IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name> Takeshi Shibue
<Date> 22/08/2024

Outline

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

Executive Summary

- Summary of methodologies
  - Space X Data Collection using Space X API
  - Space X Data Collection with Web Scraping
  - Space X Data Wrangling
  - Space X Exploratory Data Analysis using SQL
  - SpaceX EDA DataViz Using Python Pandas and Matplotlib
  - SpaceX Launch Sites Analysis with Folium interactie Visual Analytics and Ploty Dash
- Summary of all results
  - EDA results
  - Interactive Visual Analytics and Dashboards
  - Predictive Analysis

Introduction

- Project Background and Context:

SpaceX advertises Falcon 9 rocket launches at a cost of $62 million, significantly lower than other providers, who charge upwards of $165 million per launch. A substantial portion of these savings is attributed to SpaceX's ability to reuse the first stage of the Falcon 9 rocket. By accurately predicting whether the first stage will land successfully, we can better estimate the cost of a launch. This information is crucial for competing companies that may want to bid against SpaceX for rocket launches.

- Problems You Want to Find Answers To:

In this capstone project, we aim to predict the success of the Falcon 9 first stage landing using data from Falcon 9 rocket launches, as advertised on the SpaceX website.

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

Description of Data Collection Methods:.

- SpaceX API: The initial data collection was conducted using the SpaceX API, a RESTful API, by making GET requests. To facilitate this process, a series of helper functions were defined to streamline the extraction of information using identification numbers from the launch data. The data was then requested directly from the SpaceX API URL, ensuring accurate and structured retrieval.
- Data Consistency: To ensure consistency in the JSON results returned by the API, the SpaceX launch data was parsed and decoded into JSON format following the GET request. This data was subsequently converted into a Pandas DataFrame, providing a structured and easily analyzable format.
- Web Scraping: In addition to API data, historical Falcon 9 launch records were collected via web scraping from the 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 then converted into a Pandas DataFrame for further analysis.

Data Collection – SpaceX API

- Data was collected using the SpaceX API, a RESTful API, by making GET requests to retrieve launch information. The SpaceX launch data was then parsed and decoded into a JSON format. This JSON data was subsequently converted into a Pandas DataFrame for structured analysis and easier manipulation.
- Below is the GitHub URL.

https://github.com/takeshi298/IBM-Data-Sicence/blob/main/1.SpaceX-Complete%20the%20Data%20Collection%20API%20Lab.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:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex
```

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

```
response.status_code
```

```
200
```

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

```
# Use json_normalize meethod to convert the json result into a dataframe
respjson = response.json()
data = pd.json_normalize(respjson)
```

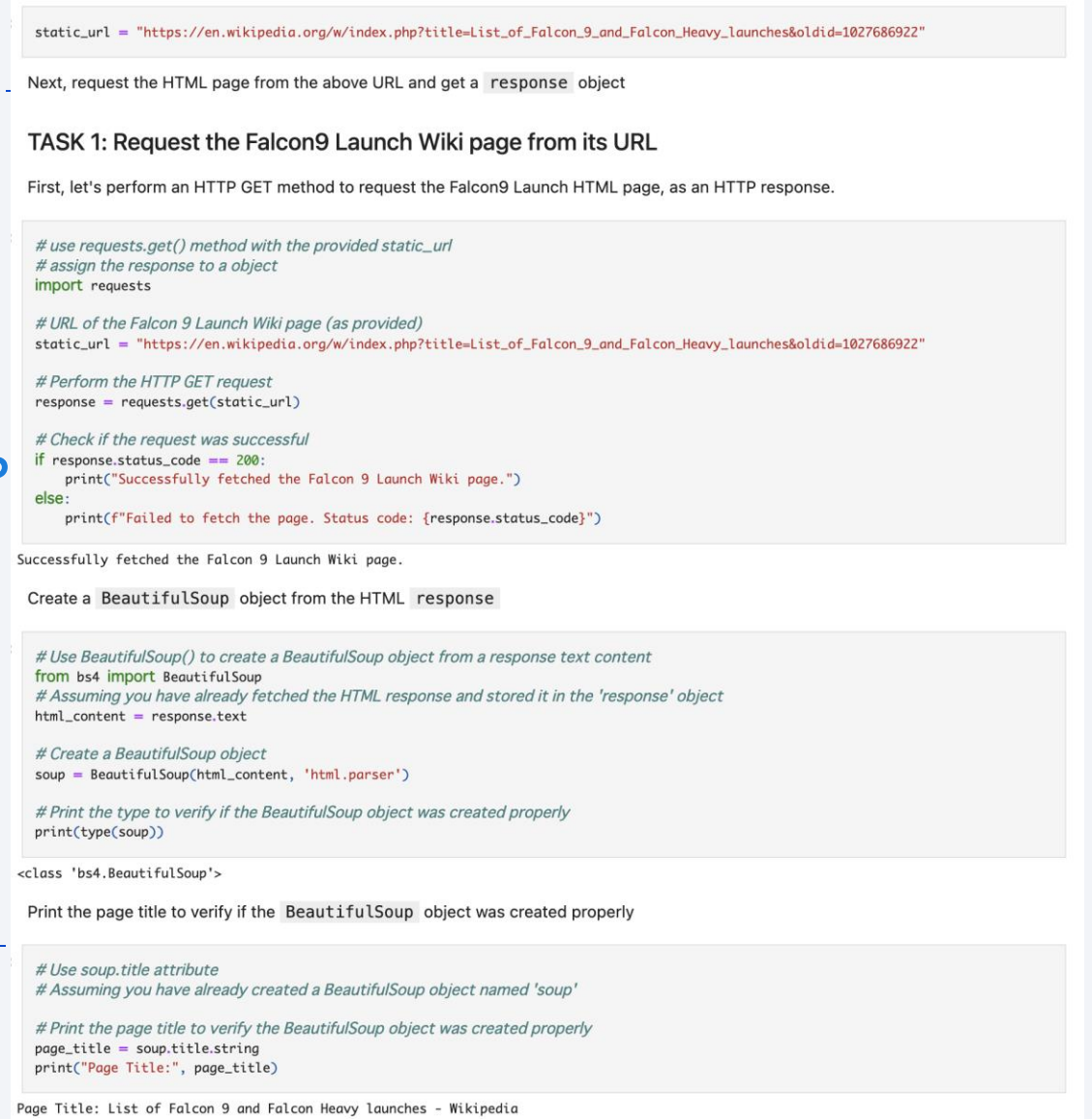
Using the dataframe `data` print the first 5 rows

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

Data Collection - Scraping

- Web scraping was utilized to collect historical launch records for Falcon 9 from a Wikipedia page. Using BeautifulSoup and the Requests library, the HTML table containing the Falcon 9 launch data was extracted. The parsed data was then converted into a Pandas DataFrame for further analysis.
- Below is the Github URL of the completed web scraping.

https://github.com/takeshi298/IBM-Data-Sicence/blob/main/2.SpaceX%20-%20Web%20scraping%20Falcon%209%20and%20Falcon%20Heavy%20Launches%20Records%20from%20Wikipedia.ipynb



```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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

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.

```
# use requests.get() method with the provided static_url
# assign the response to a object
import requests

# URL of the Falcon 9 Launch Wiki page (as provided)
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

# Perform the HTTP GET request
response = requests.get(static_url)

# Check if the request was successful
if response.status_code == 200:
    print("Successfully fetched the Falcon 9 Launch Wiki page.")
else:
    print(f"Failed to fetch the page. Status code: {response.status_code}")
```

```
Successfully fetched the Falcon 9 Launch Wiki page.
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
from bs4 import BeautifulSoup
# Assuming you have already fetched the HTML response and stored it in the 'response' object
html_content = response.text

# Create a BeautifulSoup object
soup = BeautifulSoup(html_content, 'html.parser')

# Print the type to verify if the BeautifulSoup object was created properly
print(type(soup))
```

```
<class 'bs4.BeautifulSoup'>
```

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

```
# Use soup.title attribute
# Assuming you have already created a BeautifulSoup object named 'soup'

# Print the page title to verify the BeautifulSoup object was created properly
page_title = soup.title.string
print("Page Title:", page_title)
```

```
Page Title: List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

Data Wrangling

- After creating a Pandas DataFrame from the collected data, the dataset was filtered using the ‘BoosterVersion’ column to retain only the Falcon 9 launches. Missing data in the ‘LandingPad’ and ‘PayloadMass’ columns were addressed; specifically, the missing values in the ‘PayloadMass’ column were imputed with the column's mean value.
- Additionally, Exploratory Data Analysis (EDA) was performed to uncover patterns within the data and to identify the appropriate labels for training supervised machine learning models.

Below is the GitHub URL of the completed data wrangling .

https://github.com/takeshi298/IBM-Data-Sicence/blob/main/3.SpaceX%20-%20Data%20wrangling.ipynb

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`:

```
# 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()
```

```
1    60
0    30
Name: Class, 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

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

	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, focusing on the following tasks:

- **Exploratory Data Analysis (EDA):** Conducted in-depth analysis to uncover patterns and relationships within the data.
- **Data Preparation and Feature Engineering:** Prepared the data for modeling by creating new features and refining existing ones.
- **Visualization Techniques:**
  - Utilized scatter plots to visualize relationships between key variables such as Flight Number and Launch Site, Payload and Launch Site, Flight Number and Orbit Type, and Payload and Orbit Type.
  - Employed bar charts to illustrate the success rates across different orbit types.
  - Created line plots to track and visualize the yearly trend of launch successes.

