IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Manuel Anibal Mora Jiménez
16/7/2024

Outline

Executive Summary

- This project analyzed Falcon 9 launch data by collecting it from Wikipedia and the SpaceX API. Exploratory data analysis identified landing success patterns using pandas, matplotlib, and seaborn. Data was standardized and one-hot encoded for predictive modeling. Four machine learning algorithms (Logistic Regression, SVM, Decision Trees, KNN) were trained and optimized using GridSearchCV. The best model was selected based on test accuracy. An interactive dashboard with Plotly Dash was created to visualize results and explore key variable relationships.
- The analysis identified the launch sites with the highest success rates and the most favorable payload mass ranges. The best-performing machine learning model was K-Nearest Neighbors, with an 83.33% accuracy in predicting landing success. The interactive dashboard allows users to explore the relationships between launch site, payload mass, booster version, and landing outcome, providing valuable insights for optimizing SpaceX's launch operations.

Introduction

- The space industry is constantly evolving, and SpaceX has become a leader in democratizing access to space. This project explores how machine learning can help optimize SpaceX's launch operations by predicting the landing success of the Falcon 9's first stage. By analyzing historical launch data, key patterns determining landing success were identified, and a machine learning model was developed to accurately predict this crucial outcome.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - **Web Scraping**: Data was extracted from the Wikipedia page "List of Falcon 9 and Falcon Heavy launches" using BeautifulSoup and requests, retrieving details like date, launch site, payload, booster version, and landing outcome.
  - **SpaceX API**: The SpaceX API was used to supplement Wikipedia data, providing additional information such as payload mass, orbit, and booster flight numbers, with requests used to obtain JSON responses.
- 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

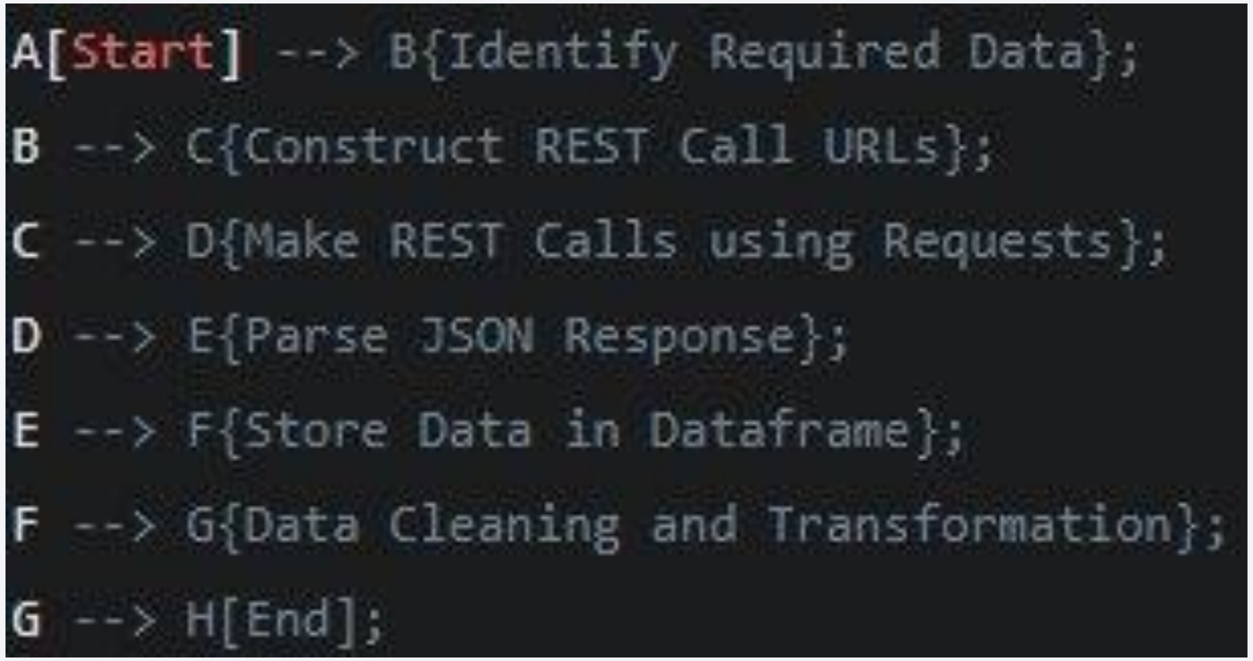
The collected data was processed for machine learning model training, involving:

- **Data Cleaning**: Removing rows with missing or inconsistent data and correcting variable coding errors.
- **Data Preparation**: Standardizing numerical variables for consistent scaling and applying one-hot encoding to convert categorical variables (e.g., launch site, orbit, booster version) into numerical format.

This cleaning and preparation process resulted in a clean and consistent dataset for training the machine learning models.

Data Collection – SpaceX API

- Key Phrases:
- SpaceX REST API
- REST Calls
- JSON
- Requests library:
- https://github.com/morajimenez18/Capstone.git

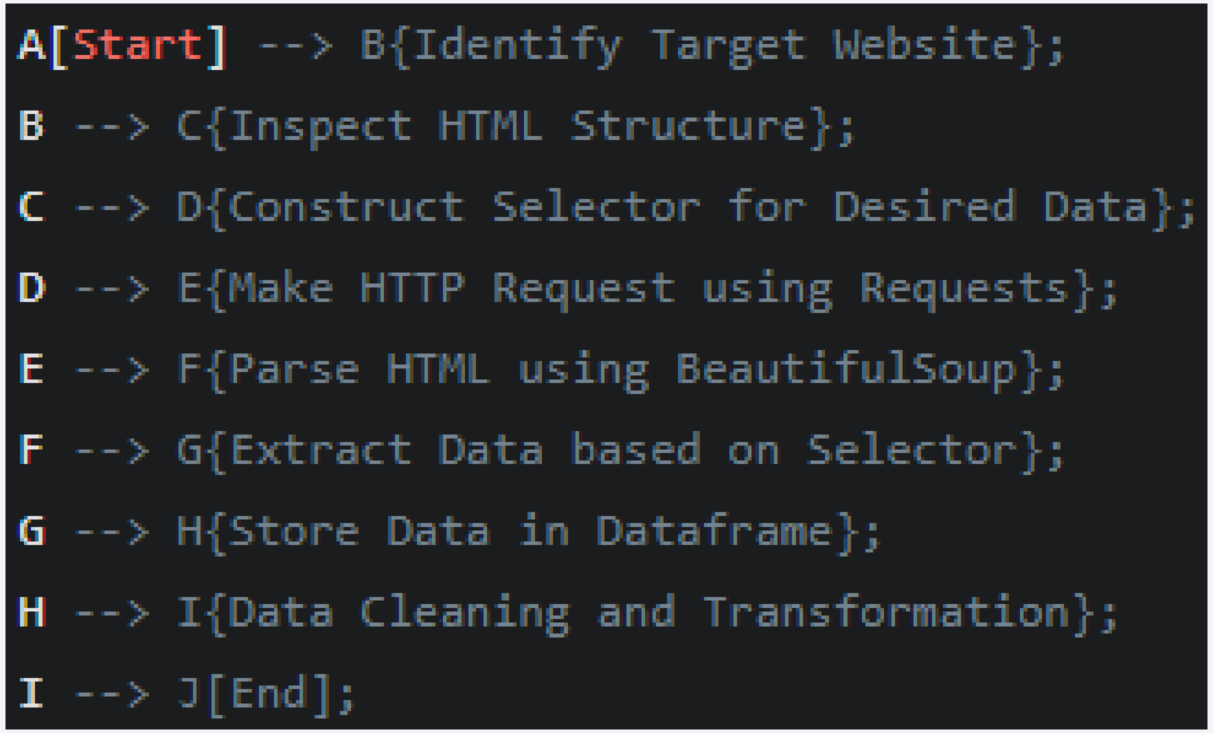


```
A[Start] --> B{Identify Required Data};
B --> C{Construct REST Call URLs};
C --> D{Make REST Calls using Requests};
D --> E{Parse JSON Response};
E --> F{Store Data in Dataframe};
F --> G{Data Cleaning and Transformation};
G --> H[End];
```

Data Collection - Scraping

- Key Phrases:
- Web Scraping
- HTML
- BeautifulSoup library
- Requests library

- https://github.com/morajimenez18/Capstone.git



```
A[Start] --> B{Identify Target Website};
B --> C{Inspect HTML Structure};
C --> D{Construct Selector for Desired Data};
D --> E{Make HTTP Request using Requests};
E --> F{Parse HTML using BeautifulSoup};
F --> G{Extract Data based on Selector};
G --> H{Store Data in Dataframe};
H --> I{Data Cleaning and Transformation};
I --> J[End];
```

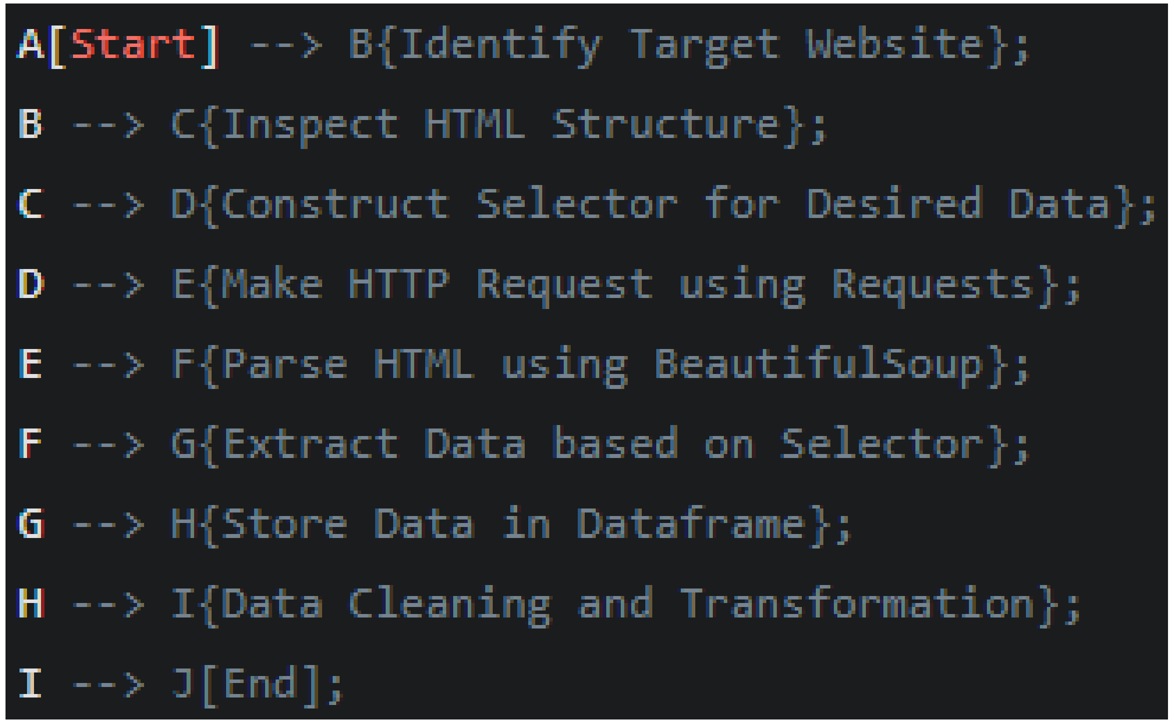
Data Wrangling

- The collected data was processed for machine learning model training, involving:
- **Data Cleaning**: Rows with missing or inconsistent data were removed, and errors in variable coding were corrected.
- **Data Preparation**: Numeric variables were standardized to ensure consistent scaling, and one-hot encoding was applied to convert categorical variables (such as launch site, orbit, and booster version) into a numerical format interpretable by machine learning models.
- This cleaning and preparation process resulted in a clean and consistent dataset for training the machine learning models.

Data Wrangling

- Key Phrases
- Data Wrangling
- Data Cleaning
- Data Transformation
- Pandas library

- https://github.com/morajimenez18/Capstone.git



```
A[Start] --> B{Identify Target Website};
B --> C{Inspect HTML Structure};
C --> D{Construct Selector for Desired Data};
D --> E{Make HTTP Request using Requests};
E --> F{Parse HTML using BeautifulSoup};
F --> G{Extract Data based on Selector};
G --> H{Store Data in Dataframe};
H --> I{Data Cleaning and Transformation};
I --> J[End];
```

EDA with Data Visualization

Several graphs were created to visually analyze the collected data and gain key insights.

