



IBM Developer  
SKILLS NETWORK

# Winning Space Race with Data Science

Katalin Kizman  
11-04-2024



# Outline

---

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

# Executive Summary

---

- Summary of methodologies
  - Data Collection via SpaceX API
  - Data Collection with Web scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA) with SQL
  - Exploratory Data Analysis (EDA) with Data Visualization
  - Interactive Visual Analytics with Folium and Dashboard
  - Prediction with Machine Learning
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

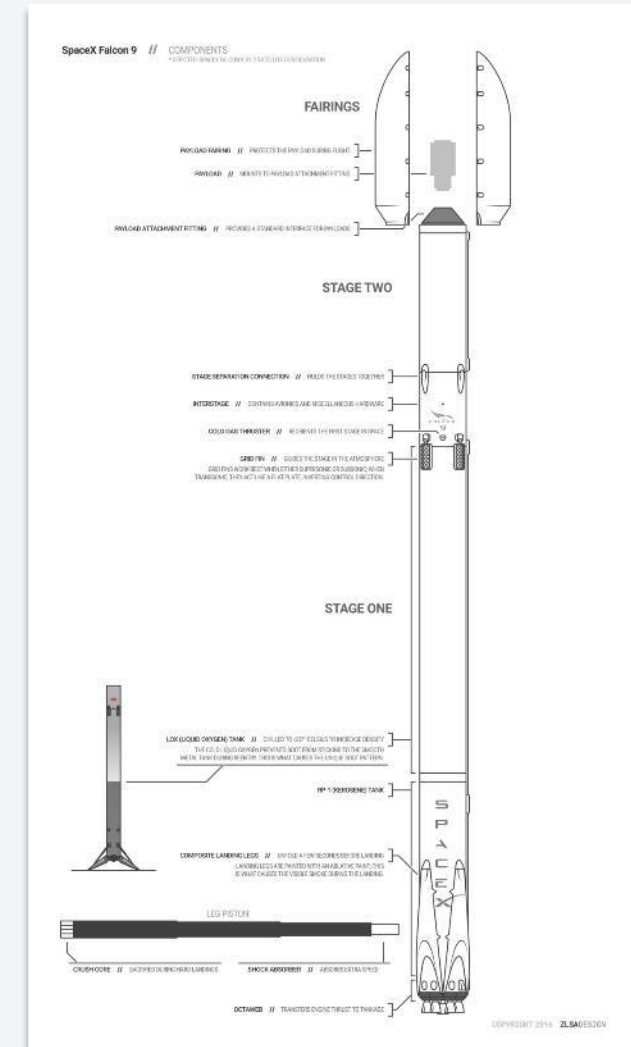
# Introduction

SpaceX stands out in the aerospace industry thanks to its revolutionary idea of making the first stage of the rocket reusable. SpaceX offers Falcon 9 rocket launches at a cost of **\$62 million**, while other providers charge upwards of **\$165 million** per launch. Most of the savings come from recovering the first stage of the launch by re-landing the rocket for use on the next mission.

As a data scientist working for a competitor of SpaceX, the goal of this project is to create a machine learning pipeline to predict the landing outcome of the first stage in the future. Having this information will help our company to bid against SpaceX for a rocket launch.

Solving this problem involves:

- Collecting information on historical SpaceX launches
- Determining the key factors that contribute to the success of a landing
- Finding the best performing machine learning algorithm for prediction





Section 1

# Methodology

# Methodology

---

## Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX REST API and web scraping from Wikipedia
- Perform data wrangling
  - Data was cleaned and then processed using one-hot encoding for categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build models to predict landing outcomes using Logistic regression, Support Vector Machine (SVM), Decision Tree and K-nearest Neighbors (KNN)

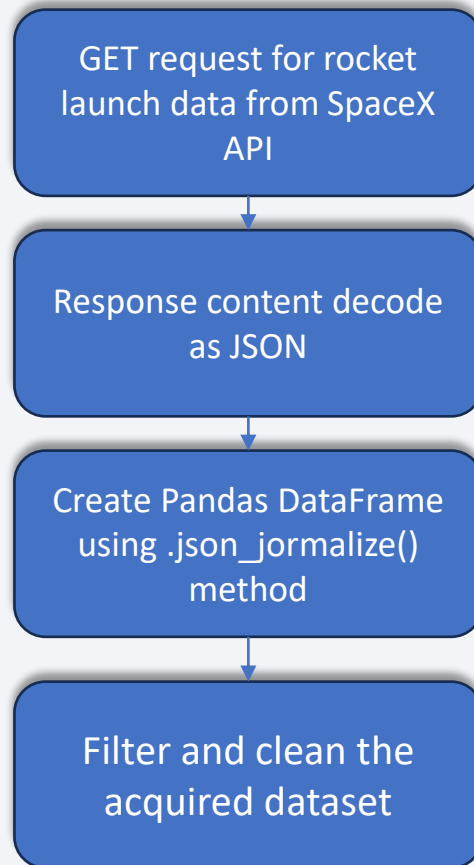
# Data Collection

---

As mentioned, the dataset was collected utilizing the SpaceX REST API and Web scraping from Wikipedia

- REST API
  - After GET request the response content has been decoded as a JSON using `.json()` method
  - The content of the above mentioned JSON object then converted into **Pandas DataFrame** using `.json_normalize()` method
- Web Scraping
  - SpaceX rocket launch history records extracted from HTML tables using **BeautifulSoup**
  - After parsing relevant information from the tables it has converted into **Pandas DataFrame** for further analysis

# Data Collection – SpaceX API



Link to notebook:  
[https://github.com/kizike/capstone\\_project/blob/main/1\\_DataCollection/jupyter-labs-spacex-data-collection-api.ipynb](https://github.com/kizike/capstone_project/blob/main/1_DataCollection/jupyter-labs-spacex-data-collection-api.ipynb)

```
spacex_url="https://api.spacexdata.com/v4/launches/past"

response = requests.get(spacex_url)

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

# Lets take a subset of our dataframe keeping only the features we want and the flight number,
# and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with
# 2 extra rocket boosters and rows that have multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value
# in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting
# the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

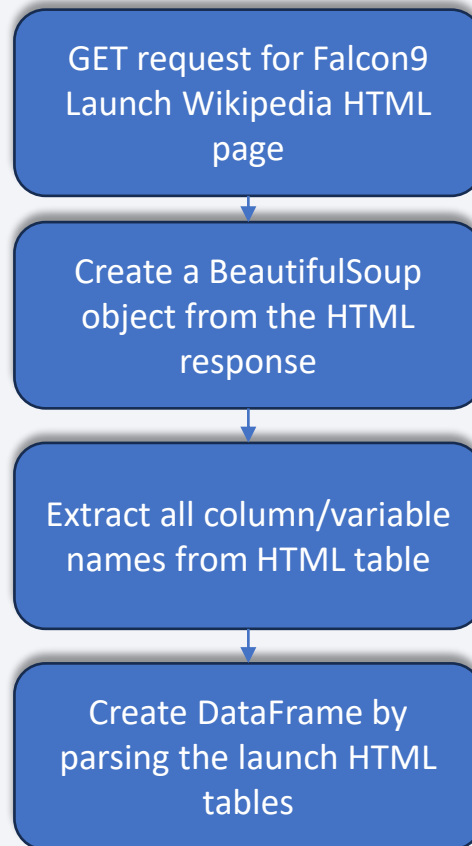
# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]

# Hint data['BoosterVersion']!= 'Falcon 1'
data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']

# Calculate the mean value of PayloadMass column
payloadmass_mean = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9.replace({'PayloadMass':np.nan}, payloadmass_mean, inplace=True)
```



