IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Denver G. Magtibay>
<January 2023>

Outline

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

Executive Summary

In the era of commercial space exploration, SPACE Y is on a mission to make space travel a reality for everyone. Born from the belief that the wonders of the universe should be within reach, we are committed to redefining accessibility in space exploration.

Methodologies:

1. **Data Collection:** *Curating Comprehensive Datasets Through API Integration and Web Scraping*
2. **Data Wrangling:** *Meticulous Data Cleaning, Structuring, and Refinement*
3. **Exploratory Data Analysis (EDA):** *In-Depth Analysis with SQL, Pandas, and Matplotlib*
4. **Interactive Visual Analytics and Dashboard Creation:** *Engaging Stakeholders with Folium and Plotly Dash*
5. **Predictive Analysis (Classification):** *Informed Decision-Making through Machine Learning Models*

Results:

1. **Optimized Hyperparameters:** *Identifying the Best Hyperparameters for Each Classification Algorithm*
2. **Performance Assessment:** *Determining the Superior Method through Test Data Evaluation*

Introduction

In this project, we're exploring a cool space idea: figuring out if SpaceX's Falcon 9 rocket's first stage will land safely. You see, SpaceX is known for its cost-friendly rocket launches, only about 62 million dollars, way less than what others charge, which can go over 165 million dollars. The trick? SpaceX can reuse the first stage of the rocket. So, if we can predict whether that first stage will land well, we can figure out how much a launch will cost. This info is super handy if another company wants to compete with SpaceX for a rocket launch. So, buckle up for a journey into predicting space success and making space travel more wallet-friendly!

Section 1

Methodology

Executive Summary

Data Collection Methodology:

- Data gathered from SpaceX REST API and web scraping from wiki pages.

Data Wrangling:

- Convert collected data (JSON, HTML) into a Pandas dataframe for visualization and analysis.

Exploratory Data Analysis (EDA):

- Load data into a database.
- Execute SQL queries for data selection and sorting.
- Conduct EDA using Python and Pandas, creating scatter plots and bar charts for analysis.

Interactive Visual Analytics (Folium and Plotly Dash):

- Build an interactive map with Folium to analyze launch site proximity.
- Use Plotly Dash to create an interactive dashboard containing pie charts and scatter plots.

Predictive Analysis (Classification Models):

- Train different classification models.
- Split data into training and testing sets.
- Perform grid search to find optimal hyperparameters.
- Utilize machine learning to build a predictive model for Falcon 9 rocket landings.

SpaceX API Web Scraping

Data Collection

- The data collection involved retrieving information from both the SpaceX REST API and conducting web scraping on relevant wiki pages.

Data Collection – SpaceX API

Data Collection - SpaceX API → 1. SpaceX REST API endpoint → 2. Get request using the requests library

↓

3. Extract information about booster name, launch site, payload mass and landing site → 4. Get the past launch data as a JSON object → 5. Convert the JSON to a dataframe

↓

6. Filter the dataframe to only include Falcon 9 launches → 7. Deal with missing values → 8. Data wrangling

Data Collection API Notebook:
https://github.com/WarDen-Git/testrepo/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

Request the Falcon 9 Launch Wiki page from its URL.

→ Create a BeautifulSoup object from the response.

→ Extract all column/variable names from the HTML table header.

→ Create a data frame by parsing the launch HTML tables.

→ Conduct data wrangling to enhance the structure and quality of the collected data.

Data Collection Scraping Notebook: https://github.com/WarDen-Git/testrepo/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

1. Identify missing values.
2. Differentiate numerical and categorical columns.
3. Calculate the count of launches for each site.
4. Determine the count and occurrences of each orbit.
5. Assess the count and occurrences of mission outcomes per orbit type.
6. Generate a landing outcome label based on the information in the outcome column.

Data Wrangling Notebook:
https://github.com/WarDen-Git/testrepo/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

Flight Number vs. Payload Mass **Scatter Plot**	• To explore any potential correlation between the flight number and payload mass, identifying trends or patterns over multiple launches.
Flight Number vs. Launch Site **Scatter Plot**	• To visually analyze if there's any clustering or grouping of launches at different sites over the course of multiple flight numbers.
Payload Mass vs. Launch Site **Scatter Plot**	• To examine how payload mass varies across different launch sites, identifying any notable differences or trends.
Orbit Type vs. Success Rate **Bar Chart**	• To compare the success rates for different orbit types, providing a clear overview of mission outcomes.
Flight Number vs. Orbit Type **Scatter Plot**	• To visualize the distribution of different orbit types over various flight numbers, revealing patterns or shifts in mission objectives.
Payload Mass vs. Orbit Type **Scatter Plot**	• To explore the relationship between payload mass and orbit type, identifying potential dependencies or preferences.
Year vs. Average Success Rate **Line Chart:**	• To observe trends in the average success rate over the years, providing insights into the overall performance and progress of SpaceX launches.

EDA with Data Visualization Notebook:
https://github.com/WarDen-Git/testrepo/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

SQL Queries:

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

%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;

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

2. Display 5 records where launch sites begin with the string 'KSC'

*%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'KSC%' LIMIT 5;*

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)
2017-03-16	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

EDA with SQL Notebook:
https://github.com/WarDen-Git/testrepo/blob/main/jupyter-labs-eda-sql-edx_sqllite.ipynb

EDA with SQL

SQL Queries:

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

%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TotalPayloadMass FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)'

TotalPayloadMass
45596

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

%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AveragePayloadMass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1'

AveragePayloadMass
2928.4

EDA with SQL Notebook:
https://github.com/WarDen-Git/testrepo/blob/main/jupyter-labs-eda-sql-edx_sqllite.ipynb

EDA with SQL

SQL Queries:

5. List the date where the successful landing outcome in drone ship was achieved.

%sql SELECT MIN(Date) AS MinLandingDate FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)’

MinLandingDate
2016-04-08

6. List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000

%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000

Booster_Version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

EDA with SQL Notebook:
https://github.com/WarDen-Git/testrepo/blob/main/jupyter-labs-eda-sql-edx_sqllite.ipynb

EDA with SQL

SQL Queries:

7. List the total number of successful and failure mission outcomes

%sql SELECT Mission_Outcome, COUNT(*) AS TotalMissions FROM SPACEXTABLE GROUP BY Mission_Outcome;

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

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

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

EDA with SQL Notebook:
https://github.com/WarDen-Git/testrepo/blob/main/jupyter-labs-eda-sql-edx_sqllite.ipynb

EDA with SQL

SQL Queries:

9. List the records which will display the month names, succesful landing_outcomes in ground pad, booster versions, launch_site for the months in year 2017

%sql SELECT substr(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr(Date, 0, 5) = '2017' AND Landing_Outcome = 'Success (ground pad)';

Month	Landing_Outcome	Booster_Version	Launch_Site
02	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
05	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
06	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
08	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
09	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
12	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40

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 OutcomeCount FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY OutcomeCount DESC;

Landing_Outcome	OutcomeCount
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

EDA with SQL Notebook:
https://github.com/WarDen-Git/testrepo/blob/main/jupyter-labs-eda-sql-edx_sqllite.ipynb

Build an Interactive Map with Folium

Folium.marker	Folium.circle	MarkerCluster	MousePosition	Folium.polyline
Purpose: Used to place markers on the map at specific locations. Markers were added to highlight specific points of interest or locations on the map. These markers often contain pop-up information to provide details about the marked location.	Purpose: Creates circular shapes on the map. Circles were added to represent areas of interest or influence, providing a visual representation of a defined radius around a central point. This is useful for displaying regions or zones on the map.	Purpose: Groups nearby markers into clusters to improve map readability. When there are numerous markers in close proximity, MarkerCluster helps organize and display them dynamically. As users zoom in, the clusters break apart to reveal individual markers, preventing map clutter.	Purpose: Displays the coordinates of the mouse pointer on the map. MousePosition was added to provide real-time latitude and longitude information as users move their cursor across the map. This can be helpful for precise location identification.	Purpose: Draws lines on the map, connecting specified points. Polylines were added to illustrate paths or connections between different locations. This is useful for visualizing routes, boundaries, or any sequential relationships on the map.

Launch Sites Locations Analysis with Folium Notebook:
https://github.com/WarDen-Git/testrepo/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

1. **Launch Site Drop-down Input Component:**

   Purpose: Enables the selection of different launch sites from a dropdown menu.

   This input component allows users to focus on specific launch sites of interest. It enhances user interactivity by providing a means to filter data based on launch locations, making the dashboard more versatile for users interested in particular sites.

2. **Callback Function for Success Pie Chart:**

   Purpose: Renders a pie chart visualizing launch success counts based on the selected launch site from the dropdown.

   The pie chart provides a quick overview of the distribution of launch successes for the chosen site. This visual representation allows users to easily grasp the success rates and assess the performance of different launch sites.

