IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Scarlet Ruiz
2024-01-09

Outline

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

Executive Summary

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

Introduction

- **Project background and context**

  SpaceX advertises the cost of Falcon 9 rocket launches on its website as 62 million dollars, while other providers charge upwards of 165 million dollars each. Much of the savings is due to SpaceX's ability to reuse the first stage of the rocket. By determining if the first stage will land successfully, we can estimate the cost of a launch. This information can be useful for other companies that want to compete with SpaceX for rocket launch bids.

- **Problems you want to find answers to**
- In this capstone project, we will predict whether the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches listed 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

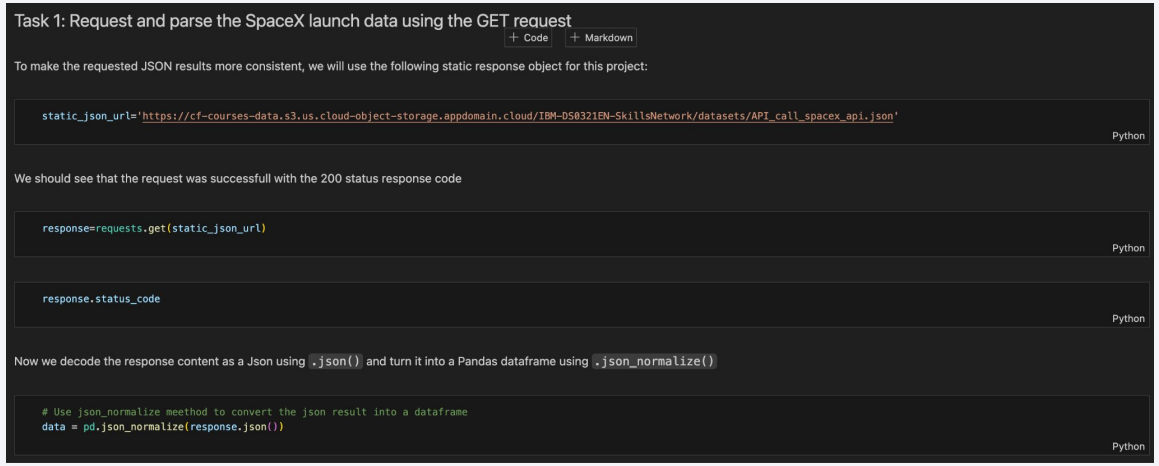
Data was first collected using the SpaceX API (a RESTful API) by making a GET request to the SpaceX API. This involved defining a series of helper functions to facilitate the use of the API for extracting information using identification numbers in the launch data and then requesting rocket launch data from the SpaceX API URL.

To make the requested JSON results more consistent, the SpaceX launch data was requested and parsed using the GET request. The response content was then decoded as a JSON result, which was subsequently converted into a Pandas DataFrame.

Additionally, web scraping was performed to collect Falcon 9 historical launch records from a Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches." The launch records are stored in HTML format. Using BeautifulSoup and Requests libraries, the Falcon 9 launch HTML table records were extracted from the Wikipedia page, parsed, and converted into a Pandas DataFrame.

Data Collection – SpaceX API

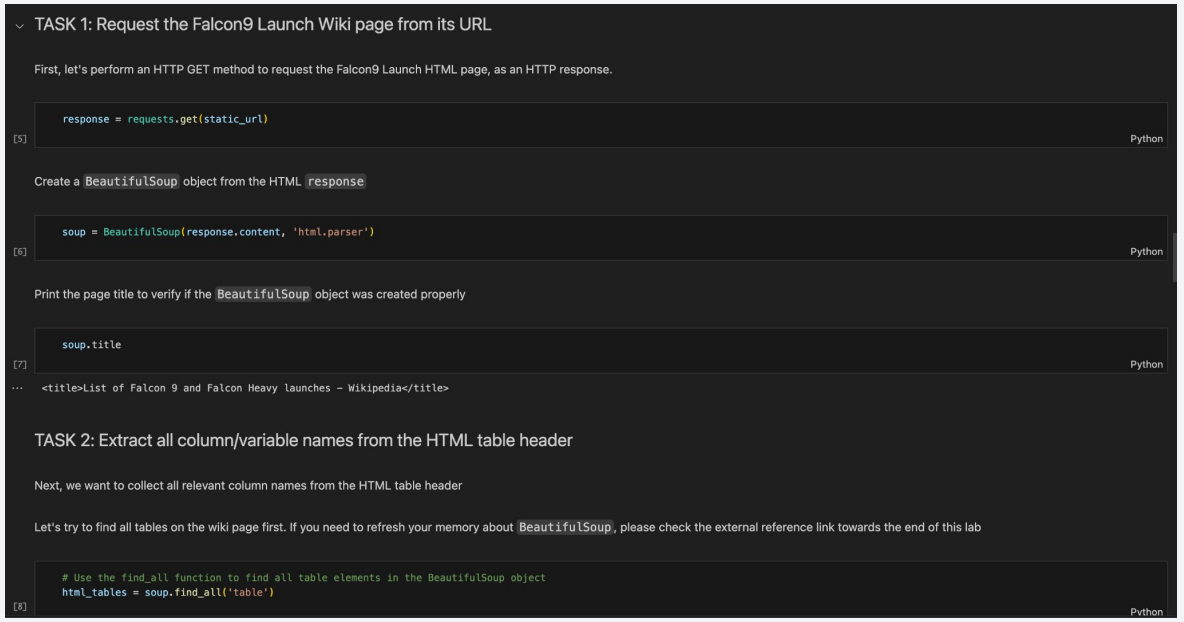
Data was collected using the SpaceX API (a RESTful API) by making a GET request to the SpaceX API. The SpaceX launch data was then requested and parsed using the GET request, and the response content was decoded as a JSON result, which was subsequently converted into a Pandas DataFrame.



SpaceX API calls notebook:
https://github.com/scar0510/Applied-Data-Science-Capstone/blob/main/1.jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

Performed web scraping to collect Falcon 9 historical launch records from Wikipedia using BeautifulSoup and Requests libraries. The Falcon 9 launch records were extracted from the HTML table on the Wikipedia page, and a DataFrame was created by parsing the launch HTML.



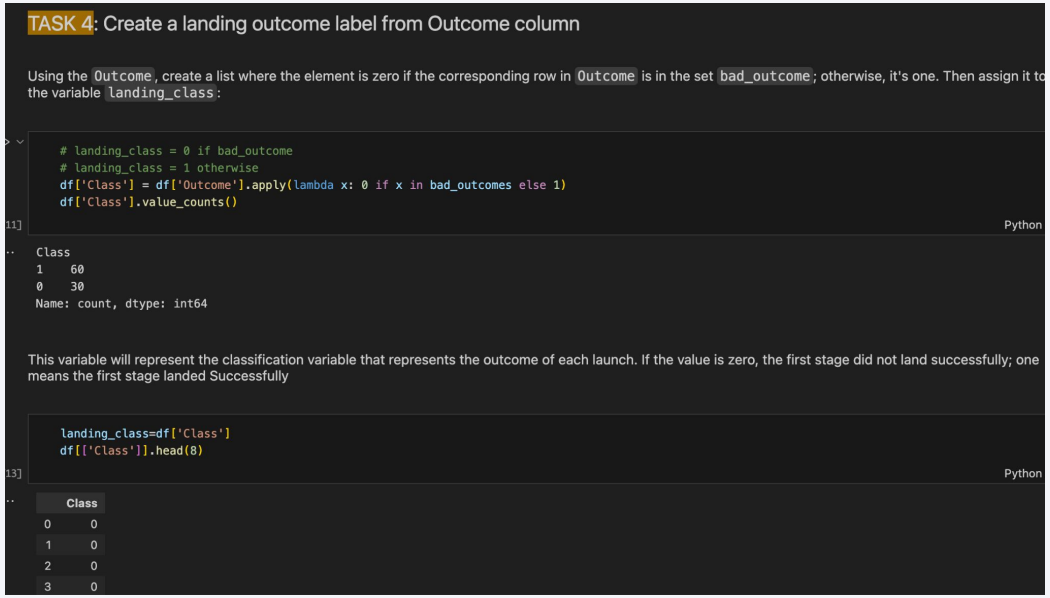
Web Scraping notebook:

https://github.com/scar0510/Applied-Data-Science-Capstone/blob/main/2.jupyter-labs-webscraping.ipynb

Data Wrangling

After obtaining and creating a Pandas DataFrame from the collected data, the data was filtered using the BoosterVersion column to retain only the Falcon 9 launches. Missing data values in the LandingPad and PayloadMass columns were handled. For the PayloadMass, missing data values were replaced using the mean value of the column

Also performed some Exploratory Data Analysis (EDA) to identify patterns in the data and determine the appropriate labels for training supervised models



Web Wrangling notebook:

https://github.com/scar0510/Applied-Data-Science-Capstone/blob/main/3.%20labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

Performed data analysis and feature engineering using Pandas and Matplotlib, which included:

- Exploratory Data Analysis
- Preparing Data and Feature Engineering

Used scatter plots to visualize the relationships between:

- Flight Number and Launch Site
- Payload and Launch Site
- Flight Number and Orbit type
- Payload and Orbit type

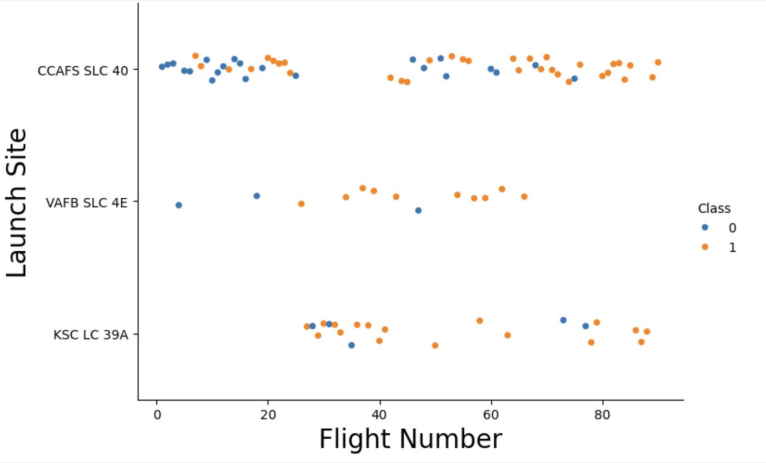
Used a bar chart to visualize the relationship between the success rate of each orbit type.

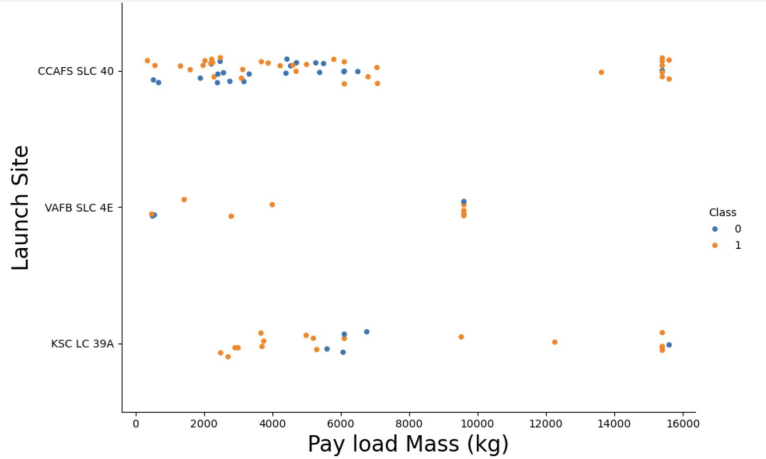
Line plot to Visualize the launch success yearly trend.

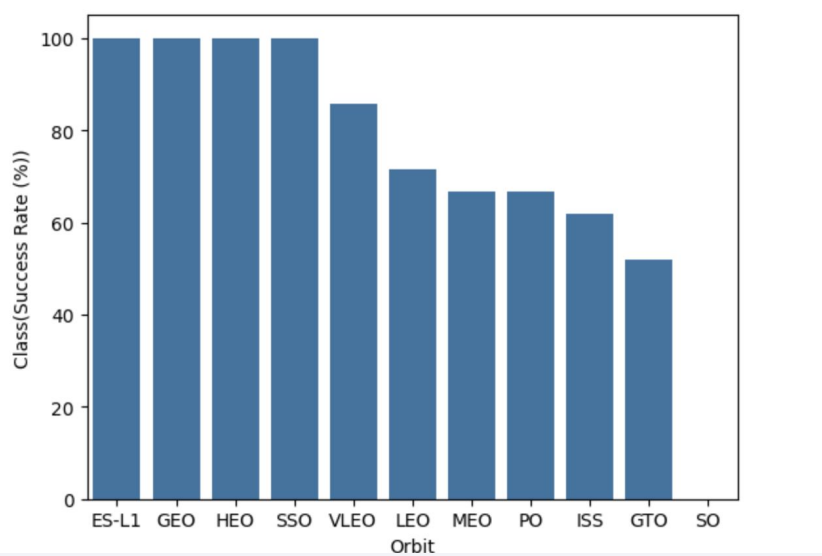
EDA Notebook:

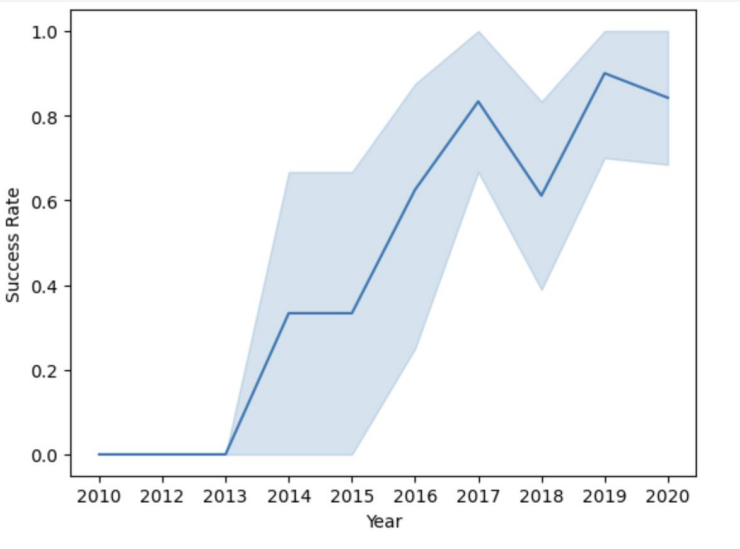
https://github.com/scar0510/Applied-Data-Science-Capstone/blob/main/5.%20edadataviz.ipynb

EDA with Data Visualization









EDA with SQL

The following SQL queries were performed for EDA

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

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

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

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';
```

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';
```

EDA with SQL

- List the date when the first successful landing outcome in ground pad was achieved

```
%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;
```