Below is the GitHub URL of the completed EDA with data visualization.

https://github.com/takeshi298/IBM-Data-Sicence/blob/main/5.SpaceX%20-%20EDA%20with%20Visualization%20Lab.ipynb

EDA with SQL

The following SQL queries were performed for EDA :

Task 1

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

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

Task 2

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

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

Task 3

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

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

Task 4

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

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

EDA with SQL

The following SQL queries were performed for EDA :

Task 5

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

Hint:Use min function

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

Task 6

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

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

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT(*) as Count FROM SPACEXTABLE GROUP BY "Mission_Outcome";
```

Below is the GitHub URL.

https://github.com/takeshi298/IBM-Data-Sicence/blob/main/4.SpaceX%20-%20Complete%20the%20EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

Created an interactive Folium map to visualize all the launch sites. The map includes various objects such as markers, circles, and lines to indicate the success or failure of launches at each site. Additionally, a launch outcome dataset was created, categorizing the results as either a failure (0) or a success (1).

Below is the GitHub URL.

https://github.com/takeshi298/IBM-Data-Sicence/blob/main/6.SpaceX%20-%20Interactive%20Visual%20Analytics%20withFolium%20lab.ipynb

Build a Dashboard with Plotly Dash

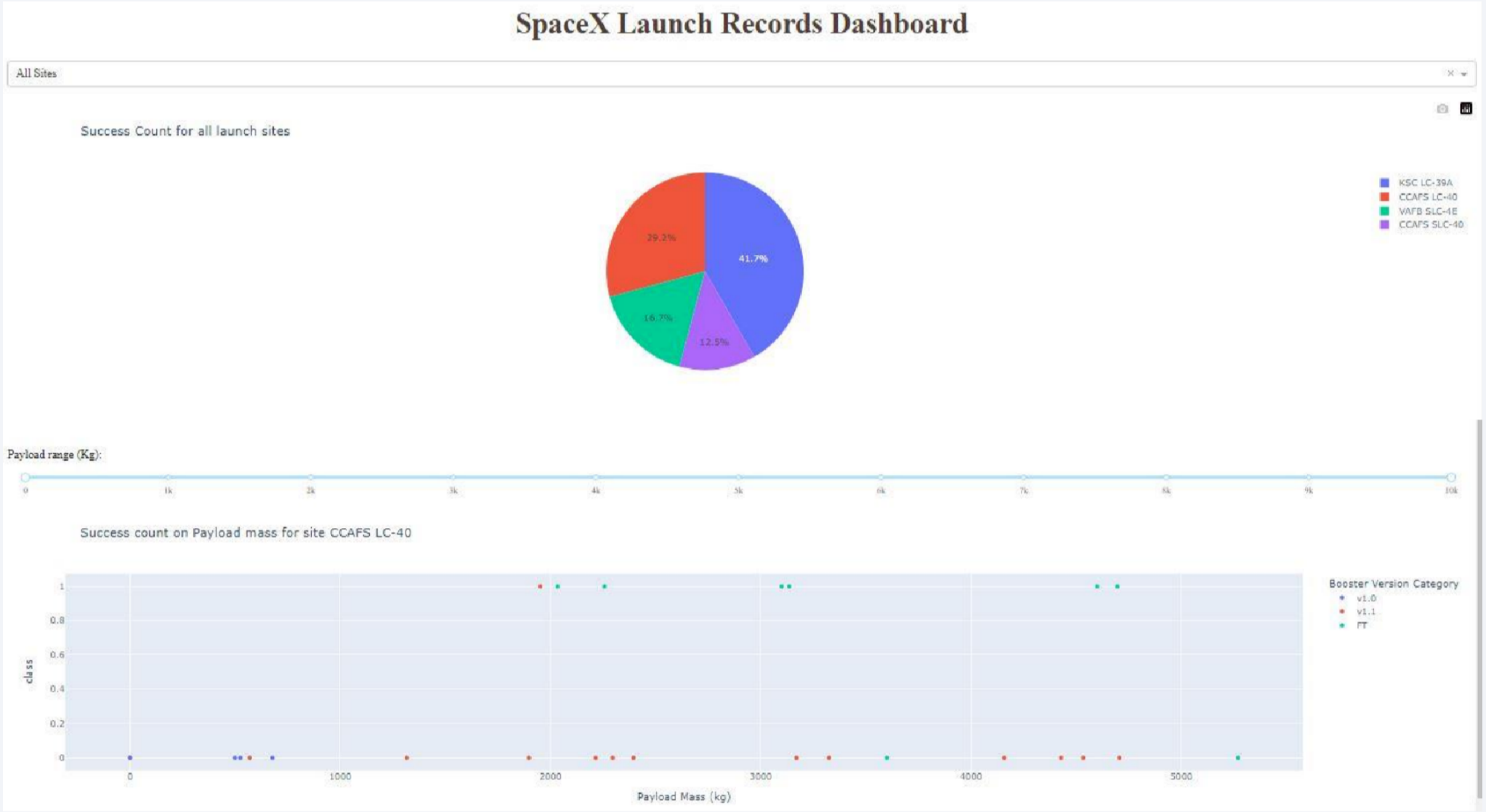
Developed an interactive dashboard application using Plotly Dash by implementing the following features:

- **Launch Site Dropdown Input:** Added a dropdown menu for selecting the launch site, allowing users to filter the data based on the chosen site.
- **Success Pie Chart:** Implemented a callback function to dynamically render a success pie chart based on the selected launch site from the dropdown.
- **Payload Range Slider:** Integrated a range slider to enable users to select and filter payload ranges interactively.
- **Success-Payload Scatter Plot:** Added a callback function to render a scatter plot, visualizing the relationship between payload mass and launch success based on the selected payload range.

Below is the GitHub URL.

https://github.com/takeshi298/IBM-Data-Sicence/blob/main/7.SpaceX%20-%20%20Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dash.ipynb.py

SpaceX Dash App



Predictive Analysis (Classification)

Summary of Building, Evaluating, and Optimizing the Best Performing Classification Model:

- After loading the data into a Pandas DataFrame, I began with Exploratory Data Analysis (EDA) to identify and prepare the training labels:
- **Label Preparation:** Created a NumPy array from the 'Class' column by applying the to_numpy() method, and assigned it to the variable Y as the target outcome variable.
- **Feature Standardization:** Standardized the feature dataset X using the StandardScaler() function from Scikit-learn's preprocessing module to ensure consistent scaling across all features.
- **Data Splitting:** Split the data into training and testing sets using the train_test_split function from Scikit-learn's model_selection module, setting the test_size parameter to 0.2 and the random_state to 2 to maintain consistency in the results.

Predictive Analysis (Classification)

To identify the best-performing machine learning model among SVM, Classification Trees, K-Nearest Neighbors, and Logistic Regression, the following approach was taken:

- **Model Initialization:** Created an object for each of the algorithms under consideration.