- Scatter Plot (Flight Number vs Payload Mass)
- Scatter Plot (Launch Site vs Payload Mass)
- Bar Chart (Success Rate by Orbit Type)
- Scatter Plot (Flight Number vs Orbit)
- Scatter Plot (Payload Mass vs Orbit)
- Line Chart (Annual Success Rate Trend)
- Pie Chart (Success Rate by Launch Site)
- Scatter Plot (Payload Mass vs Class)

These graphs allowed for visualizing variable relationships, identifying patterns, discovering key insights, and making informed decisions for machine learning model training.

https://github.com/morajimenez18/Capstone.git

EDA with SQL

- **Get unique launch site names:**
- `SELECT DISTINCT "Launch_Site" FROM SPACEXTBL;`
- **Get 5 records where launch sites start with "CCA":**
- `SELECT * FROM SPACEXTBL WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;`
- **Get the total payload mass transported by NASA (CRS) boosters:**
- `SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass FROM SPACEXTBL WHERE "Customer" LIKE '%NASA (CRS)%';`
- **Get the average payload mass transported by F9 v1.1 boosters:**
- `SELECT AVG("PAYLOAD_MASS__KG_") AS Average_Payload_Mass FROM SPACEXTBL WHERE "Booster_Version" = 'F9 v1.1';`
- **Get the date of the first successful landing on a ground pad:**
- `SELECT MIN("Date") AS First_Successful_Landing FROM SPACEXTBL WHERE "Landing_Outcome" = 'Success (ground pad)';`

- **Get booster names that successfully landed on a drone ship and carried payload mass between 4000 kg and 6000 kg:**
- `SELECT DISTINCT "Booster_Version" FROM SPACEXTBL WHERE "Landing_Outcome" = 'Success (drone ship)' AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000;`
- **Get the total number of successful and failed mission outcomes:**
- `SELECT "Landing_Outcome", COUNT(*) AS Total_Outcomes FROM SPACEXTBL WHERE "Landing_Outcome" IN ('Success (ground pad)', 'Success (drone ship)', 'Success', 'Failure', 'Failure (parachute)', 'Uncontrolled (ocean)', 'Controlled (ocean)', 'Failure (drone ship)', 'Precluded (drone ship)', 'No attempt') GROUP BY "Landing_Outcome";`
- **Get booster versions that carried the maximum payload mass:**
- `SELECT "Booster_Version" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);`
- **Get month names, landing outcomes (failed on drone ship), booster versions, and launch sites for the year 2015:**
- `SELECT CASE WHEN substr("Date", 6, 2) = '01' THEN 'January' WHEN substr("Date", 6, 2) = '02' THEN 'February' ... END AS Month_Name, "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTBL WHERE substr("Date", 0, 5) = '2015' AND "Landing_Outcome" LIKE 'Failure (drone ship)%';`
- **Sort the count of landing outcomes ("Failure (drone ship)" or "Success (ground pad)") between dates 2010-06-04 and 2017-03-20 in descending order:**
- `SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count FROM SPACEXTBL WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' AND "Landing_Outcome" IN ('Failure (drone ship)', 'Success (ground pad)') GROUP BY "Landing_Outcome" ORDER BY Outcome_Count DESC;`

Build an Interactive Map with Folium

Map Objects:

- Markers: A total of 56 markers were created to represent each SpaceX launch on the map, referencing data from the spacex_df dataframe.
    - Color: Green markers were used for successful launches (class = 1), and red markers were used for failed launches (class = 0).
    - Purpose: To visually display the location of each launch and its outcome, allowing for easy identification of successful and unsuccessful launches on the map.
    - Clustering: The MarkerCluster plugin was used to group markers in areas with multiple launches. This significantly improved map readability, especially in regions with high launch density.
- Circles: A total of 4 circles were created, one for each launch site, based on data from the launch_sites_df dataframe.
    - Purpose: To visually highlight the approximate location of each launch site and emphasize their geographic area.
    - Radius: A radius of 1000 km was used for each circle to provide a clear visual representation of the site's general influence zone.
- Polylines: Lines were drawn between each launch site and nearby locations of interest, such as coastlines, railways, or highways. The user was guided to identify the nearest location using the MousePosition tool, and the distance was calculated accordingly.
    - Purpose: To visually determine the proximity of launch sites to various points of interest and analyze any potential patterns in their selection.
    - Distance: The distance between points was calculated using a calculate_distance() function.
    - Lines: A line was drawn between the launch site and the point of interest using folium.PolyLine(). Here's a code snippet:

Build a Dashboard with Plotly Dash

Dashboard Elements:

- Dropdown Input: A dropdown menu was added to allow the user to select a specific launch site.
  - Purpose: To enable the user to filter the data displayed in the charts by site, allowing for focused analysis of specific locations..
- Range Slider Input: A range slider was added to allow the user to select a range of payload mass values.
  - Purpose: To enable the user to explore the relationship between payload mass and launch success within a specific range. This helps identify potential trends and correlations for different payload weight classes.
- Pie Chart: A pie chart was created to visualize the total count of successful and failed launches.
  - Purpose: To provide an overview of the overall success rate for all launches or for a specific site selected by the user.
- Scatter Plot: A scatter plot was created to analyze the correlation between payload mass and launch success.
  - Purpose: To visually identify any trends or relationships between the payload mass and the outcome of the landing (success or failure). The user can also see the distribution of different booster versions by color.

Key Interactions:

- Site Selection: The dropdown menu allows users to explore the launch success rate and payload-success correlation for different launch sites.
- Payload Range Filtering: The range slider enables users to focus on specific payload mass ranges to identify trends and relationships that might not be visible when viewing all data.

Predictive Analysis (Classification)

- Select Classification Models:
  - Choose several suitable classification models to evaluate (e.g., Logistic Regression, Support Vector Machine, Decision Tree, K-Nearest Neighbors).
- Prepare Training and Test Data:
  - Split the cleaned and transformed data into training and test sets.
  - Ensure that the training data is used to train the model, and the test data is reserved for evaluating the model's performance on unseen data.
- Train Models with Default Hyperparameters:
  - Initialize each chosen classification model with its default hyperparameters.
  - Train each model on the training data.
- Evaluate Model Performance:
  - Evaluate the performance of each model on the test data using appropriate metrics (e.g., accuracy, precision, recall).
  - Analyze the initial performance to identify any strengths and weaknesses of each model.

- Optimize Hyperparameters using GridSearchCV:
  - Define a grid of hyperparameter values to explore.
  - Use GridSearchCV to train the models with different combinations of hyperparameters and evaluate their performance using cross-validation.
  - The GridSearchCV technique helps find the best hyperparameter combination for each model, maximizing its performance on the training data.
- Retrain Models with Optimized Hyperparameters:
  - Train each model again using the best hyperparameter combination identified by GridSearchCV.
- Evaluate Model Performance on Test Data:
  - Evaluate the performance of the retrained models on the test data using the chosen evaluation metrics.
  - Compare the performance of the models before and after hyperparameter optimization.
- Select Best Performing Model:
  - Choose the model that achieves the best performance on the test data based on the chosen evaluation metrics.
  - The model with the highest accuracy, precision, or recall is typically considered the best-performing model.