# Data Collection - Scraping



Link to notebook:  
[https://github.com/kizike/capstone\\_project/blob/main/1\\_DataCollection/jupyter-labs-webscraping.ipynb](https://github.com/kizike/capstone_project/blob/main/1_DataCollection/jupyter-labs-webscraping.ipynb)

```
# use requests.get() method with the provided static_url
response = requests.get(static_url)
html = response.text

soup = BeautifulSoup(html, 'html.parser')

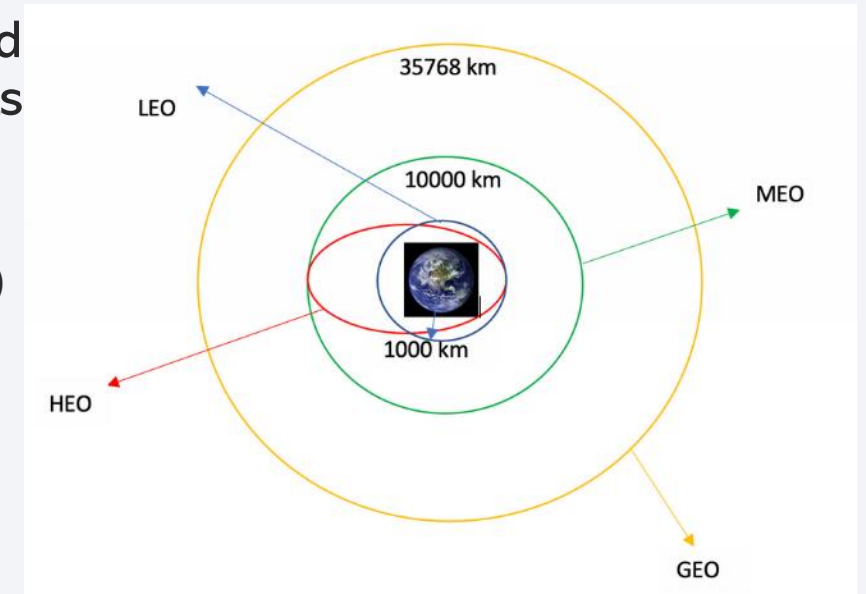
extracted_row = 0
#Extract each table
for table_number, table in enumerate(soup.find_all('table',
    "wikitable plainrowheaders collapsible")):
    # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
            else:
                flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictionary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            launch_dict['Flight No.'].append(flight_number)

            datatimelist=date_time(row[0])
            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            #print(date)
            launch_dict['Date'].append(date)
```

# Data Wrangling

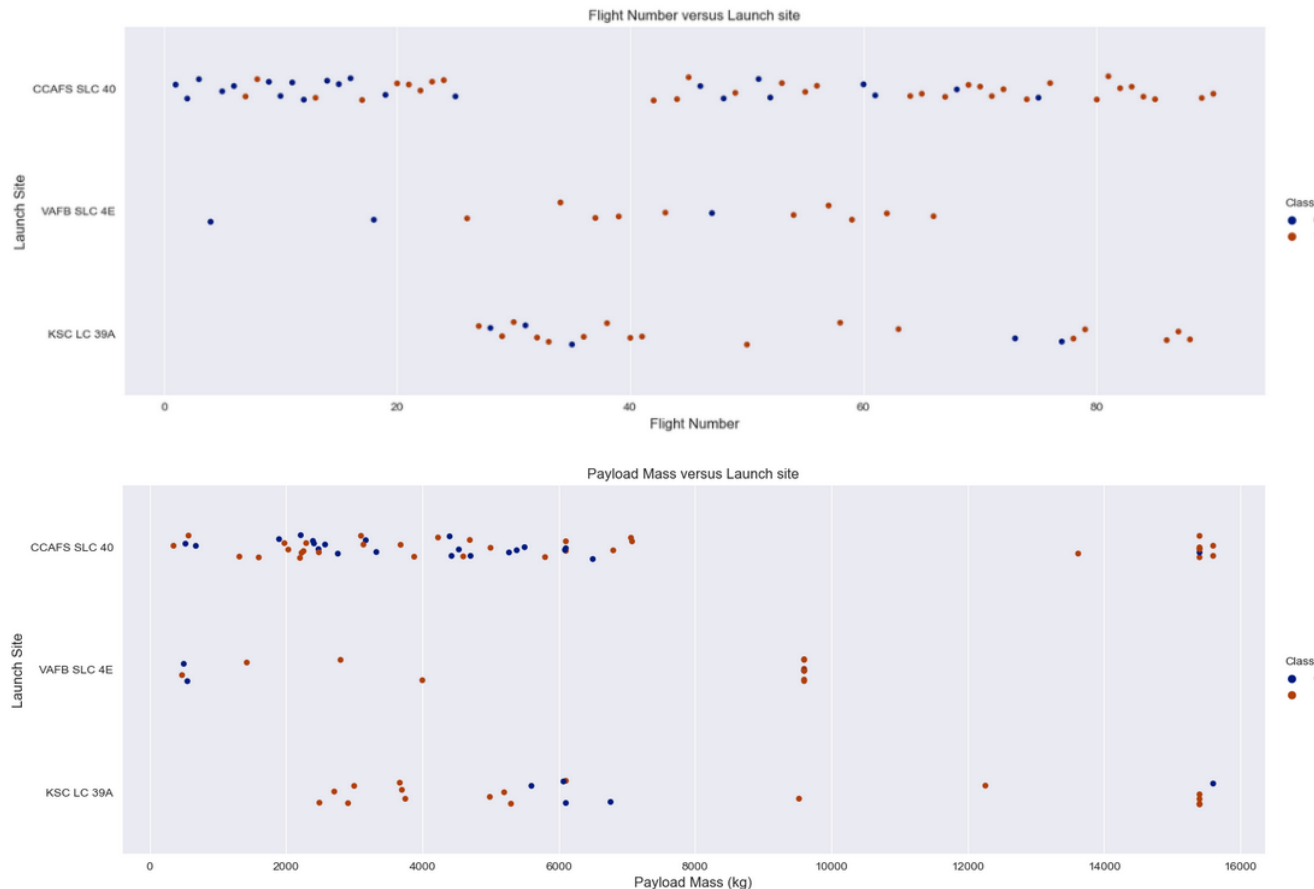
Data wrangling ensures that raw data is transformed, mapped and validated to be ready for further analysis such as Exploratory Data Analysis (EDA) or machine learning

- Data discovery/examination
- Data cleaning (handling missing values, remove unnecessary data, etc.)
- Structuring data
- Checking data consistency, quality
- The number of launches at each site was **calculated**, then the number and occurrence of mission outcome per orbit was calculated
- **Landing outcome labels** were **created** from the Outcome column for further analysis, visualization and ML
- Results were **exported** to a **CSV** file



Link to notebook:  
[https://github.com/kizike/capstone\\_project/blob/main/1\\_DataWrangling/labs-jupyter-spacex-Data%20Wrangling.ipynb](https://github.com/kizike/capstone_project/blob/main/1_DataWrangling/labs-jupyter-spacex-Data%20Wrangling.ipynb)

# EDA with Data Visualization



Various scatterplots, barcharts and lineplots have been created.