3. **Range Slider to Select Payload:**

   Purpose: Allows users to select a payload range using a slider.

   The range slider offers a flexible way to filter data based on payload values. Users can easily explore variations in payload ranges and observe any patterns or trends in the dataset related to mission outcomes.

4. **Callback Function for Success-Payload Scatter Plot:**

   Purpose: Renders a scatter plot visualizing the correlation between payload and mission success for the selected launch site.

   The scatter plot aids in exploring potential relationships between payload characteristics and mission outcomes. By dynamically updating based on the selected launch site and payload range, users can visually assess whether certain payload features are associated with higher or lower success rates.

Interactive Dashboard with Plotly Notebook:
https://github.com/WarDen-Git/testrepo/blob/main/Hands_on_Lab__Build_an_Interactive_Dashboard_with_Plotly_Dash.ipynb

Predictive Analysis (Classification)

1. Created a NumPy array (Y) from the 'Class' column in the data to serve as the target variable for classification.

2. Standardized the feature data (X) to ensure uniform scale and prevent any single feature from dominating the model.

3. Split the data into training and test sets using the train_test_split function for evaluating model performance on unseen data.

4. Conducted a hyperparameter search to optimize logistic regression, support vector machine, decision tree, and k-nearest neighbor classifiers.

5. Computed accuracy and generated confusion matrices for each classifier to assess their performance on the dataset.

6. Identified the best-performing model based on the evaluation metrics, selecting the classifier with the highest accuracy or meeting specific performance criteria.

Predictive Analysis Notebook:
https://github.com/WarDen-Git/testrepo/blob/main/module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