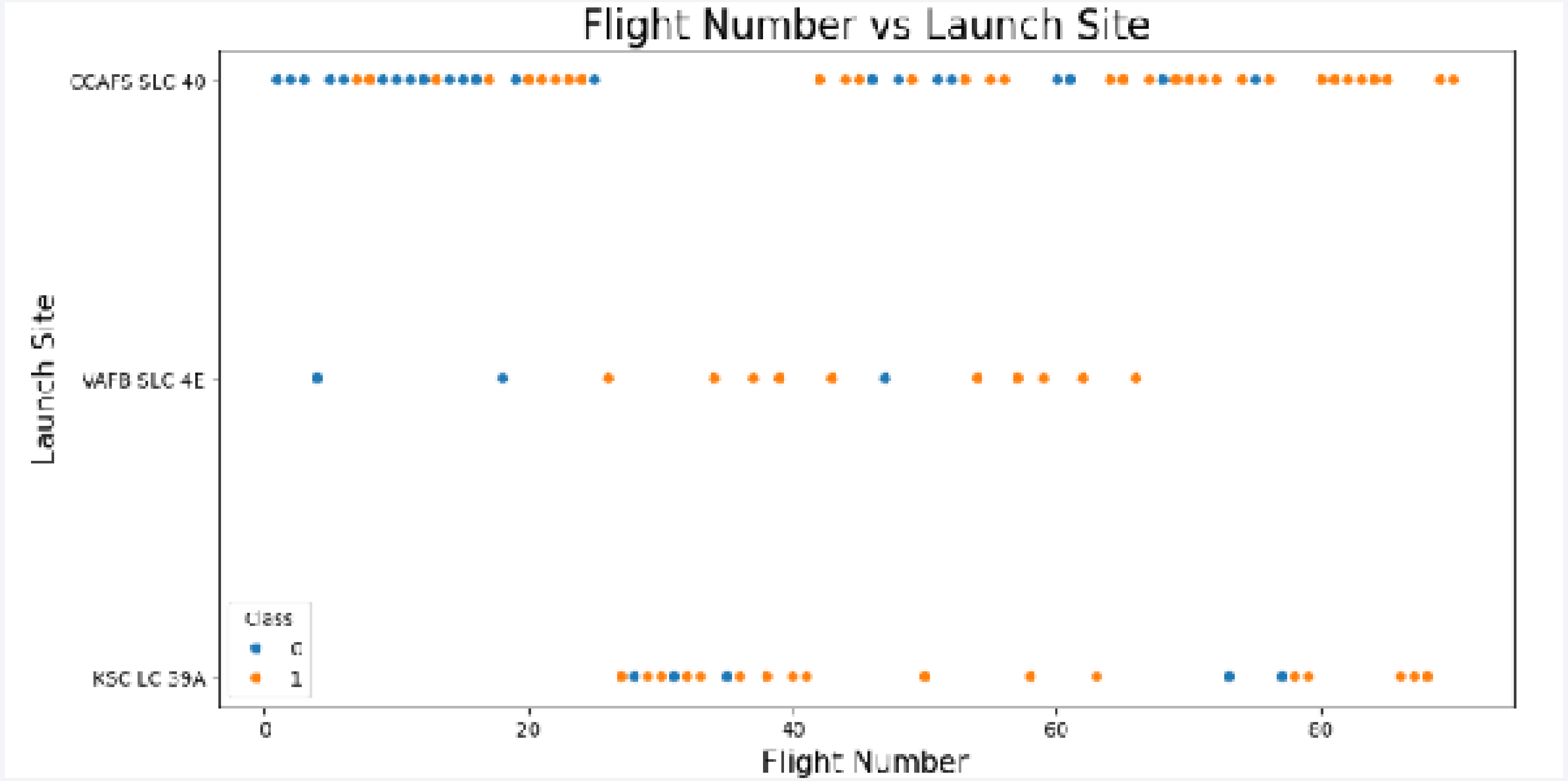
Results

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

Section 2

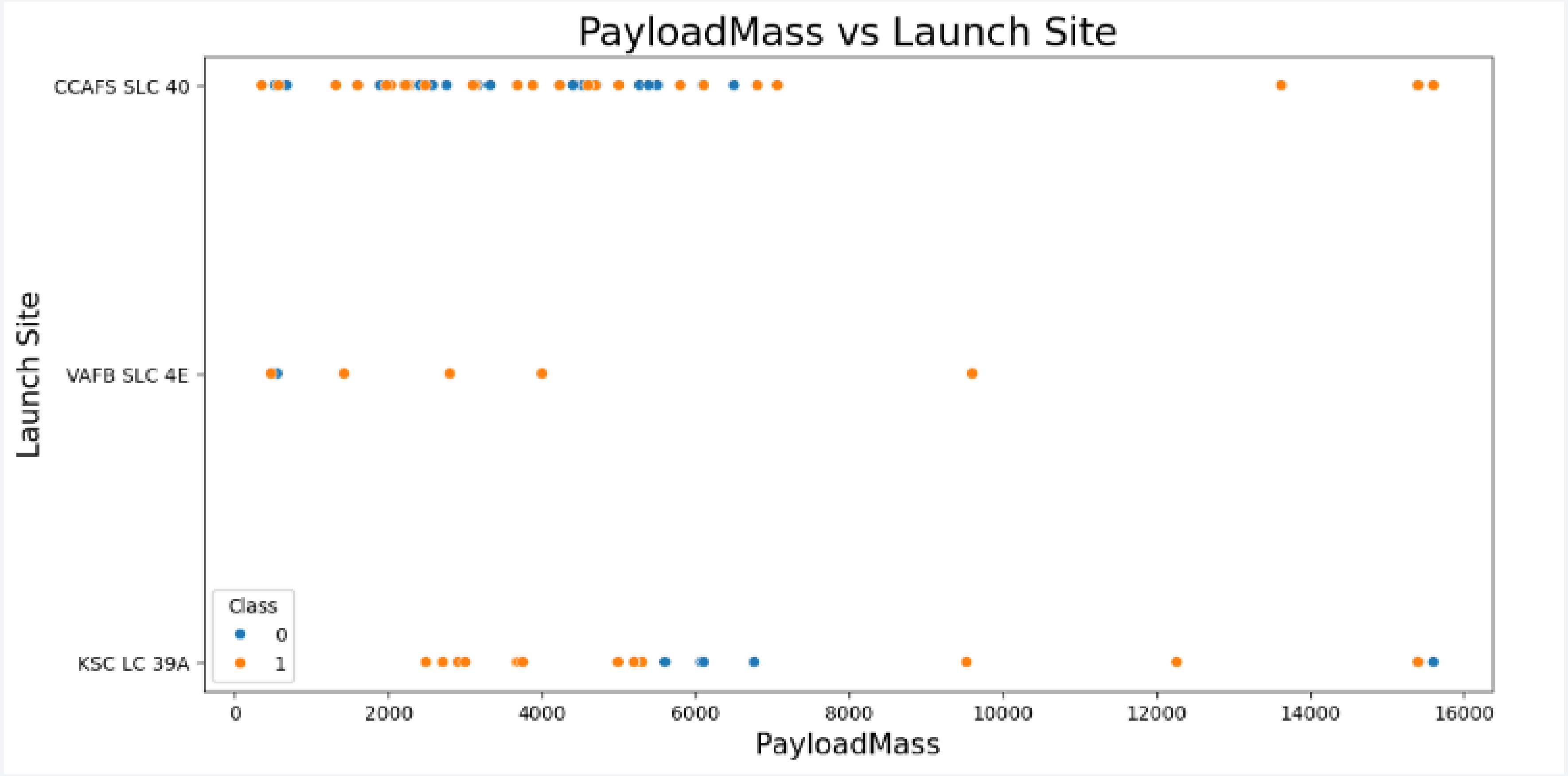
Insights drawn from EDA

Flight Number vs. Launch Site



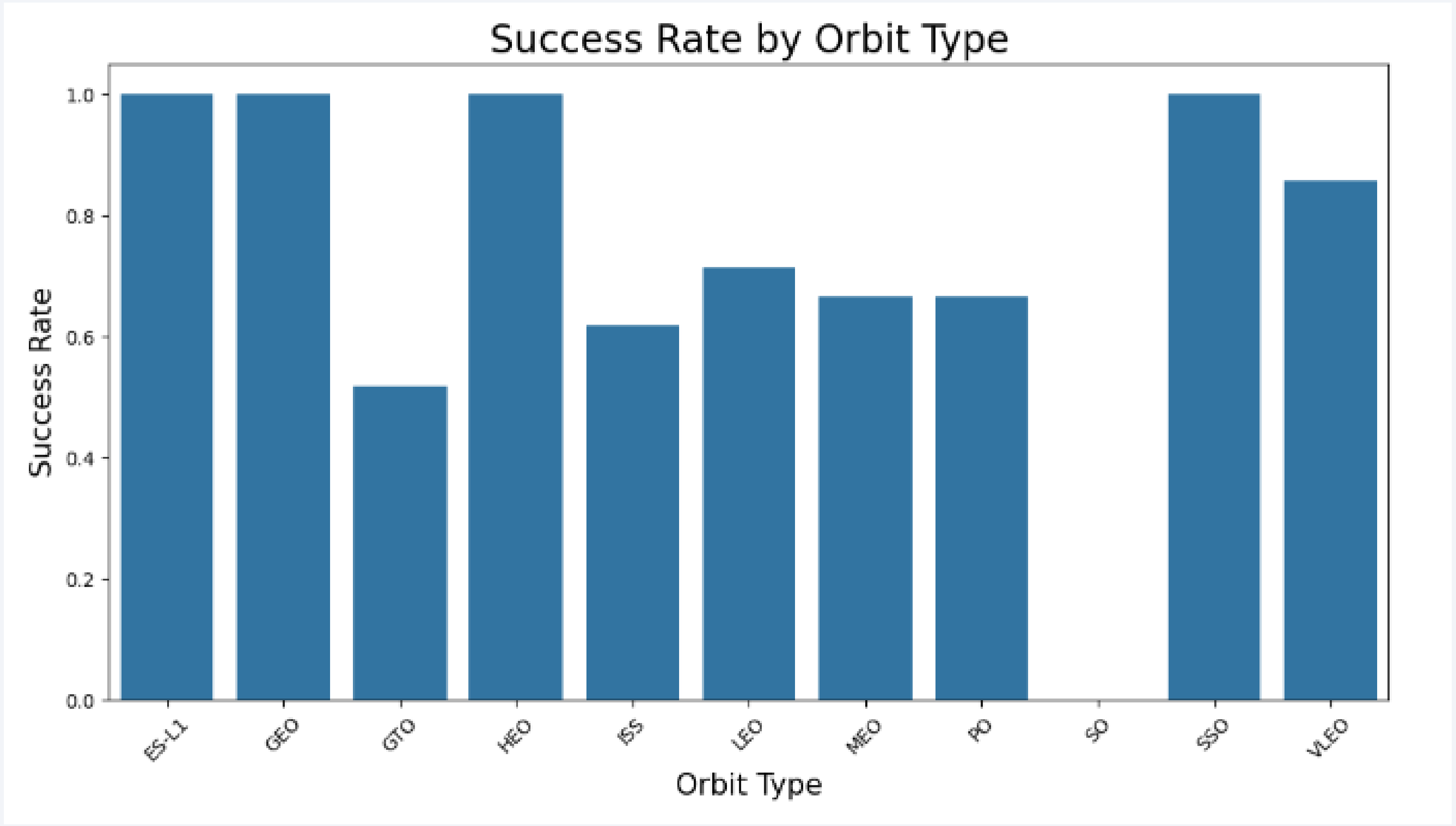
This scatter plot shows where each launch took off from as a function of flight number and the color indicates whether the first stage landing was successful.

Payload vs. Launch Site



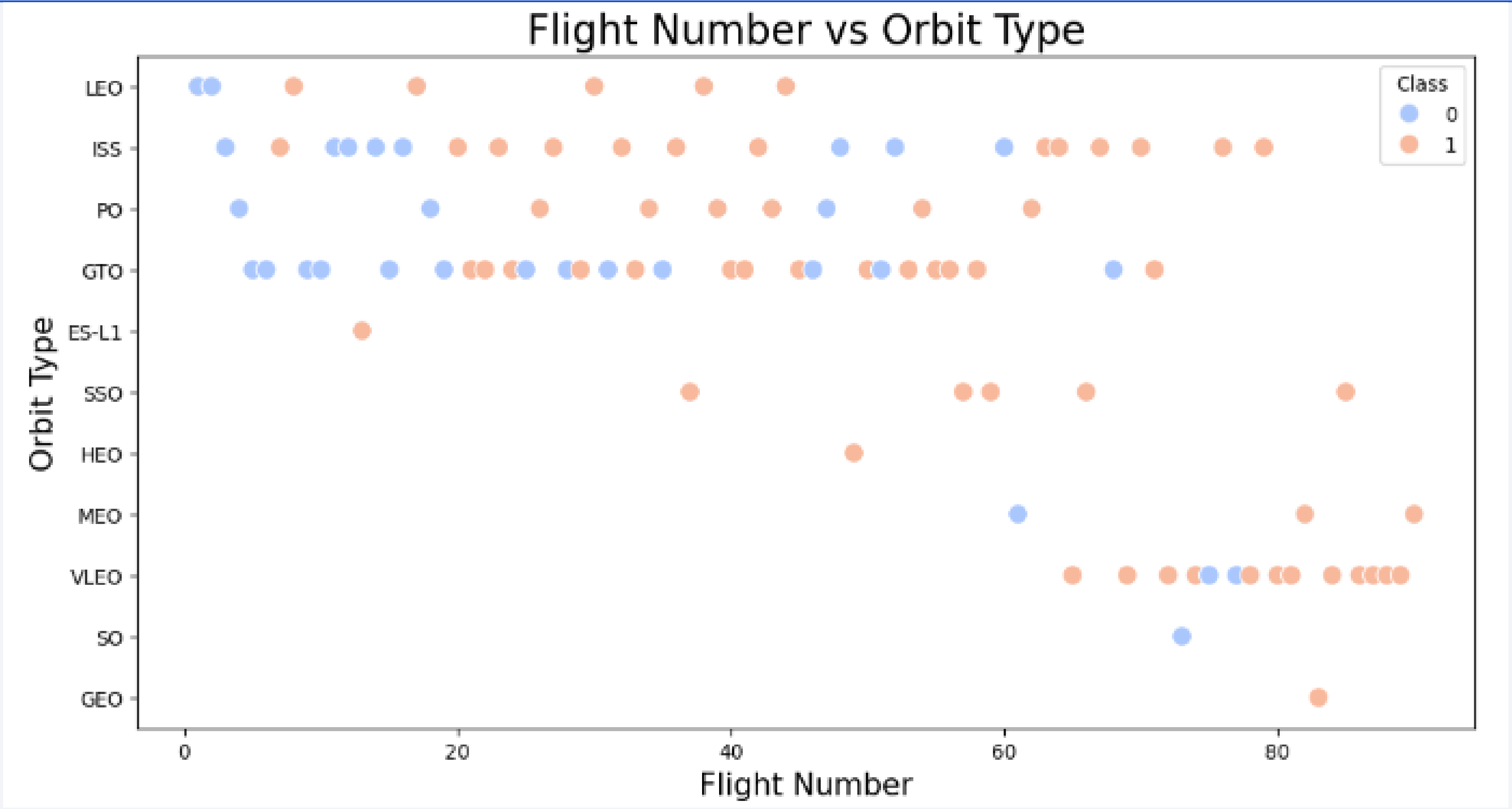
This scatter plot shows where each launch took off from as a function of payload mass, and the color indicates whether the first stage landing was successful.

Success Rate vs. Orbit Type



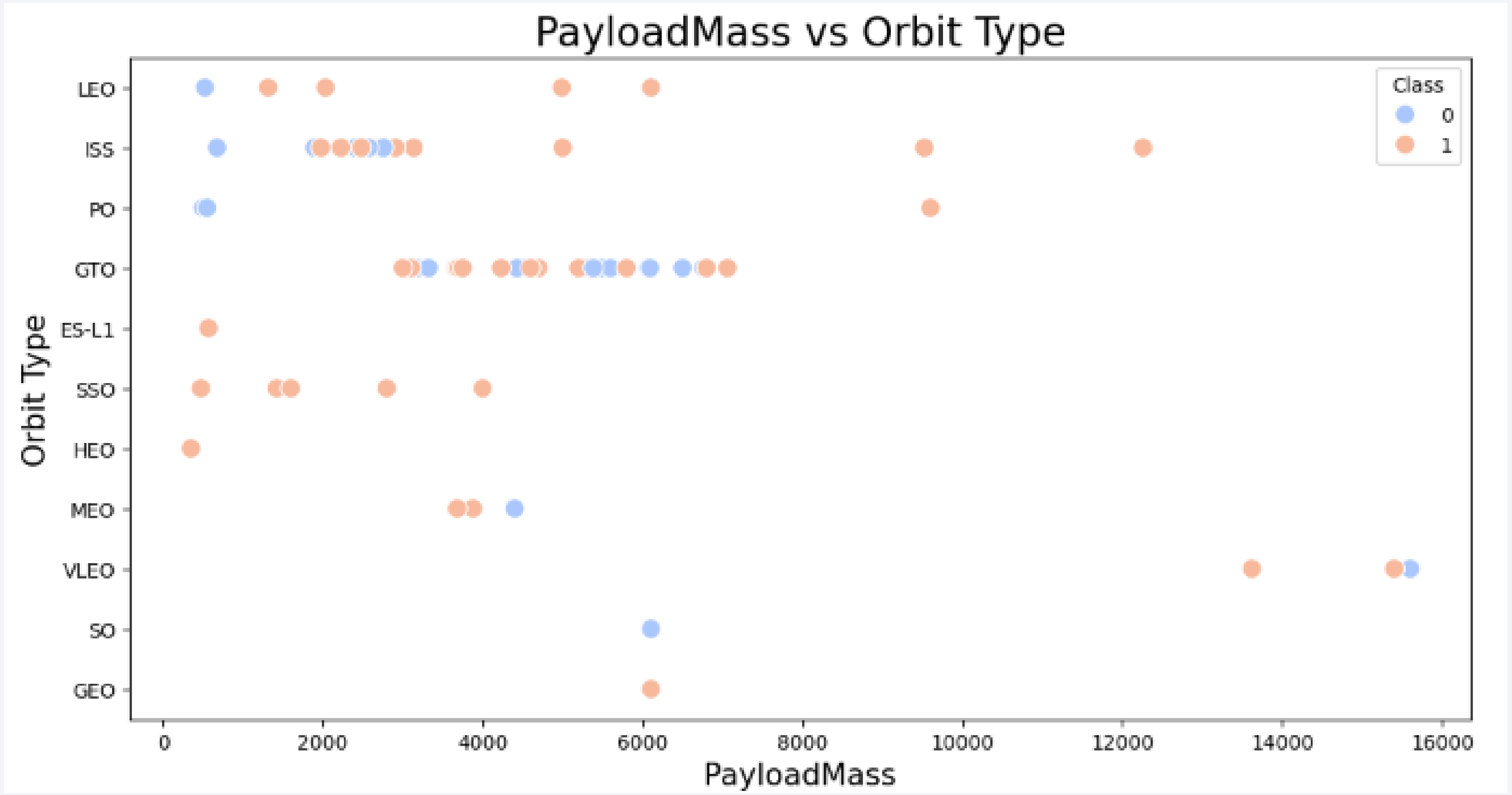
This bar chart shows the success rate of the different orbits.