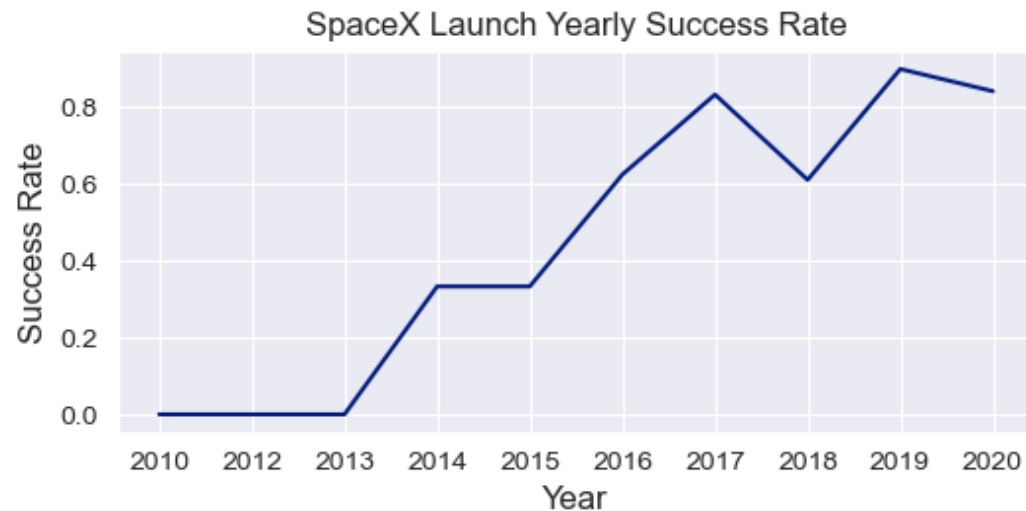
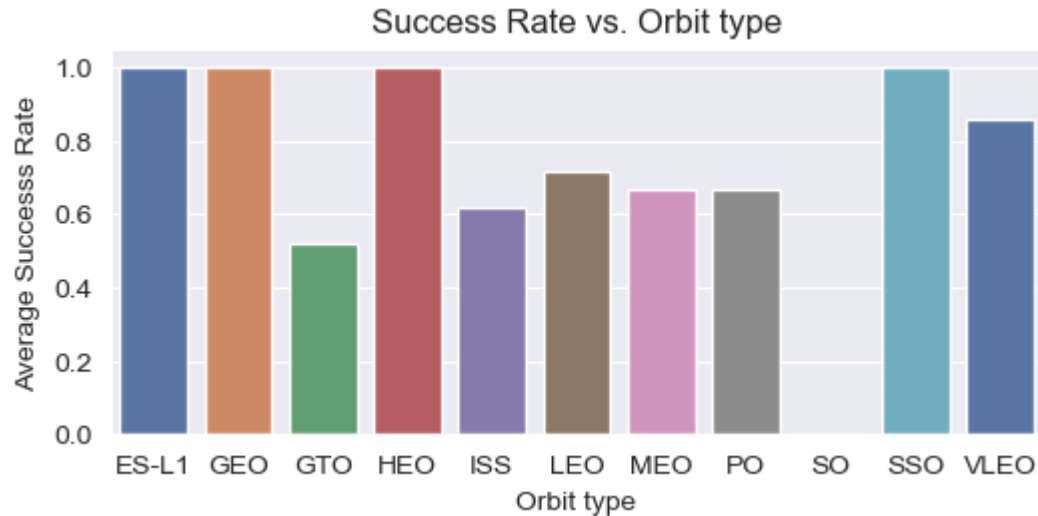
**Scatterplots** were created to investigate the relationship between:

- *Flight Number vs. Launch Site*
- *Flight Number vs. Orbit type*
- *Payload Mass (kg) vs. Launch Site*
- *Payload Mass (kg) vs. Orbit type*

By looking at these graphs, it's easier to see the factors that contribute most to the success of the landing results.

Link to notebook:  
[https://github.com/kizike/capstone\\_project/blob/main/2\\_EDA/jupyter-labs-eda-dataviz.ipynb](https://github.com/kizike/capstone_project/blob/main/2_EDA/jupyter-labs-eda-dataviz.ipynb)

# EDA with Data Visualization



After gaining insights from the scatter plots, categorical variables were examined using bar charts and trends over time were examined using line graph.

In this case, the **bar chart** was used to determine which orbit type had a high success rate.

Then the **line graph** has shown the yearly trend of launch success.

Link to notebook:  
[https://github.com/kizike/capstone\\_project/blob/main/2\\_EDA/jupyter-labs-eda-dataviz.ipynb](https://github.com/kizike/capstone_project/blob/main/2_EDA/jupyter-labs-eda-dataviz.ipynb)

# EDA with SQL

---

SQL queries were executed to gain a better insight into the data and to answer the following questions:

- Display the names of the launch sites
- Display 5 records where launch sites begin with the string "CCA"
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass
- List the record which will display the month names, failure landing\_outcomes in drone ship, booster versions, launch\_site for month in year 2015
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the data 2010-06-04 and 2017-03-20, in descending order



# Build an Interactive Map with Folium

---

Folium was used to create an interactive map showing the exact locations of the launch sites and NASA. Then, several objects were marked, and other important information was visualized on the map:

- **Circle markers** labelled with the name of the site were added to the map to indicate the positions of the launch sites based on their latitude and longitude coordinates
- **Red** and **green** markers were created and placed on this map for each site based on Success/Failed attributes of the launches to see which site have high success rate
- The distances between a launch site to its proximities (nearest railway, highway, coastline and city) were calculated and displayed on the map with colored lines showing the distance in kilometers to help understand of the positioning characteristics of the launch sites.

# Build a Dashboard with Plotly Dash


---

An interactive dashboard was created using Plotly Dash to allow users to explore the data to gain more valuable insights. The dashboard features include:

- **Dropdown list** with launch sites to select
- **Pie chart** can be plotted to show the total success of the launches (overall or by sites)
- **Scatter plots** can be generated to show the correlation between payload mass (kg) and the launch success (with changable payload mass range) for different booster version
  - Slider added for adjusting Payload Mass Range

# Predictive Analysis (Classification)

---

- 
- **Pre-process** data to make it ready for machine learning models
    - **Create** Numpy array
    - **Standardizing** data using **StandardScaler()**
  - **Split** the data into training and test set using **train\_test\_split()**
  - **Create** GridSearchCV object with cv=10 for parameter optimization
  - **Use GridSearchCV** on the following algorithms: Logistic Regression, SVM, Decision Tree, KNN using the training set
  - **Evaluate** models using confusion matrices and accuracy scores
  - **Determine** the best performing model based on the accuracy and other characteristics of the models

# Results

---

The results includes:

- Exploratory data analysis (EDA) results
- Interactive analytics demo in screenshots
- Predictive analysis results



The background of the slide is a complex, abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks and bands of lighter blue and vibrant red. These streaks vary in thickness and intensity, creating a sense of motion and depth. A faint, semi-transparent grid pattern is also visible, particularly in the upper right quadrant, where it intersects with the colored streaks. The overall effect is a high-tech, digital aesthetic.

Section 2

# Insights drawn from EDA



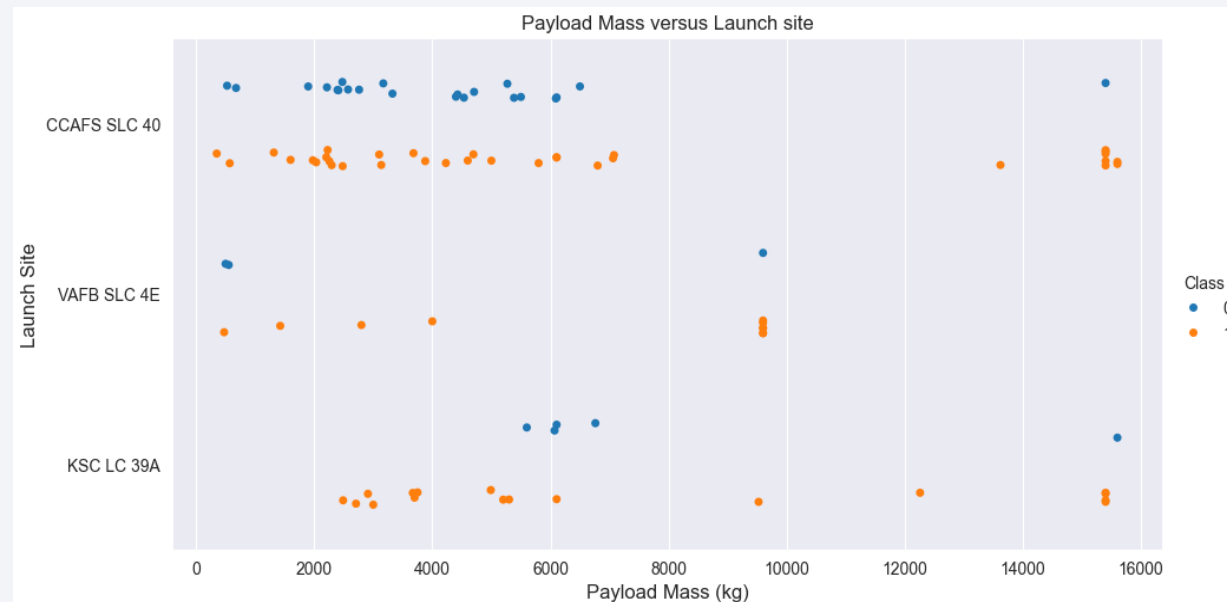
# Flight Number vs. Launch Site

- **Earlier flights** have a **lower** success rate (blue = failed launch)
- **Later flights** have a **higher** success rate (orange = successful launch)
- CCAFS SLC 40 site have the most launch
- KSC LC 39A and VAFB SLC 4E sites have higher success rate compared to CCAFS SLC40
- It can be concluded that as the **flight number increased**, so did the **success rate**