- **Hyperparameter Tuning with GridSearchCV:** For each model, a GridSearchCV object was instantiated with a defined set of parameters to search over. The GridSearchCV was configured with cv=10 to perform 10-fold cross-validation.
- **Model Training:** Fitted the training data into the GridSearchCV object for each model to find the optimal hyperparameters.
- **Evaluation:** After fitting, the best parameters for each model were identified using the best_params_ attribute of the GridSearchCV object. The accuracy on the validation data was obtained using the best_score_ attribute.
- **Testing and Visualization:** Calculated the accuracy of each model on the test data using the score method. Finally, confusion matrices were plotted for each model using the test data and their respective predictions to visually assess performance.

Predictive Analysis (Classification)

The table below presents the accuracy scores of the test data for each method, allowing for a comparison between SVM, Classification Trees, K-Nearest Neighbors, and Logistic Regression to determine which model performed best on the test data.

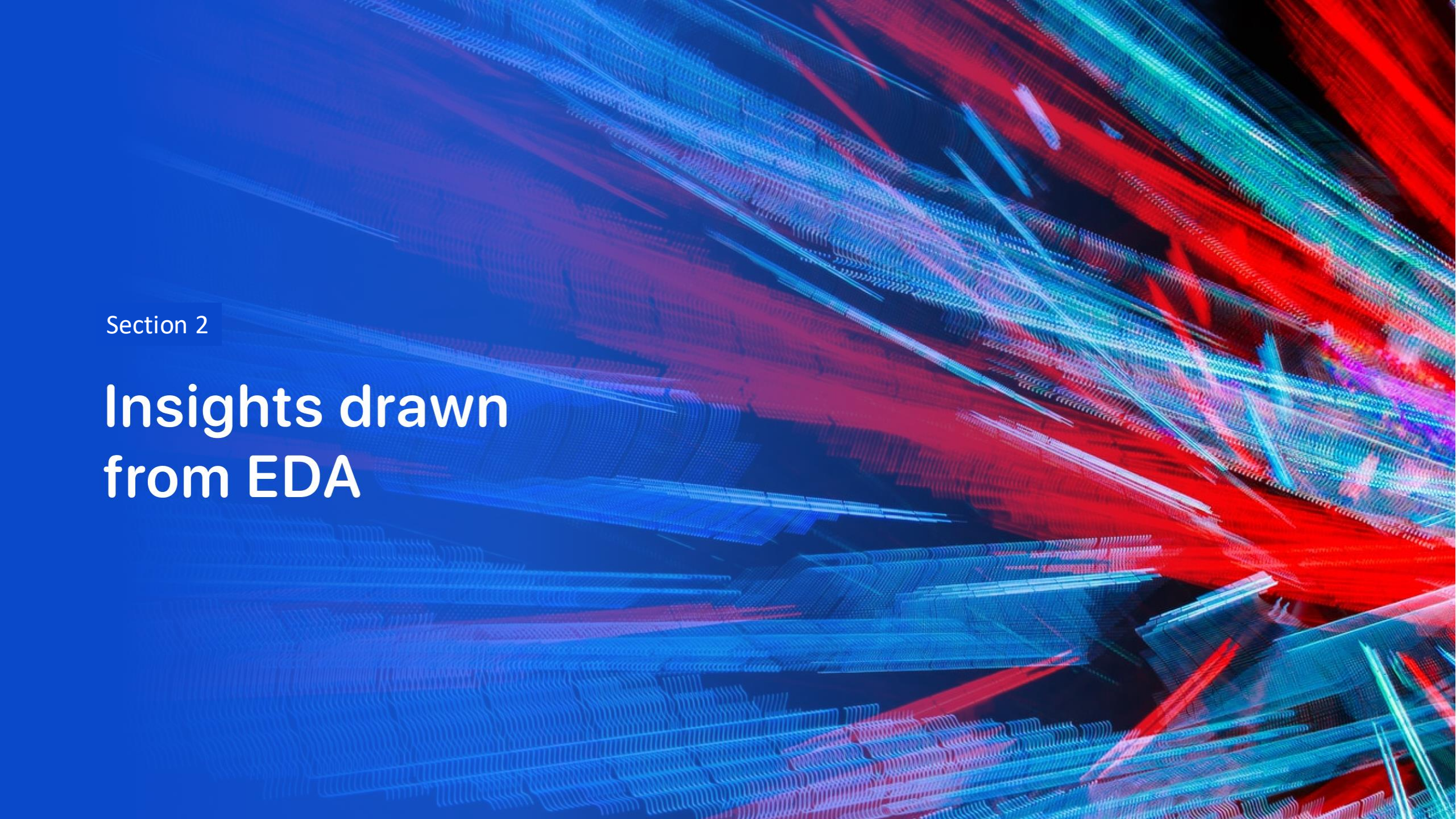
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

Below is the GitHub URL.

https://github.com/takeshi298/IBM-Data-Sicence/blob/main/8.SpaceX%20-%20Machine%20Learning%20Prediction.ipynb

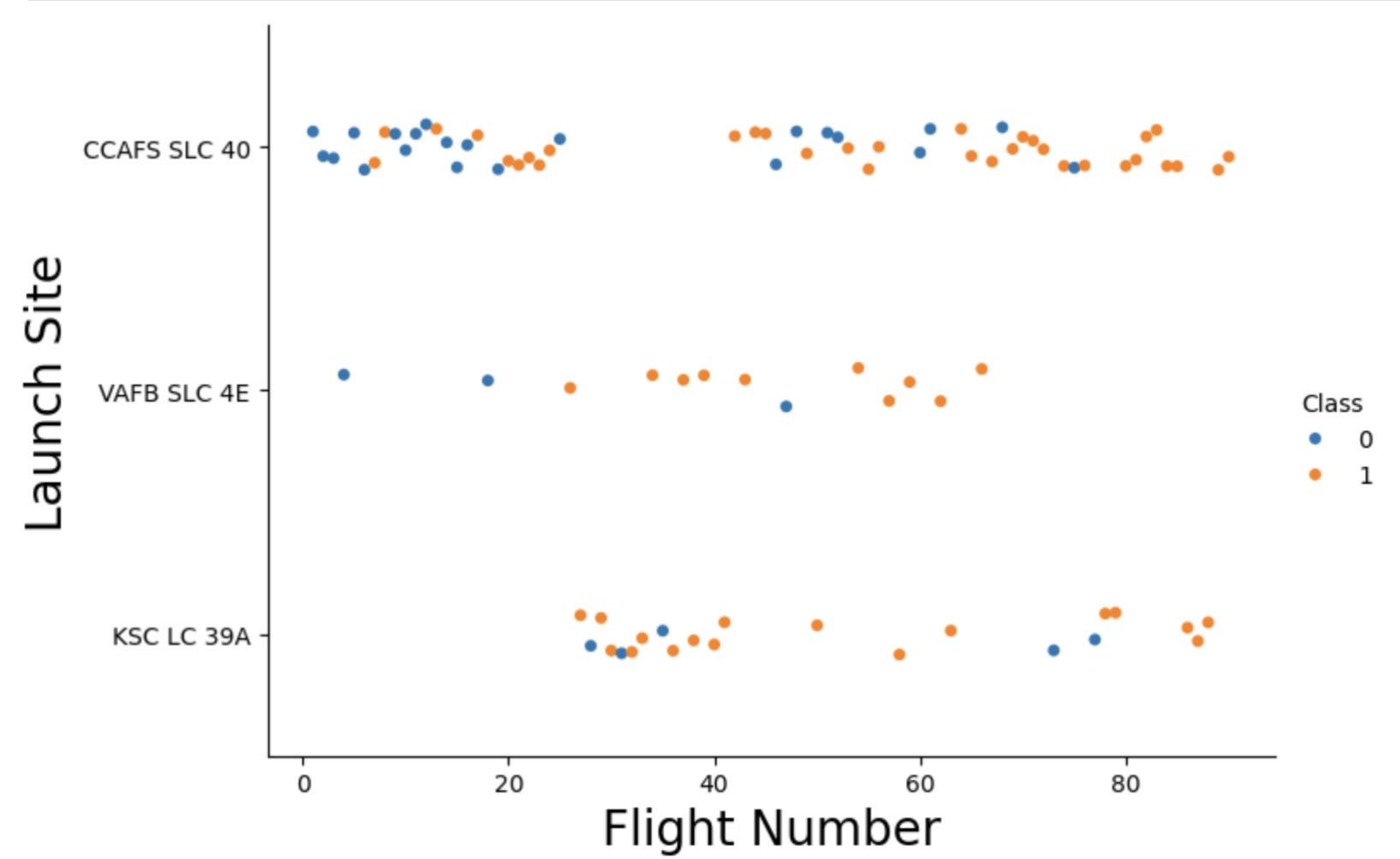
Results

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


Section 2
Insights drawn from EDA

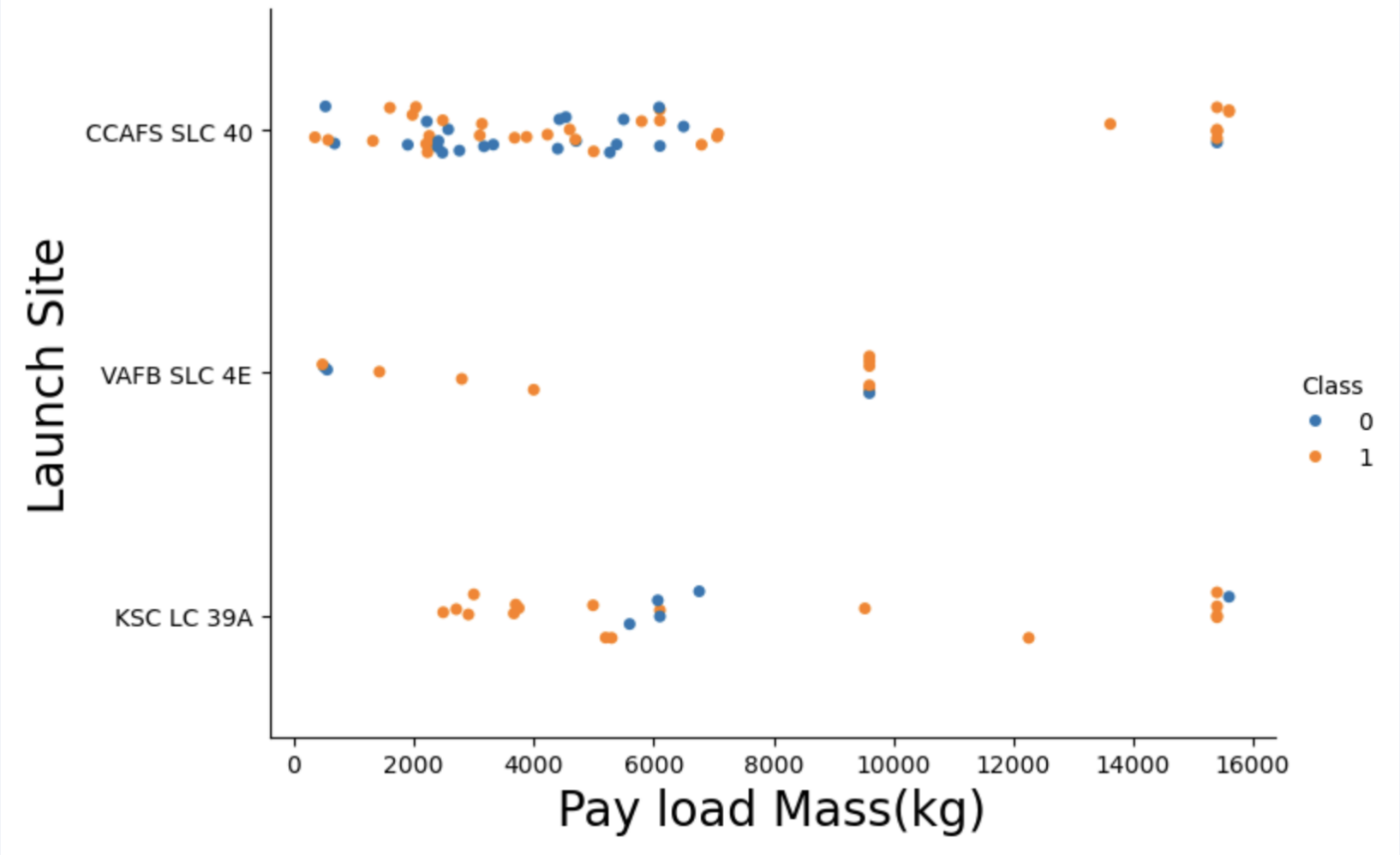
Flight Number vs. Launch Site

The relationship between Flight Number and Launch Site



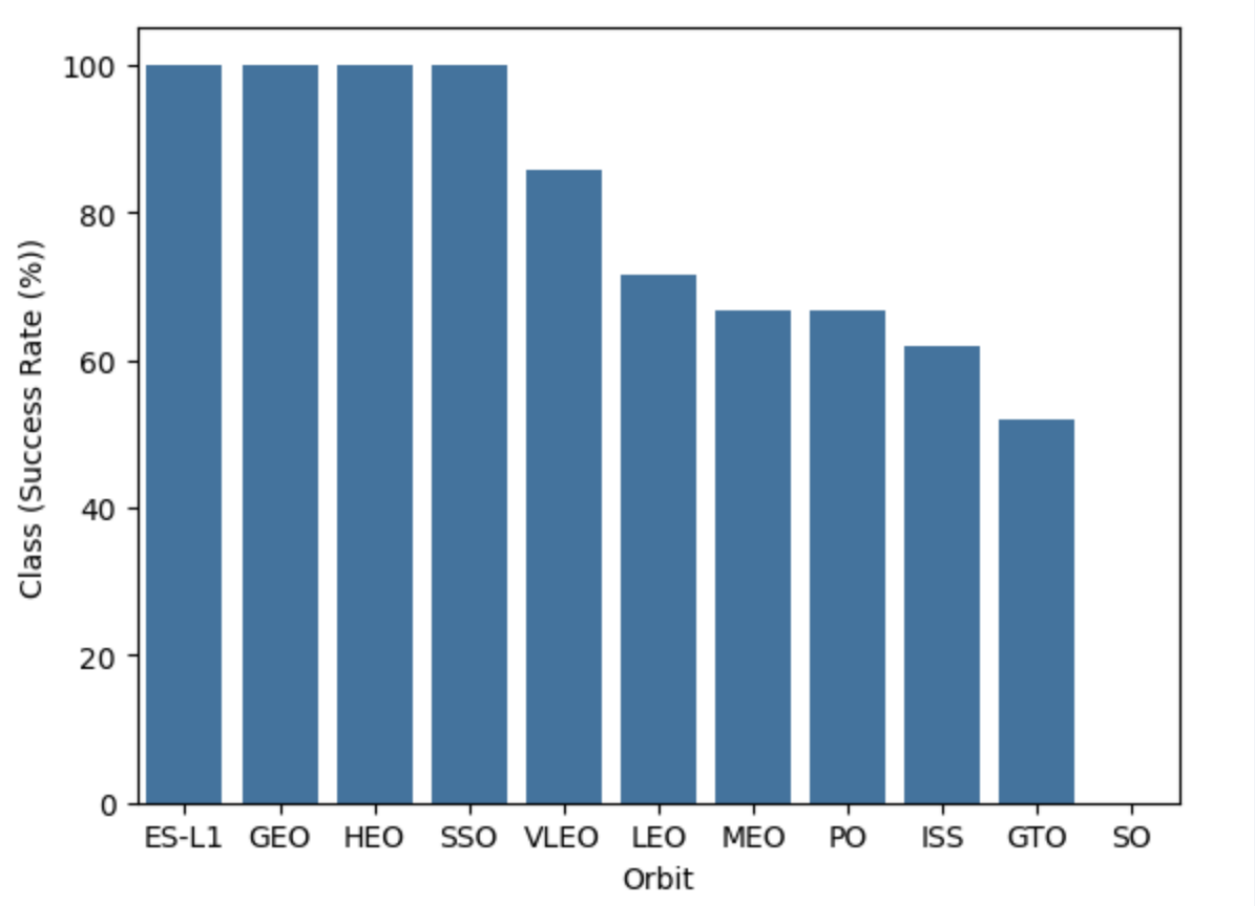
Payload vs. Launch Site

The relationship between Payload Mass and Launch Site



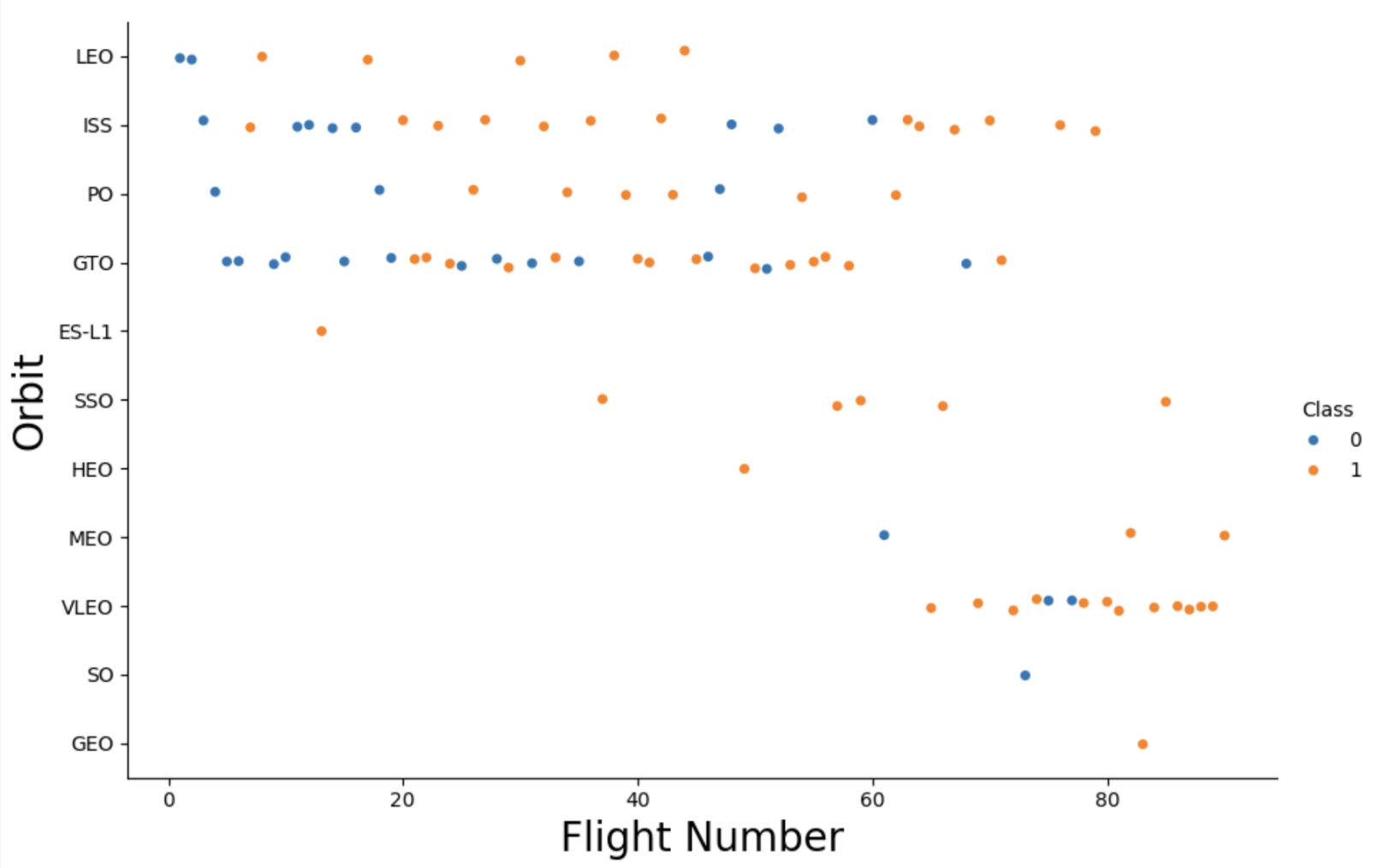
Success Rate vs. Orbit Type

The relationship between success rate of each orbit type



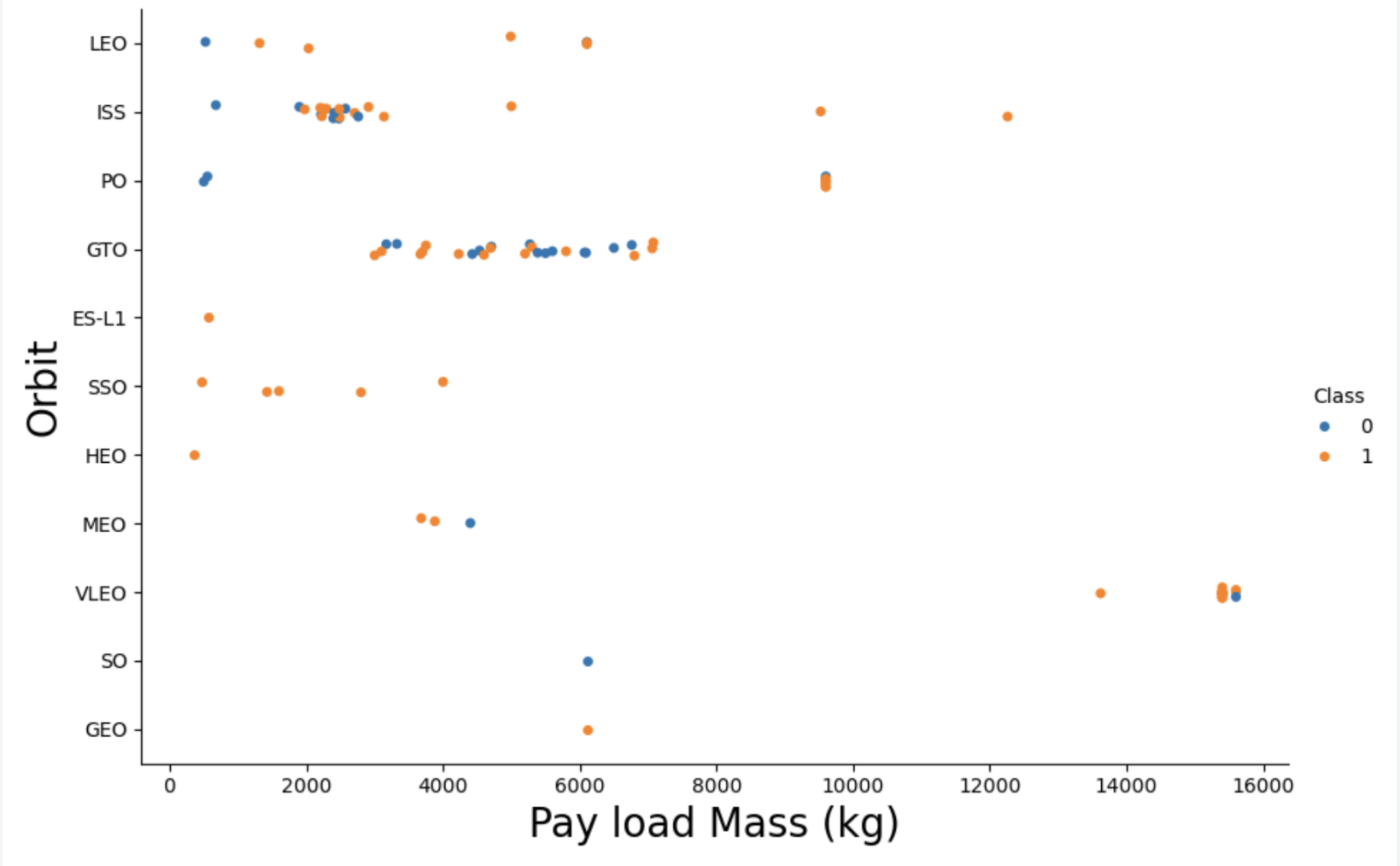
Flight Number vs. Orbit Type

The relationship between FlightNumber and Orbit type



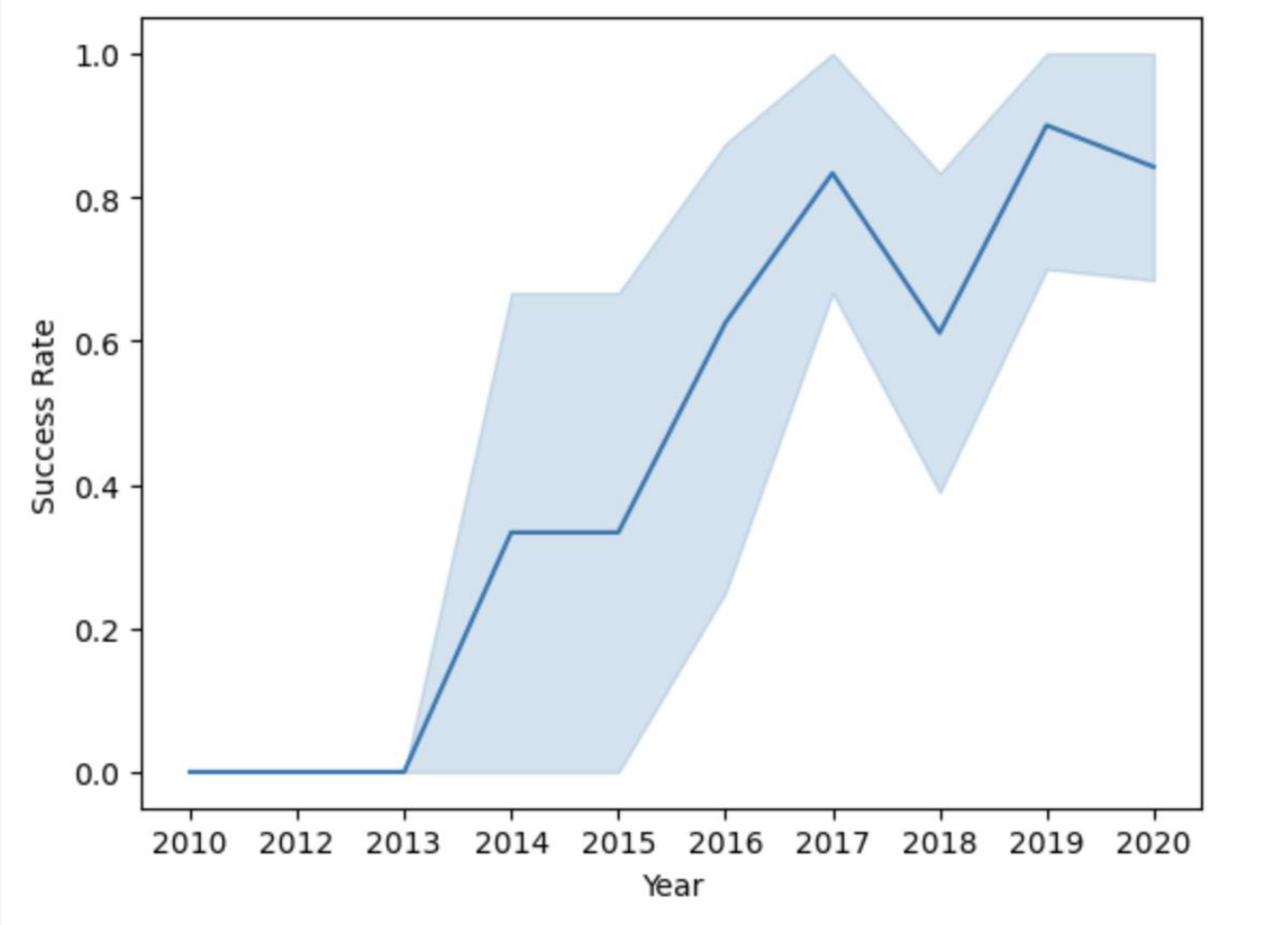
Payload vs. Orbit Type

The relationship between Payload Mass and Orbit type



Launch Success Yearly Trend

The launch success yearly trend



All Launch Site Names

Utilized the SELECT DISTINCT statement to retrieve unique launch sites from the LAUNCH_SITE column in the SPACEXTBL table.

Task 1

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

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

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

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

Launch Site Names Begin with 'CCA'

Employed the LIKE command with the % wildcard in the WHERE clause to select and display all records where the launch site names begin with the string 'CCA'.

Task 2

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

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

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (p
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 (p
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

Total Payload Mass

Applied the SUM() function to calculate and display the total payload mass in the PAYLOAD_MASS_KG column for the customer 'NASA (CRS)'.

Task 3

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

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

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

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

Utilized the AVG() function to calculate and display the average payload mass carried by the booster version F9 v1.1.

Task 4

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

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

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

Avg_Payload_Mass
2928.4

First Successful Ground Landing Date

Applied the MIN() function to retrieve and display the earliest date of the first successful landing on a ground pad with the outcome '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") as First_Successful_Landing FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)';
```

