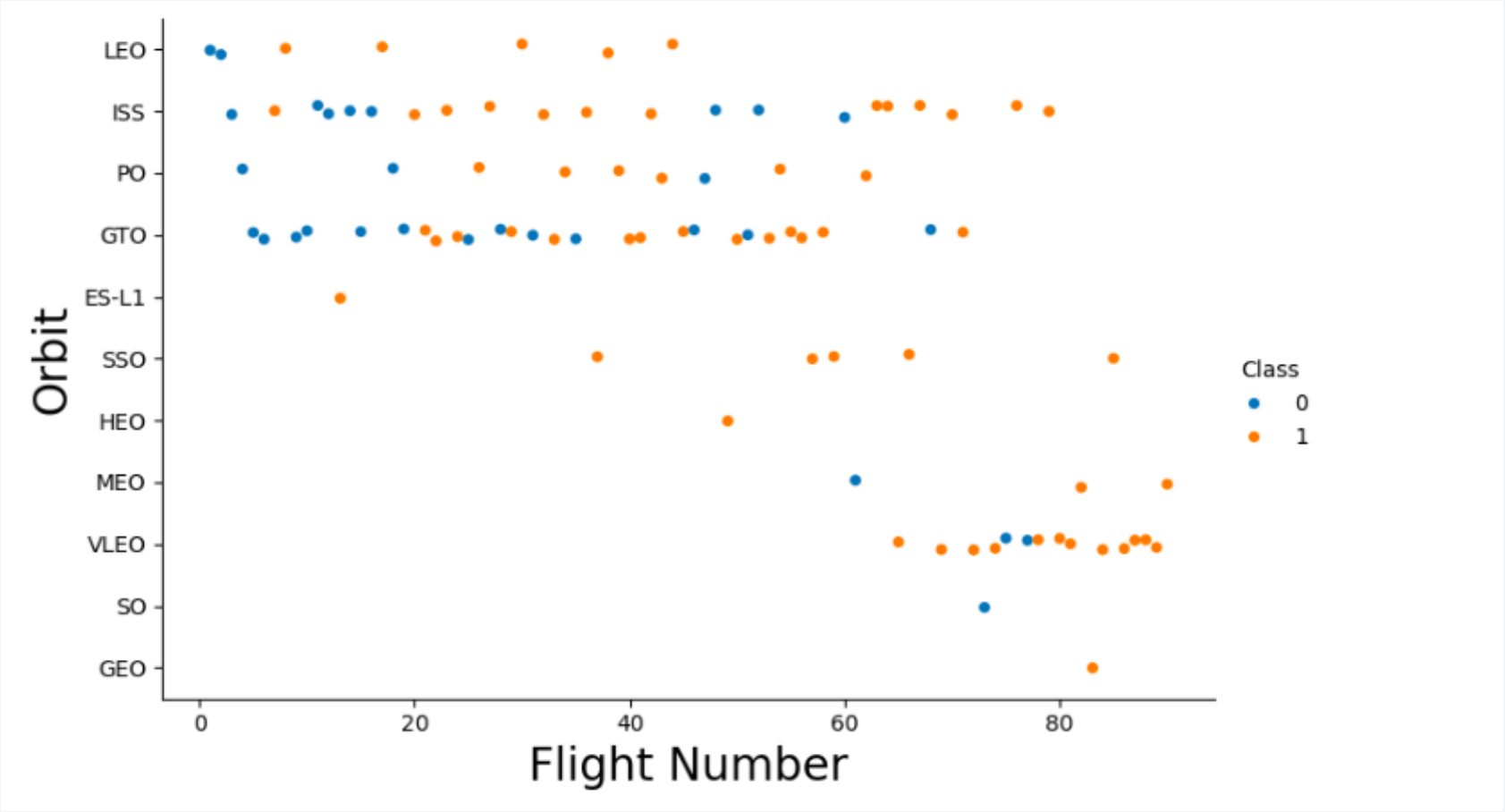
Success Rate vs. Orbit Type

- Bar chart for the success rate of each orbit type



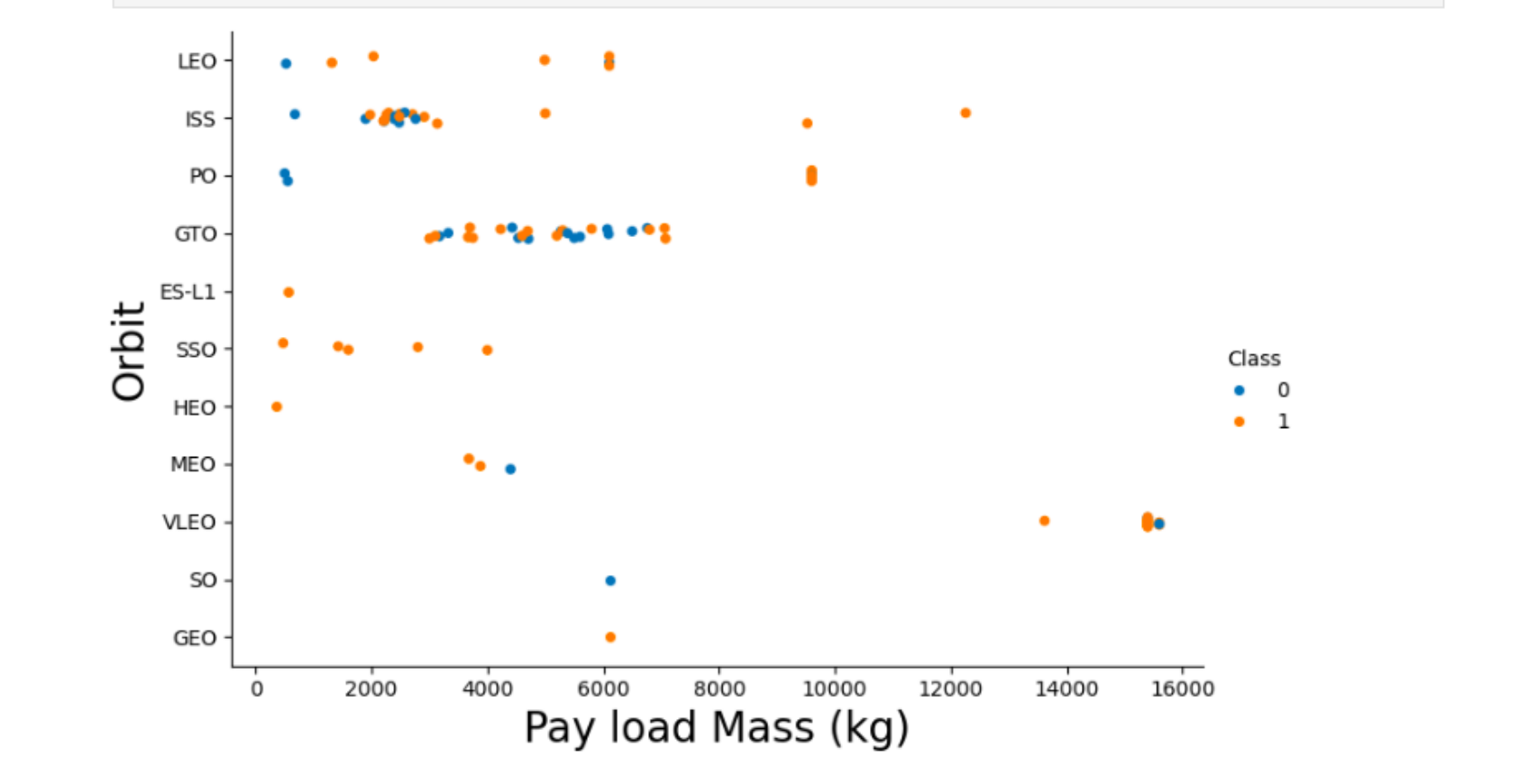
Flight Number vs. Orbit Type

- A scatter point of Flight number vs. Orbit type



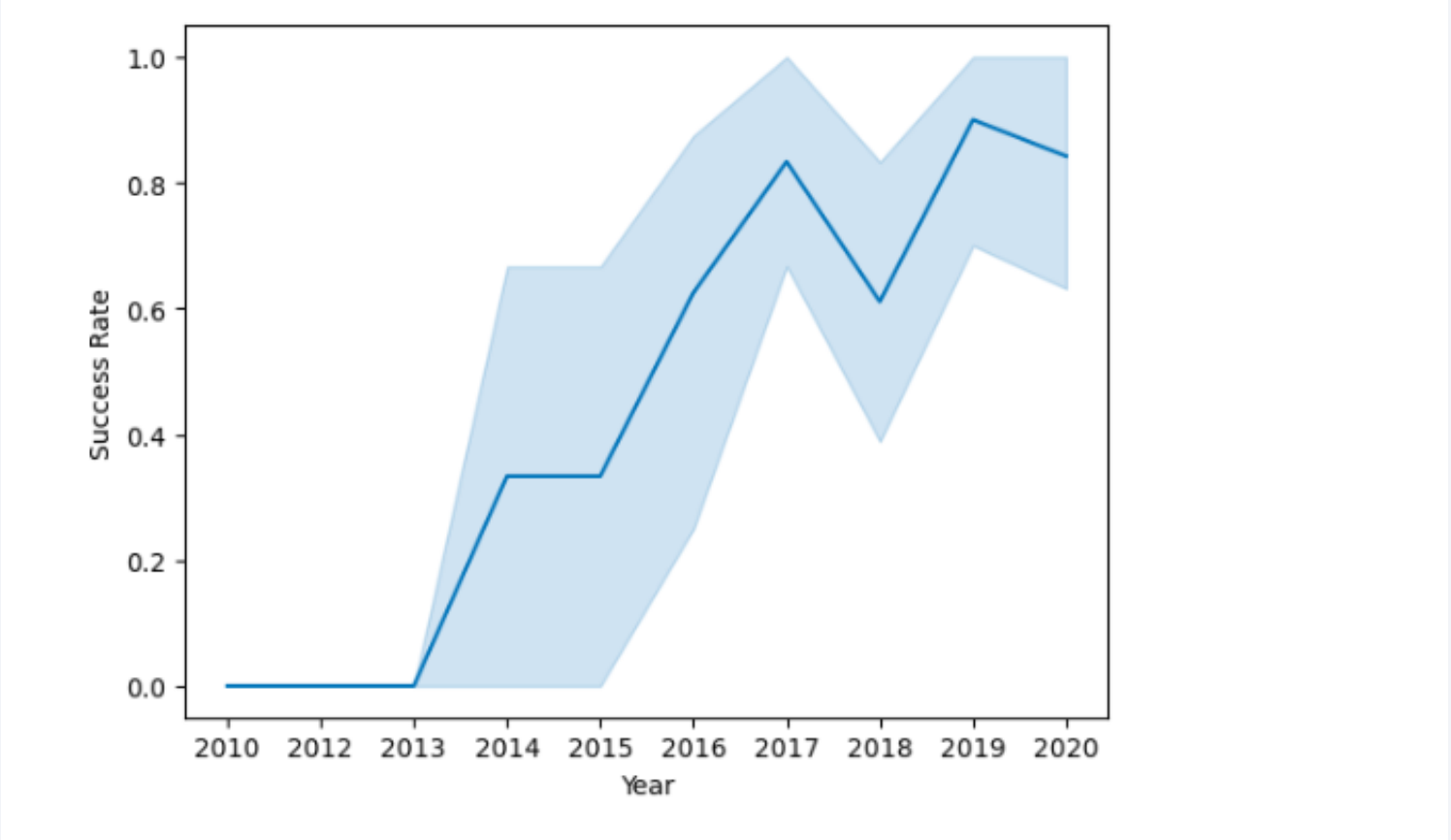
Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type



Launch Success Yearly Trend

- Line chart showing yearly average success rate
- The success rate kept going from 2013 to 2020



All Launch Site Names

- Retrieve the names of distinct launch sites.
- Utilized the SELECT DISTINCT statement to retrieve only the unique launch sites from the Launch_site column of the SPACEXTBL table.