Flight Number vs. Orbit Type



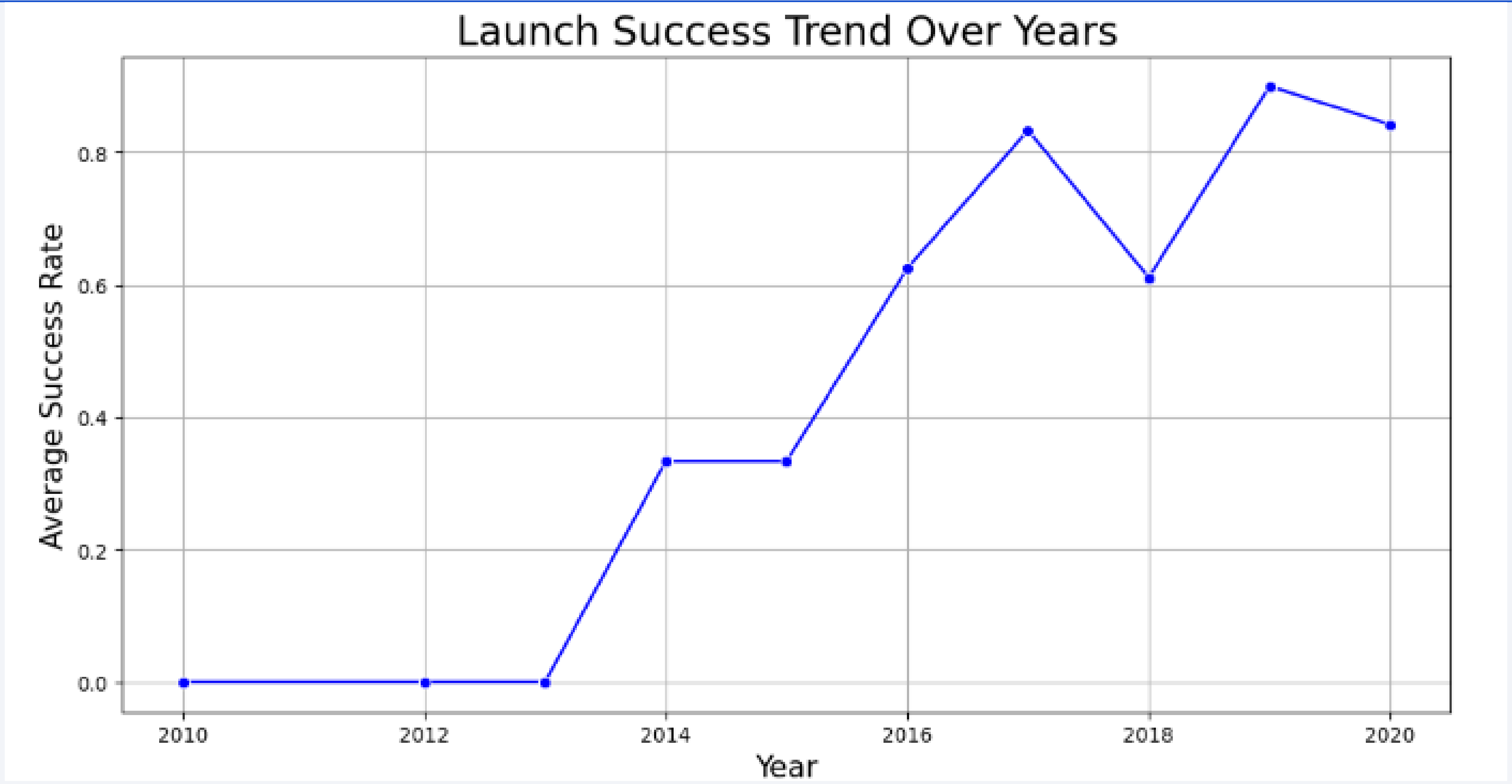
This scatter plot shows Orbit type as a function of Flight Number, and the color indicates whether the first stage landing was successful.

Payload vs. Orbit Type



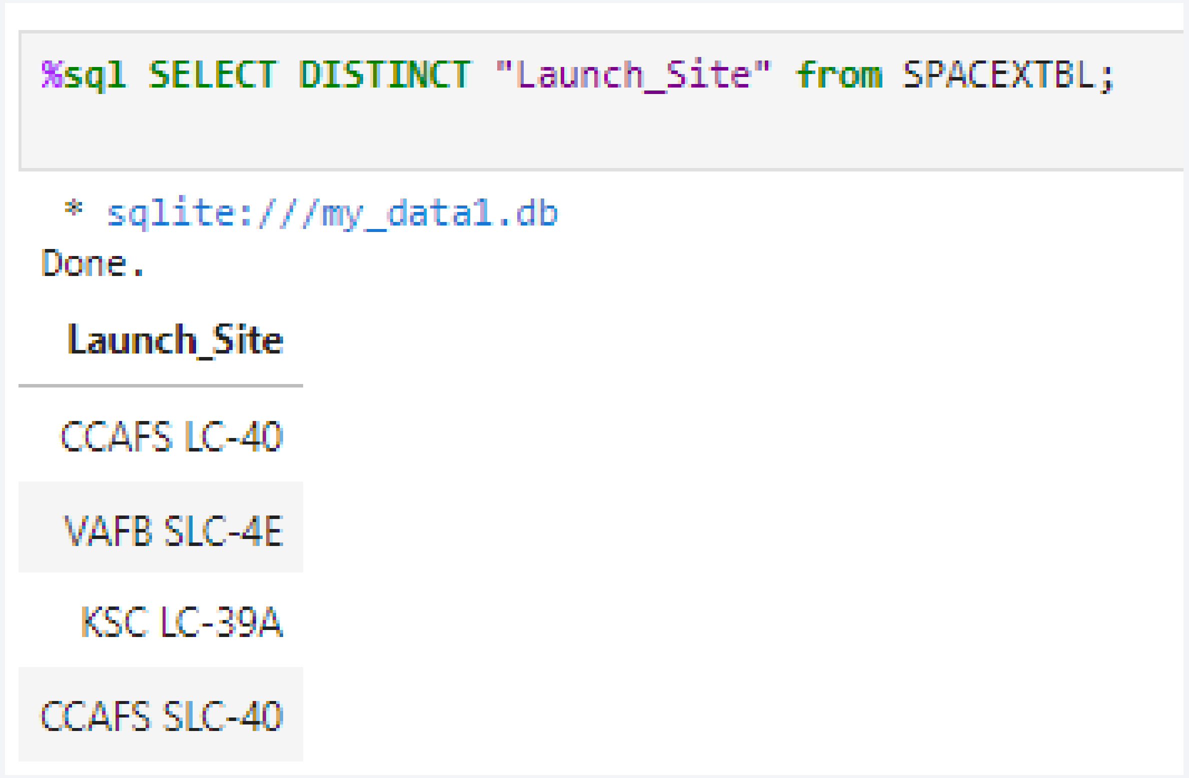
This scatter plot shows Orbit type as a function of Payload mass, and the color indicates whether the first stage landing was successful.

Launch Success Yearly Trend



This line graph shows the trend of success over the years.

All Launch Site Names

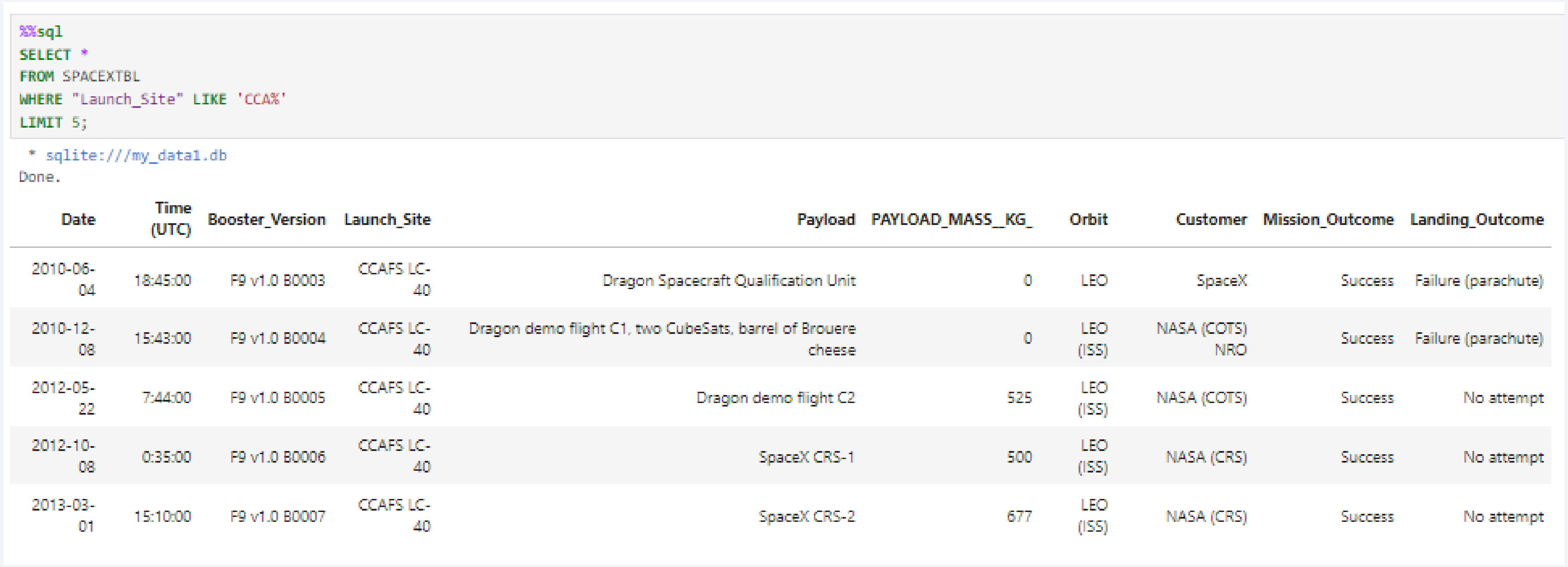


```
%sql SELECT DISTINCT "Launch_Site" from SPACEXTBL;
```

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

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

Launch Site Names Begin with 'CCA'

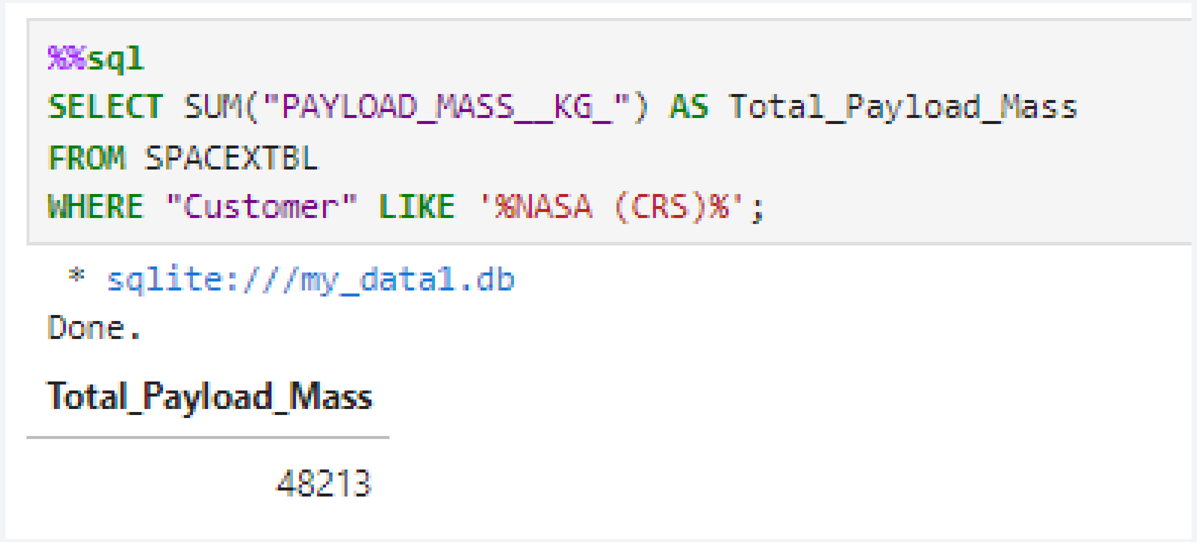


```
%%sql
SELECT *
FROM SPACEXTBL
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

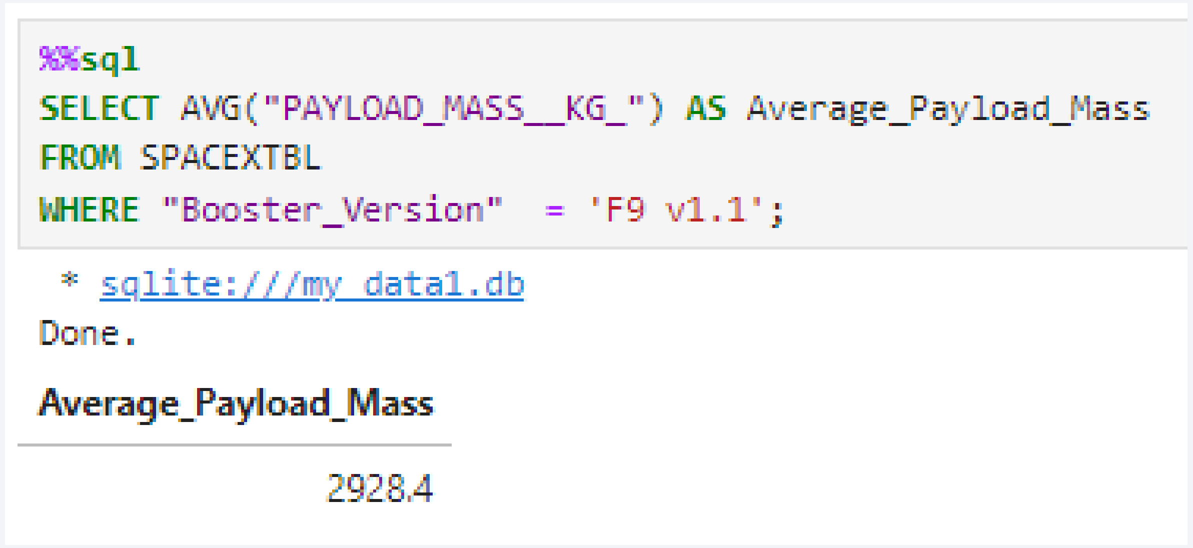