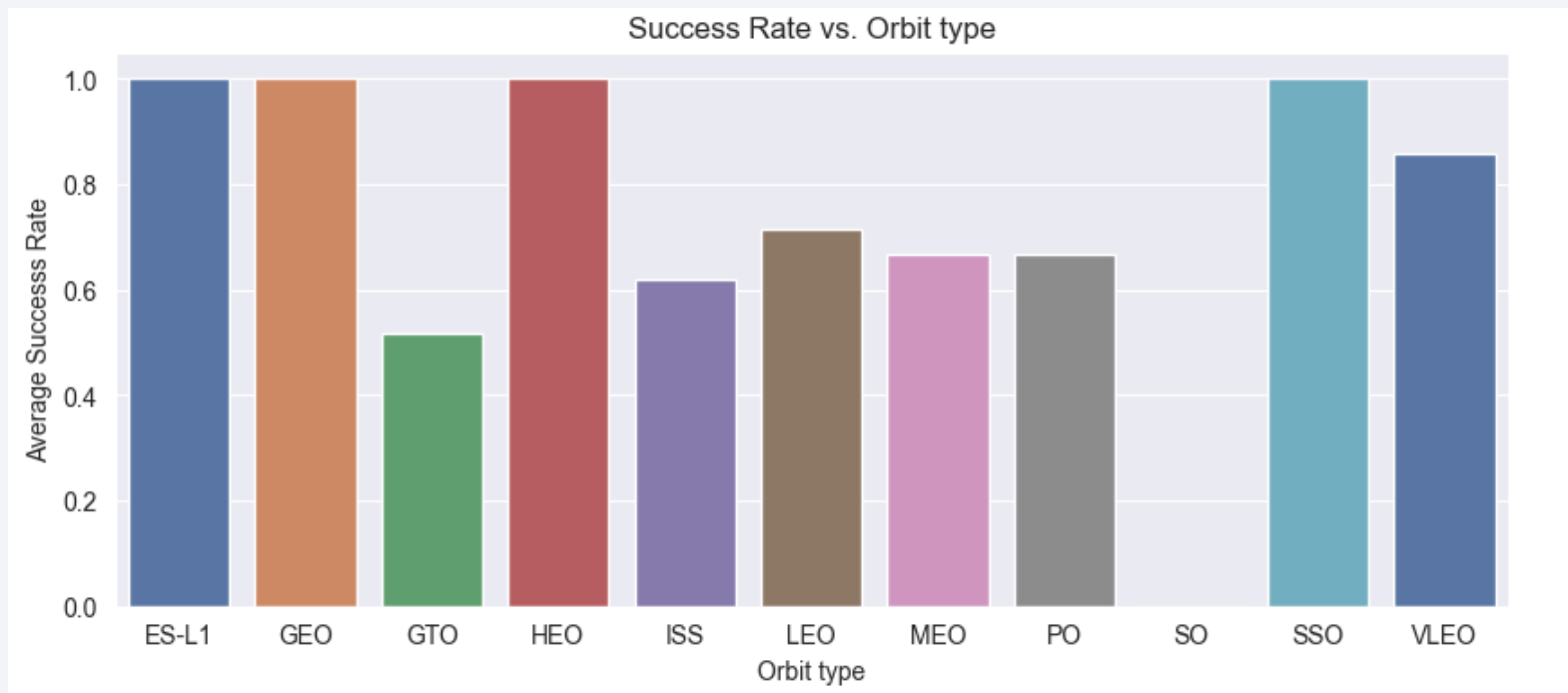
# Payload vs. Launch Site

- In general, launches with high payload mass have a higher success rate
- At the **VAFB SLC 4E site**, the launch payload mass **did not exceed ~10,000 kg**
- Only the KSC LC 39A and CCAFS SLC40 sites have had high payload mass launches



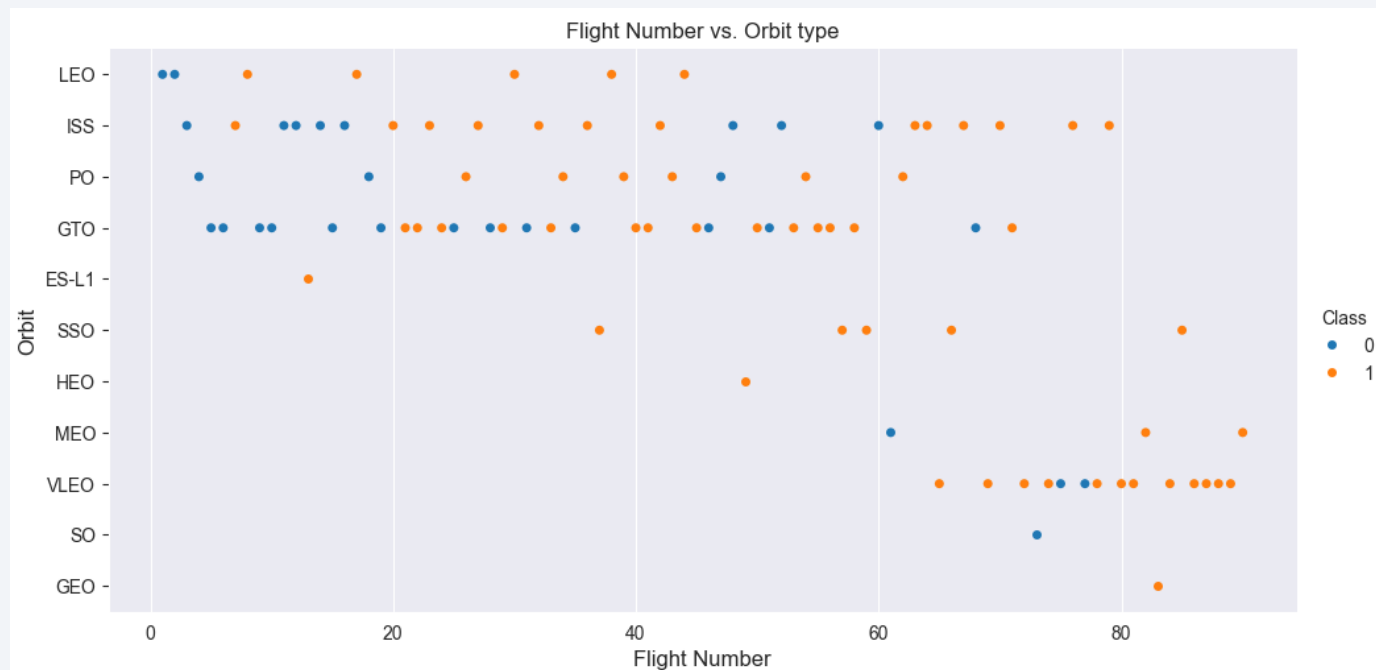
# Success Rate vs. Orbit Type

- **100% success rate:** ES-L1, GEO, HEO and SSO
- **50%-80 % success rate:** GTO, ISS, LEO, MEO, PO and VLEO
- **0 % success rate:** SO
- **GTO** orbit type has a **least success rate** compared to the **mid range**