- List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

- 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);
```

EDA with SQL Notebook:

https://github.com/scar0510/Applied-Data-Science-Capstone/blob/main/4.jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Created a folium map to mark all the launch sites, and created map objects such as markers, circles, and lines to indicate the success or failure of launches for each launch site.
- Created a launch set outcomes (failure=0 or success=1).

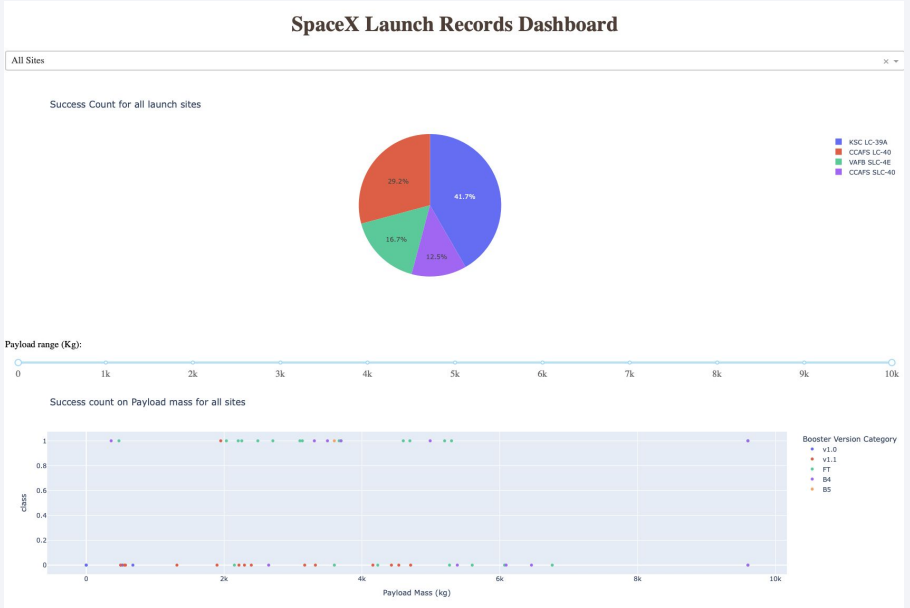
- Interactive Map Notebook:

  https://github.com/scar0510/Applied-Data-Science-Capstone/blob/main/6.%20lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

Built an interactive dashboard application with Plotly Dash by:

- Adding a Launch Site Drop-down Input Component
- Adding a callback function to render a success pie chart based on the selected site dropdown
- Adding a Range Slider to select payload
- Adding a callback function to render the success-payload-scatter-chart scatter plot



Interactive Dashboard code:

https://github.com/scar0510/Applied-Data-Science-Capstone/blob/main/7.%20Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dash%20-%20spacex_dash_app

Predictive Analysis (Classification)

Summary of how I built, evaluated, improved, and found the best performing classification model:

1. After loading the data as a Pandas DataFrame, I performed exploratory data analysis and determined the training labels by:
2. Creating a NumPy array from the column Class in the data, by applying the method to_numpy(), then assigned it to the variable Y as the outcome variable.
3. Standardizing the feature dataset (X) by transforming it using the preprocessing.StandardScaler() function from Sklearn.
4. Splitting the data into training and testing sets using the train_test_split function from sklearn.model_selection with the test_size parameter set to 0.2 and random_state set to 2.

Predictive Analysis (Classification)

Summary of how I built, evaluated, improved, and found the best performing classification model:

1. After loading the data as a Pandas DataFrame, I performed exploratory data analysis and determined the training labels by:
2. Creating a NumPy array from the column Class in the data, by applying the method to_numpy(), then assigned it to the variable Y as the outcome variable.
3. Standardizing the feature dataset (X) by transforming it using the preprocessing.StandardScaler() function from Sklearn.
4. Splitting the data into training and testing sets using the train_test_split function from sklearn.model_selection with the test_size parameter set to 0.2 and random_state set to 2.

Predictive Analysis (Classification)

In order to find the best machine learning model/method that performs best using the test data between SVM, Classification Trees, k-Nearest Neighbors, and Logistic Regression:

1. First, created an object for each of the algorithms, then created a GridSearchCV object and assigned them a set of parameters for each model.
2. For each of the models under evaluation, the GridSearchCV object was created with cv=10, then fit the training data into the GridSearch object for each to find the best hyperparameters.
3. After fitting the training set, output the GridSearchCV object for each of the models, then displayed the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_.
4. Finally, used the method score to calculate the accuracy on the test data for each model and plotted a confusion matrix for each using the test and predicted outcomes.

Predictive Analysis (Classification)

Predictive analysis notebook:

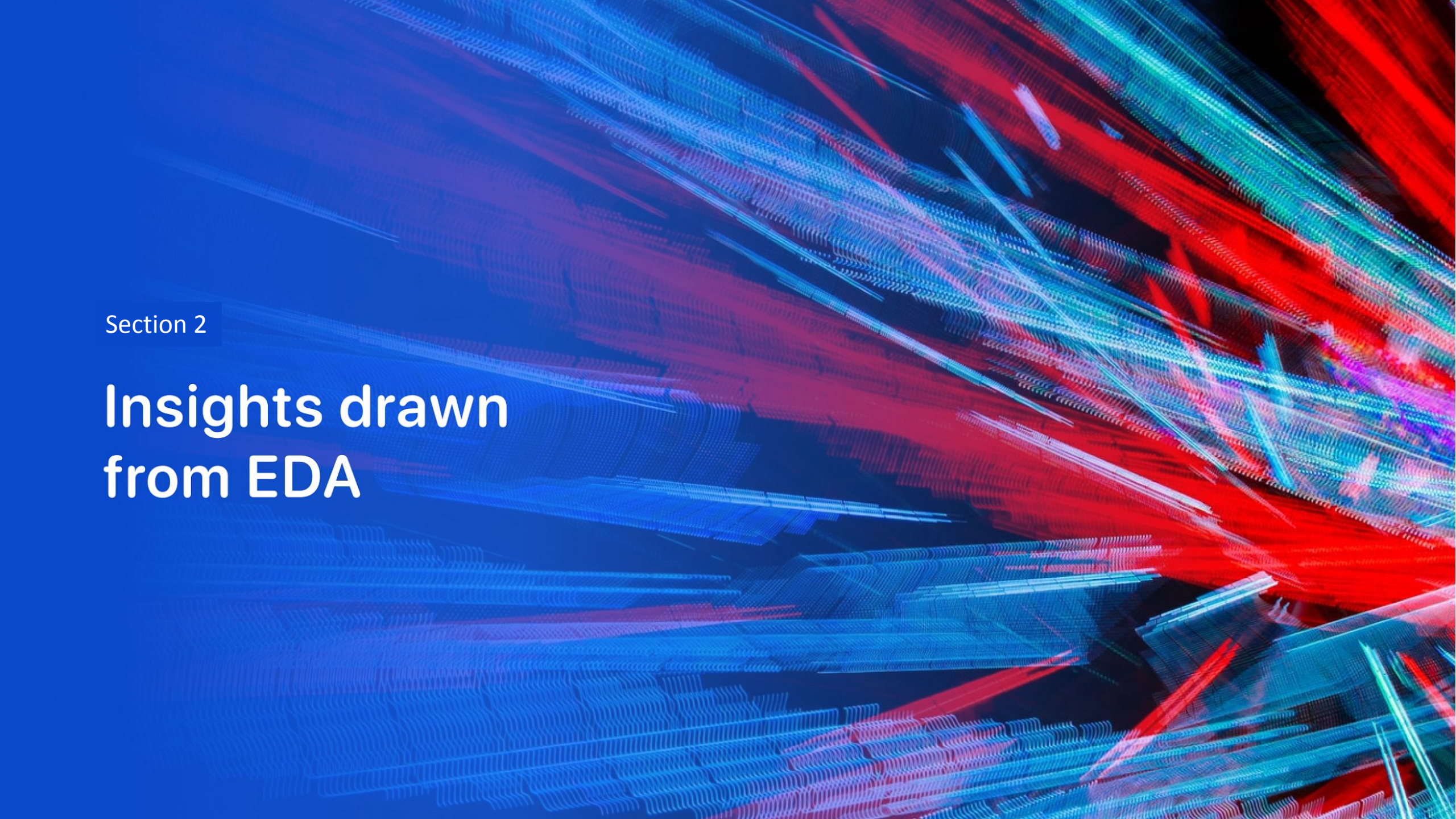
https://github.com/scar0510/Applied-Data-Science-Capstone/blob/main/8.%20SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

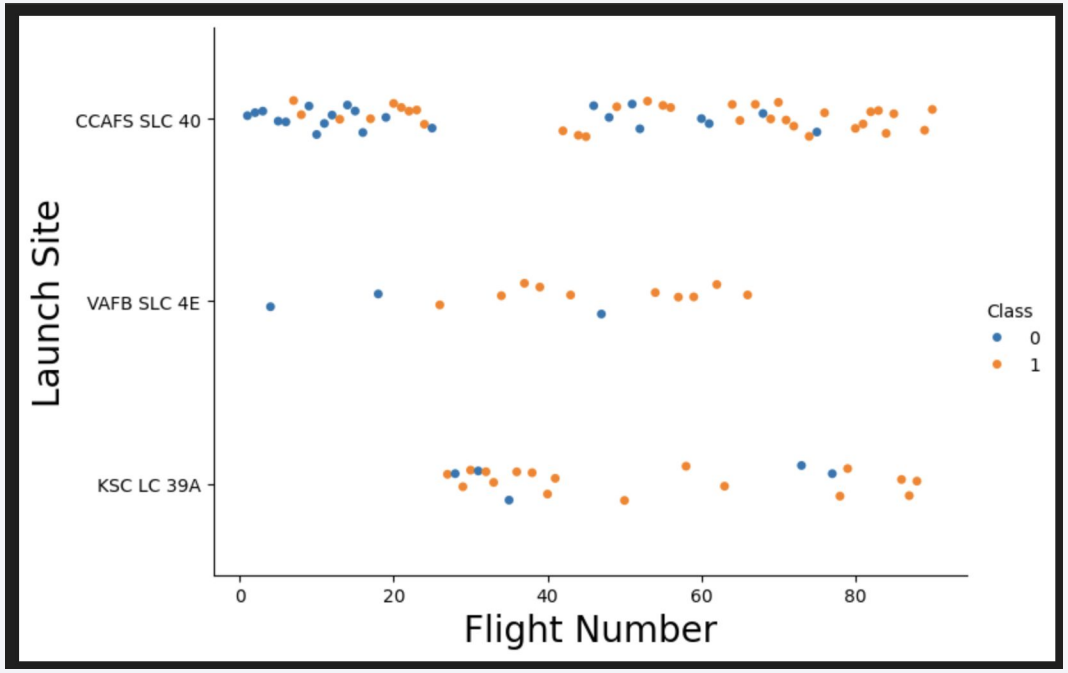
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

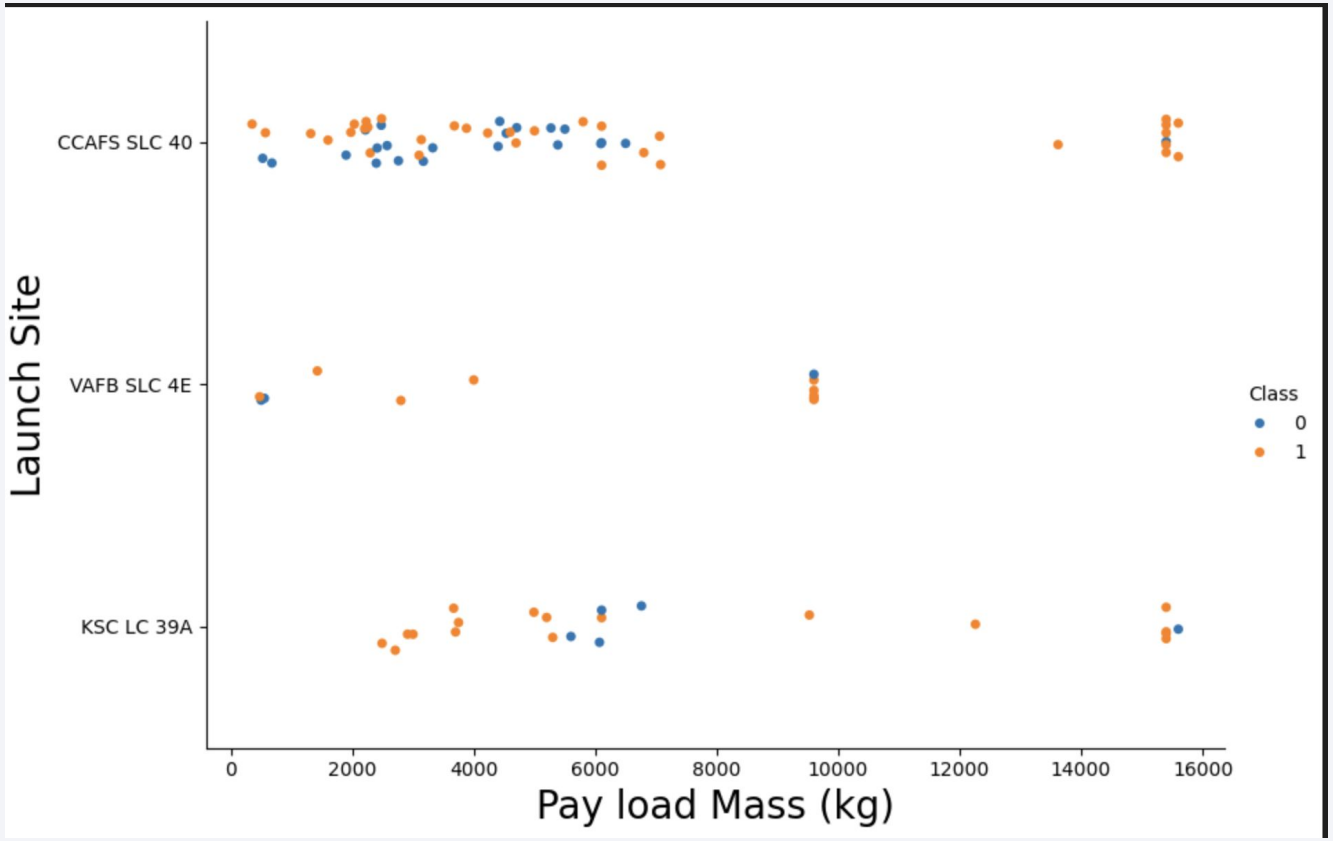


Flight Number vs. Launch Site



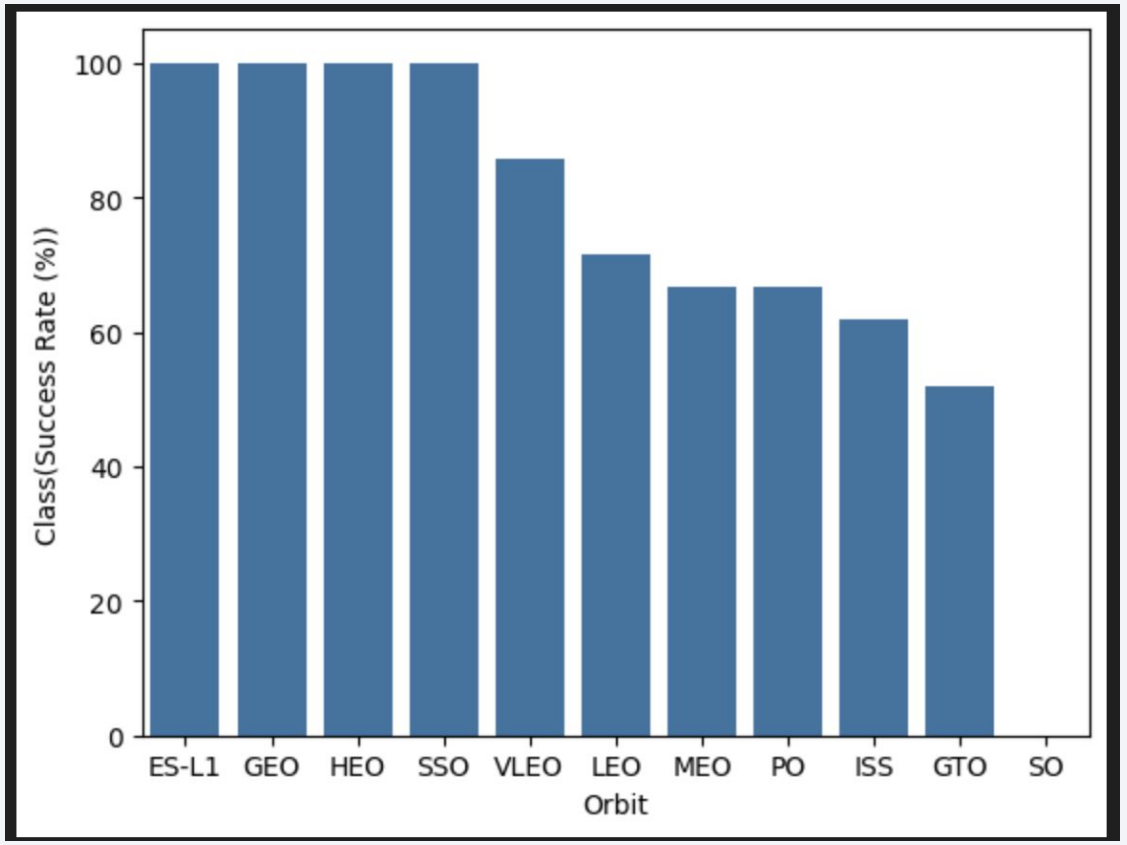
We can deduce that as the flight number increases in each of the three launch sites, so does the success rate. This indicates that more experience and iterations lead to improved performance and higher success rates over time.

Payload vs. Launch Site



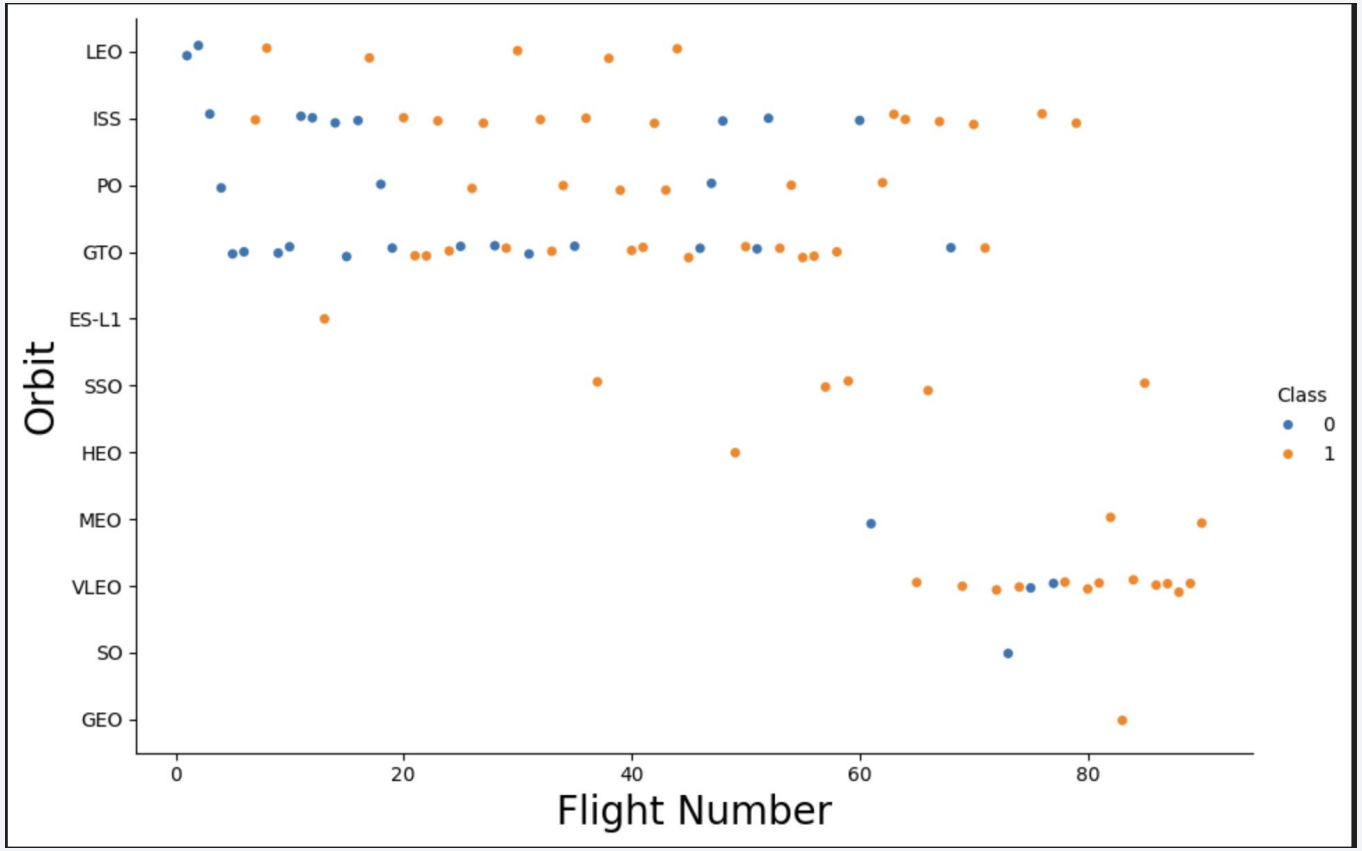