Task 1

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

```
%sql select Distinct Launch_Site from SPACEXTBL;
```

```
 * 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'

- Find 5 records where launch sites begin with `CCA`
- Employed the 'LIKE' command with the "%" wildcard in the 'WHERE' clause to filter and display all records where launch sites begin with the string "CCA".

Task 2

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

```
%sql select * from SPACEXTBL where Launch_Site like "cca%" limit 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Ou
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (par
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 (par
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No a
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No a
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No a

Total Payload Mass

Task 3

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

In [26]:
```
%sql select sum (PAYLOAD_MASS__KG_) As "TotalMass_in_Kgs",Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)'
```

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

Out[26]:

TotalMass_in_Kgs	Customer
45596	NASA (CRS)

- Calculate the total payload carried by boosters from NASA
- Utilized the sum() function to calculate and display the total value of the Payload_Mass_KG column for NASA (CRS).

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Applied the avg() function to calculate and display the average payload mass carried by booster version F9 v1.1.

Task 4

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

```
[n [ ]:
# %sql SELECT * FROM 'SPACEXTBL'
```

```
[28]:
%sql select avg(PAYLOAD_MASS__KG_) as "PayLoadMass_in_Kgs" , Customer, Booster_Version FROM 'SPACEXTBL' WHERE Boc
```

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

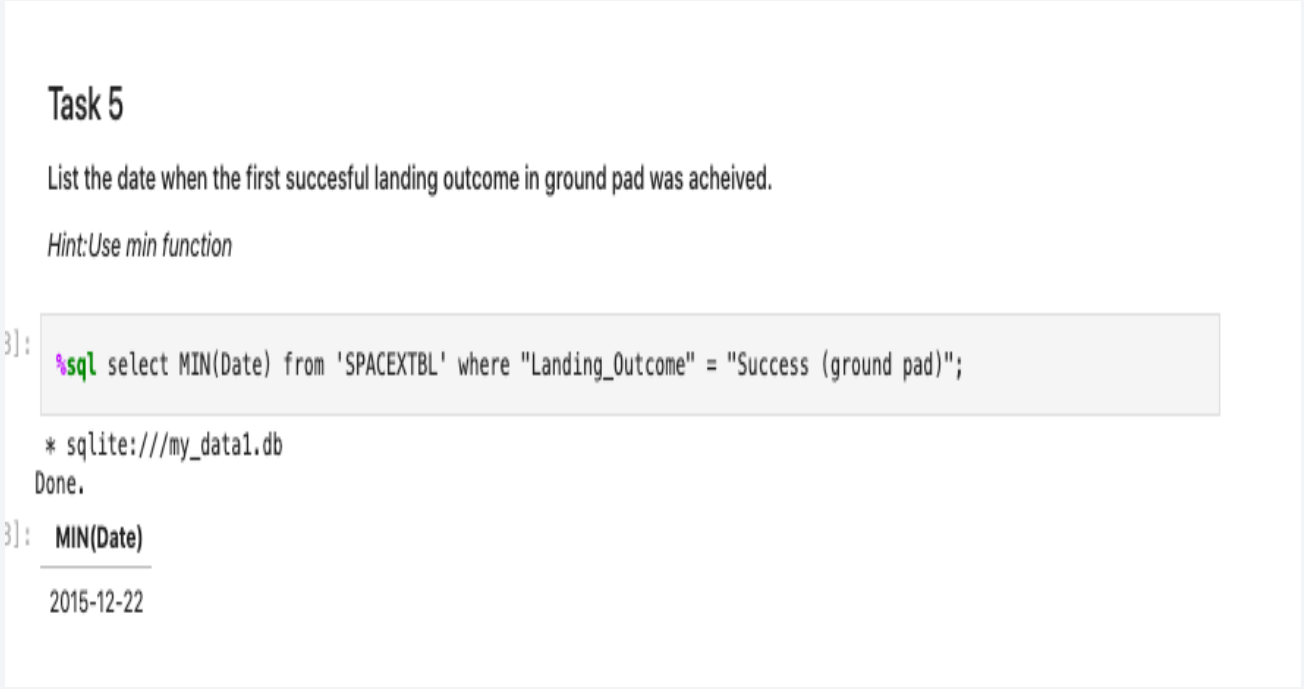
ut[28]:

PayLoadMass_in_Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

Task 5

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Utilized the min() function to retrieve and display the earliest (oldest) date of the first successful landing outcome on the ground pad labeled 'success (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) from 'SPACEXTBL' where "Landing_Outcome" = "Success (ground pad)";
```

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

MIN(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

```
In [38]: %sql SELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (drone ship)" AND
 * sqlite:///my_data1.db
Done.
```

Out[38]:

Booster_Version	Payload
F9 FT B1022	JCSAT-14
F9 FT B1026	JCSAT-16
F9 FT B1021.2	SES-10
F9 FT B1031.2	SES-11 / EchoStar 105

- List of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
- Utilized the SELECT DISTINCT statement to retrieve and list unique booster names with operators between 4000 and 6000, filtering only boosters with payloads in this range and a landing outcome of 'success (drone ship)'.

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Utilized the COUNT() function with the GROUP BY statement to retrieve the total number of mission outcomes.

Task 7

List the total number of successful and failure mission outcomes

```
39]: %sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total_Successful FROM SPACEXTBL GROUP BY "Mission_Outc
 * sqlite:///my_data1.db
Done.
```

39]:

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

Boosters Carried Maximum Payload

- List of boosters which have carried the maximum payload mass
- Utilized a subquery to retrieve the maximum payload and listed all boosters that carried the maximum payload of 15,600 kg.

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX
```

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

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

2015 Launch Records

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

- Utilized the SUBSTR() function to extract month and year from the Date column and filtered records for the year 2015 with failed landing_outcomes on a drone ship, displaying booster versions and launch sites.