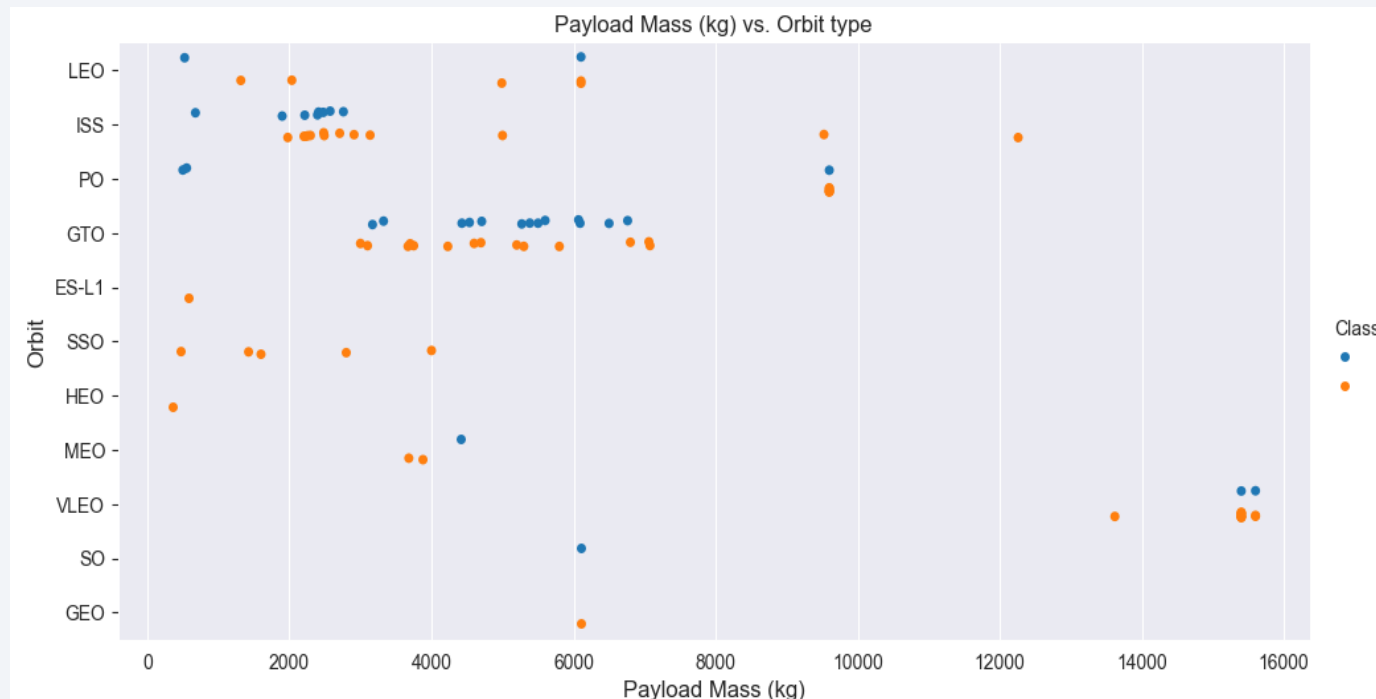
# Flight Number vs. Orbit Type

- Typically, the **success rate** for most orbits **increases** as the **number of flights increases**
- This **correlation** is particularly **evident** for the **LEO orbit**
- Seems like **no correlation** between flight number and success rate for **GTO orbit**



# Payload vs. Orbit Type

- **Successful landing rate** in case of heavy payloads are **better** for **PO, LEO and ISS** orbits
- **No correlation** between **payload mass** and **success rate** can be observed for **GTO orbit**

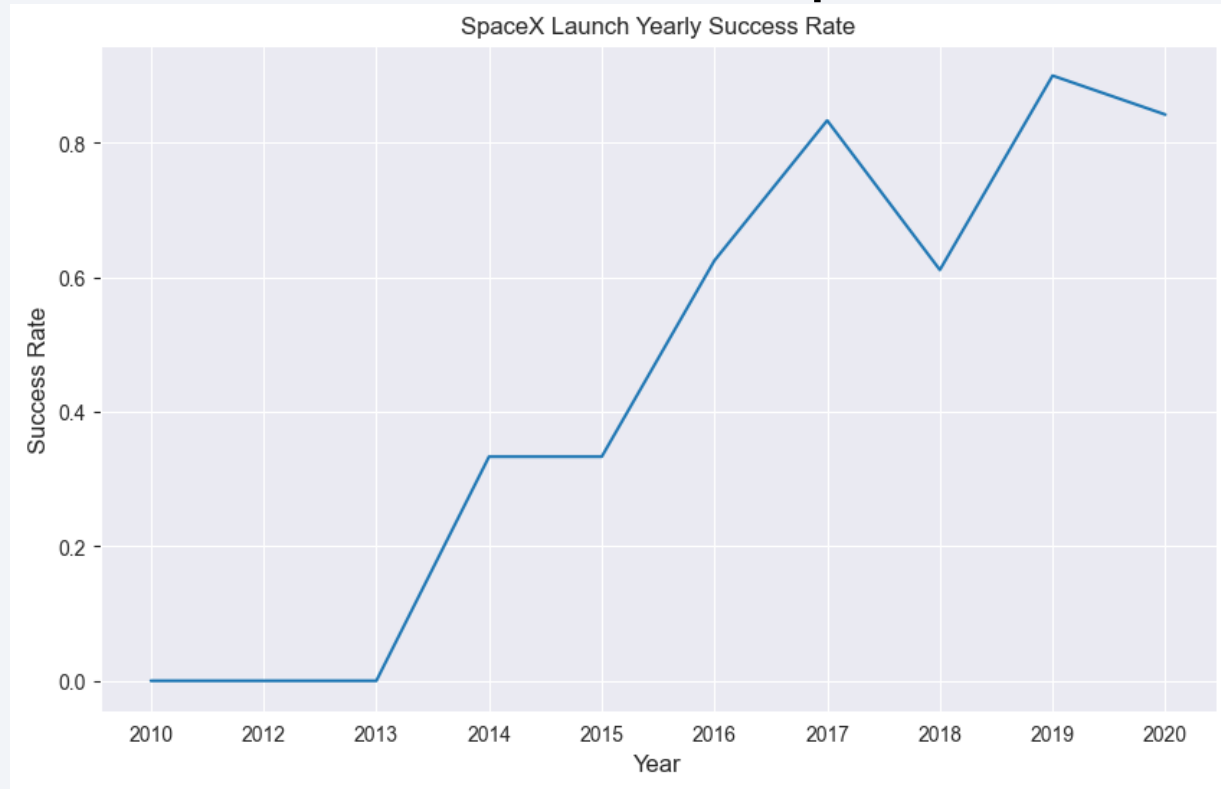




# Launch Success Yearly Trend

---

- The data shows an **upward trend** in success rates from **2013 to 2017**
- The success rate was **stable** in **2014**
- Success rate **decreased** in 2017-2018, however, it **improved** from 2018-2019



# All Launch Site Names

---

- **DISTINCT** keyword were used to get the unique launch site names from SpaceX data

```
%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'

- **LIKE** and **LIMIT** keywords were used to get 5 records where launch sites begin with '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_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

---

- **Total Payload Mass** launched by **NASA (CRS)** were calculated as **45596 kg** using the below query

```
%sql SELECT SUM("PAYLOAD_MASS_KG_") FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)'
```

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

```
Done.
```

```
SUM(PAYLOAD_MASS_KG_)
```

```
45596
```

# Average Payload Mass by F9 v1.1

---

- **Average Payload Mass** carried by **booster F9 v1.1** were calculated as **2928.4 kg** using the below query

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

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

```
Done.
```

```
AVG(PAYLOAD_MASS_KG_)
```

```
2928.4
```



# First Successful Ground Landing Date

---

- First Successful ground landing date where found using **MIN()** keyword
- The **first successful landing on ground pad** was on **2015-12-22**

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

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

```
Done.
```

```
MIN(Date)
```

```
2015-12-22
```

## Successful Drone Ship Landing with Payload between 4000 and 6000

---

- **WHERE** clause was used to filter to boosters that had successful landing on drone ship, and it was combined with an **AND** condition using the **BETWEEN** keyword to find the the records where the payload mass was greater than 4000 and less than 6000

```
%sql SELECT Booster_Version,"PAYLOAD_MASS_KG_",Landing_Outcome FROM SPACEXTABLE WHERE ("PAYLOAD_MASS_KG_" BETWEEN 4000 and 6000) AND Landing_Outcome = 'Success (drone ship)';  
* sqlite:///my_data1.db  
Done.
```

| Booster_Version | PAYLOAD_MASS_KG_ | Landing_Outcome      |
|-----------------|------------------|----------------------|
| F9 FT B1022     | 4696             | Success (drone ship) |
| F9 FT B1026     | 4600             | Success (drone ship) |
| F9 FT B1021.2   | 5300             | Success (drone ship) |
| F9 FT B1031.2   | 5200             | Success (drone ship) |

# Total Number of Successful and Failure Mission Outcomes

---

- Total number of successful and failure mission outcomes were calculated:
  - 99 Success
  - 1 Success (payload status unclear)
  - 1 Failure in flight

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

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

```
Done.
```

| Mission_Outcome                  | COUNT(Mission_Outcome) |
|----------------------------------|------------------------|
| Failure (in flight)              | 1                      |
| Success                          | 98                     |
| Success                          | 1                      |
| Success (payload status unclear) | 1                      |

# Boosters Carried Maximum Payload

- The booster versions which have carried the maximum payload mass were queried using **subquery** in the **WHERE** clause and **MAX()** function:

```
%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 |
|---------------|

# 2015 Launch Records

---

- Booster version, launch site names and months for failed landing outcomes in drone ship in 2015 were displayed using **WHERE** clause, **LIKE**, **AND** condition and **substr()** function

```
%sql SELECT substr(Date, 6,2) AS Month,"Booster_Version", "Launch_Site", "Landing_Outcome" FROM SPACEXTABLE WHERE (Landing_Outcome LIKE 'Failure%') AND (substr(Date,0,5)='2015');
```

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

```
Done.
```

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

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

---

- Ranked count of landing outcomes between 2010-06-04 and 2017-03-20 were queried in descending order

```
%sql SELECT Landing_Outcome,COUNT(Landing_Outcome) AS Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY Count DESC;
```

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

```
Done.
```

| Landing_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     |

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The image is a composite of a dark blue sky and a view of the Earth's surface, which is covered in a dense network of city lights and clouds. The lights are concentrated in the lower right portion of the image, while the upper left shows a clear view of the Earth's horizon and the surrounding space.