If you observe the payload mass versus launch site, you find that there are no rocket launches for heavy payloads at certain launch sites. This indicates that some launch sites may have restrictions or limitations on handling heavier payloads, or they are primarily used for launching lighter payloads.

Success Rate vs. Orbit Type



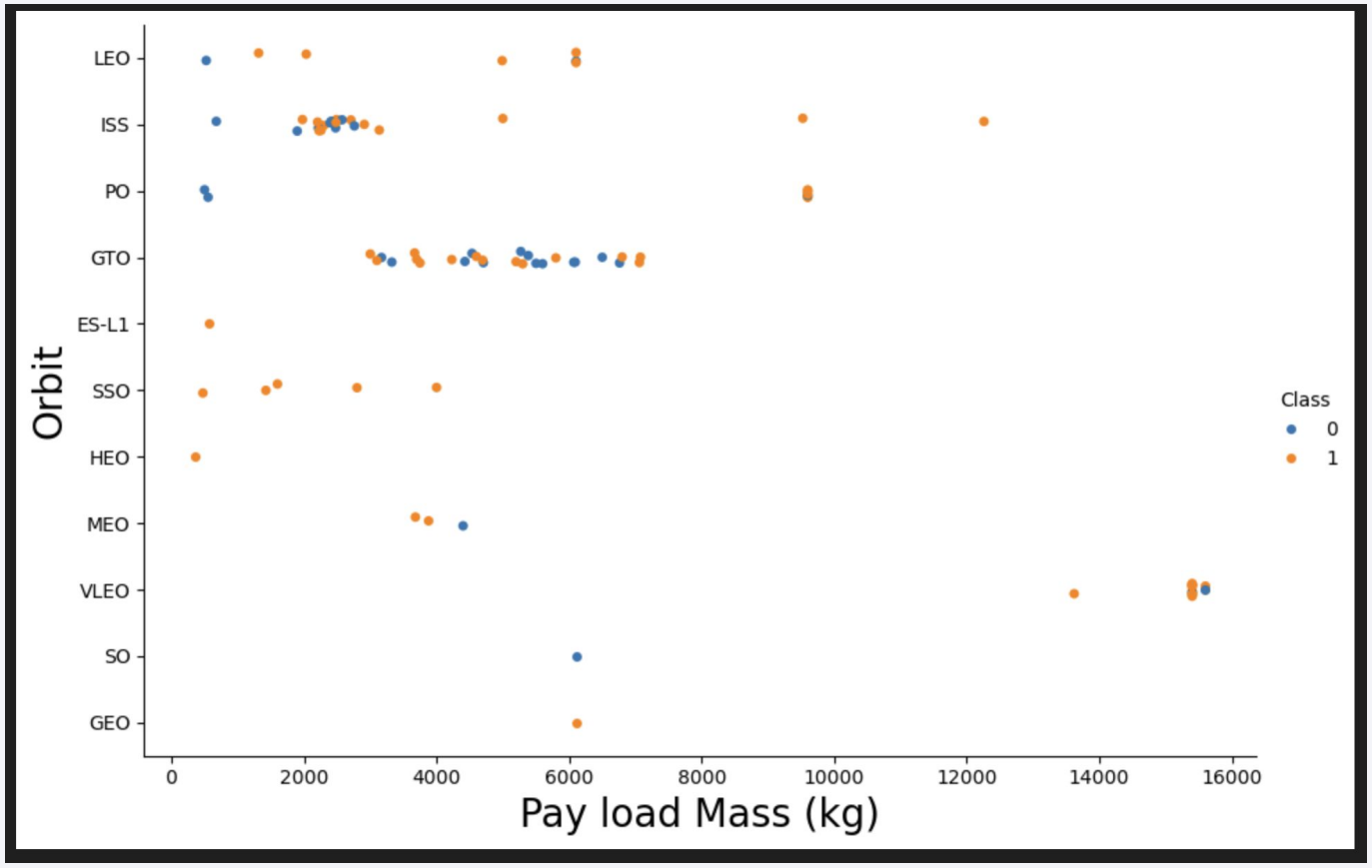
In the LEO orbit, success appears to be correlated with the number of flights. Conversely, in the GTO orbit, there doesn't seem to be any noticeable relationship between success and flight number.

Flight Number vs. Orbit Type



It can be observed that in the LEO orbit, success seems to be linked to the number of flights. On the other hand, there appears to be no clear relationship between flight number and success in the GTO orbit.

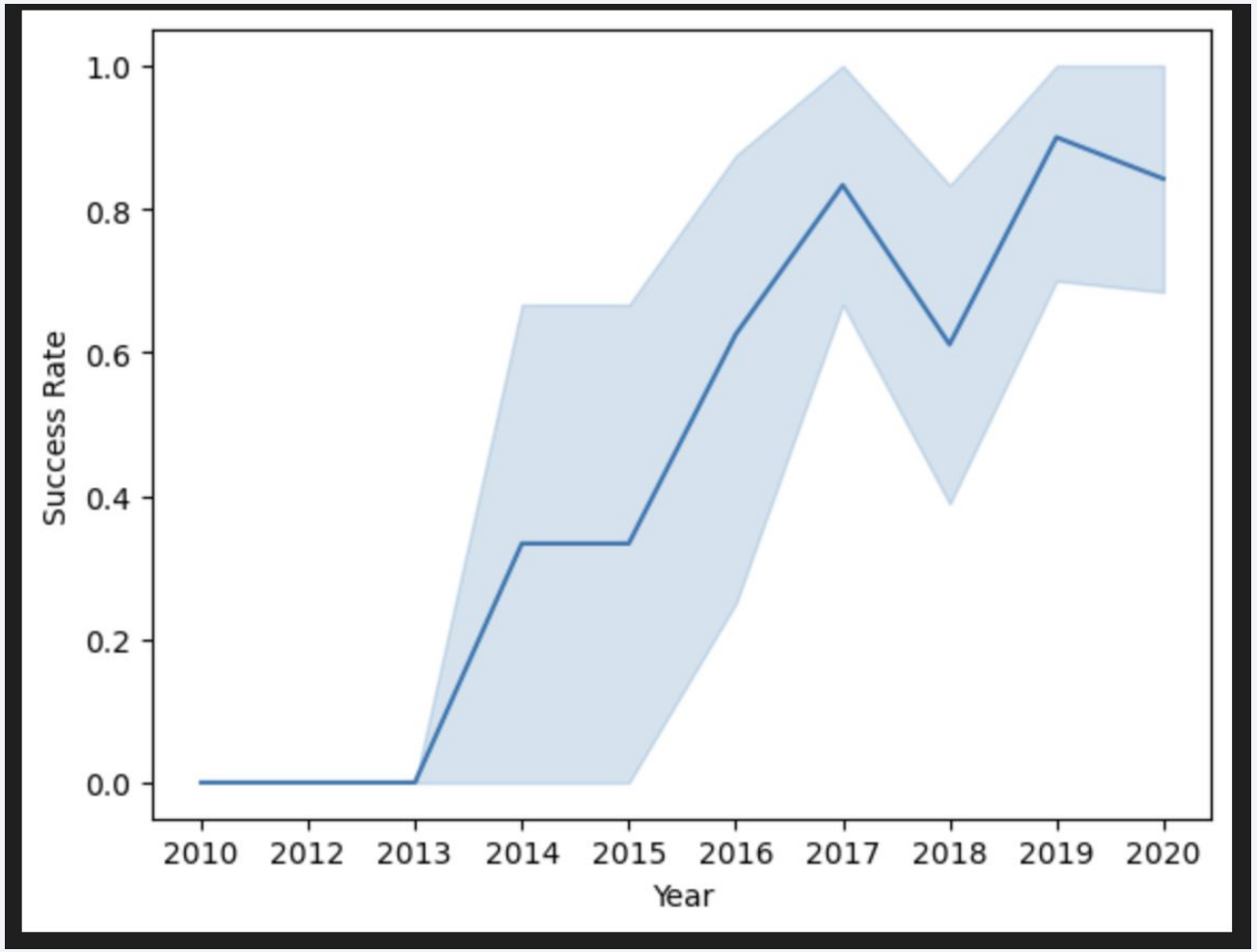
Payload vs. Orbit Type



For heavy payloads, the success rate of landings, or positive landing rate, is higher for Polar, LEO, and ISS missions.

However, for GTO missions, it is difficult to distinguish between successful and unsuccessful landings, as both outcomes have nearly equal probabilities.

Launch Success Yearly Trend



Since 2013, the launch success rate has shown a steady upward trend, consistently improving each year until 2020. This increase reflects advancements in technology, better mission planning, and enhanced reliability in launch systems

All Launch Site Names

- Find the names of the unique launch sites

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

Python

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

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

The SQL query retrieves a list of unique launch site names from the SPACEXTBL table. This is achieved by selecting the distinct values in the LAUNCH_SITE column and renaming the output column to "Launch_Sites"

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

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

Python

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

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

The SQL query retrieves 5 records from the SPACEXTBL table where the Launch_Site column values begin with the string 'CCA'. It uses the LIKE operator with 'CCA%' to filter the records and limits the result to 5 rows.

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';
```