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, "Booster_Version", "Launch_Site", "Landing_Outcome" FROM SPACEXTBL WHERE
```

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

Month	Booster_Version	Launch_Site	Landing_Outcome
01	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- Landing outcomes ranking (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order
- Utilized the COUNT( ) function with GROUP BY to rank landing outcomes, such as "Failure (drone ship)" and "Success (ground pad)," within the date range 2010-06-04 to 2017-03-20, ordering results in descending order of occurrence.

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 SPACEXTBL WHERE Date BETWEEN '2010-06-04' AND '201
```

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

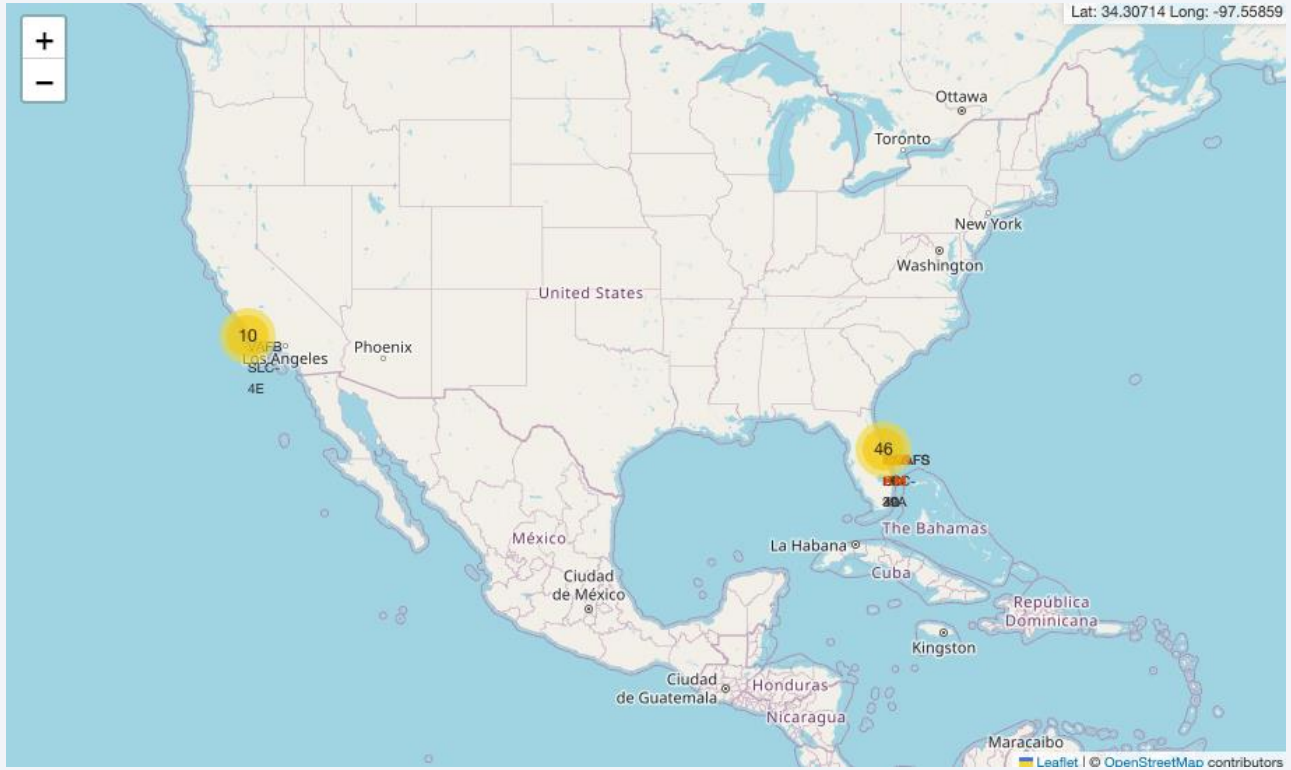
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