Section 3

# Launch Sites Proximities Analysis



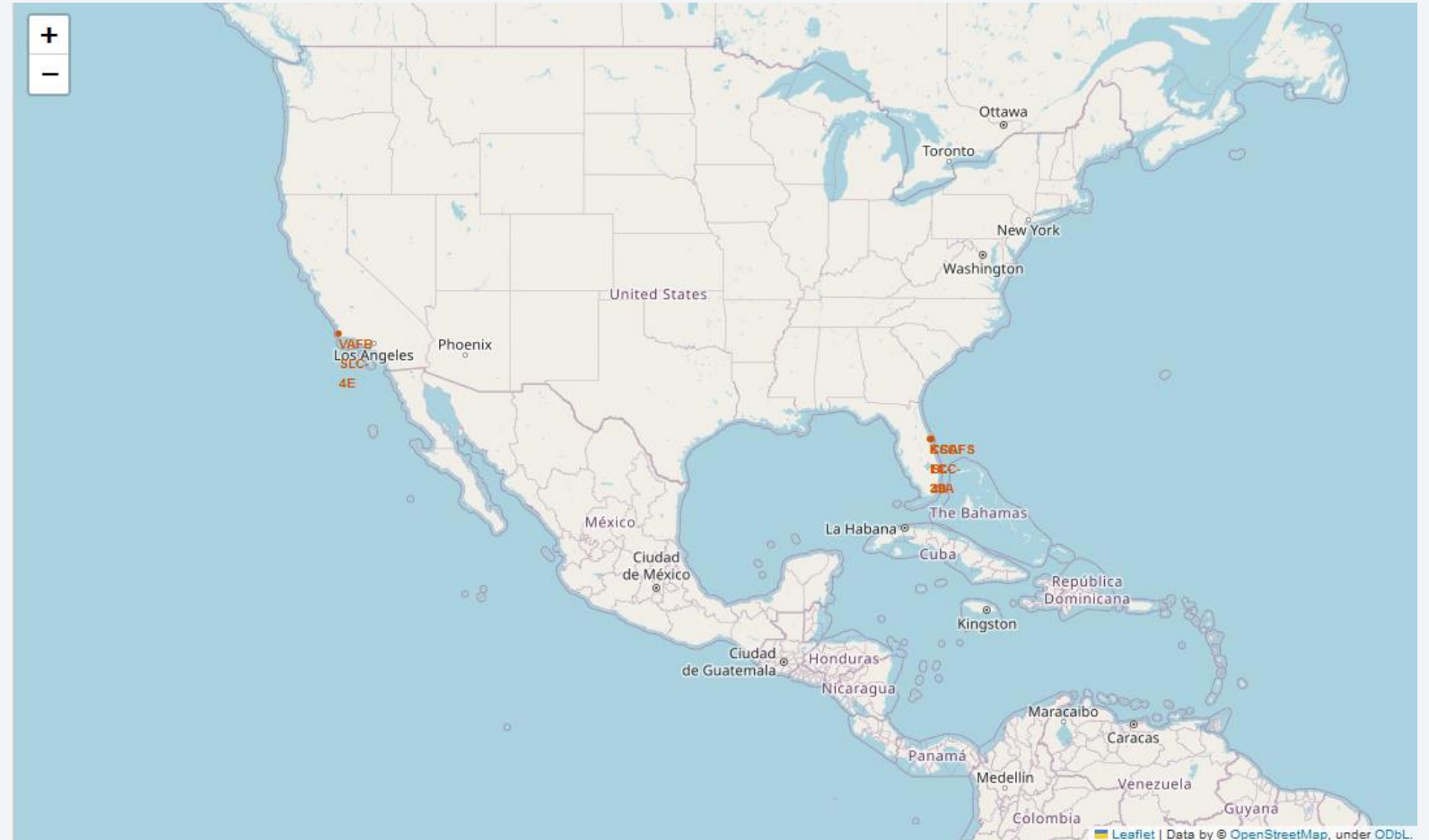
# Launch Sites on Map

All launch sites are relatively **close to the equator**.

The reason behind this is that launching from this position allows to take optimum advantage of the Earth's substantial rotational speed.

All launch sites are **close to the coastlines**.

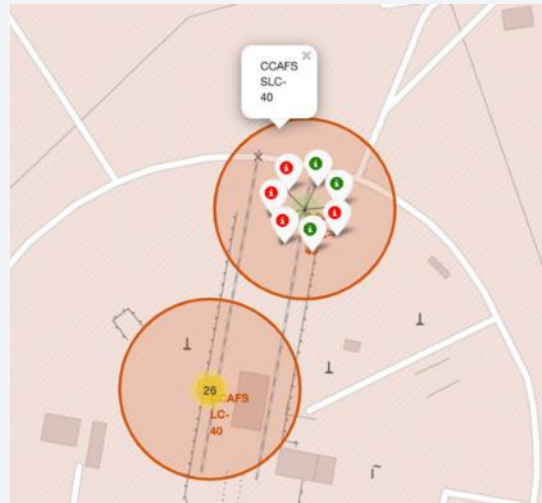
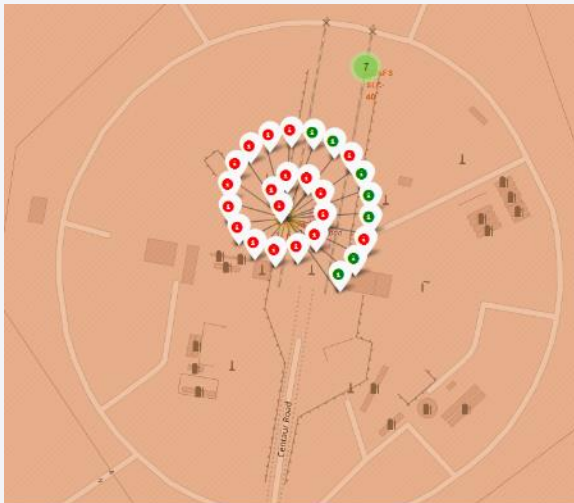
The launch site must also have a clear pathway during the launch to avoid any damage to populated areas in case of any accidents.