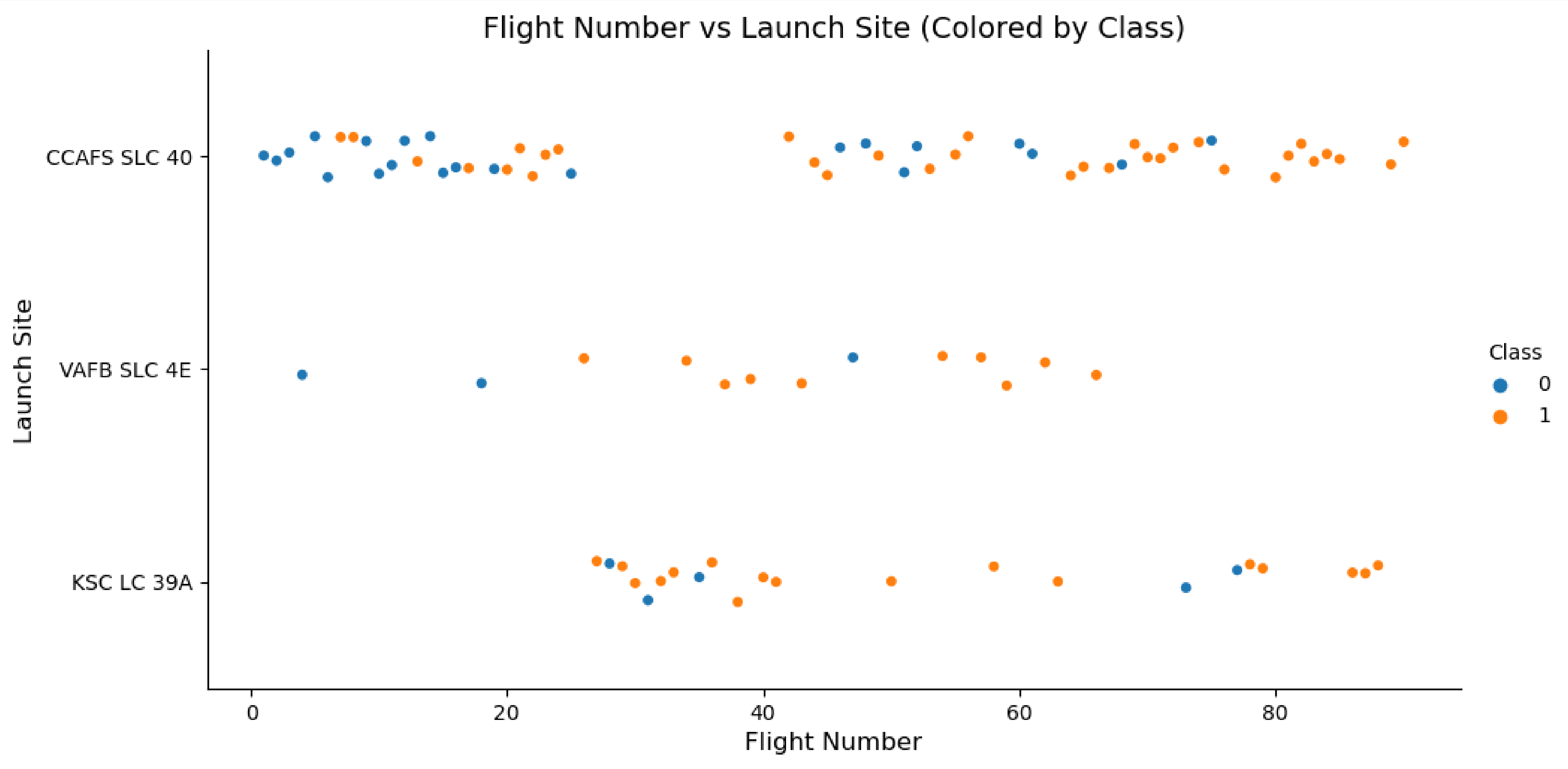
Results

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

Section 2

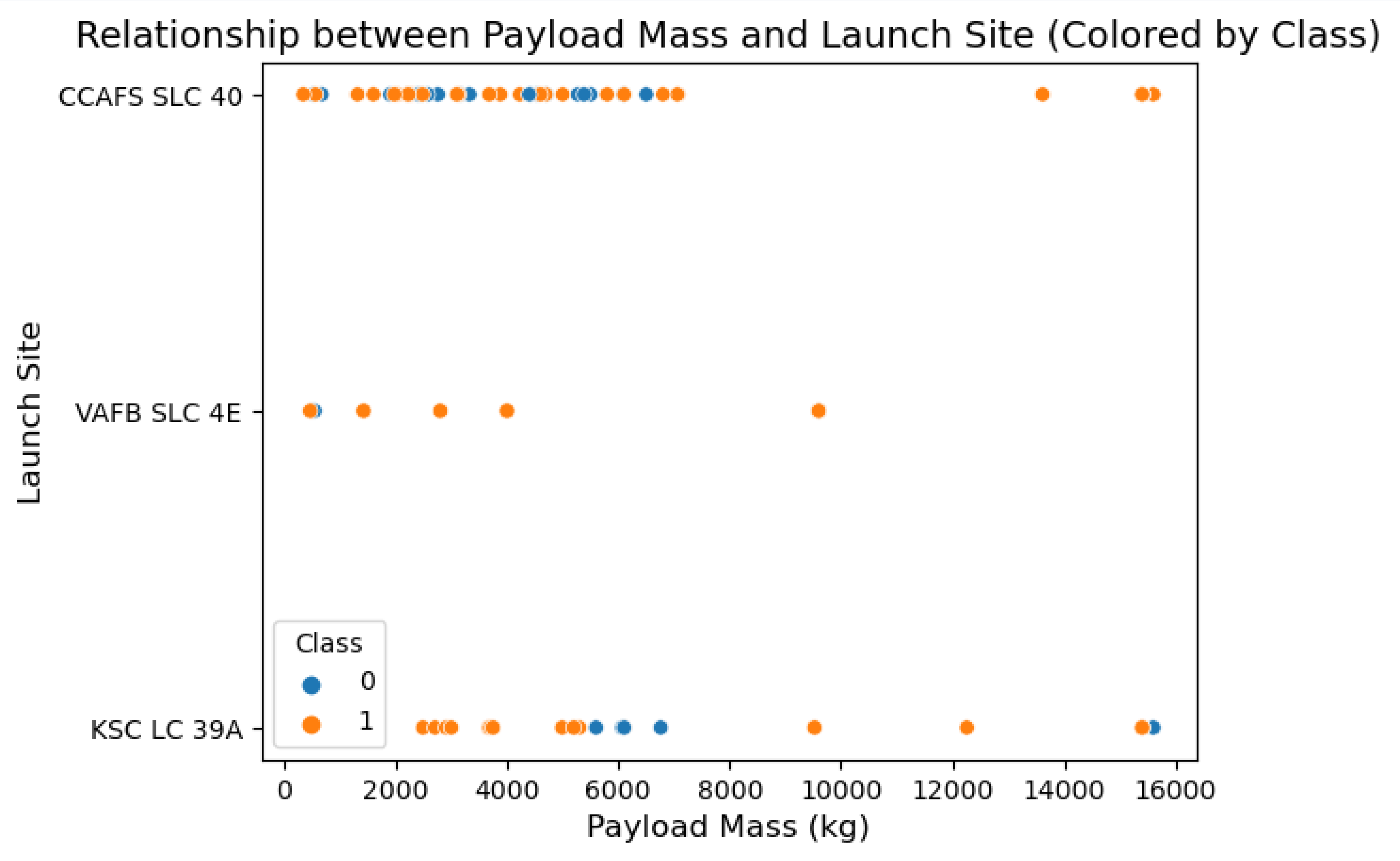
Insights drawn from EDA

Flight Number vs. Launch Site



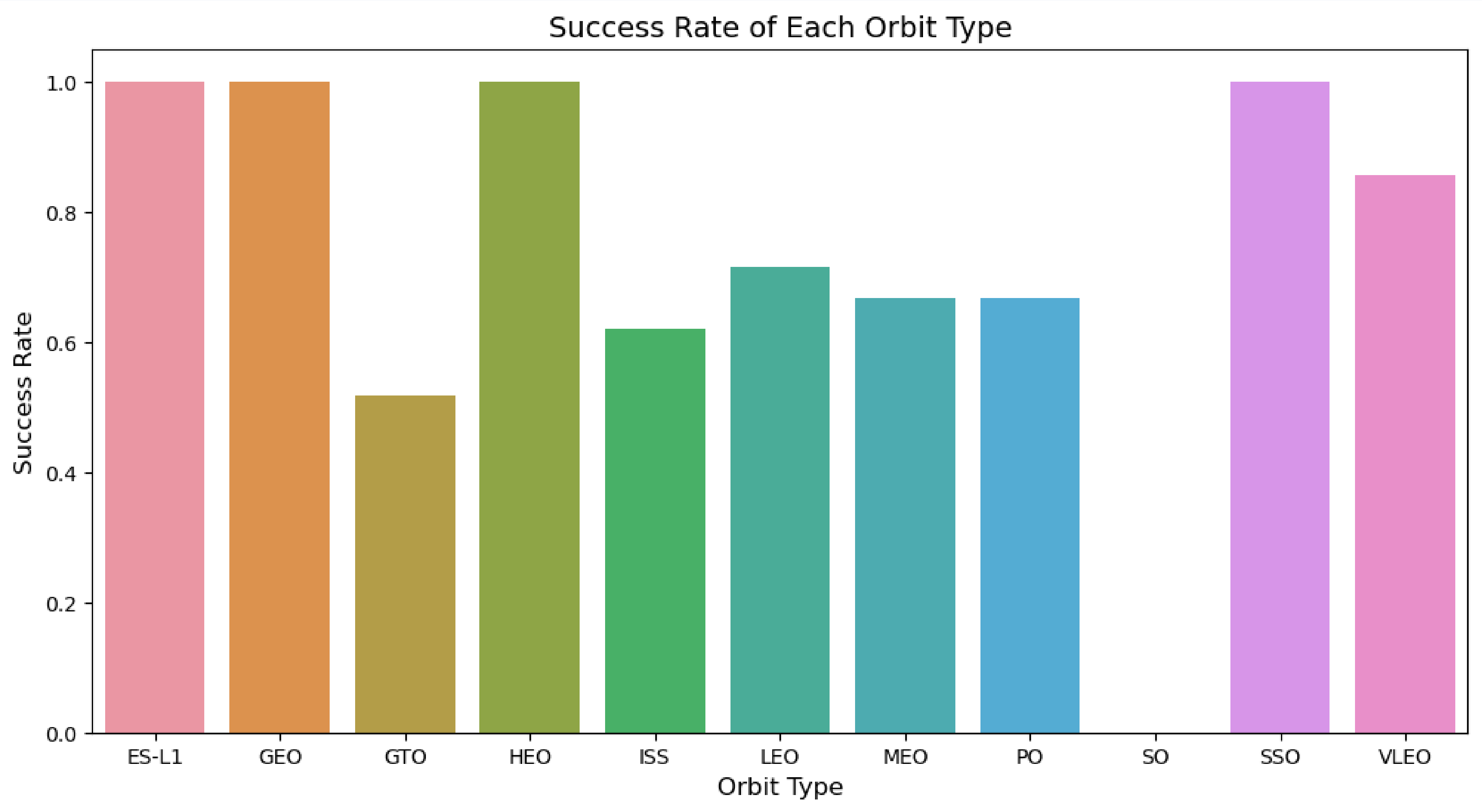
If the number of flights exceeds 80, the success rate is 100%.

Payload vs. Launch Site



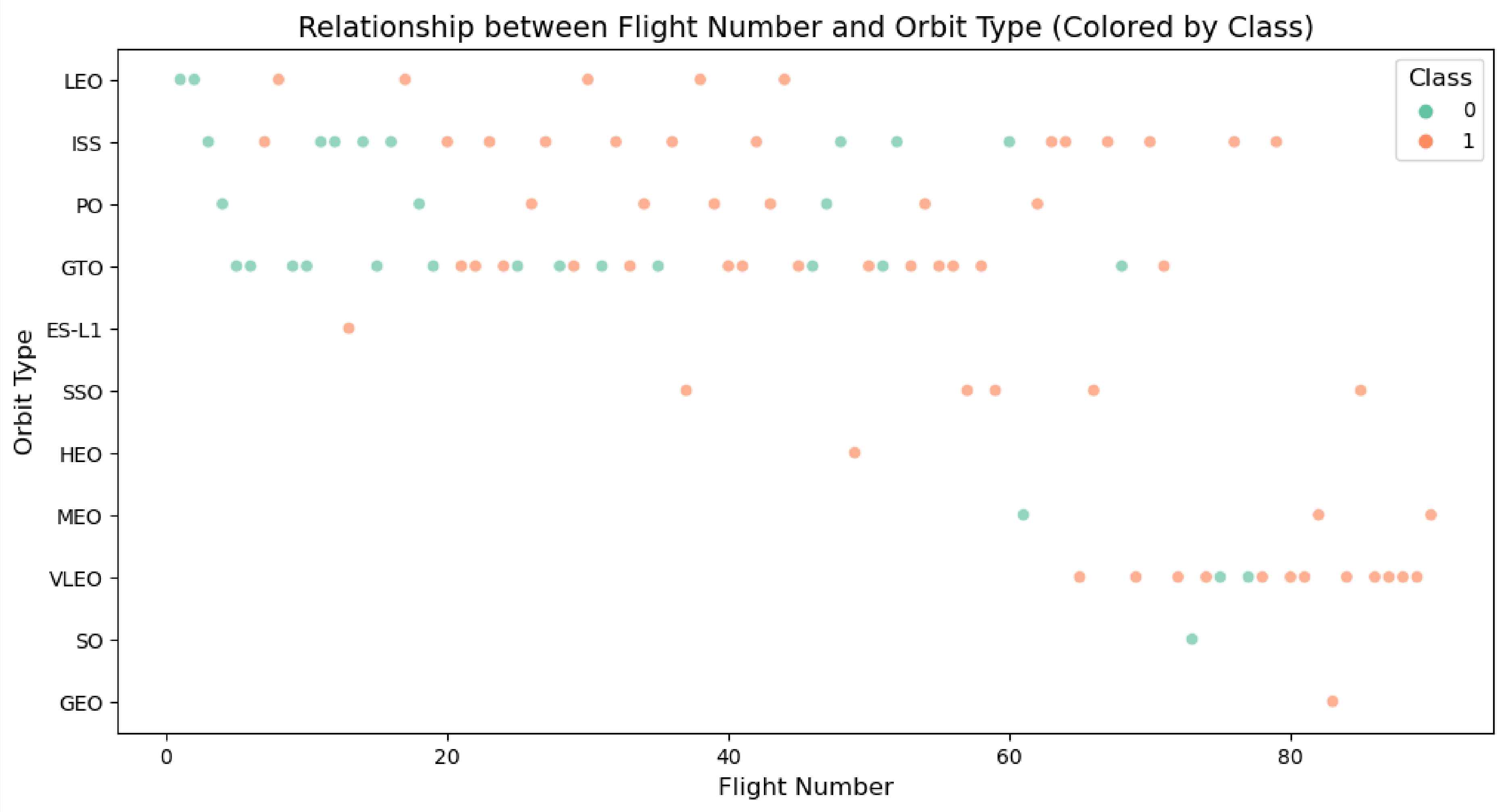
At the VAFB-SLC launch site, no rockets have been launched with a heavy payload mass exceeding 10,000.