```
%%sql
SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass
FROM SPACEXTBL
WHERE "Customer" LIKE '%NASA (CRS)%';
```

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

Total_Payload_Mass
48213

Average Payload Mass by F9 v1.1

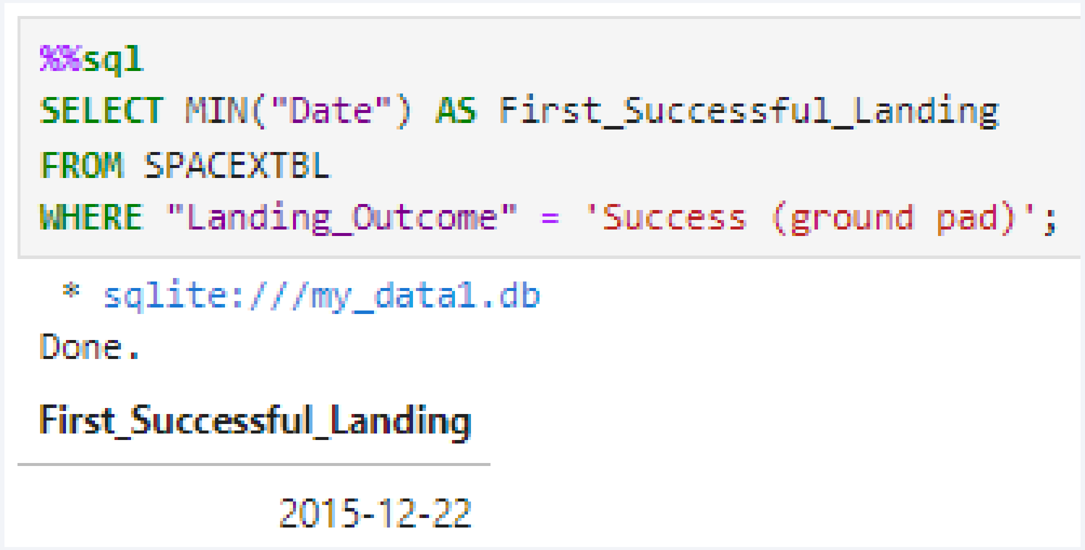


```
%%sql
SELECT AVG("PAYLOAD_MASS__KG_") AS Average_Payload_Mass
FROM SPACEXTBL
WHERE "Booster_Version"  = 'F9 v1.1';
```

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

Average_Payload_Mass
2928.4

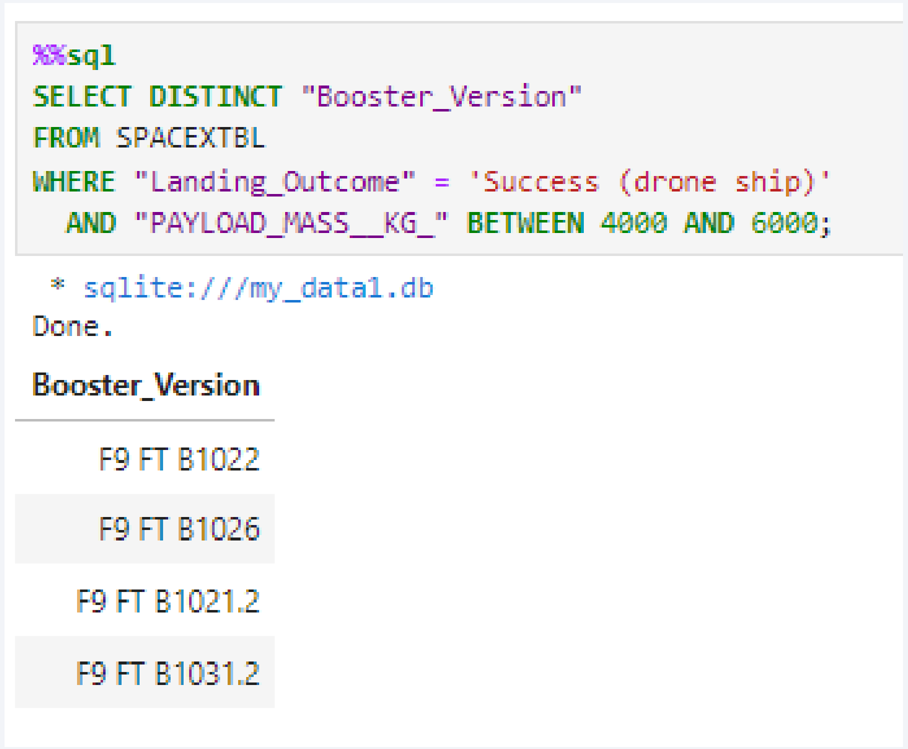
First Successful Ground Landing Date


```
%%sql
SELECT MIN("Date") AS First_Successful_Landing
FROM SPACEXTBL
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



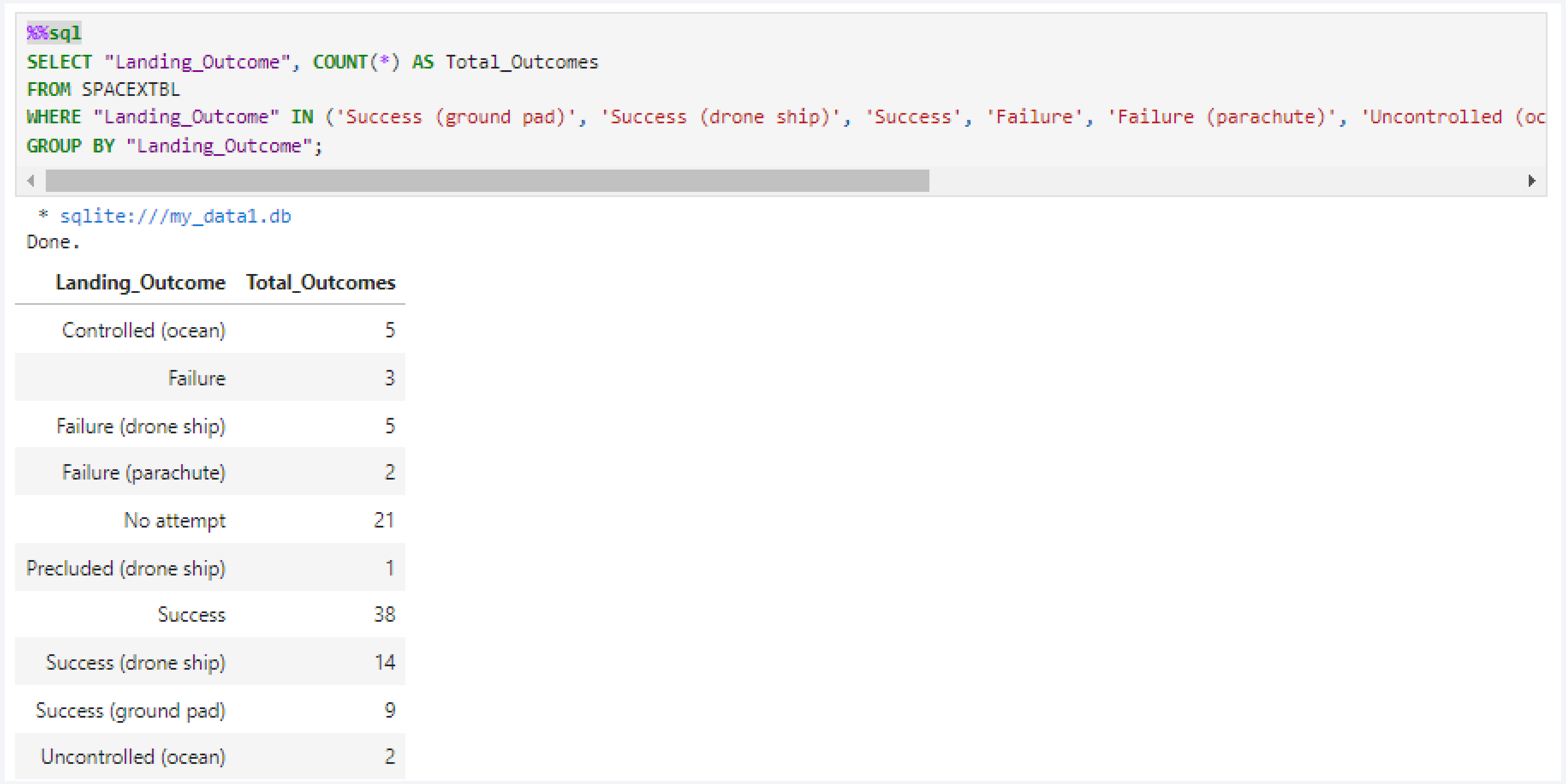
```
%%sql
SELECT DISTINCT "Booster_Version"
FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Success (drone ship)'
  AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000;
```

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

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

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes



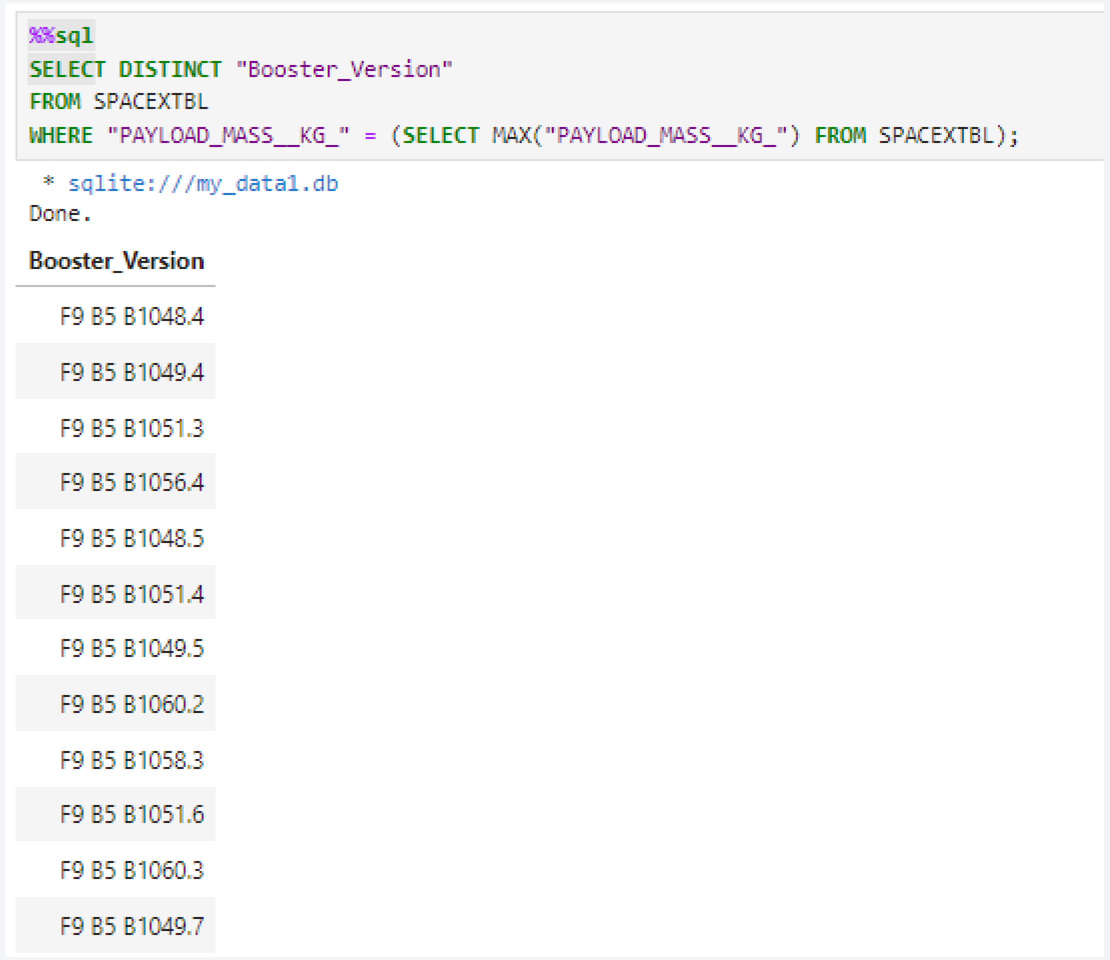
```
%%sql
SELECT "Landing_Outcome", COUNT(*) AS Total_Outcomes
FROM SPACEXTBL
WHERE "Landing_Outcome" IN ('Success (ground pad)', 'Success (drone ship)', 'Success', 'Failure', 'Failure (parachute)', 'Uncontrolled (oc
GROUP BY "Landing_Outcome";
```