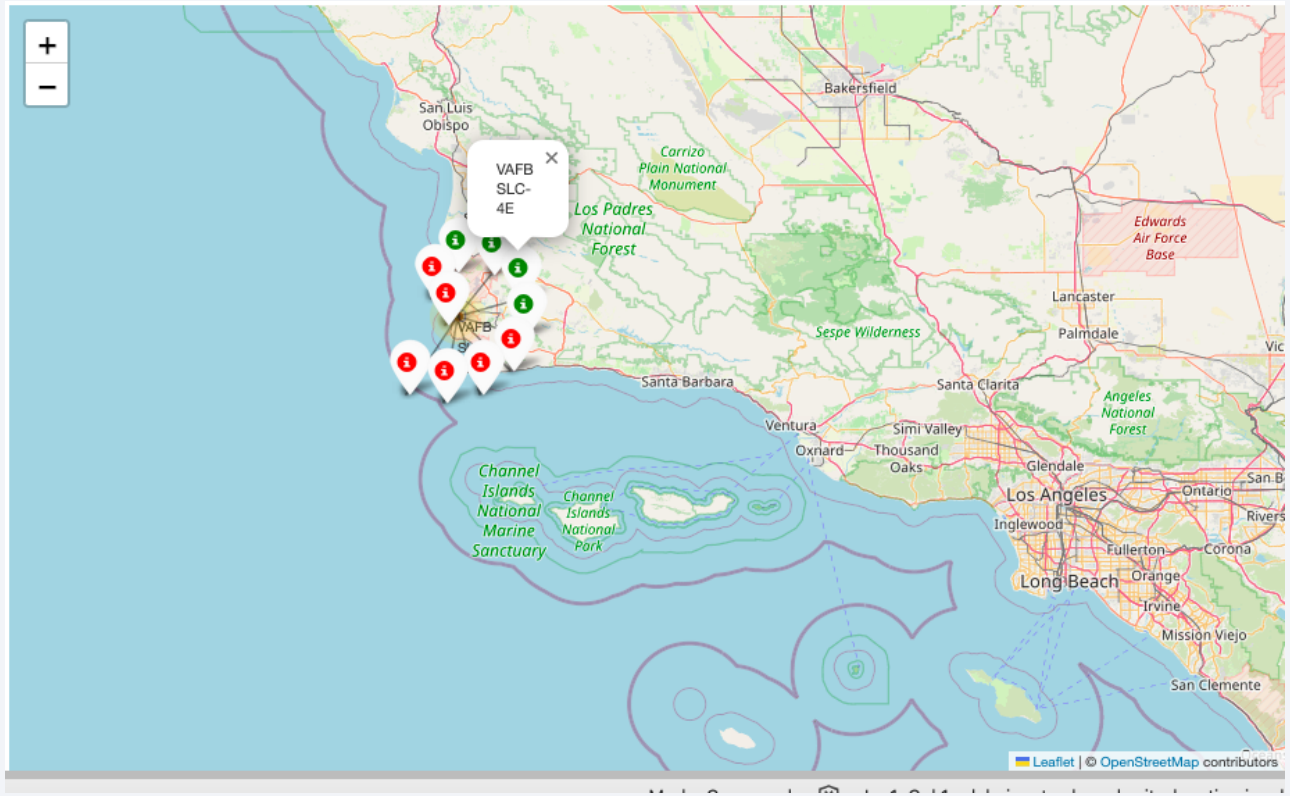
Visualizing all Launch sites Outcomes

- I explored the folium map, visualizing the launch sites locations on a global scale. The map displays markers for launch sites like KSC LC-39A, VAFB SLC-4E, and CCAFS SLC-40, located on the coasts of California and Florida. Centered on NASA Johnson Space Center, it provides a clear view of the key sites.



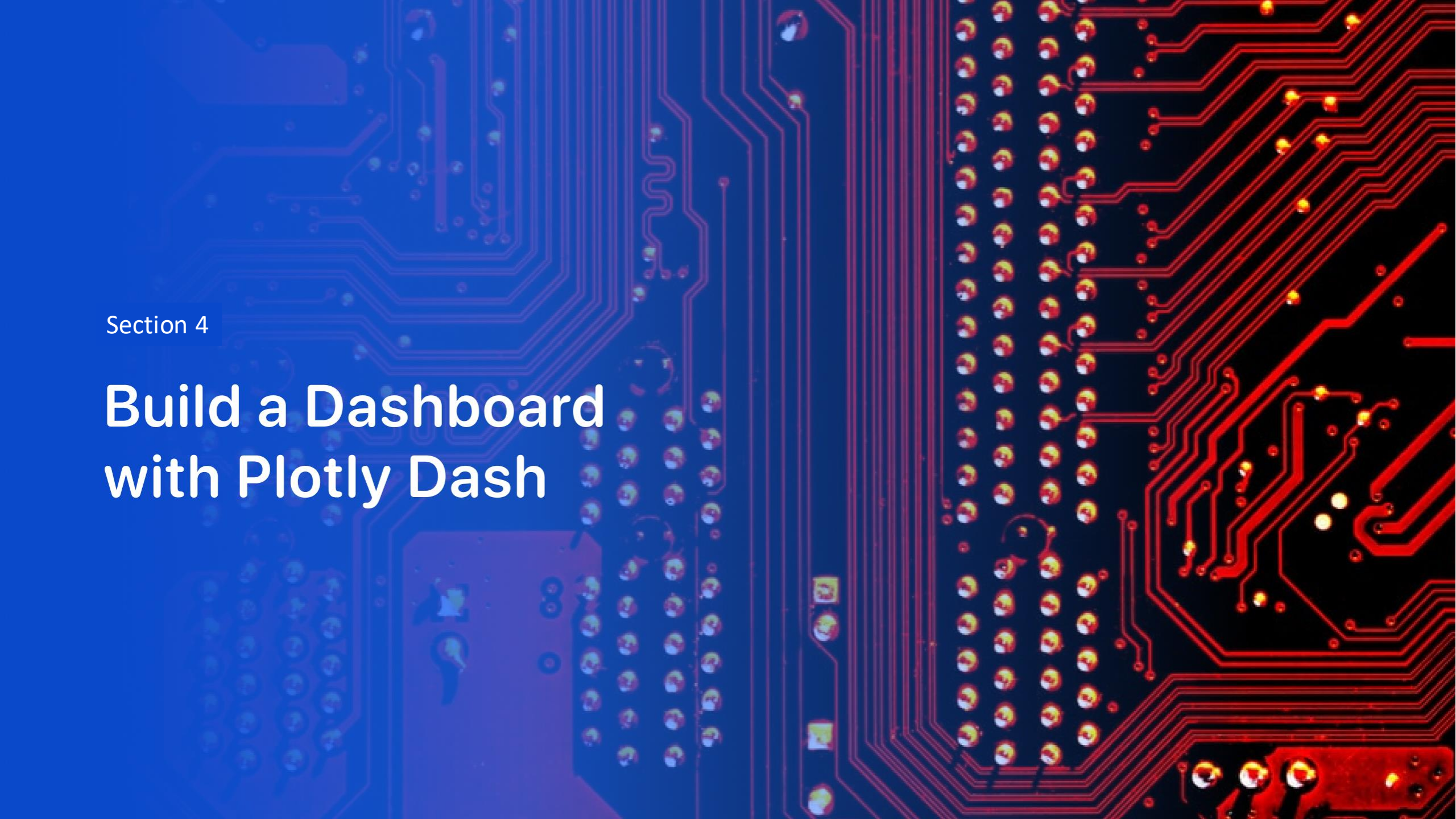
Visualizing Launch Outcomes with Color-Coded Markers on the Folium Map

- The VAFB SLC-4E launch site on the West Coast (California) has a lower success rate of 4/10 compared to the KSC LC-39A launch site on the East Coast (Florida).