Success Rate vs. Orbit Type



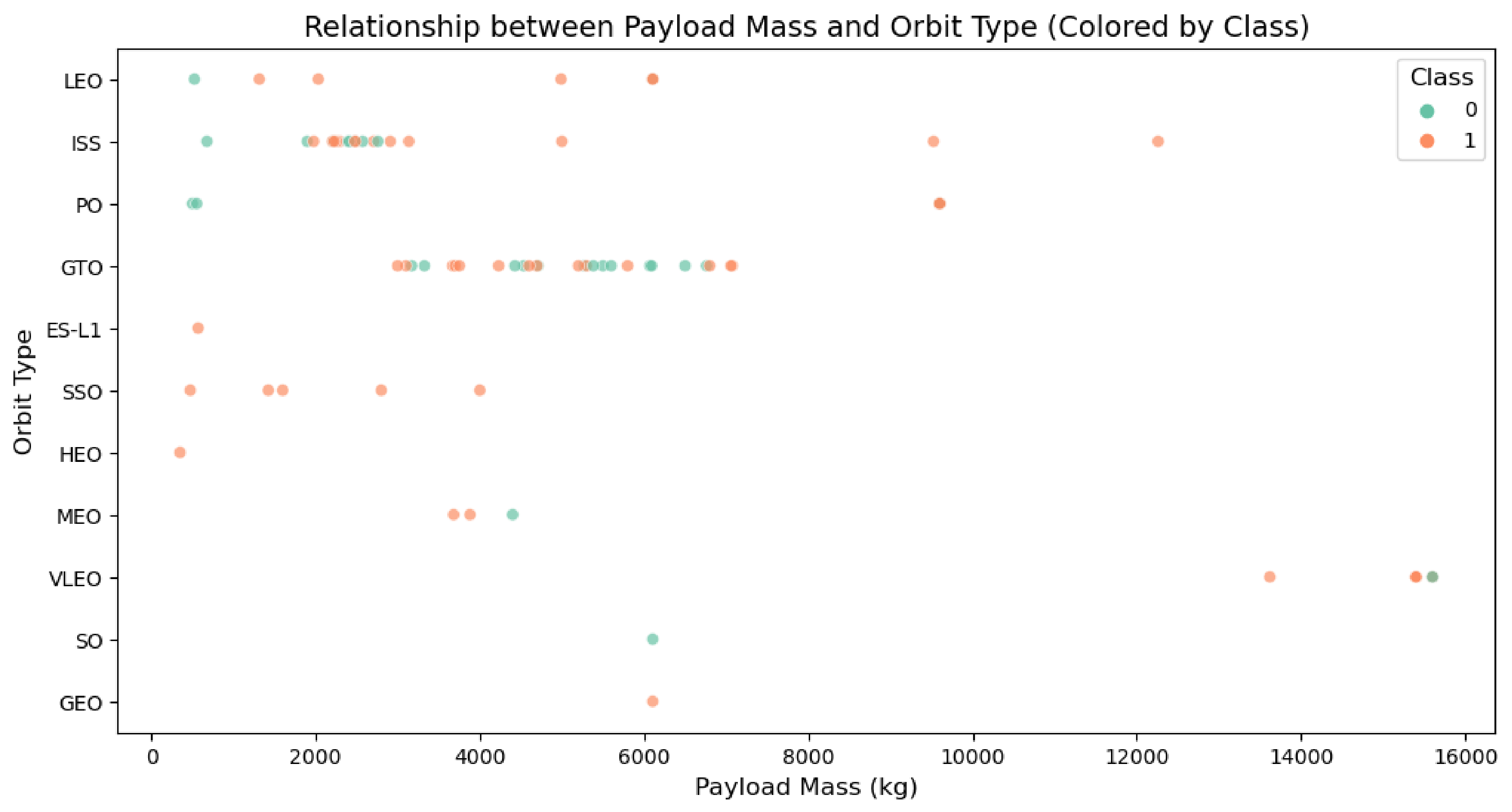
The ES-L1, GEO, HEO, and SSO orbits exhibit the highest success rates.

Flight Number vs. Orbit Type



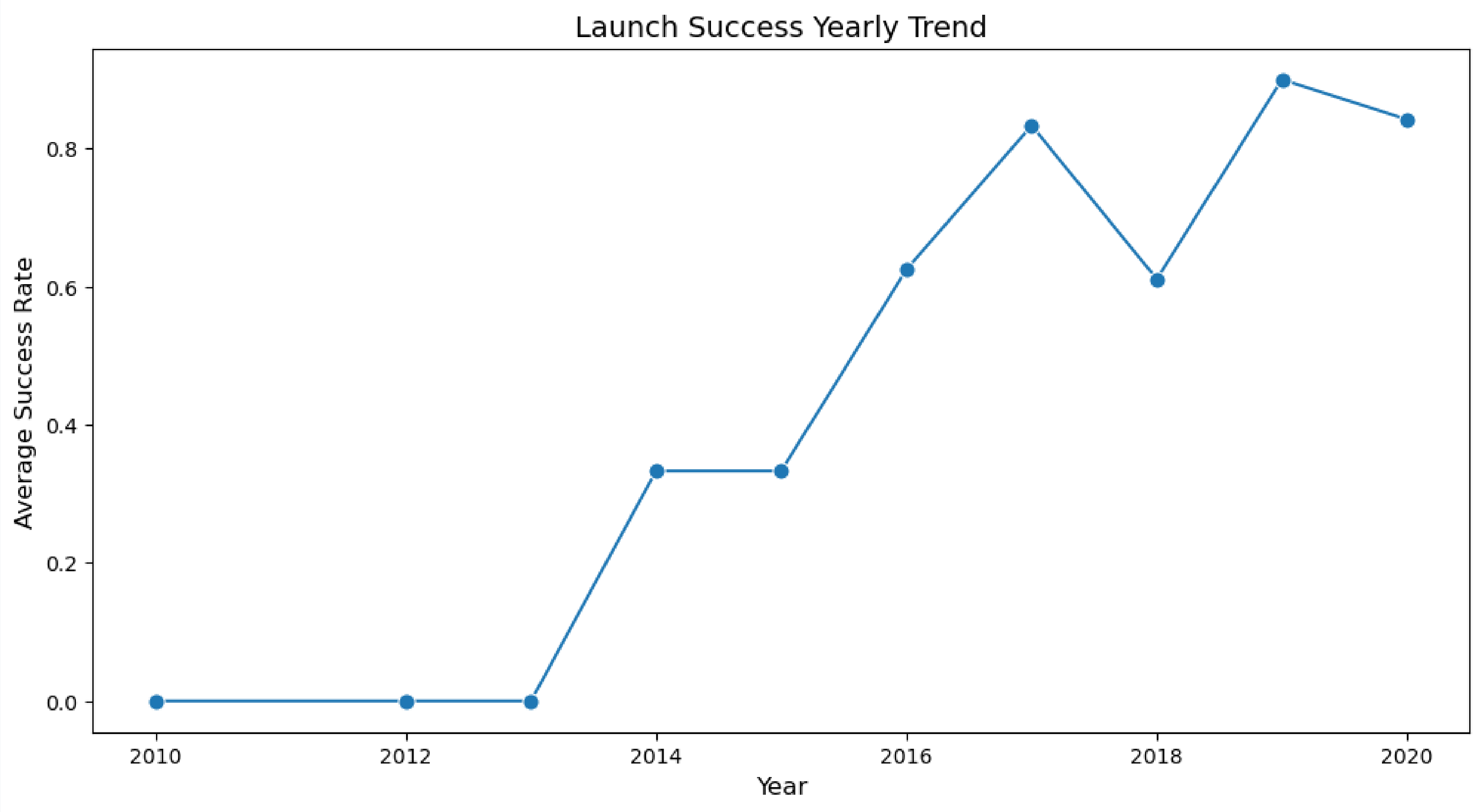
If the number of flights exceeds 80, the success rate is 100%.

Payload vs. Orbit Type



For heavy payloads, Polar, LEO, and ISS orbits show a higher rate of successful landings. However, distinguishing the landing outcomes is less clear for GTO, as both positive landing rates and negative landing (unsuccessful mission) rates are observed in this orbit.

Launch Success Yearly Trend



The success rate has been consistently increasing from 2013 until 2020.

All Launch Site Names

```
%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

The SQL query %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE; retrieves a list of unique launch site names from the "Launch_Site" column in the "SPACEXTABLE" table. The use of DISTINCT ensures that only distinct values are returned, eliminating any duplicates in the result set. This query is designed to provide a clear and unique list of launch sites mentioned in the specified column of the table.

Launch Site Names Begin with 'KSC'

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

```
* 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)
2017-03-16	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

The SQL query %sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'KSC%' LIMIT 5; retrieves up to five rows from the "SPACEXTABLE" table where the values in the "Launch_Site" column start with 'KSC'. The LIKE operator, combined with the percent sign (%), is used for pattern matching, and in this case, it filters records where the launch site name begins with 'KSC'. The LIMIT 5 statement ensures that only a maximum of five results are returned. This query provides a snapshot of the first five records in the table that meet the specified criteria related to the launch site.