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

First_Successful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Utilized the SELECT statement to return and list the unique names of boosters with payloads between 4000 and 6000, and with a landing outcome of 'Success (drone ship)'.

Task 6

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

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

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

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

Total Number of Successful and Failure Mission Outcomes

Applied the COUNT() function in conjunction with the GROUP BY statement to return the total number of mission outcomes.

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT(*) as Count FROM SPACEXTABLE GROUP BY "Mission_Outcome";
```

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

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

Boosters Carried Maximum Payload

Used a subquery to retrieve the maximum payload and passed it to list all boosters that have 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("PAYLOAD_MASS__KG_") FROM SPACEXTBL);
```

```
 * 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

Utilized the SUBSTR() function in the SELECT statement to extract the month and year from the Date column where SUBSTR(Date, 0, 5) = '2015' for the year, and the Landing_outcome was 'Failure (drone ship)', returning the records that match these criteria.

Task 9

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

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

```
%sql SELECT substr("Date", 6, 2) as Month, "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE "Landing_Outcome"
```

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

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

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

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

Task 10

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

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

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

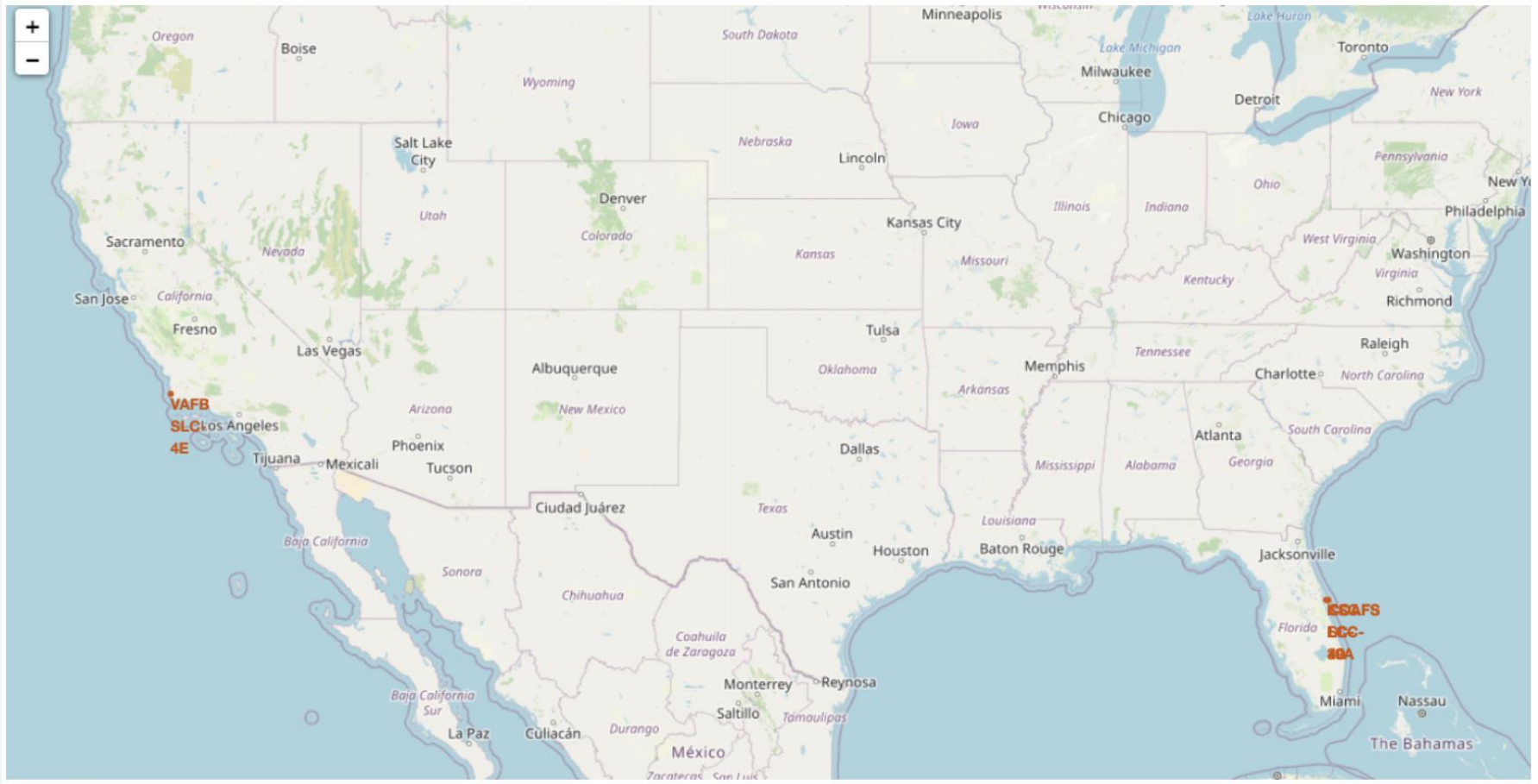
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

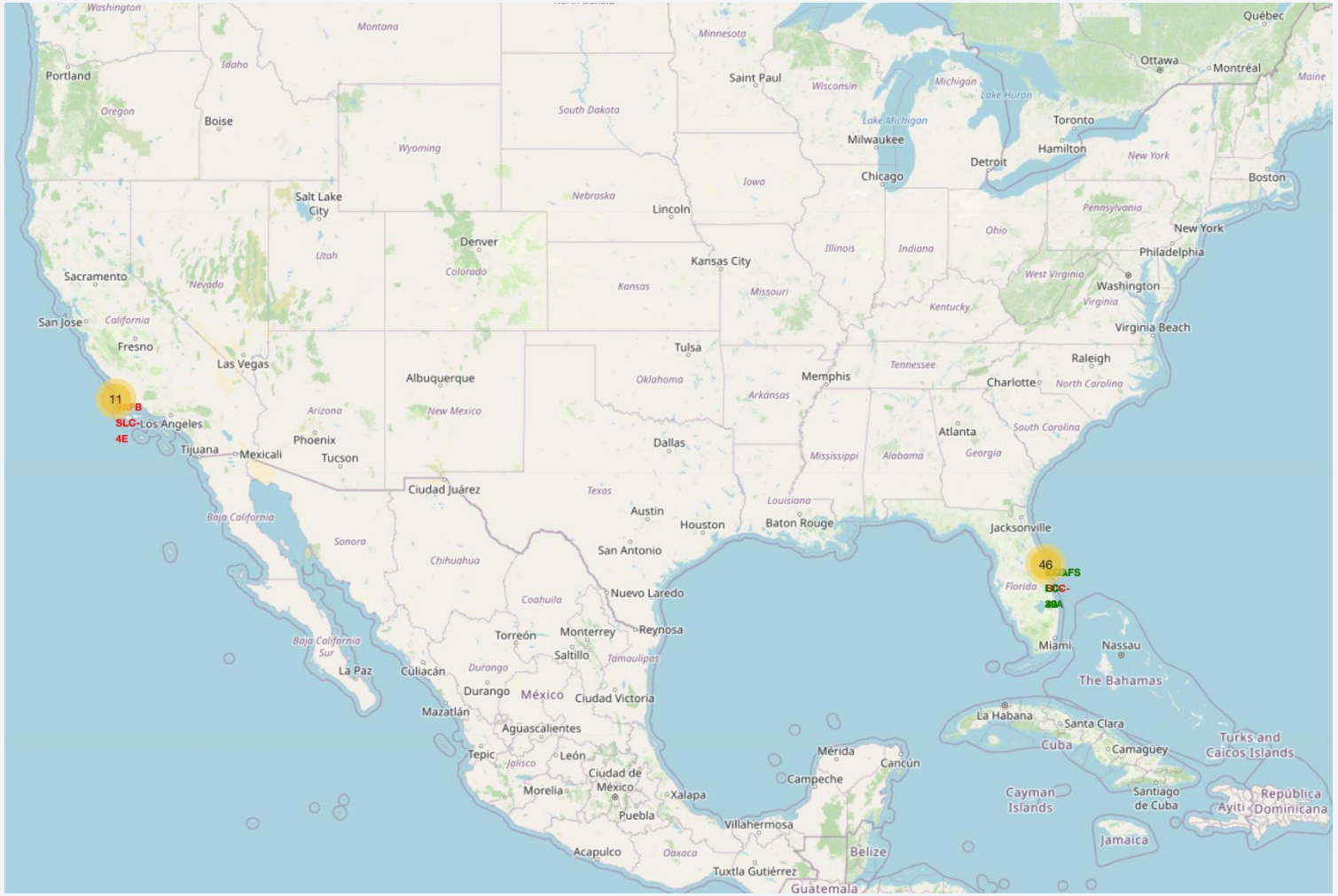
<Markers of all launch sites on map>

All launch sites are located near the Equator, positioned southward on the U.S. map. Additionally, these launch sites are in close proximity to the coast.



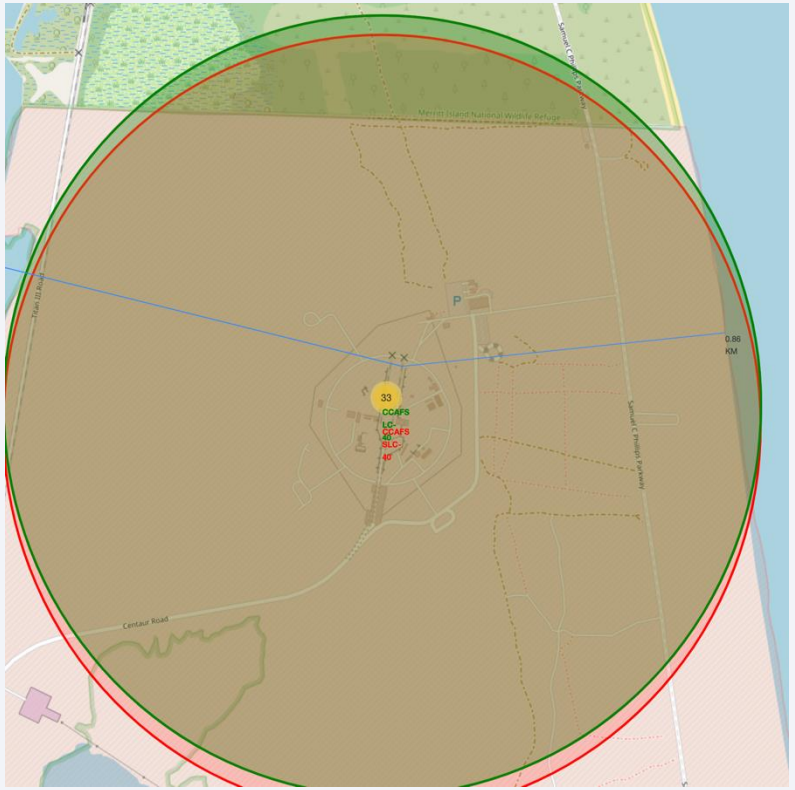
<Launch outcomes for each site on the map with color markers>

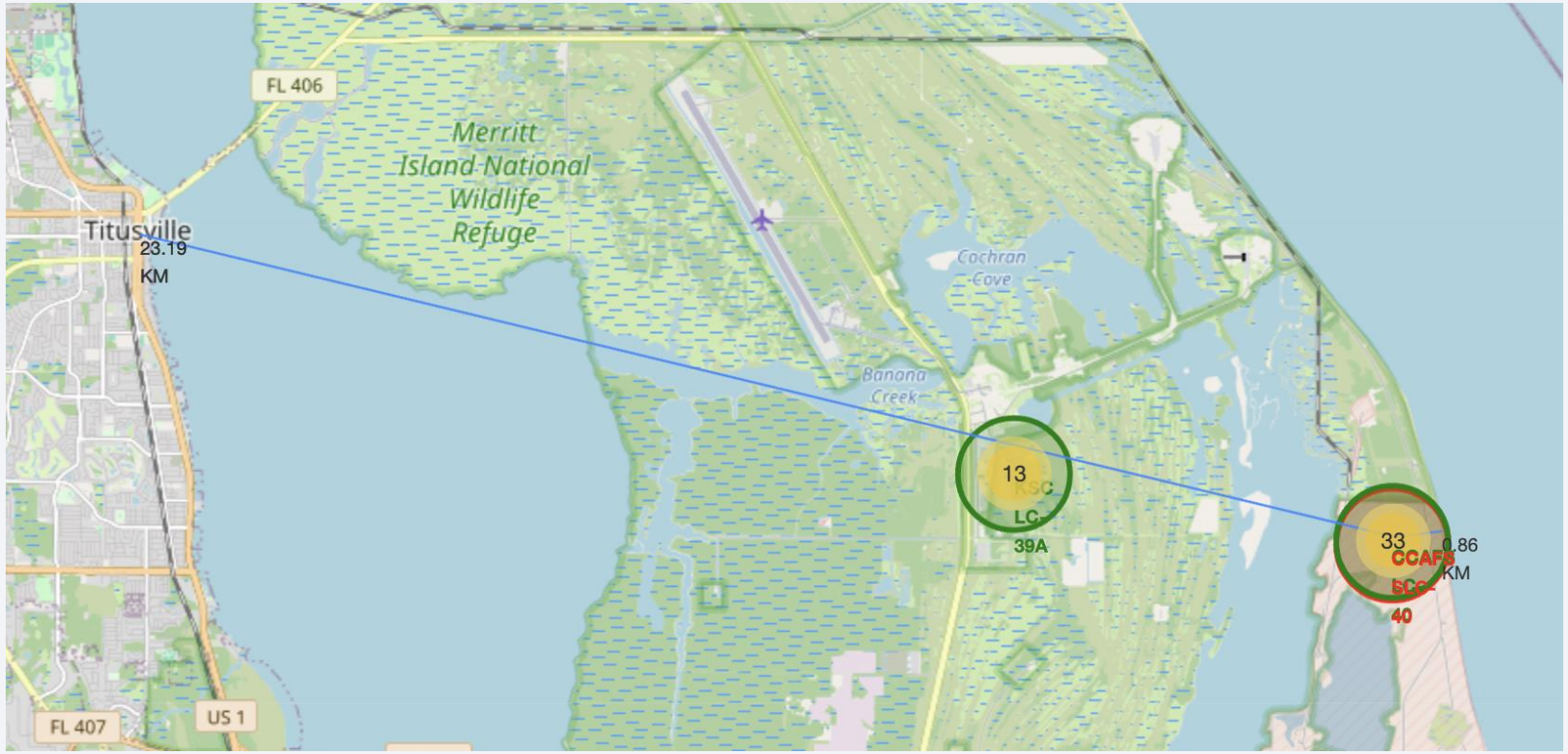
The color-labeled markers in the marker clusters allow for easy identification of launch sites with relatively high success rates.

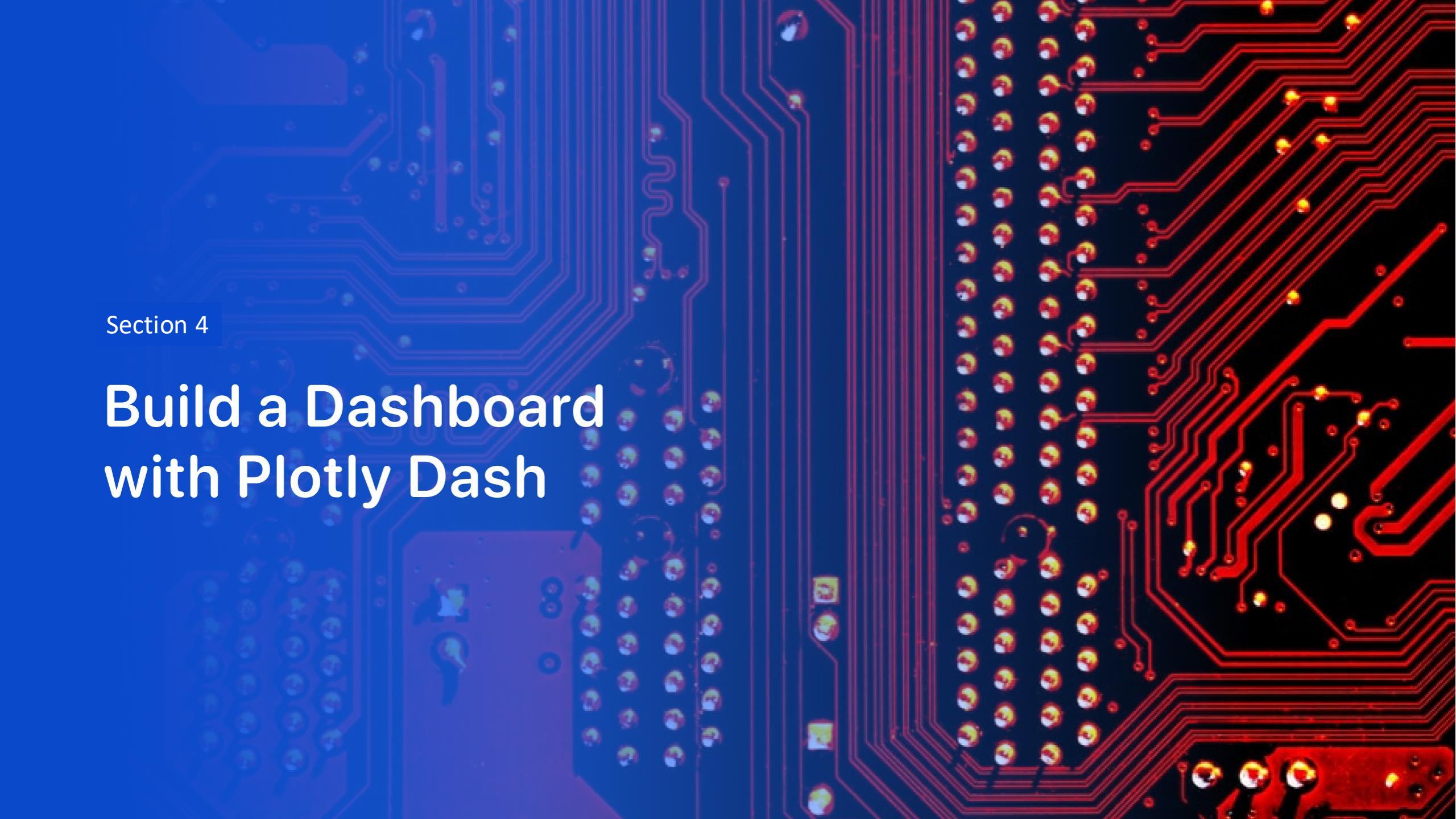


<Distances between a launch site to its proximites>

The CCAFS SLC-40 launch site is located 0.86 km from the coastline and 23.19 km from the nearest highway, Washington Avenue.

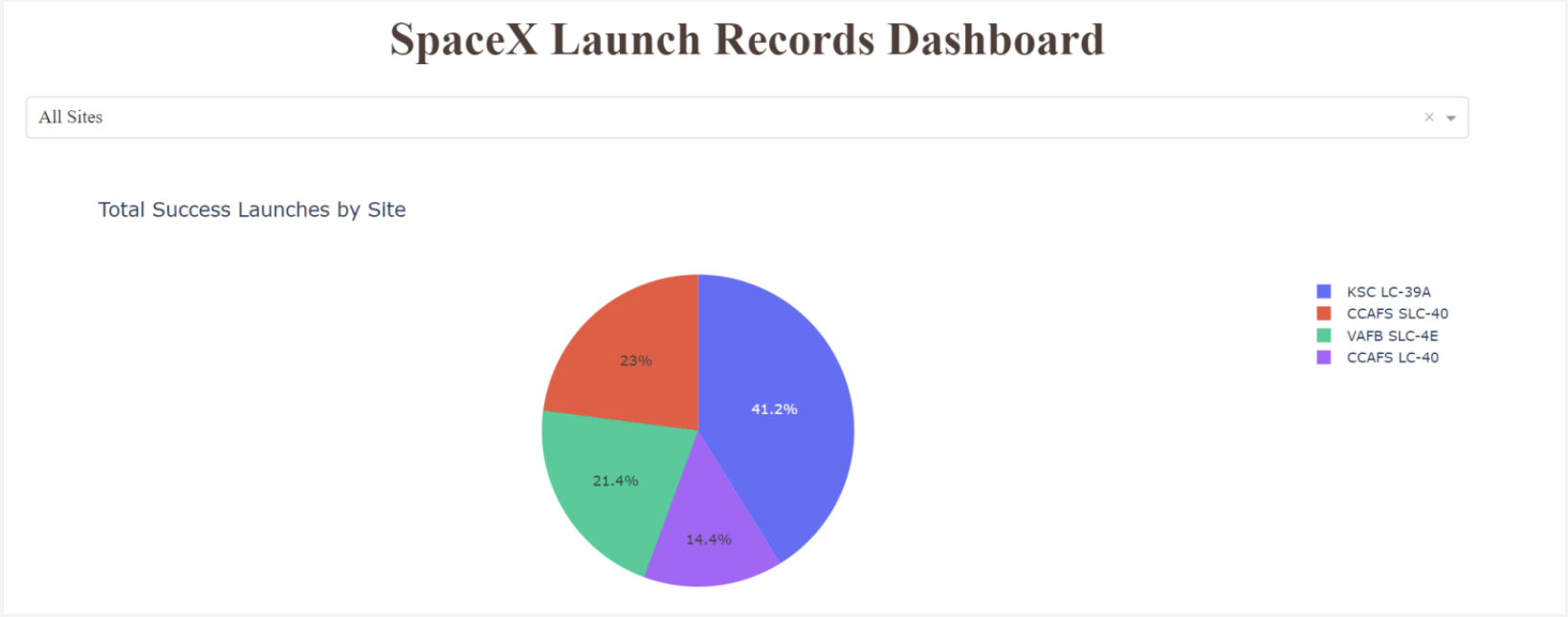





Section 4
Build a Dashboard with Plotly Dash

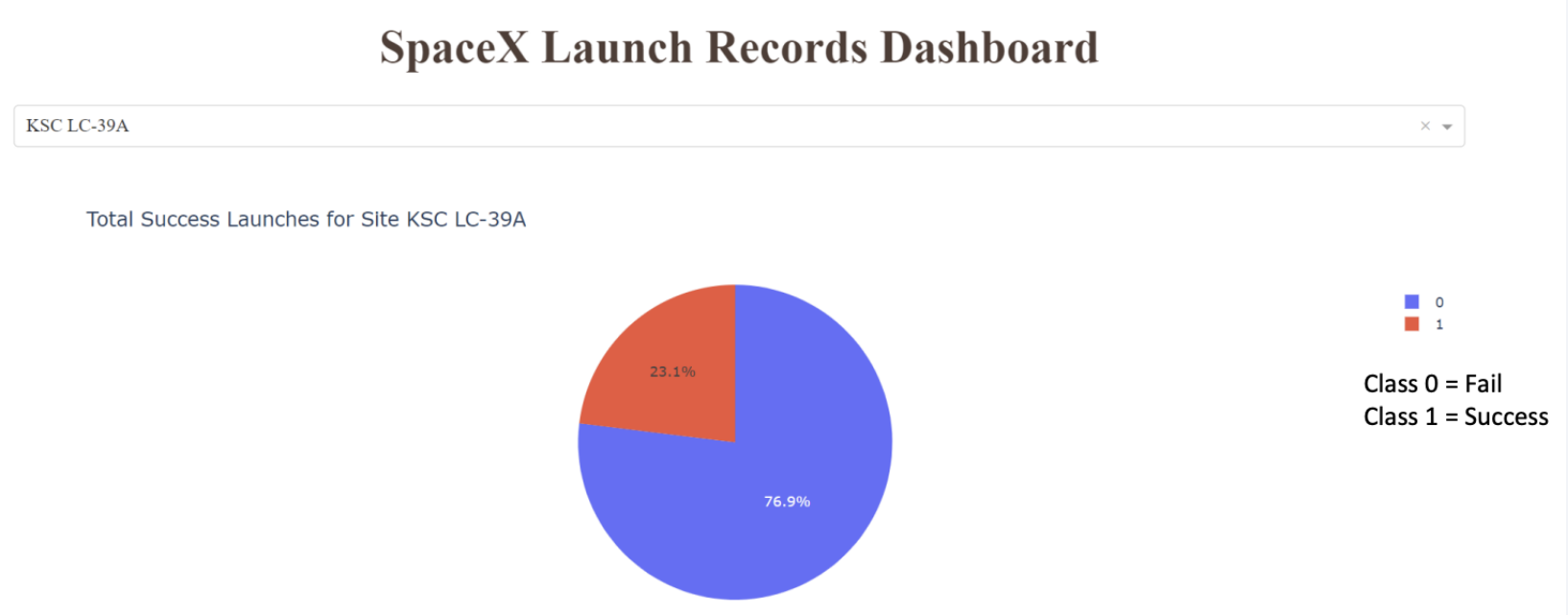
<Pie-Chart for launch success count for all sites>

Launch site KSC LC-39A boasts the highest launch success rate at 41.2%, followed by CCAFS LC-40 at 14.4%, VAFB SLC-4E at 21.4%, and finally, CCAFS SLC-23 with a success rate of 13%.



<Pie chart for the launch site with 2nd highest launch success ratio>

Launch site CCAFS LC-39had the second-highest success rate, with 76.9% of launches succeeding and 23.1% resulting in failure.



<Payload vs. Launch Outcome scatter plot for all sites>