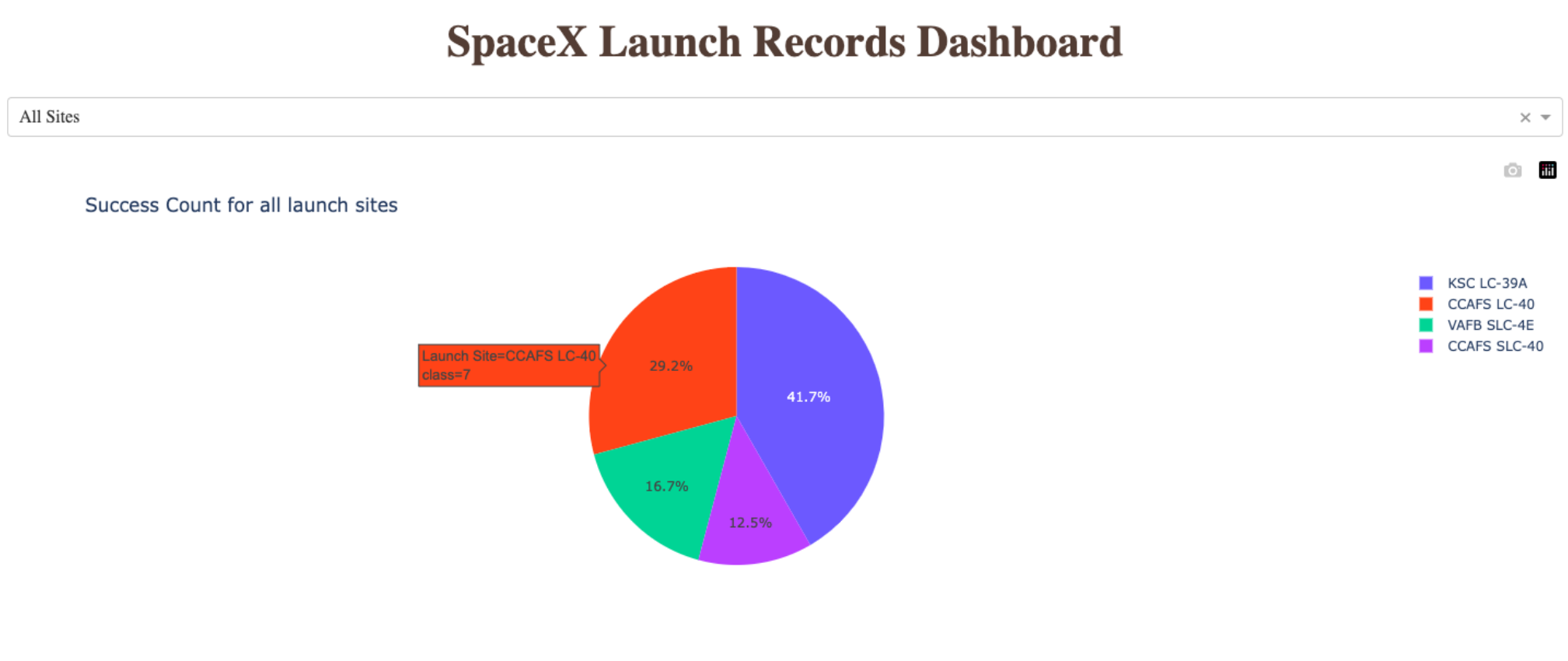


Distances from a Launch Site to Nearby Proximities

The distance from the CCAFS SLC-40 launch site to the coastline is 0.86 km.