Total Payload Mass

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

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

TotalPayloadMass
45596

The SQL query %sql SELECT SUM(PAYLOAD_MASS__KG_) AS TotalPayloadMass FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)' calculates the total payload mass, represented by the "PAYLOAD_MASS__KG_" column, for missions where the customer is identified as 'NASA (CRS)' in the "SPACEXTABLE" table. The SUM function is used to aggregate the payload masses of all relevant rows, and the result is given an alias "TotalPayloadMass" for clarity. This query allows for the retrieval of the cumulative payload mass specifically associated with NASA Commercial Resupply Service (CRS) missions in the dataset.

Average Payload Mass by F9 v1.1

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

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

AveragePayloadMass
2928.4

The SQL query %sql SELECT AVG(PAYLOAD_MASS__KG_) AS AveragePayloadMass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1' calculates the average payload mass, represented by the "PAYLOAD_MASS__KG_" column, for missions where the booster version is specified as 'F9 v1.1' in the "SPACEXTABLE" table. The AVG function is used to compute the average of payload masses across all relevant rows, and the result is assigned the alias "AveragePayloadMass" for clarity. This query provides the mean payload mass specifically for missions utilizing the Falcon 9 (v1.1) booster version in the dataset.

First Successful Ground Landing Date

```
%sql SELECT MIN(Date) AS MinLandingDate FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)'
```

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

MinLandingDate
2016-04-08

The SQL query %sql SELECT MIN(Date) AS MinLandingDate FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' retrieves the minimum (earliest) landing date, represented by the "Date" column, among missions where the landing outcome is specified as 'Success (drone ship)' in the "SPACEXTABLE" table. The MIN function is used to find the minimum date value in the subset of rows that meet the specified landing outcome criteria. The result is given the alias "MinLandingDate" for clarity, providing the earliest successful drone ship landing date recorded in the dataset.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)' AND PAYLOAD_MASS__KG_ > 4000 ANI
```

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

Booster_Version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

The SQL query %sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000 retrieves the "Booster_Version" values from the "SPACEXTABLE" table where the landing outcome is specified as 'Success (ground pad)' and the payload mass falls within the range of 4000 to 6000 kilograms. This query filters the dataset to specifically identify booster versions associated with successful ground pad landings and payloads within the specified mass range.

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT Mission_Outcome, COUNT(*) AS TotalMissions FROM SPACEXTABLE GROUP BY Mission_Outcome;
```

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

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

The SQL query %sql SELECT Mission_Outcome, COUNT(*) AS TotalMissions FROM SPACEXTABLE GROUP BY Mission_Outcome; retrieves a summary of mission outcomes and their respective counts from the "SPACEXTABLE" table. The COUNT(*) function is used to calculate the total number of occurrences for each unique value in the "Mission_Outcome" column. The GROUP BY Mission_Outcome statement ensures that the counting operation is performed for each distinct mission outcome. The result will show a list of unique mission outcomes alongside the total count of missions associated with each outcome in the dataset.

Boosters Carried Maximum Payload

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

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

Activate Windows

The SQL query %sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE); retrieves the "Booster_Version" values from the "SPACEXTABLE" table for missions where the payload mass is equal to the maximum payload mass found in the entire dataset. The subquery (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE) calculates the maximum payload mass, and the outer query filters records where the payload mass matches this maximum value. This query identifies the booster version associated with the mission(s) having the highest payload mass in the dataset.

2017 Launch Records

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

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

Month	Landing_Outcome	Booster_Version	Launch_Site
02	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
05	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
06	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
08	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
09	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
12	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40

The SQL query %sql SELECT substr(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr(Date, 0, 5) = '2017' AND Landing_Outcome = 'Success (ground pad)'; retrieves a subset of data from the "SPACEXTABLE" table for missions that occurred in the year 2017 and had a landing outcome specified as 'Success (ground pad)'. The substr(Date, 6, 2) extracts the month information from the "Date" column, and the selected columns include the extracted month, landing outcome, booster version, and launch site for these specific missions.

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

```
%sql SELECT Landing_Outcome, COUNT(*) AS OutcomeCount FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROU
```

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

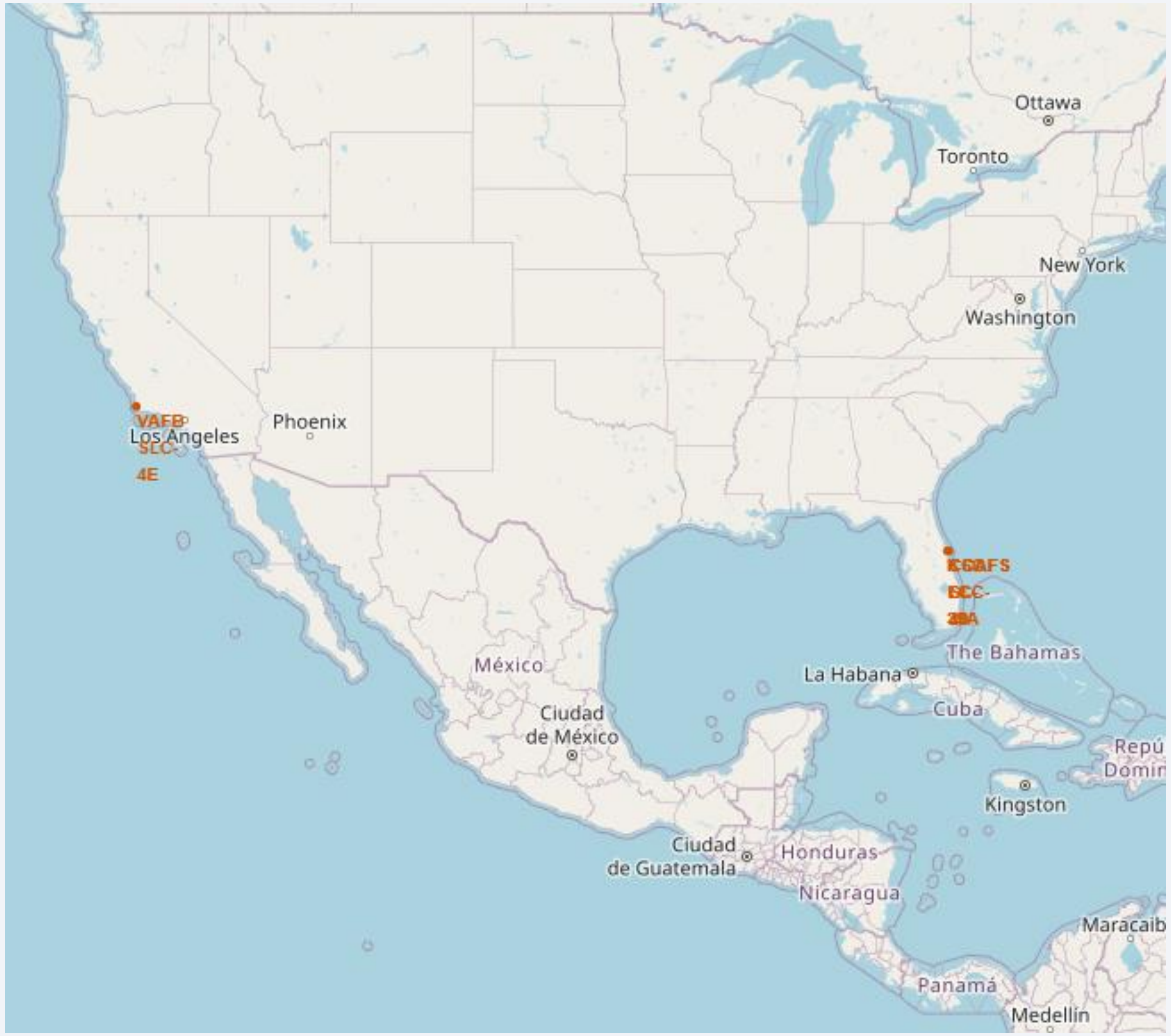
Landing_Outcome	OutcomeCount
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

The SQL query %sql SELECT Landing_Outcome, COUNT(*) AS OutcomeCount FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY OutcomeCount DESC; retrieves a summary of landing outcomes and their respective counts from the "SPACEXTABLE" table for missions that occurred between June 4, 2010, and March 20, 2017. The COUNT(*) function calculates the total number of occurrences for each unique value in the "Landing_Outcome" column, and the GROUP BY Landing_Outcome ensures the counting operation is performed for each distinct landing outcome. The result is then ordered in descending order based on the count of each landing outcome, providing a ranking of outcomes from the most to the least frequent during the specified time range.