Python

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

Total Payload Mass(Kgs)	Customer
45596	NASA (CRS)

The SQL query calculates the total payload mass carried by boosters launched by NASA (CRS). It does this by summing up the values in the PAYLOAD_MASS__KG_ column for records where the Customer column is 'NASA (CRS)'. The result is displayed with the column name "Total Payload Mass(Kgs)".

Average Payload Mass by F9 v1.1

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';
```

Python

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

Payload Mass Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

The SQL query calculates the average payload mass carried by the booster version F9 v1.1. It does this by averaging the values in the PAYLOAD_MASS__KG_ column for records where the Booster_Version column starts with 'F9 v1.1'. The result is displayed with the column name "Payload Mass Kgs".

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

```
%sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing_Outcome" = "Success (ground pad)";
```

Python

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

MIN(DATE)
2015-12-22

The SQL query retrieves the date when the first successful landing outcome on a ground pad was achieved. It does this by selecting the minimum value from the DATE column for records where the Landing_Outcome column is 'Success (ground pad)'.

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had 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;
```

Python

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

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

The SQL query retrieves the names of boosters that have had a successful landing on a drone ship and carried a payload mass between 4000 and 6000 kilograms. It selects distinct values of the Booster_Version and PAYLOAD_MASS__KG_ columns from the SPACEXTBL table where the Landing_Outcome is 'Success (drone ship)' and the PAYLOAD_MASS__KG_ is between 4000 and 6000.

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

Python

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

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

The SQL query retrieves the total number of successful and failed mission outcomes. It does this by counting the occurrences of each unique value in the Mission_Outcome column and grouping the results by Mission_Outcome. The result is displayed with the column name "Total".

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

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

Python

```
 * 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

The SQL query retrieves the names of the booster versions that have carried the maximum payload mass. It uses a subquery to find the maximum value in the PAYLOAD_MASS__KG_ column and then selects the Booster_Version, Payload, and PAYLOAD_MASS__KG_ columns from the SPACEXTBL table where the PAYLOAD_MASS__KG_ matches this maximum value.

2015 Launch Records

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

```
%sql SELECT substr(Date,6,2) AS 'Month', "Booster Version", "Launch Site", Payload, "PAYLOAD MASS  KG ", "Mission Outcome",
"Landing_Outcome" FROM SPACEXTBL WHERE substr(Date,0,5)='2015' AND "Landing_Outcome" = 'Failure (drone ship)'                ;
```

Month	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	Landing_Outcome
01	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	Success	Failure (drone ship)
04	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	Success	Failure (drone ship)

The SQL query lists records for the year 2015 with failures on a drone ship. It extracts the month using substr(Date,6,2) and filters for the year 2015 using substr(Date,0,5). The result includes Month, Booster_Version, Launch_Site, Payload, PAYLOAD_MASS__KG_, Mission_Outcome, and Landing_Outcome.

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

```
%sql SELECT * FROM SPACEXTBL WHERE "Landing_Outcome" LIKE 'Success%' AND (Date BETWEEN '2010-06-04' AND '2017-03-20') ORDER BY Date DESC;
```

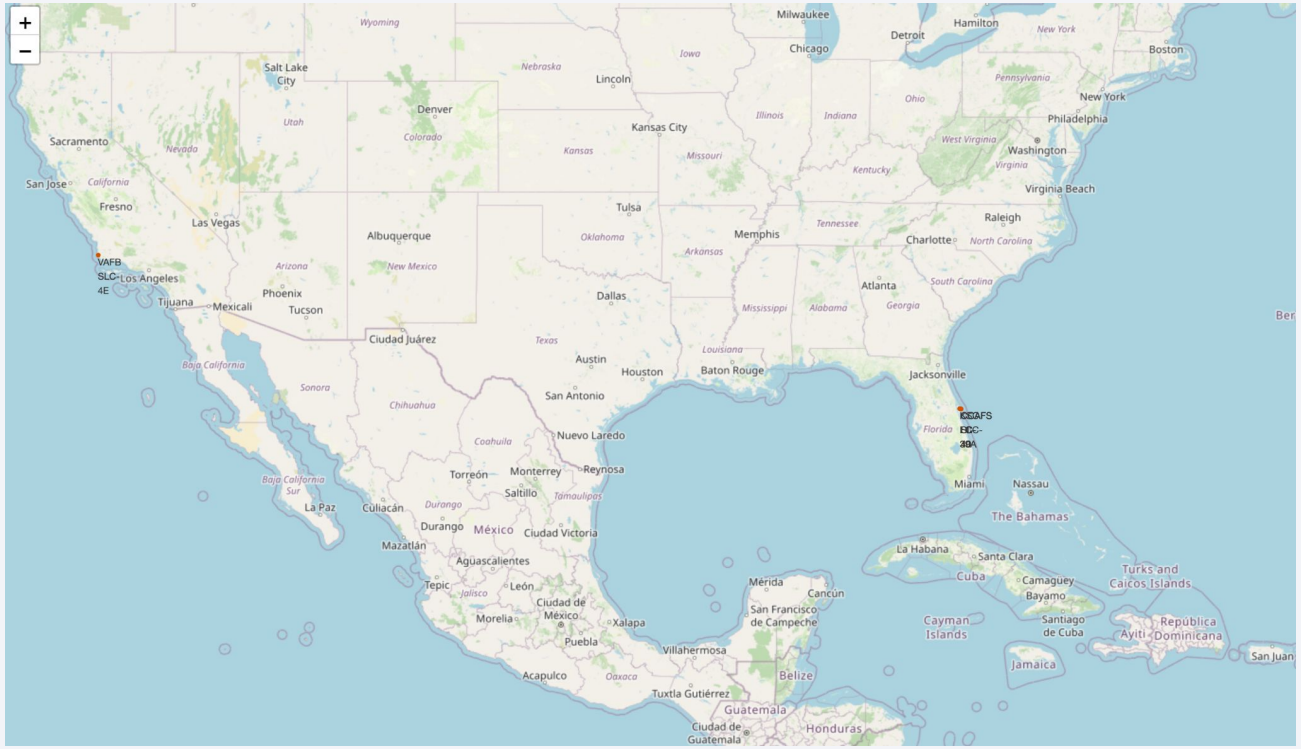
```
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-01-14	17:54:00	F9 FT B1029.1	VAFB SLC-4E	Iridium NEXT 1	9600	Polar LEO	Iridium Communications	Success	Success (drone ship)
2016-08-14	5:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-07-18	4:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2016-05-27	21:39:00	F9 FT B1023.1	CCAFS LC-40	Thaicom 8	3100	GTO	Thaicom	Success	Success (drone ship)
2016-05-06	5:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-04-08	20:43:00	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136	LEO (ISS)	NASA (CRS)	Success	Success (drone ship)
2015-12-22	1:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

The SQL query ranks the count of landing outcomes between the dates 2010-06-04 and 2017-03-20 in descending order. It groups by Landing_Outcome and orders by the count in descending order.


Section 3
Launch Sites
Proximities Analysis

All launch sites on global map

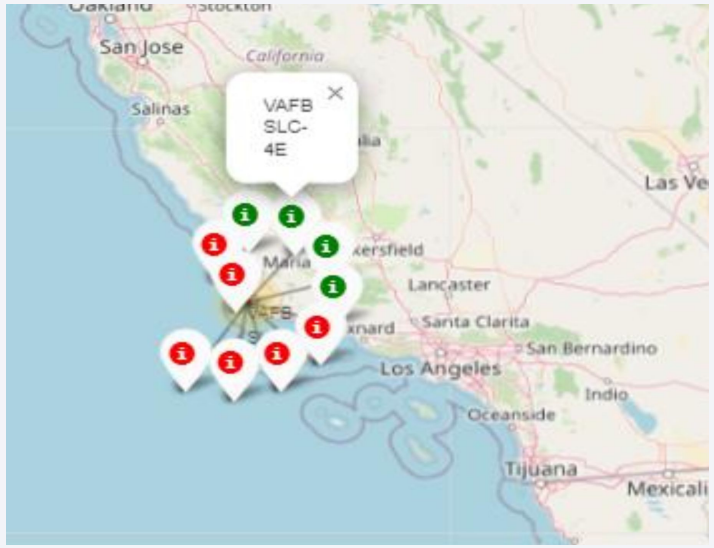


As we can see on the map, all launch sites are located in the USA, specifically in the coastal states of Florida and California, near the coast.