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

Landing_Outcome	Total_Outcomes
Controlled (ocean)	5
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	21
Precluded (drone ship)	1
Success	38
Success (drone ship)	14
Success (ground pad)	9
Uncontrolled (ocean)	2

- Total number of successful and failure mission outcomes

Boosters Carried Maximum Payload



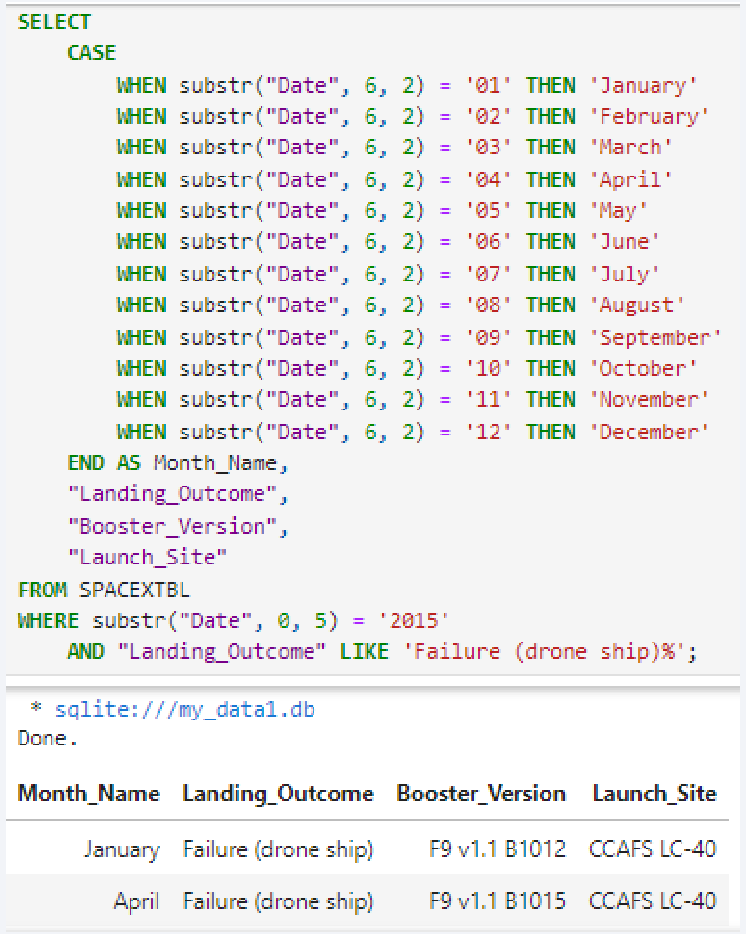
```
%%sql
SELECT DISTINCT "Booster_Version"
FROM SPACEXTBL
WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);
```

```
 * 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

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

2015 Launch Records



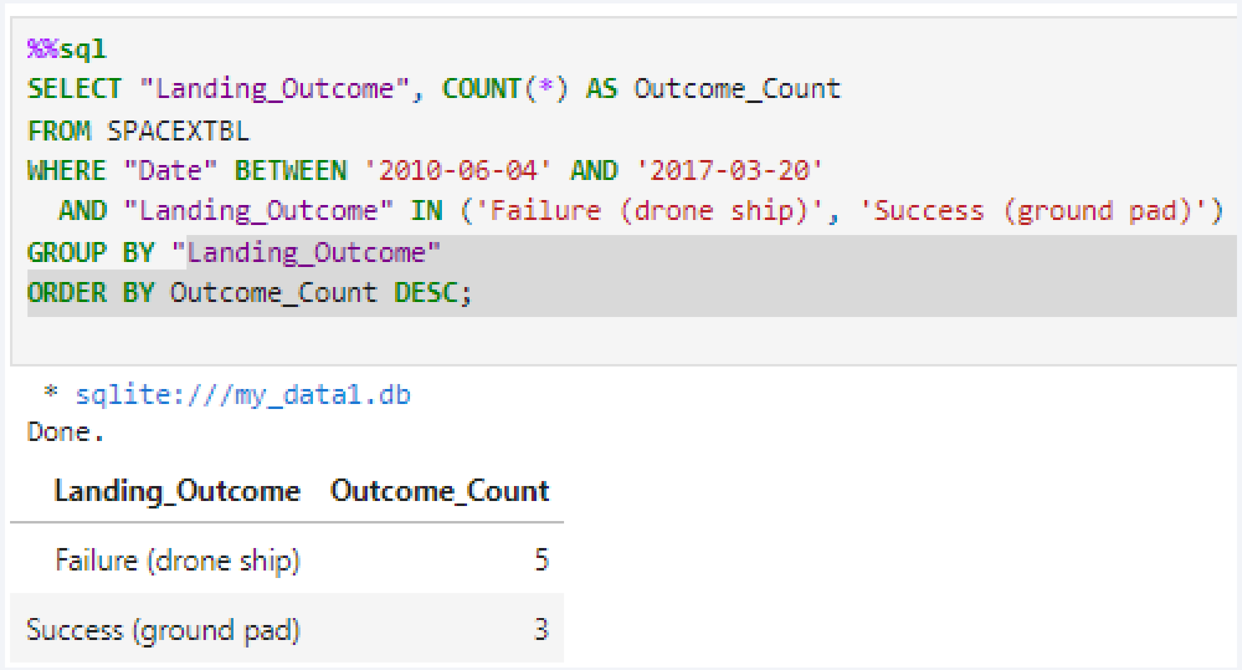
```
SELECT
    CASE
        WHEN substr("Date", 6, 2) = '01' THEN 'January'
        WHEN substr("Date", 6, 2) = '02' THEN 'February'
        WHEN substr("Date", 6, 2) = '03' THEN 'March'
        WHEN substr("Date", 6, 2) = '04' THEN 'April'
        WHEN substr("Date", 6, 2) = '05' THEN 'May'
        WHEN substr("Date", 6, 2) = '06' THEN 'June'
        WHEN substr("Date", 6, 2) = '07' THEN 'July'
        WHEN substr("Date", 6, 2) = '08' THEN 'August'
        WHEN substr("Date", 6, 2) = '09' THEN 'September'
        WHEN substr("Date", 6, 2) = '10' THEN 'October'
        WHEN substr("Date", 6, 2) = '11' THEN 'November'
        WHEN substr("Date", 6, 2) = '12' THEN 'December'
    END AS Month_Name,
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTBL
WHERE substr("Date", 0, 5) = '2015'
    AND "Landing_Outcome" LIKE 'Failure (drone ship)%';
```

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

Month_Name	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

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