Section 3

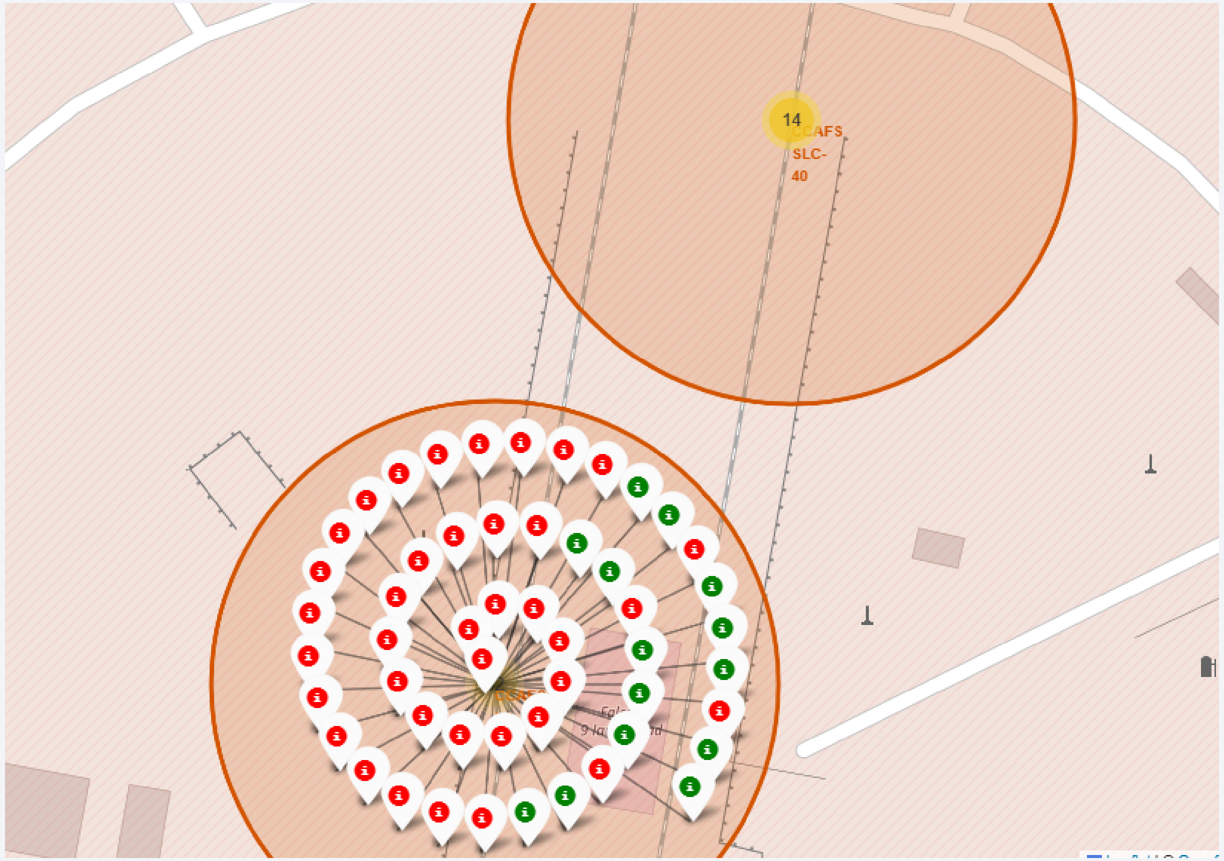
Launch Sites Proximities Analysis

All Launch Sites



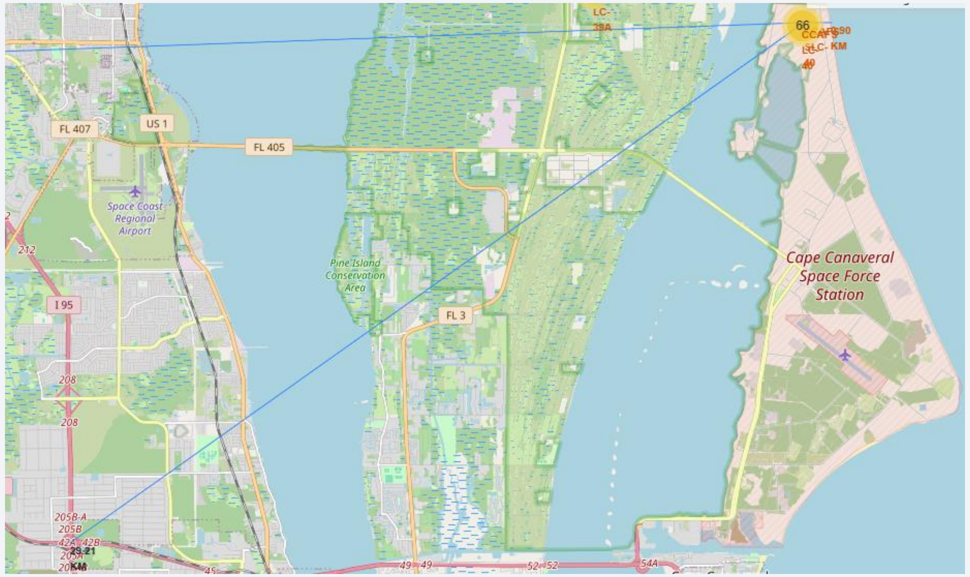
All launch sites are situated in close proximity to the coast and within restricted areas.

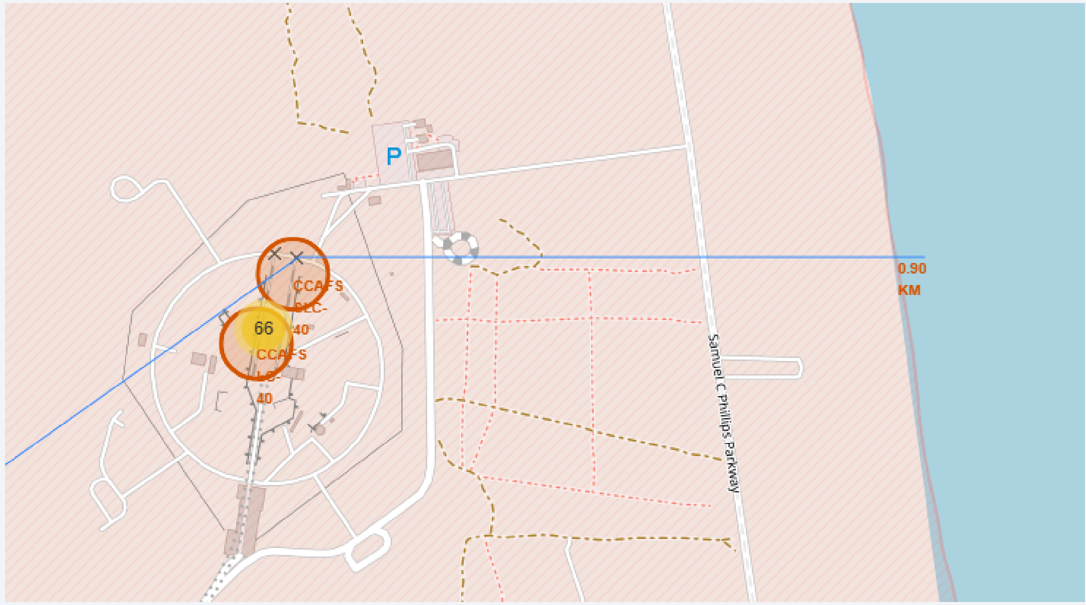
Success/Failed Launches For Each Site



Green marker indicates that a launch was successful. Otherwise, red marker means that the launch was failed.