Launch Outcomes for Each Site

Florida Sites

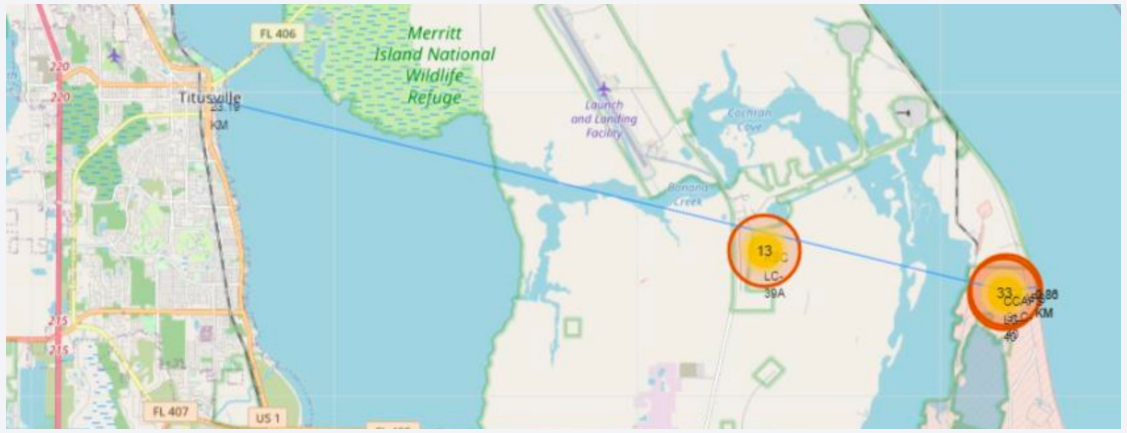
California Sites



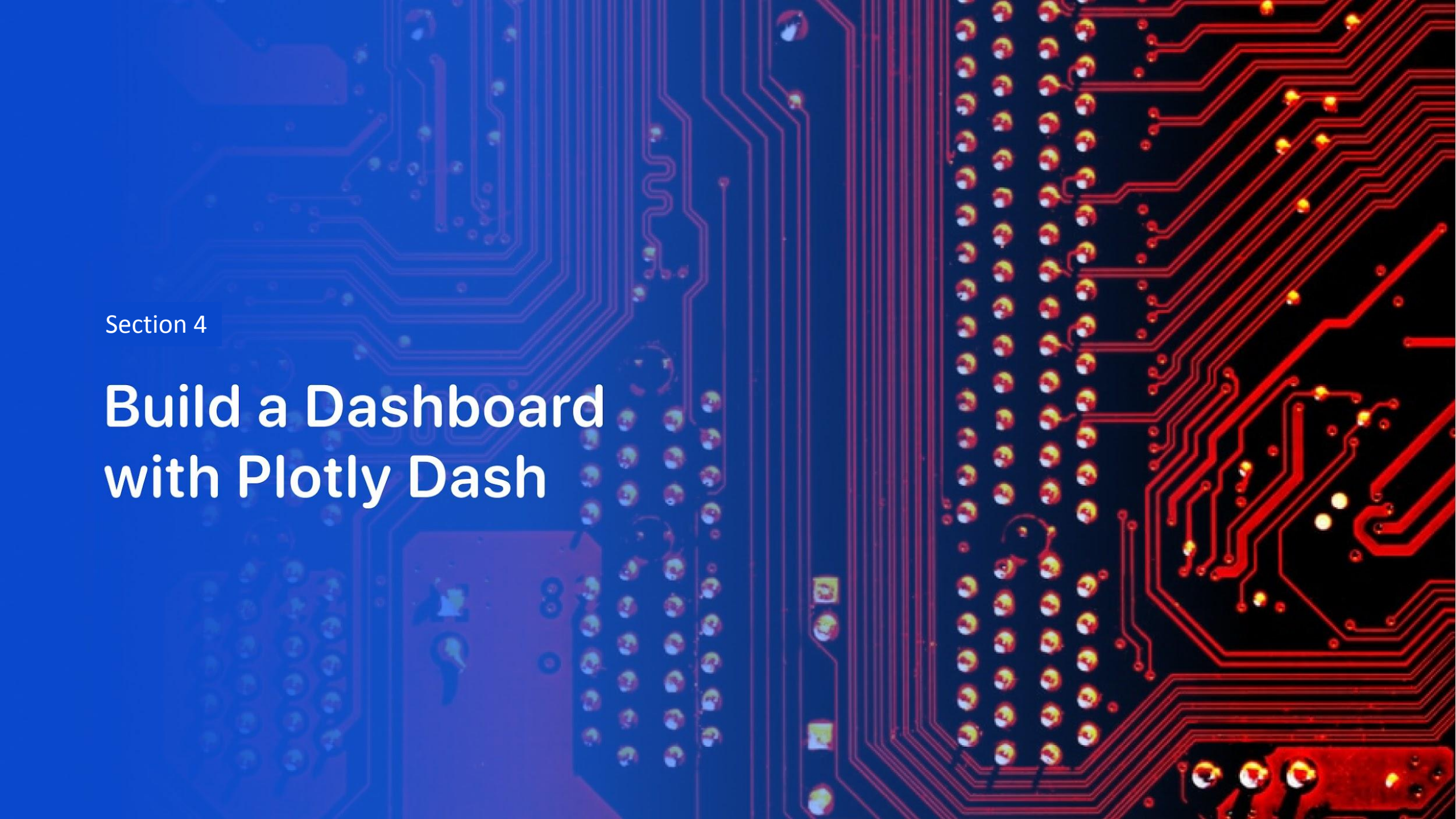
There are a lot more launches in Florida, but it seems to also have a lower success rate compared to California

Distances between a launch site to its proximities

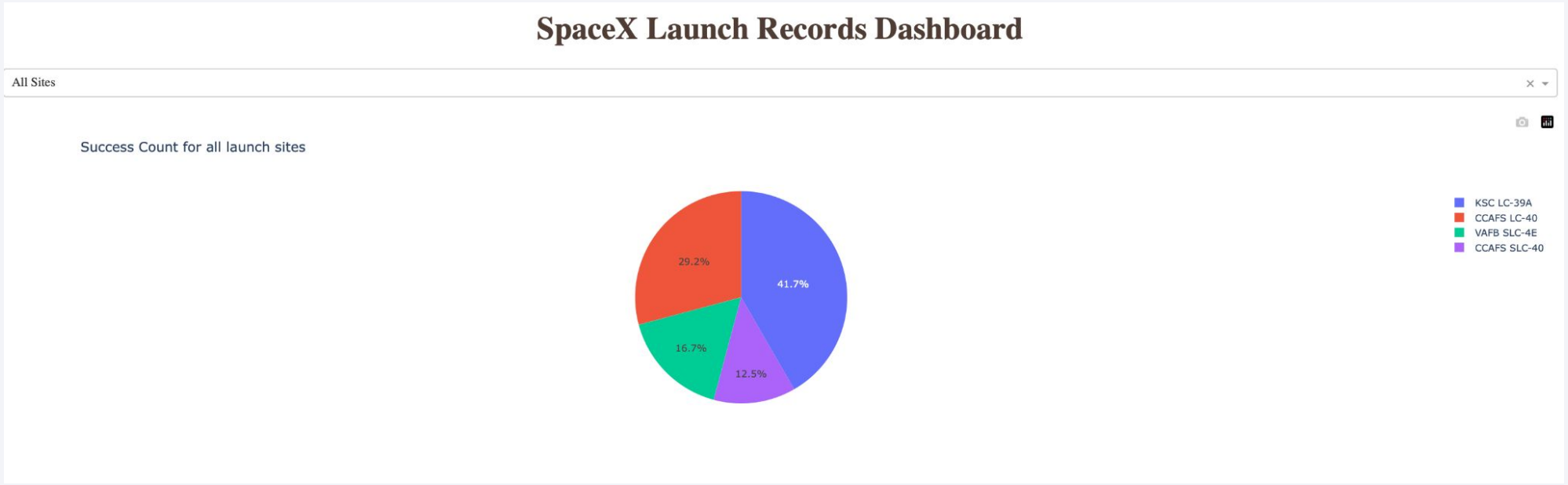
CCAFS SLC-40



As we can see on the map, all launch sites are located in the USA, specifically in the coastal states of Florida and California, near the coast. There are more launches in Florida, but it has a lower success rate. For example, the launch site CCAFS SLC-40 is 23.19 km from the nearest highway (Washington Avenue) and 0.86 km from the coastline.