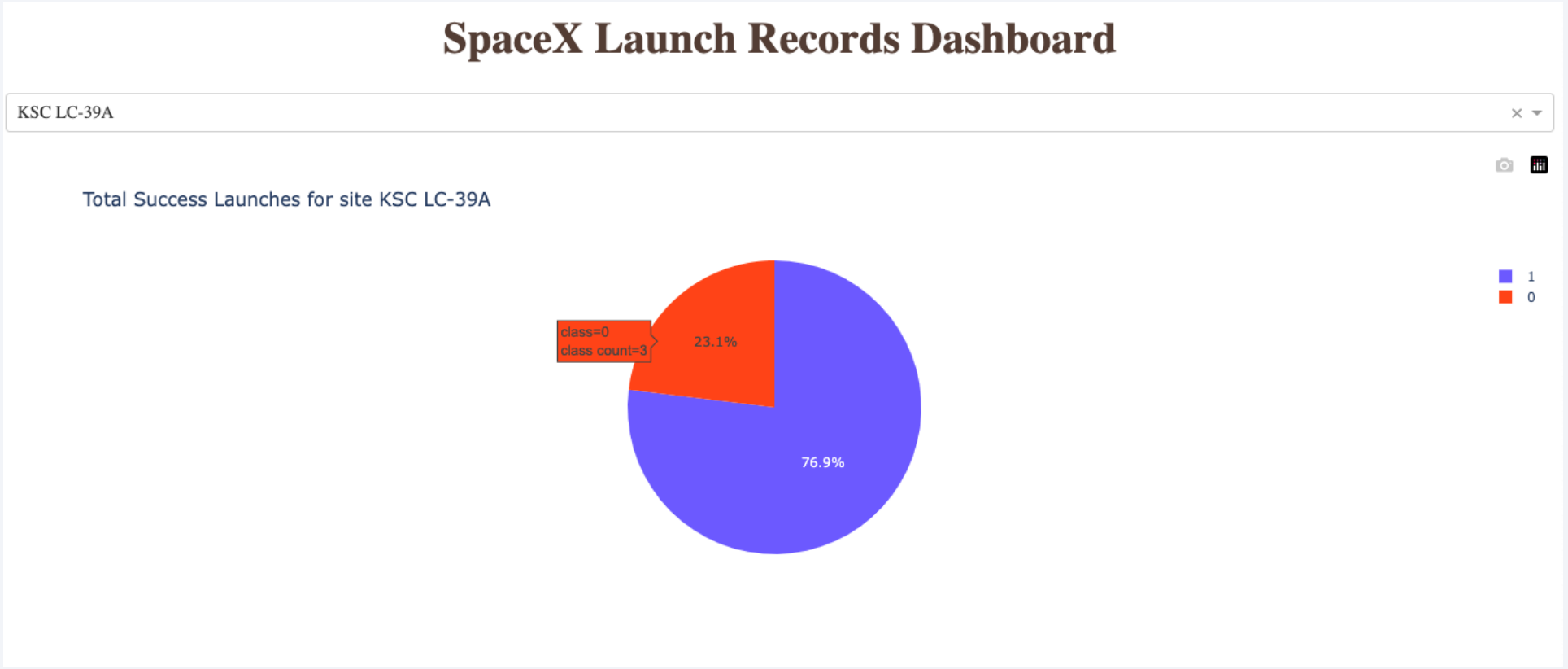

Section 4
Build a Dashboard with Plotly Dash

Launch Success Count by Site: Pie Chart



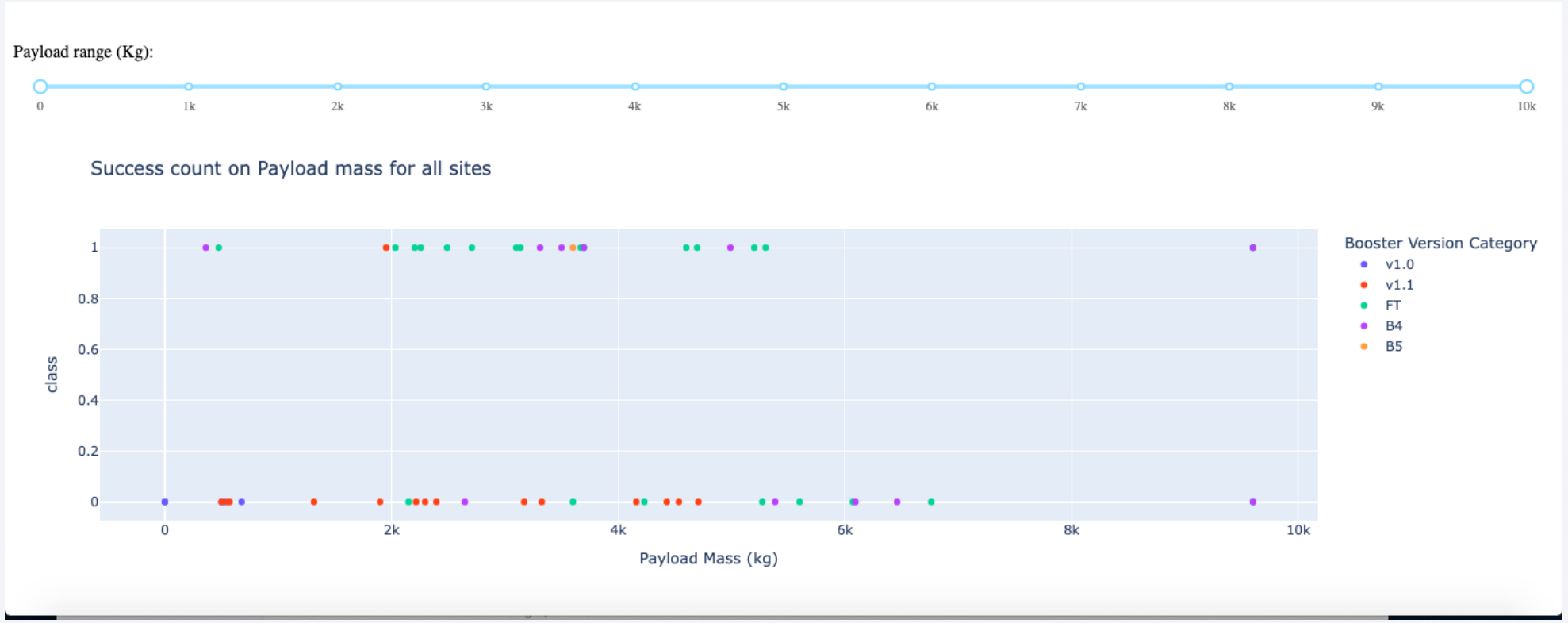
- KSC LC-39A has the highest launch success rate at 42%, followed by CCAFS LC-40 at 29%, VAFB SLC-4E at 17%, and CCAFS SLC-40 with the lowest success rate of 13%.

Pie Chart Displaying Launch Success Ratio for the Top Performing Launch Site



- Launch site KSC LC-39A had the highest success rate, with 77% successful launches and 23% failures.