A Launch Site and its Proximities



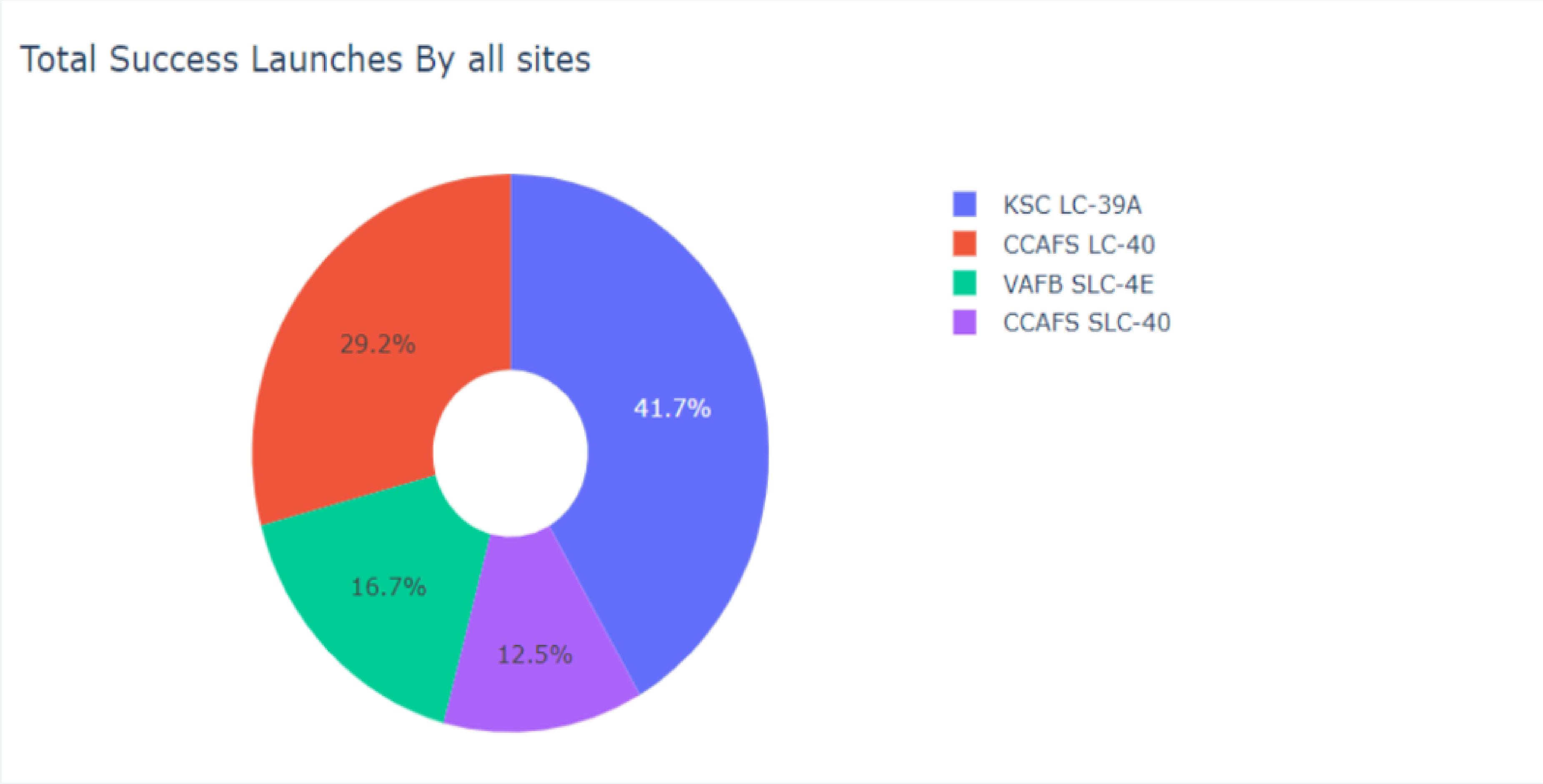


Launch sites are near to railways, roads, highways and coastline.

Section 4

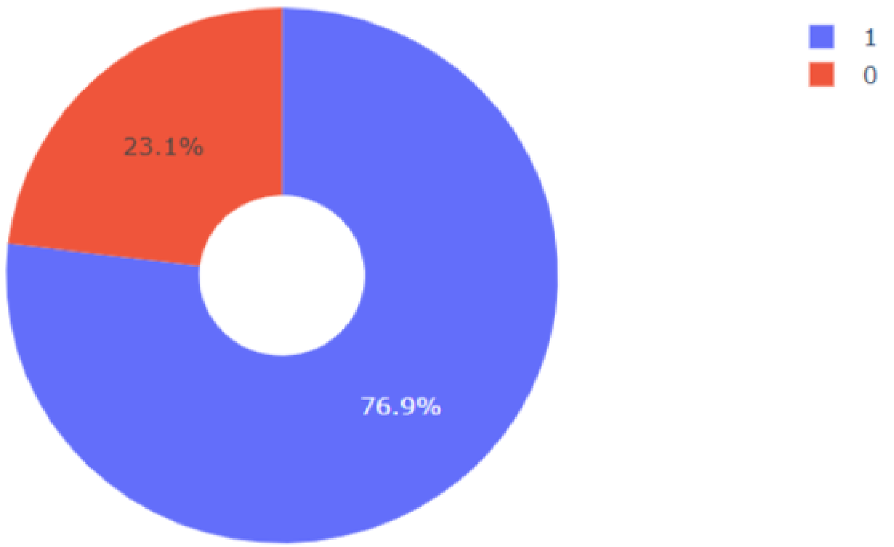
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



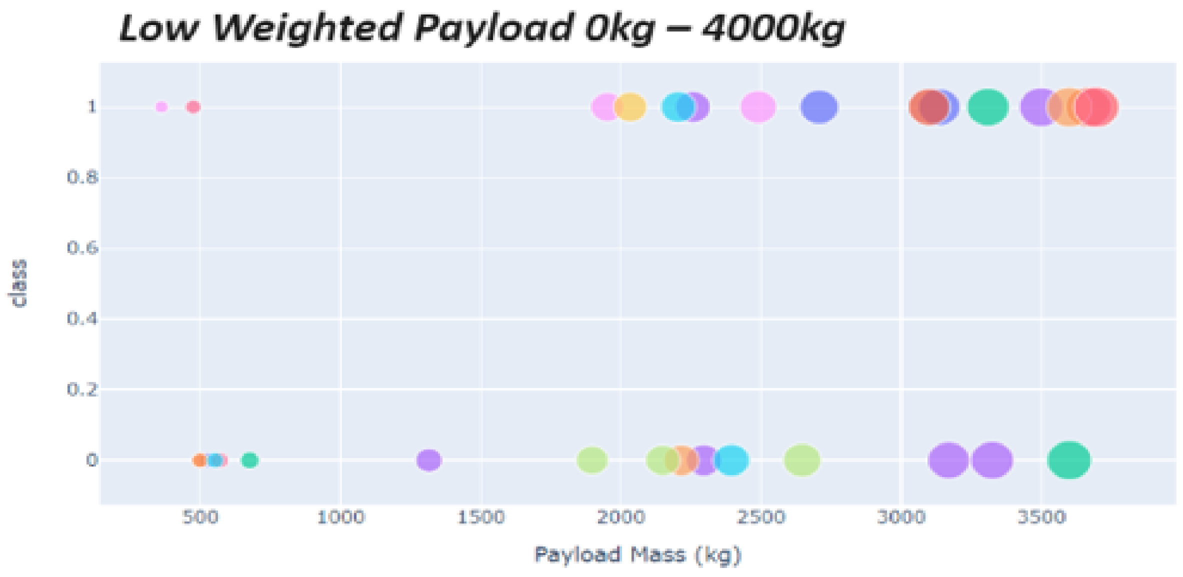
Based on chart, we conclude that KSC LC-39A has the highest number of successful launches among all the launch sites.

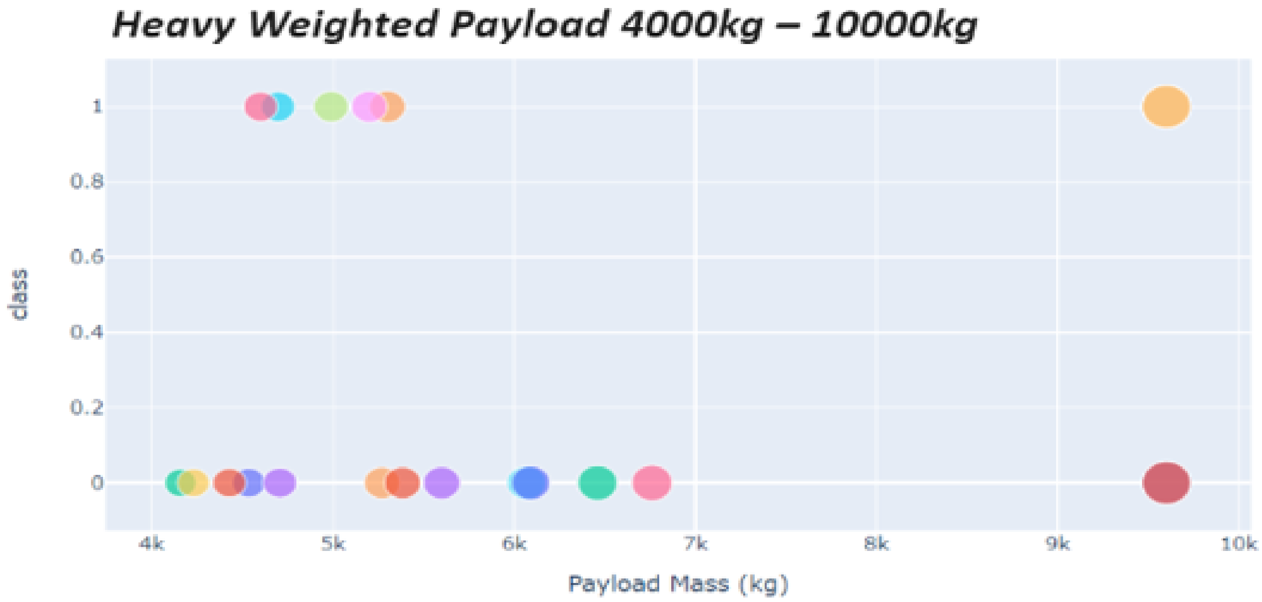
Pie chart showing the Launch site with the highest launch success ratio



The success rate for launches from KSC LC-39A is 76.9%.