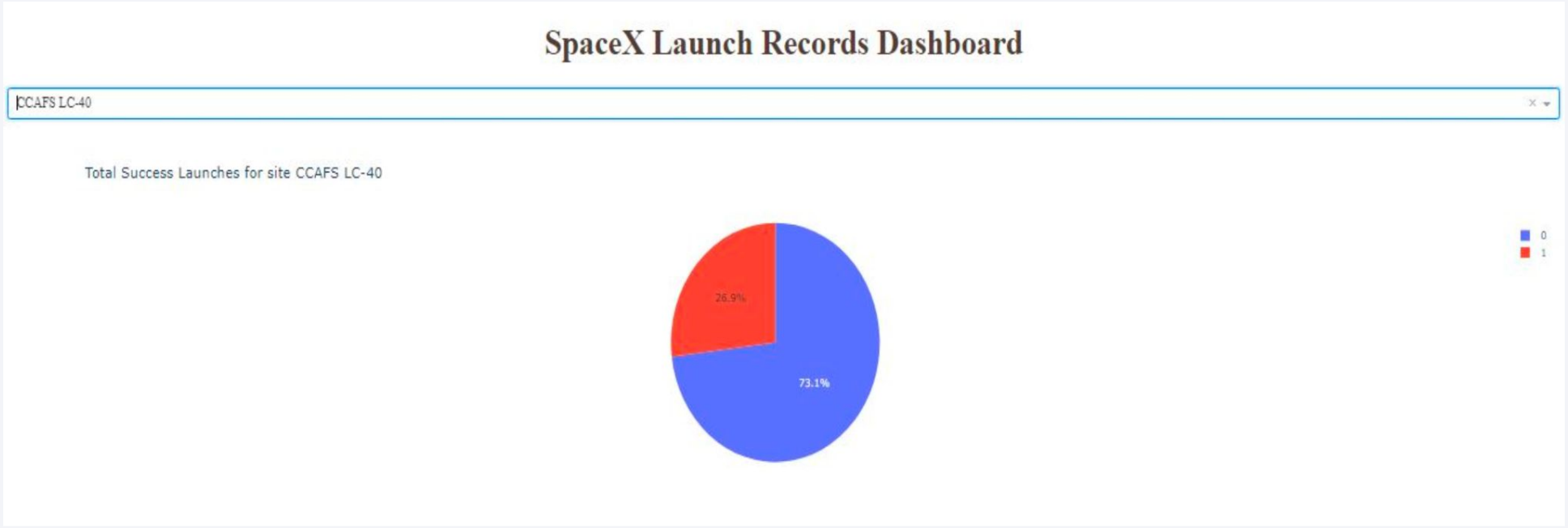

Section 4
Build a Dashboard with Plotly Dash

Success Count for All Launch Sites



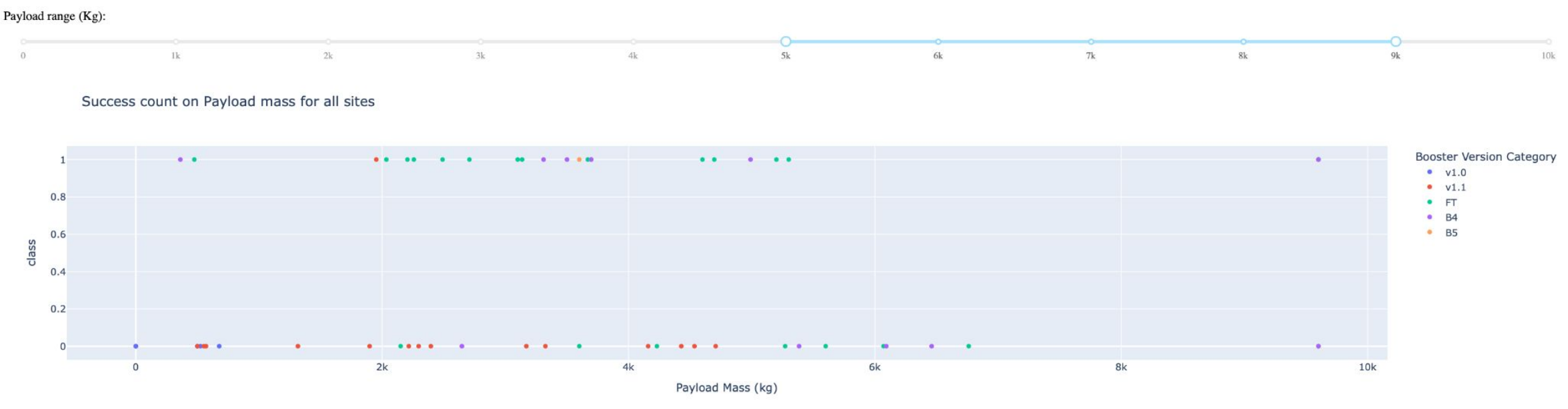
As we can see in the chart, CCAFS has a higher success count, which may be due to the larger number of launches. This site is 23.19 km from the nearest highway (Washington Avenue) and 0.86 km from the coastline

Success Rate in CCAFS LC-40



The success rate at CCAFS LC-40 is currently 73%, indicating a strong performance in recent missions. Continuous improvements and rigorous testing protocols have contributed to this high success rate.

Payload vs. Launch Outcome for all sites



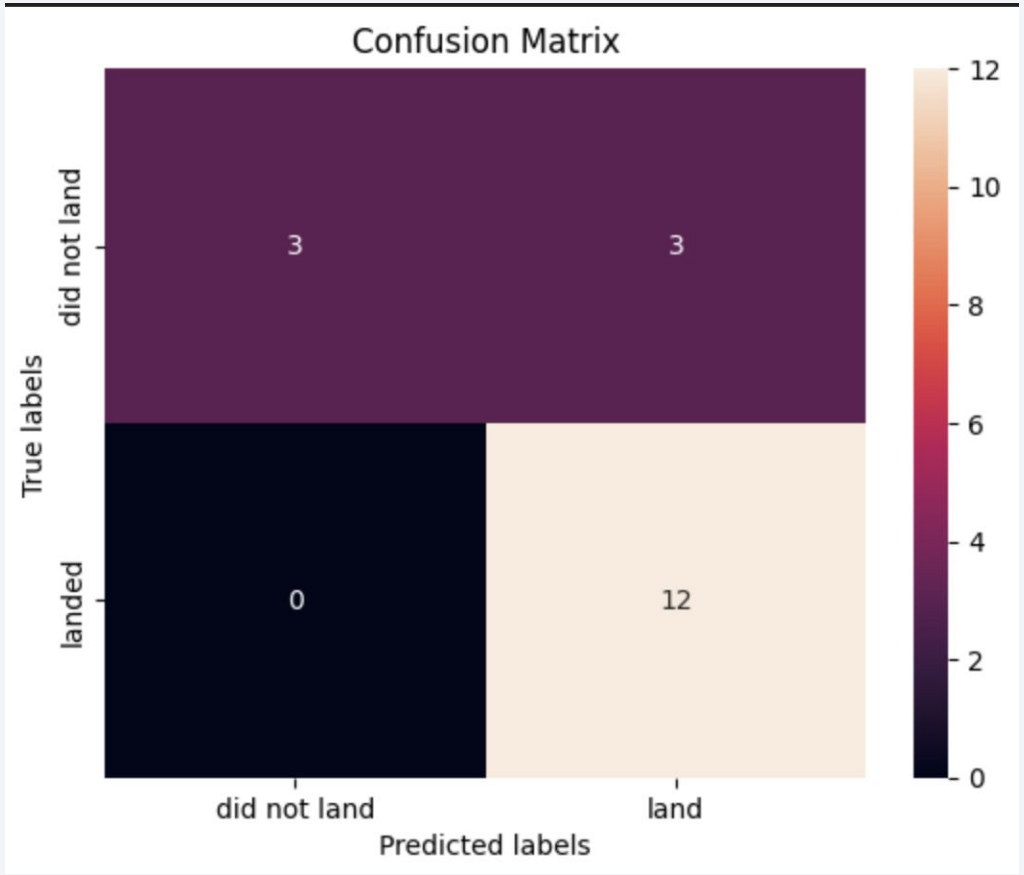
For Launch site CCAFS LC-40, the booster version FT boasts the highest success rate for payloads with a mass between 5,000 and 9,000 kg. This impressive performance highlights the reliability and capability of the FT booster in handling medium to heavy payloads effectively.

Section 5

Predictive Analysis (Classification)

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 is the high number of false positives.



Conclusions

Different launch sites exhibit varying success rates. CCAFS LC-40 has a success rate of 60%, while KSC LC-39A and VAFB SLC 4E each have a success rate of 77%.

- It can be observed that the success rate increases with the number of flights at each of the three launch sites. For example, the success rate for the VAFB SLC 4E launch site reaches 100% after the 50th flight. Both KSC LC-39A and CCAFS SLC-40 achieve a 100% success rate after the 80th flight.
- Analyzing the Payload vs. Launch Site scatter plot reveals that no rockets with a heavy payload mass (greater than 10,000 kg) are launched from the VAFB SLC 4E launch site.
- Orbits ES-L1, GEO, HEO, and SSO have the highest success rates at 100%, while the SO orbit has the lowest success rate at approximately 50%. The SO orbit has a 0% success rate.
- For LEO orbit, the success rate appears to be related to the number of flights. Conversely, there seems to be no correlation between the number of flights and success rate for the GTO orbit.
-

Thank you!