Payload Vs Launch Outcomes for all sites: Scatter Plot



- For the launch site CCAFS LC-40, the FT booster version has the highest success rate for payload masses greater than 2000 kg

Section 5

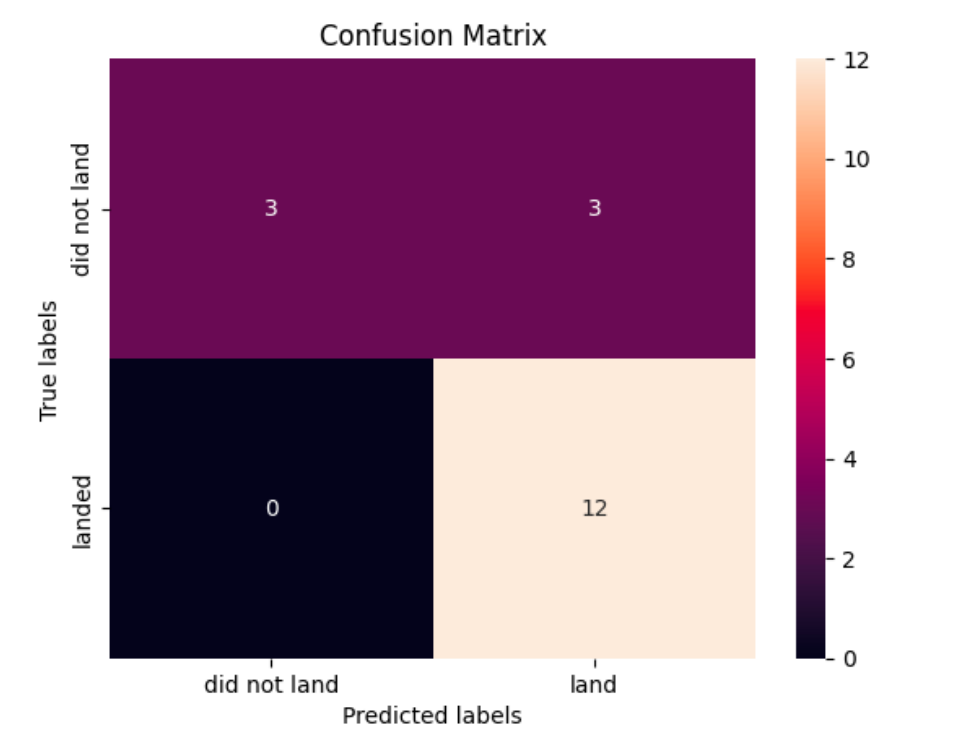
Predictive Analysis (Classification)

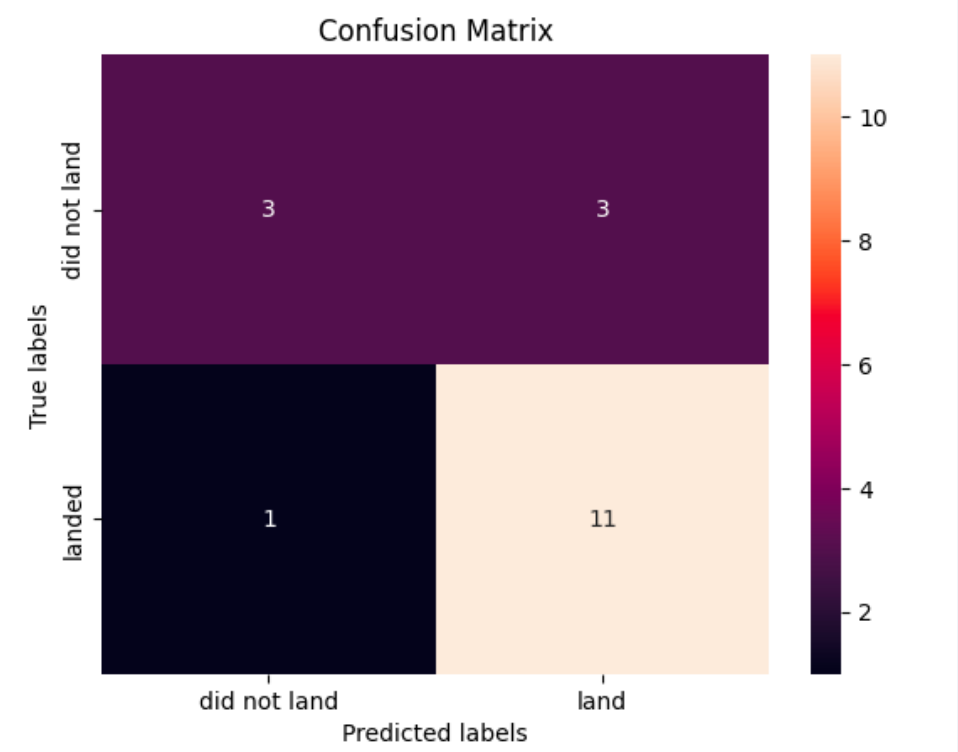
Classification Accuracy

	0	1	2	3
Method	Logistic Regression	SVM	Decision Tree	KNN
Test Data Accuracy	0.833333	0.833333	0.777778	0.833333

- The models with the highest classification accuracy are Logistic Regression, Support Vector Machine (SVM), and k-Nearest Neighbors (KNN), all of which have an accuracy of 83.33%. The model with the lowest classification accuracy is Decision Tree, with an accuracy of 77.78%.

Confusion Matrix





- Three of the four classification models—Logistic Regression, SVM, and KNN—produced identical confusion matrices and were equally able to distinguish between the different classes. However, the Decision Tree model had a different confusion matrix. A common issue across all models was the high occurrence of false positives

Conclusions

- Each launch site has a distinct success rate.
- CCAFS LC-40 has a 60% success rate, while KSC LC-39A and VAFB SLC-4E have a higher success rate of 77%.
- It’s evident that as the number of flights increases at each of the three launch sites, the success rate also improves.
- For example, after flight 50, VAFB SLC-4E achieved a 100% success rate.
- Both KSC LC-39A and CCAFS SLC-40 reached a 100% success rate after their 80th flight.
- Looking at the payload vs launch site scatter plot, it's clear that the VAFB SLC-4E launch site doesn't launch rockets with a heavy payload mass (greater than 10,000 kg).
- Orbit types such as ES-L1, GEO, HEO, and SSO have the highest success rates at 100%.
- The SO orbit has the lowest success rate at 50%, with some instances of 0% success rate.
- For LEO orbits, the success rate appears to correlate with the number of flights.
- GTO orbits show an interesting pattern where success rates seem linked to the flight count as well.

Appendix

- https://github.com/Gitangelito/Data-Science-Capstone-Coursera-Presentation/tree/main

Thank you!