Scatter plot of Payload vs Launch Outcome for all sites, With different payload selected in the range slider



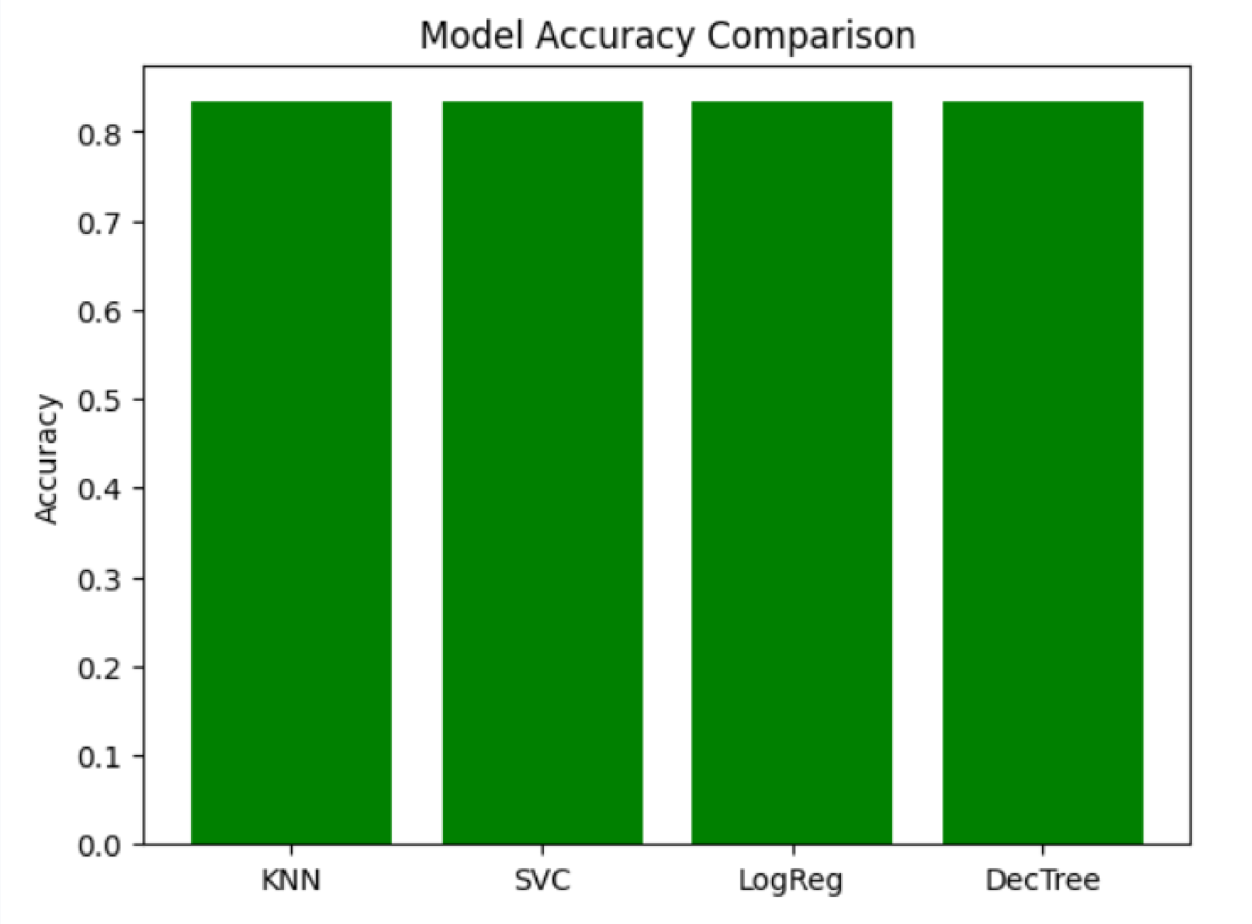


The analysis indicates that the success rate is higher for missions with lower payload weights compared to those with higher payload weights.

Section 5

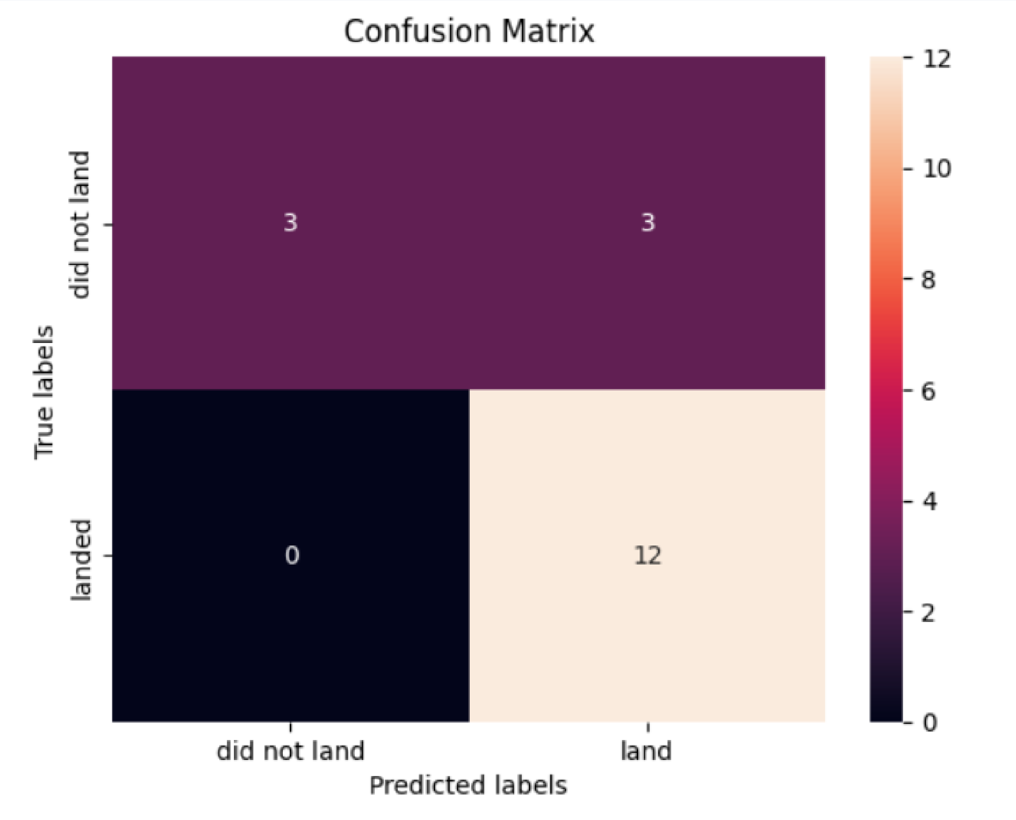
Predictive Analysis (Classification)

Classification Accuracy



All machine learning algorithms tested, including KNN, Logistic Regression, Support Vector Machine (SVM), and Decision Tree, exhibited identical accuracy levels for this task.

Confusion Matrix



The 4 models show strength in true positive predictions and true negatives, there is room for improvement in reducing false negatives to enhance overall performance, depending on the specific goals and requirements of the classification task.

Conclusions

1. **Launch Site Flight Amount Impact:** An observed trend indicates that a launch site's success rate is positively correlated with the volume of flights conducted at that site. In other words, sites with more frequent launches tend to have higher success rates.
2. **Temporal Success Rate Trend:** The launch success rate exhibited an upward trajectory starting in 2013 and continued until 2020. This temporal analysis suggests an improvement in overall launch success during this period.
3. **Orbit-specific Success Rates:** Among various orbits, including ES-L1, GEO, HEO, SSO, and VLEO, launches into these specific orbits demonstrated notably higher success rates. This highlights the significance of considering the target orbit when assessing launch success.
4. **Top Performing Launch Site:** KSC LC-39A stands out as the launch site with the most successful launches. This site consistently achieved a high level of success, indicating its reliability in hosting successful launch missions.
5. **Machine Learning Algorithm Performance:** All machine learning algorithms tested, including KNN, Logistic Regression, Support Vector Machine (SVM), and Decision Tree, exhibited identical accuracy levels for this task. This suggests that, within the context of this specific predictive task, these algorithms perform equally well and yield the same level of accuracy. Consequently, the choice of algorithm may be based on other considerations such as interpretability, computational efficiency, or specific requirements of the task at hand, given the absence of discernible accuracy differences among these models.

Appendix

- For notebooks and scripts, follow this GitHub repository link.
- https://github.com/WarDen-Git/testrepo

Thank you!