Payloads between 2,000 kg and 5,000 kg achieve the highest success rate, with 1 indicating a successful outcome and 0 indicating an unsuccessful outcome.

Section 5

Predictive Analysis (Classification)

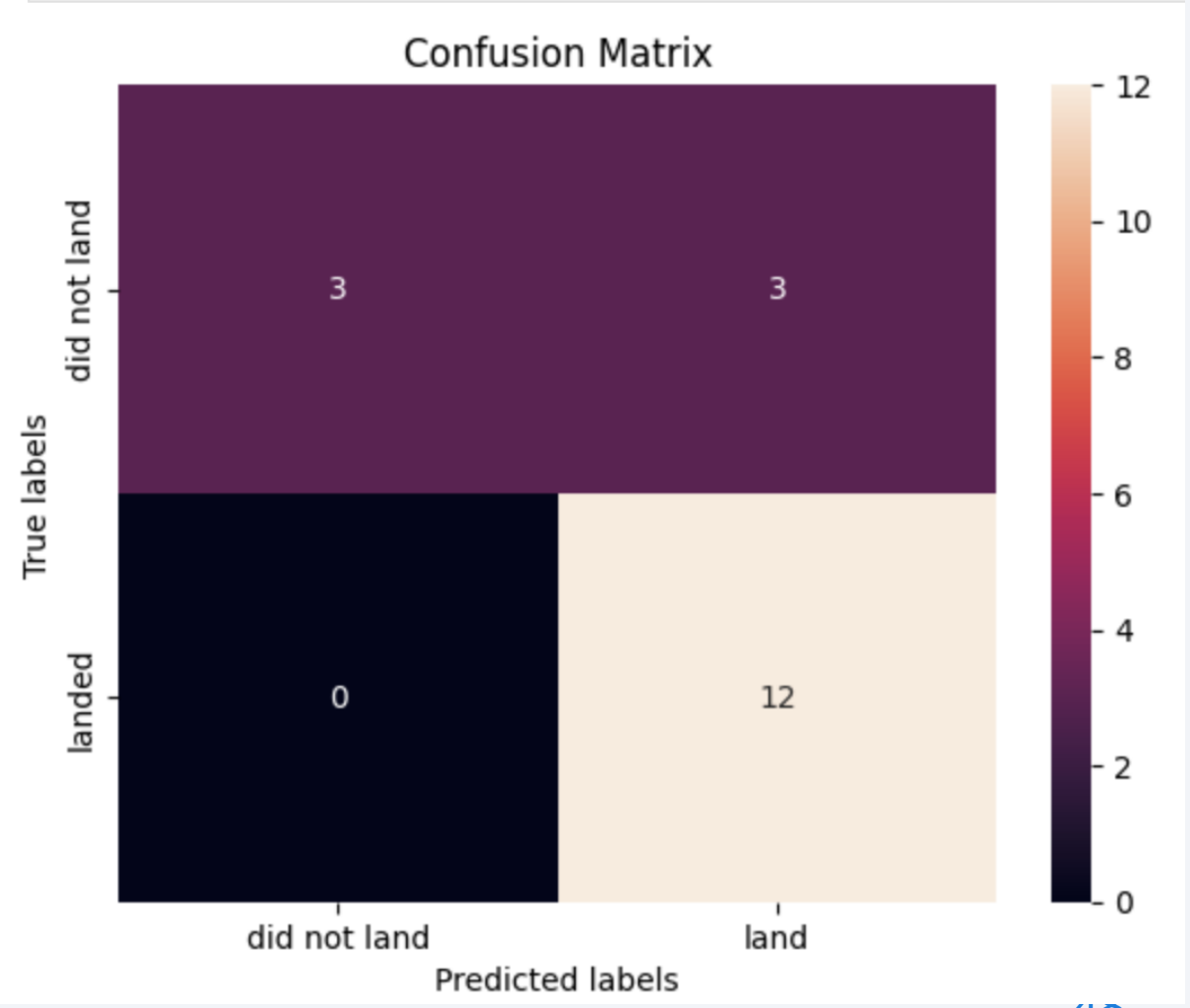
Classification Accuracy

All the methods perform equally on the test data. They all have the same accuracy of 0.83333 on the test data.

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

Confusion Matrix

All four classification models produced identical confusion matrices and were equally effective in distinguishing between the different classes. However, a significant issue across all models was the prevalence of false positives.



Conclusions

- **Varying Success Rates Across Launch Sites**: The analysis reveals that KSC LC-39A holds the highest launch success rate, followed by CCAFS LC-40, VAFB SLC-4E, and finally CCAFS SLC-40. This variation highlights the differing performance levels across SpaceX's launch facilities.
- **Correlation Between Flight Number and Success**: There is a clear trend of increasing success rates with a higher number of flights at each site, indicating that experience and iterative improvements contribute significantly to better outcomes.
- **Payload and Orbit Insights**: Payloads within the 2,000 kg to 5,000 kg range show the highest success rates, particularly in orbits like ES-L1, GEO, HEO, and SSO, which achieved a perfect 100% success rate. On the other hand, the SO orbit had the lowest success rate, highlighting the challenges associated with certain orbital paths.
- **Progress Over Time**: The success rate has seen a steady increase from 2013 through 2020, reflecting advancements in SpaceX's launch technology and operational efficiency. This trend underscores the company's growing reliability and capability in space missions.

Thank you!