```
%%sql
SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count
FROM SPACEXTBL
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
  AND "Landing_Outcome" IN ('Failure (drone ship)', 'Success (ground pad)')
GROUP BY "Landing_Outcome"
ORDER BY Outcome_Count DESC;
```

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

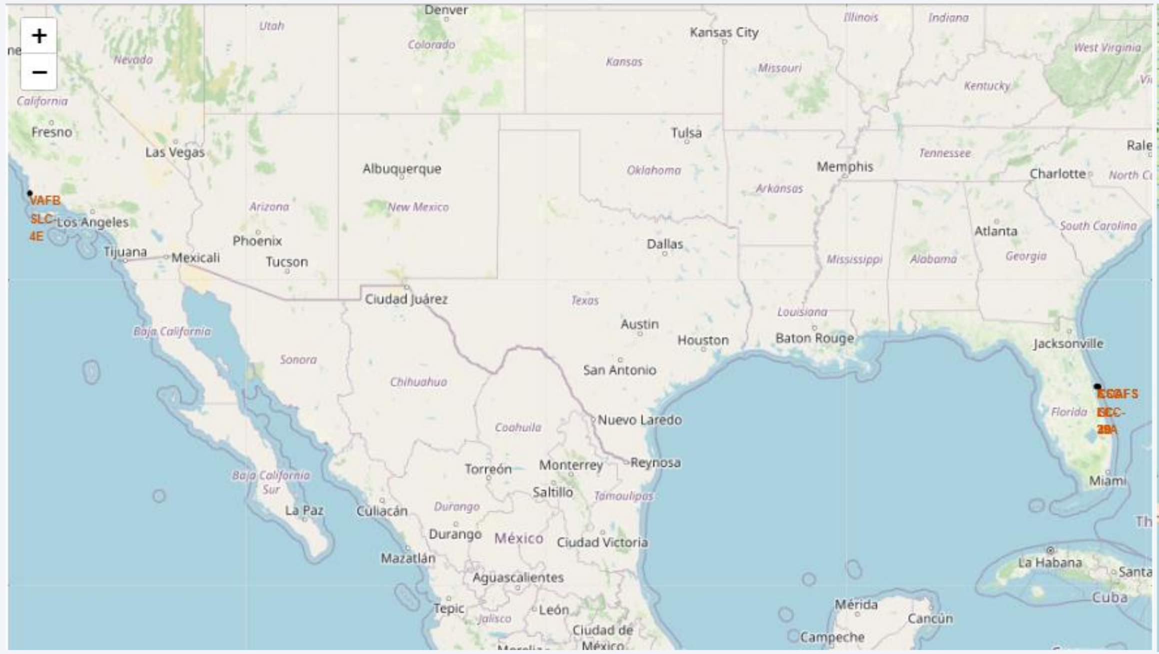
Landing_Outcome	Outcome_Count
Failure (drone ship)	5
Success (ground pad)	3

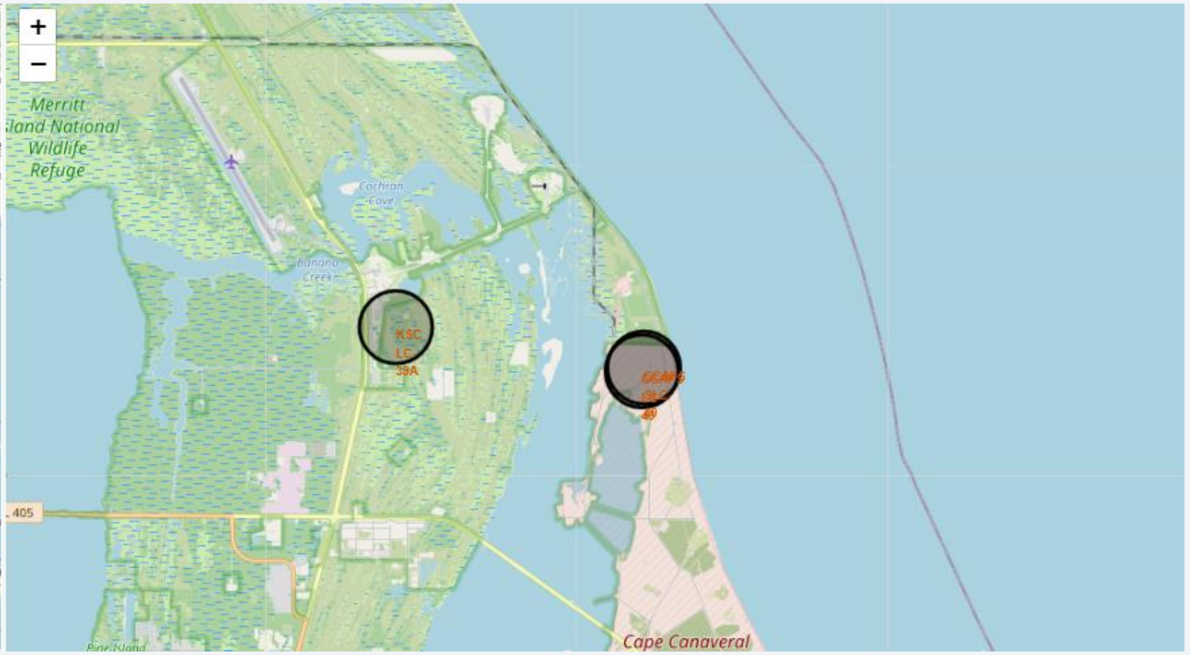
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Section 3

Launch Sites Proximities Analysis

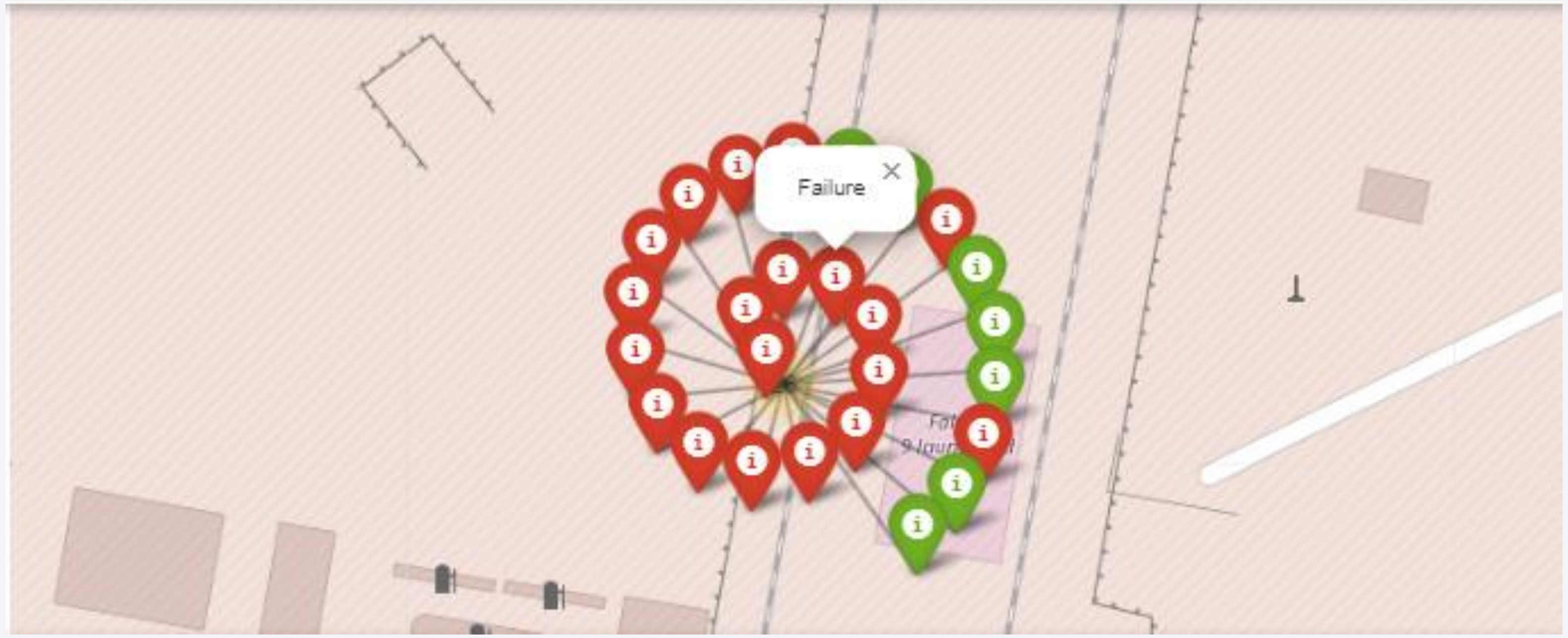
Launch site locations





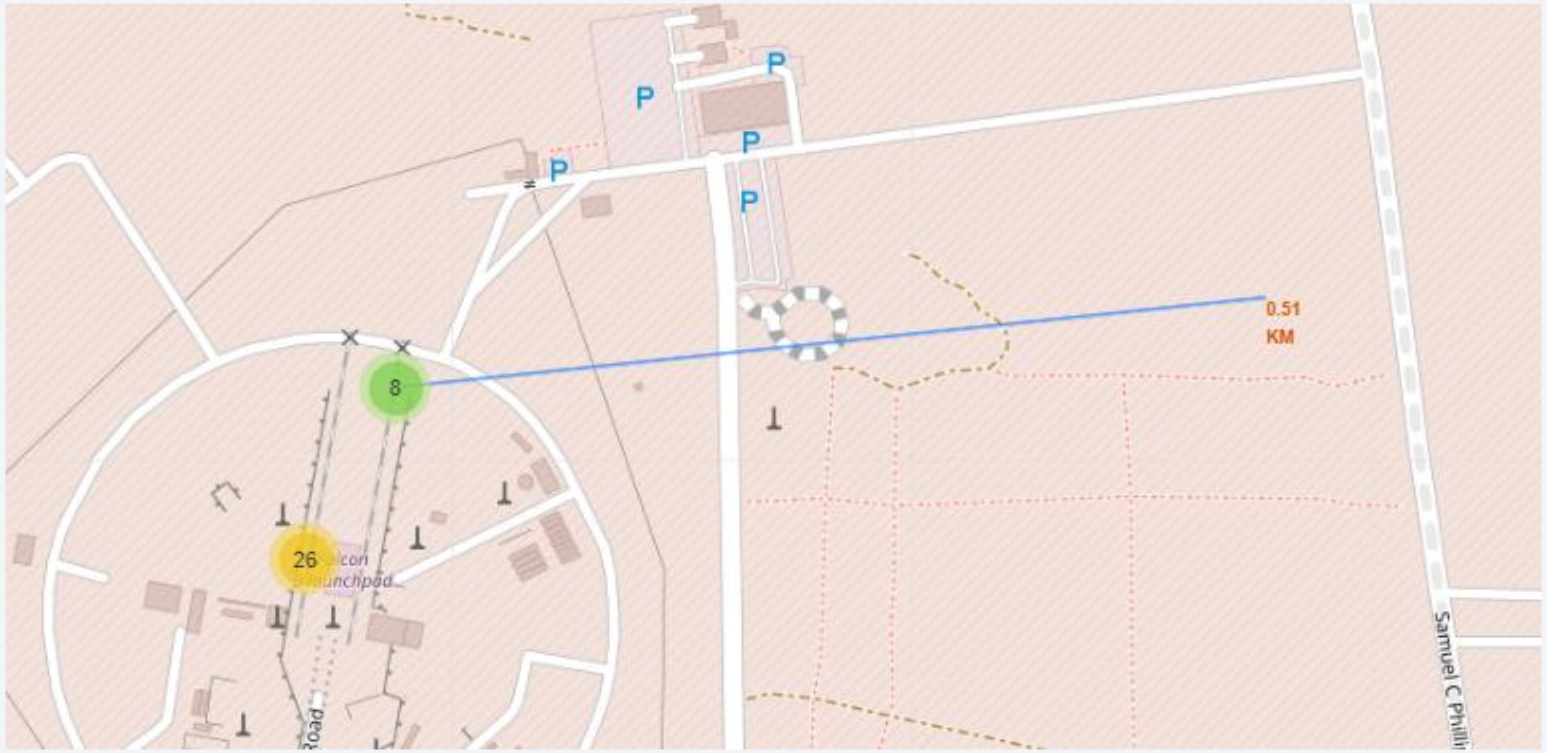
Added a circle with an area around each lauch site

Markers for each launch



In this image you can see a marker for each throw and labeled with its respective result which is also indicated by its color.

Distance measurement

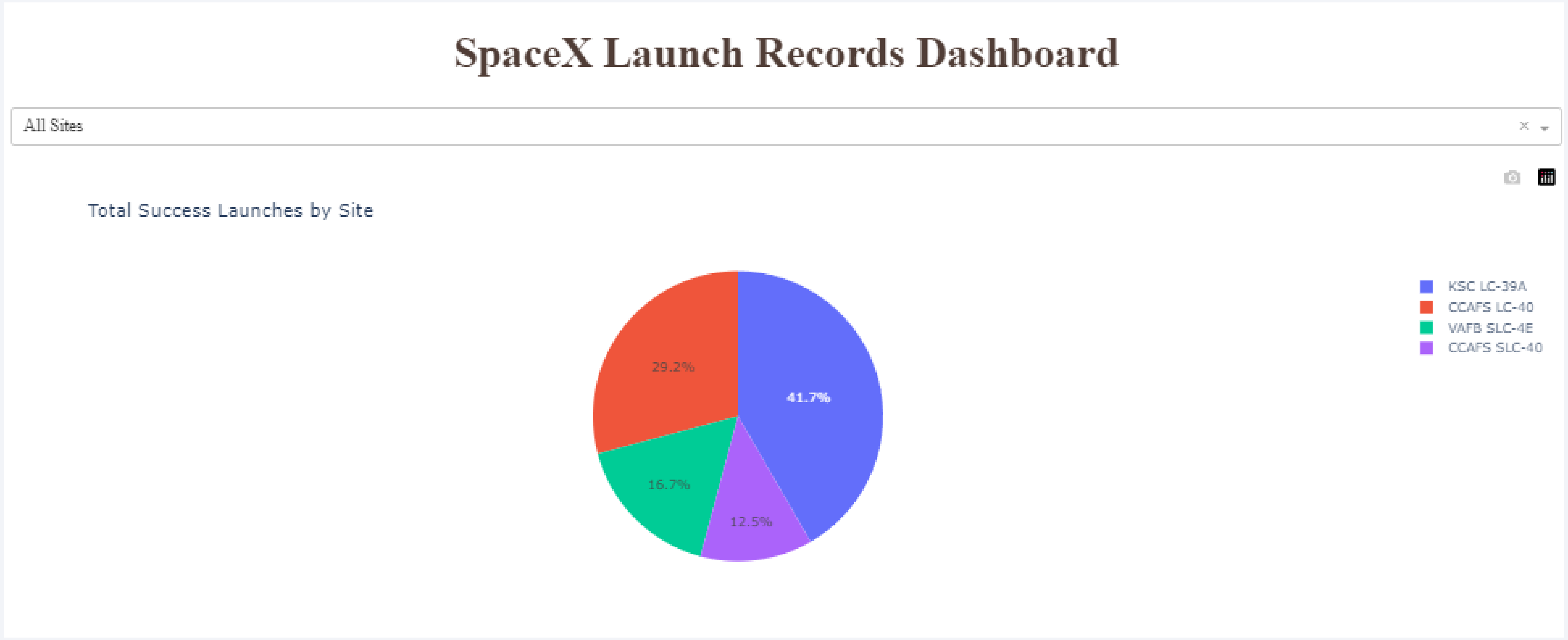


Added a functionality that allows to calculate the distance from a point x to any launch site

Section 4

Build a Dashboard with Plotly Dash

Launch success count for all sities

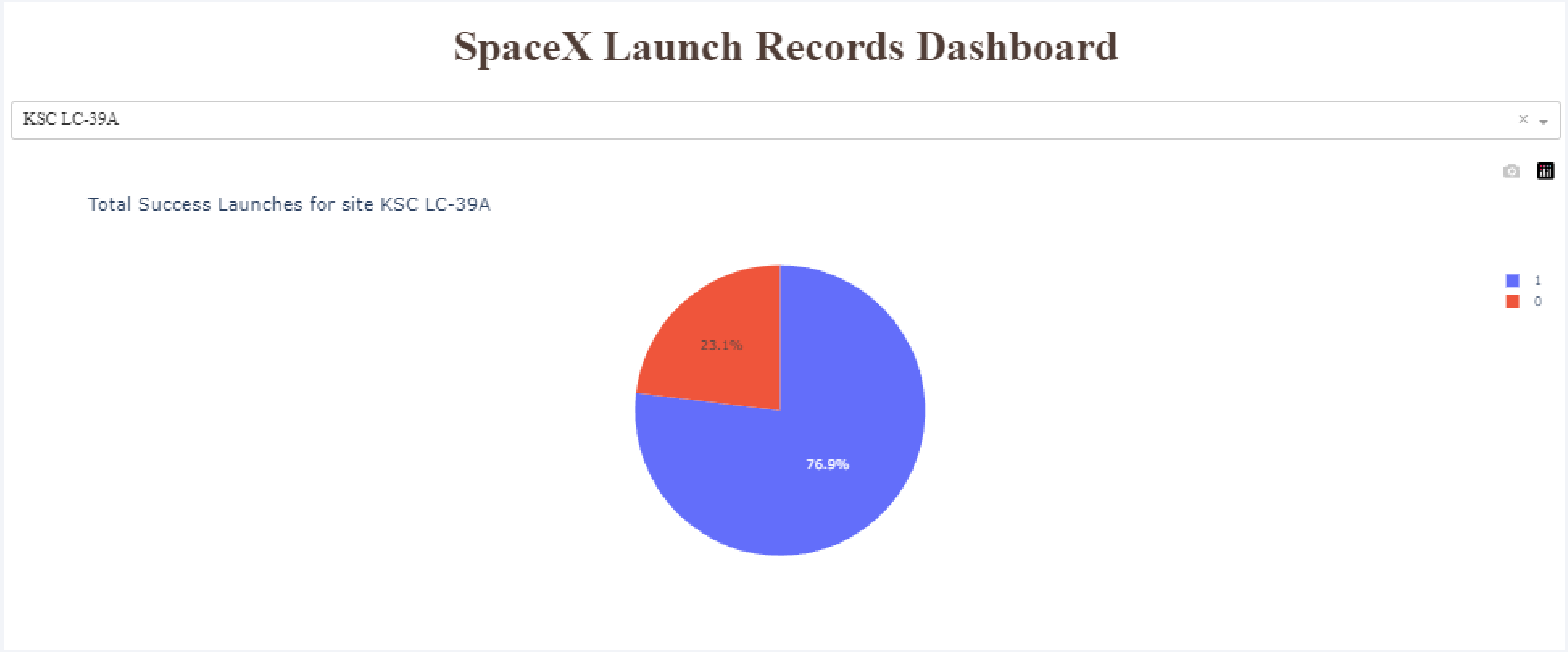


SpaceX Launch Records Dashboard

All Sites

In this pie chart you can see the success rate of each lauch site

Launch site with highest launch success ratio

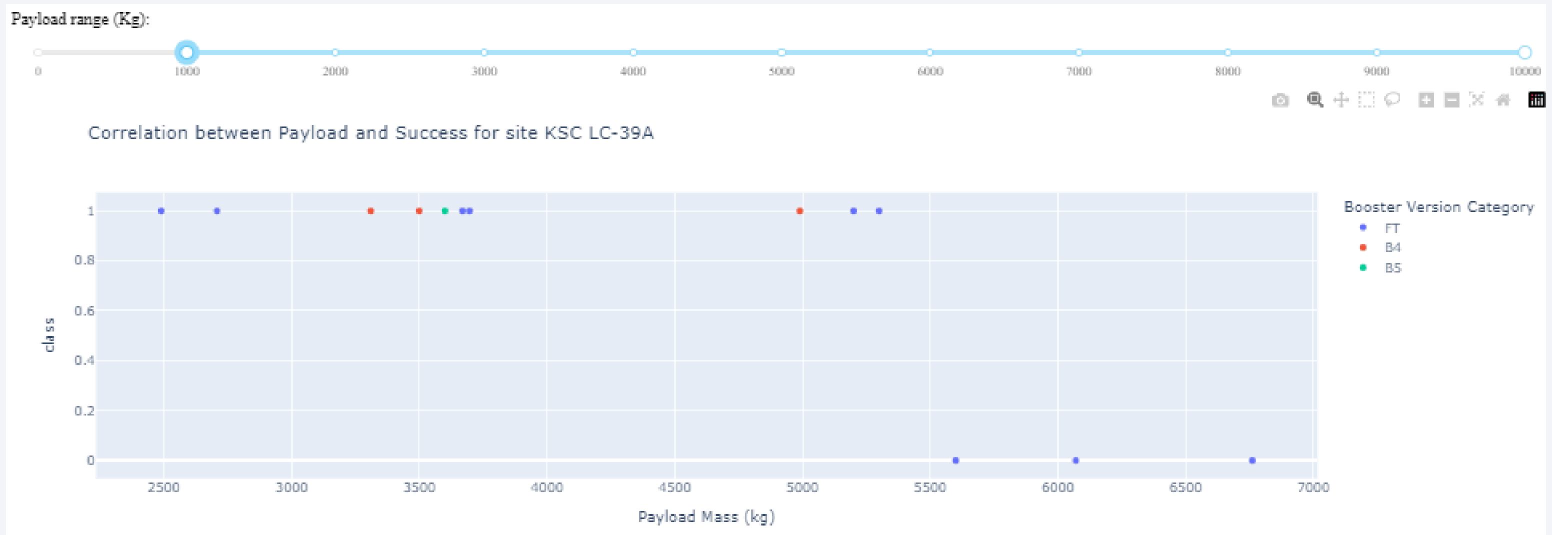

SpaceX Launch Records Dashboard

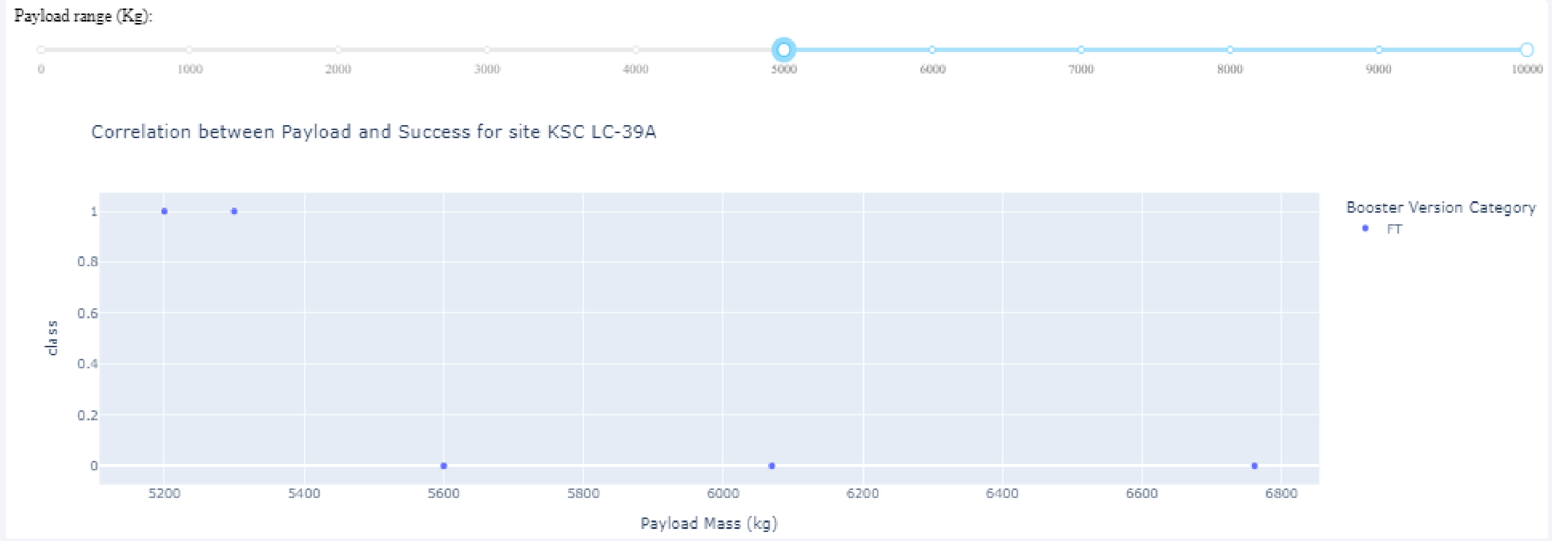
KSC LC-39A

Total Success Launches for site KSC LC-39A

As shown in the chart, the KSC LC-39A launch site has a success rate of 76.9%.

Payload vs Launch outcome scatter plot for all sites



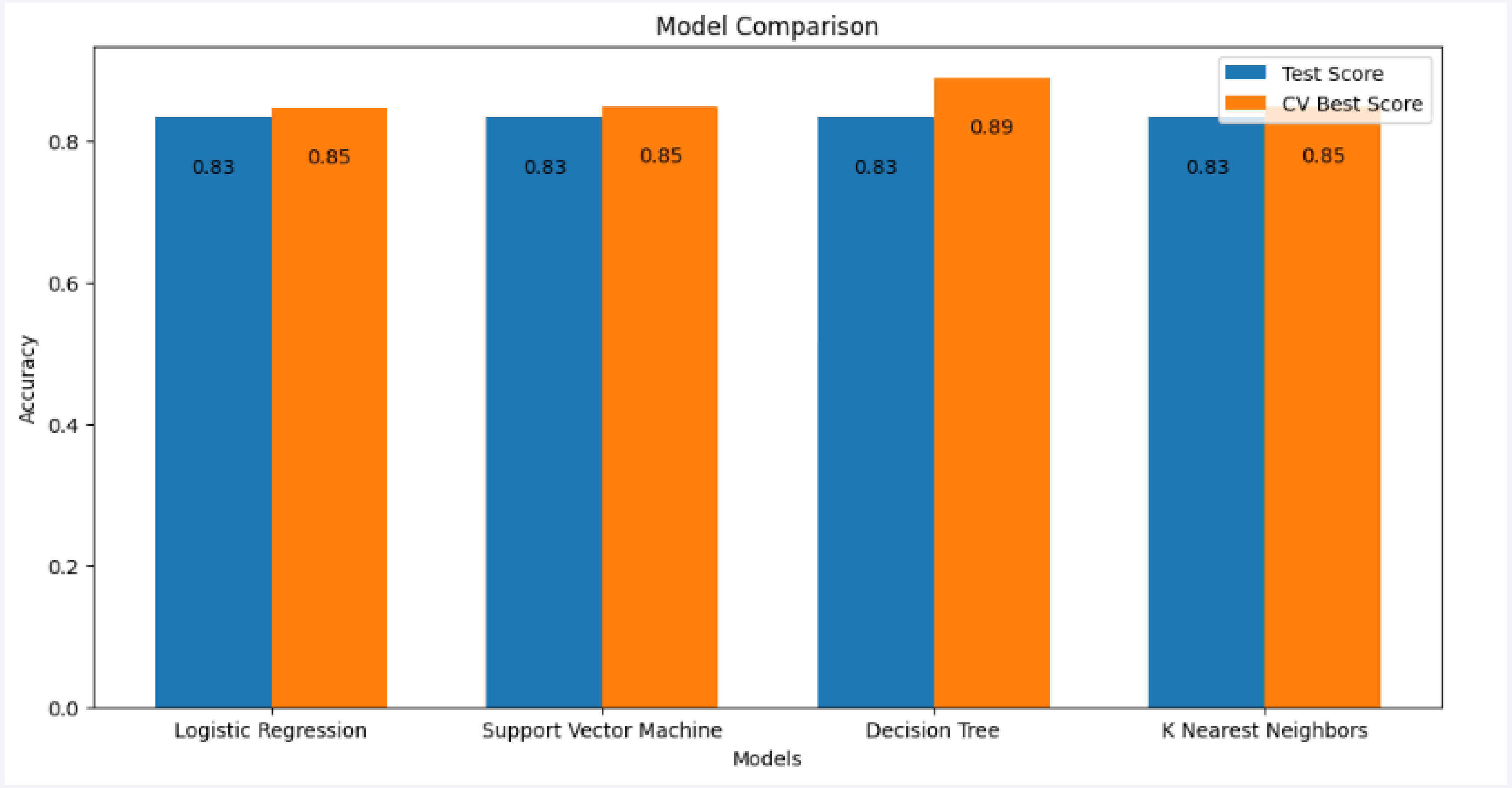


These data demonstrate that the probability of a successful landing decreases with increasing payload.

Section 5

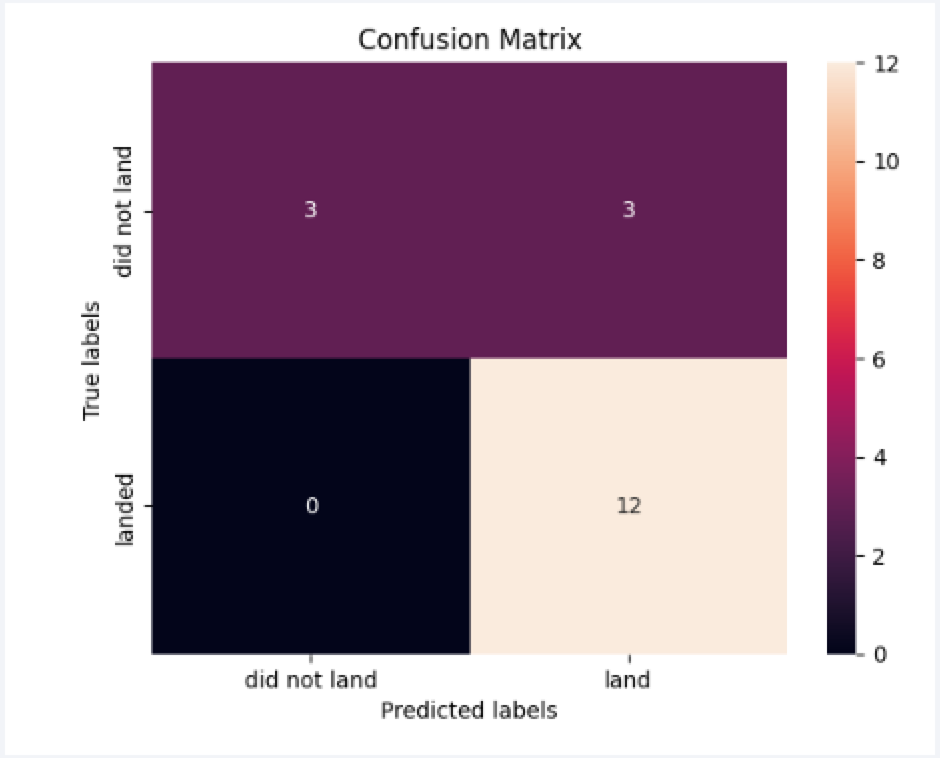
Predictive Analysis (Classification)

Classification Accuracy



In this bar chart it can be seen that the best model is the dessicion tree because it has the best accuracy in cross validation.

Confusion Matrix



In this confusion matrix we can see that the weakness of the model consists in the false negatives.

Conclusions

- The analysis consistently highlights launch site as a significant factor in landing success. Visualizations like "Launch site locations" and "Launch site with highest launch success ratio" suggest that KSC LC-39A has a higher success rate.
- The visualizations, especially "Payload vs. Launch Site" and "Payload vs Launch outcome scatter plot", strongly indicate an inverse relationship between payload mass and successful landings.
- While the document mentions using various classification models , it emphasizes the Decision Tree's accuracy in cross-validation. However, the confusion matrix reveals its weakness in false negatives.
- The document showcases an interactive dashboard built with Plotly Dash, enabling users to filter data and visualize relationships

Thank you!