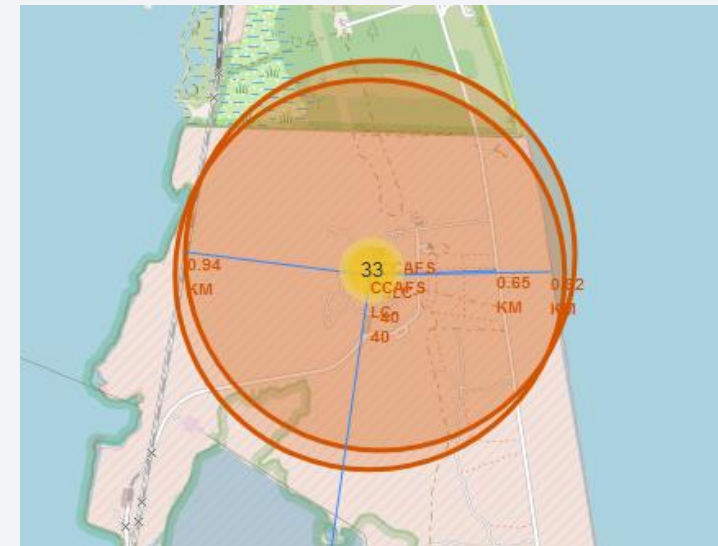
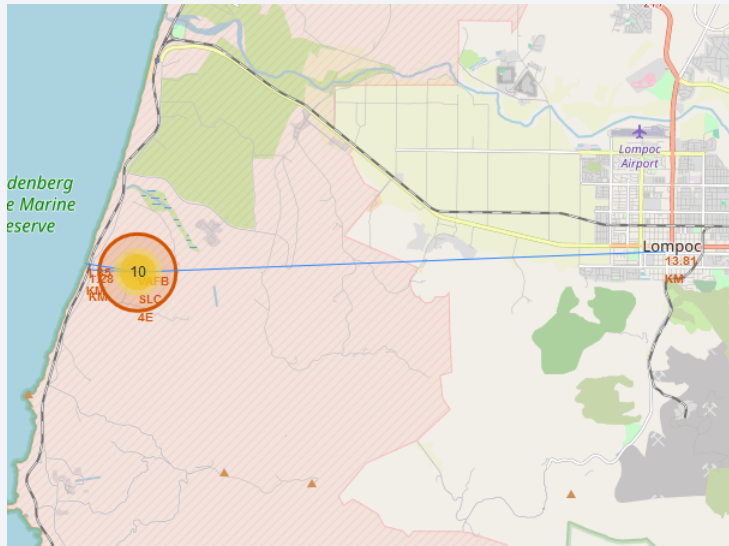
# Launch Outcomes on Map by Sites

- Outcomes:
  - Green marker indicates a successful launch
  - Red marker indicates an unsuccessful launch
- **KSC LC-39A** site has relatively high success rate
- **CCAFS SLC-40** has a **42.8 %** success rate (7/3)



# Distance to Proximities

- All launch site are relatively **close to railways, highways and the coastline** :  $\sim 1$  km
  - Due to the required transportations (heavy cargo, people and material) to maintain the complex ground facilities
  - Emergency water landing, minimizing people and property at risk from falling debris
- All launch site **keep distance from cities**:  $> 10$  km
  - For safety and security reasons





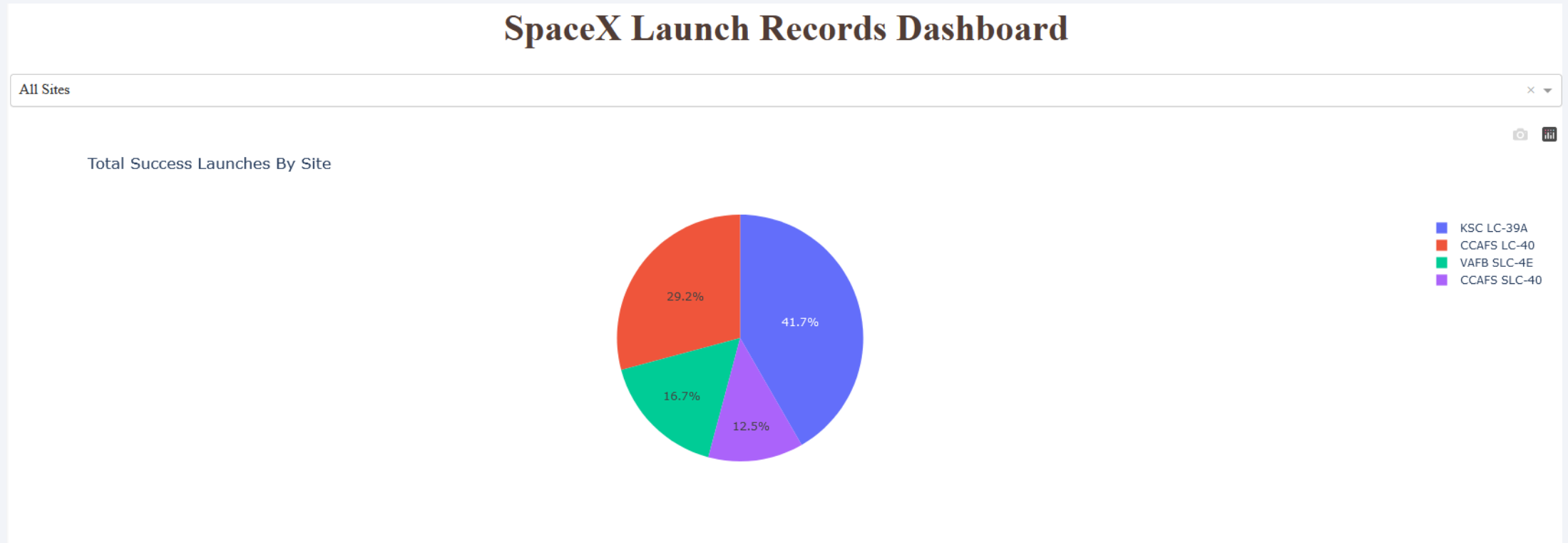
Section 4

# Build a Dashboard with Plotly Dash



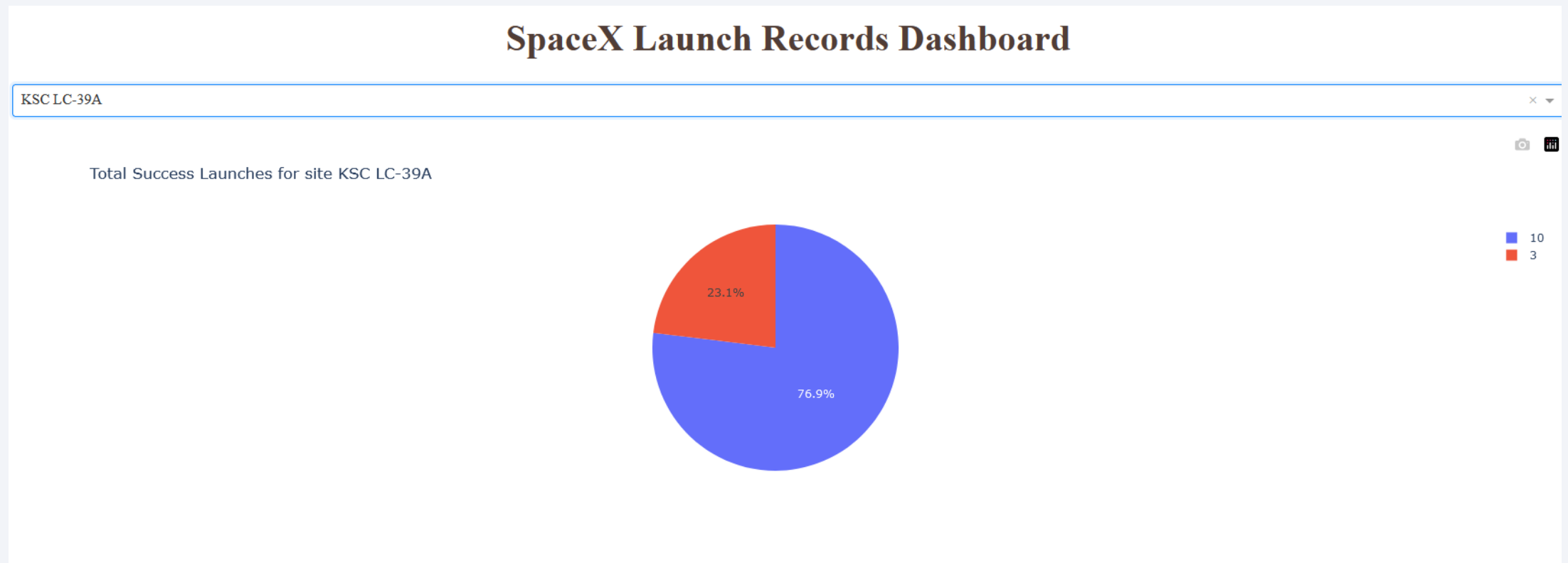
# Launch Success by Site

- **KSC LC-39A** site has the **highest success rate** amongst all sites, **41.7% (percent of total)**, based on the following piechart:



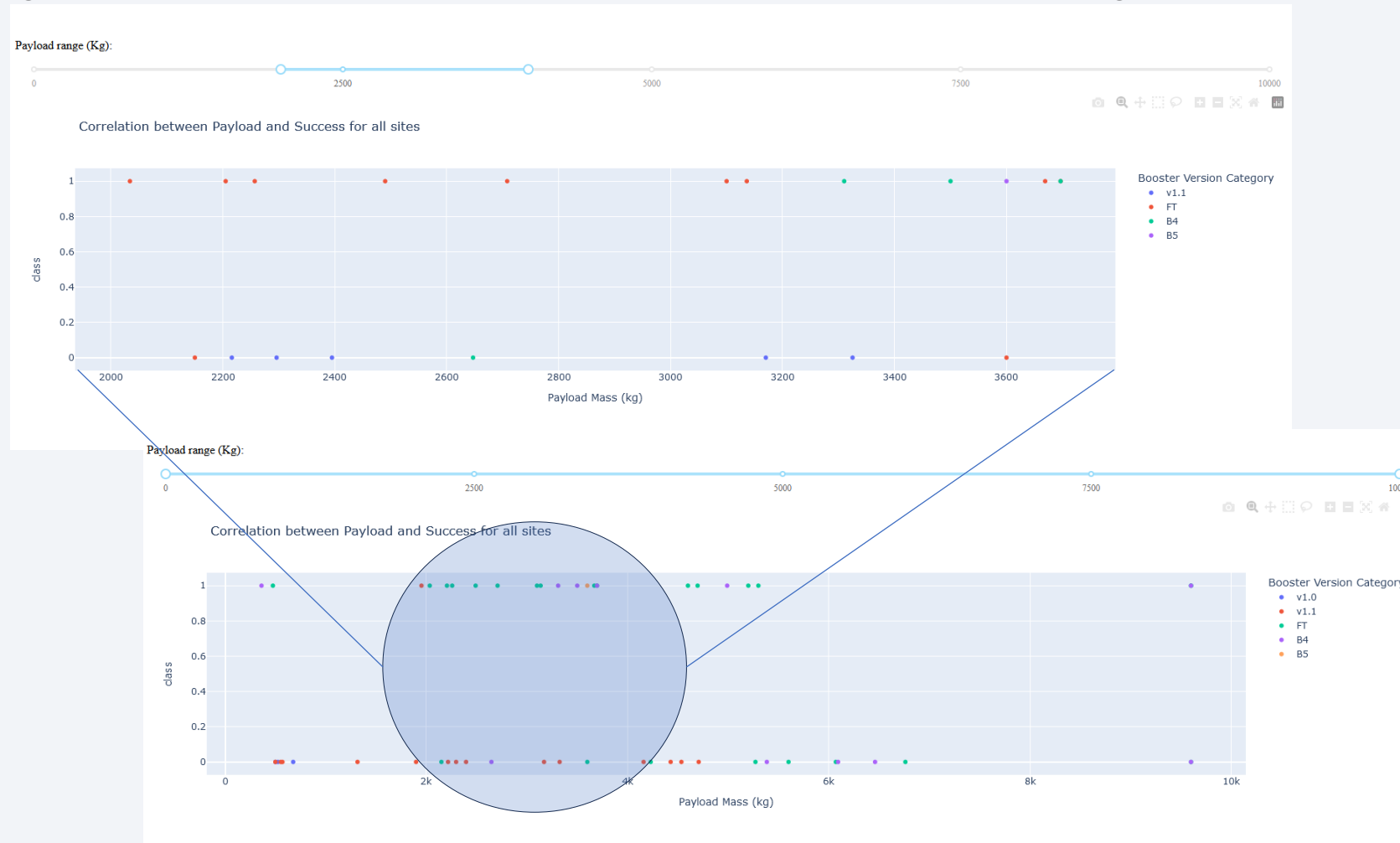
# Launch Success (KSC LC-39A)

- The KSC LC-39A site has the highest success rate: **76.9%**



# Payload vs. Launch Outcomes

- The highest launch success rate is between 2,000-4,000 kg payload mass





Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

---

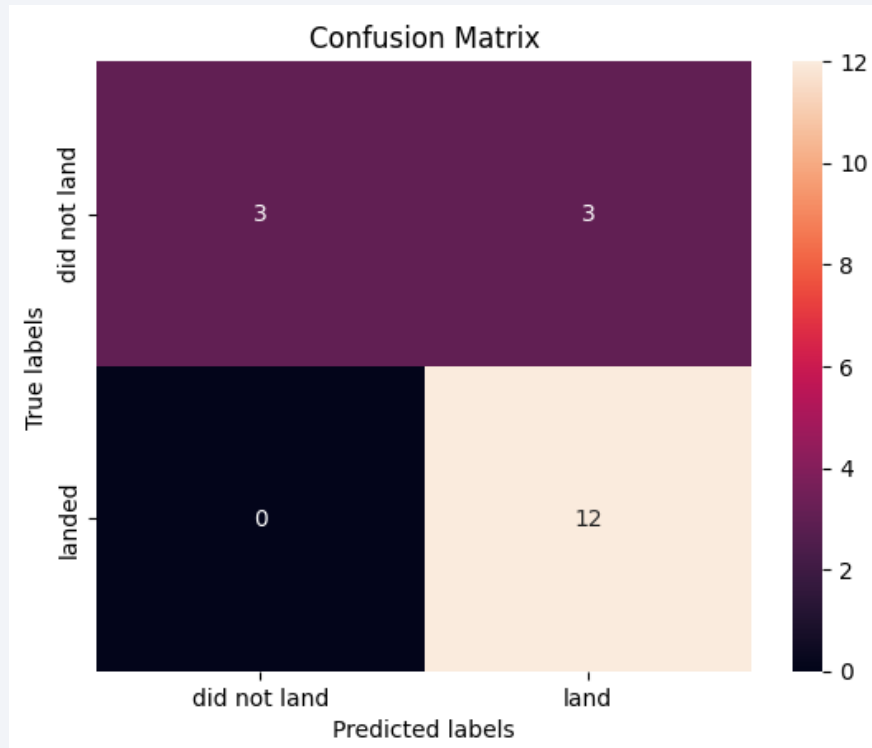
- Basically, all models performed equally, the accuracy of the models were the same.
  - **Accuracy = 0.83**
- The Decision Tree algorithm slightly outperformed the others in training accuracy – might be due to overfitting
- Best training and fitting speed: **Logistic Regression**
- The dataset was relatively **small**, which could potentially lead to overfitting. Therefore, it might worth to **avoid complex models** such as SVM.

```
print("Accuracy of Logistic Regression:", logreg_cv.score(X_test, Y_test))
print("Accuracy of SVM:", svm_cv.score(X_test, Y_test))
print("Accuracy of DecisionTree:", tree_cv.score(X_test, Y_test))
print("Accuracy of KNN:", knn_cv.score(X_test, Y_test))
```

```
Accuracy of Logistic Regression: 0.8333333333333334
Accuracy of SVM: 0.8333333333333334
Accuracy of DecisionTree: 0.8333333333333334
Accuracy of KNN: 0.8333333333333334
```

|   | ML method           | Best train score |
|---|---------------------|------------------|
| 0 | Logistic Regression | 0.846429         |
| 1 | SVN                 | 0.848214         |
| 2 | Decison Tree        | 0.875000         |
| 3 | KNN                 | 0.848214         |

# Confusion Matrix



- All confusion matrices were **identical**
- Confusion matrix results:
  - 12 True Positive
  - 3 True Negative
  - 0 False Negative
  - **3 False Positive**
- The present of False Positives (Type I error) results is not optimal
- $\text{Precision} = \frac{TP}{TP+FP} = 0.8$
- $\text{Recall} = \frac{TP}{TP+FN} = 1.0$
- $\text{F1 score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = 0.89$

# Conclusions

---

- **Model performance:** The models performed similarly on the test set. Due to the risk of overfitting on a small dataset, it is advisable to avoid selecting more complex models. A less computationally intensive model is also a good choice (reduced training and prediction time).
- **Launch site:** All of the launch sites are near the **equator** and near **coastlines**.
- **Launch success:** The launch success rate increases with time.
- **Launch sites success rate:** **KSC LC-39A** site has the highest success rate amongst all sites.
- **Payload Mass:** The higher the payload mass (kg), the higher the success rate.
- **Dataset size:** The dataset was relatively small. It is worth considering collecting more data to improve the model performance, prediction accuracy and reduce the false positive rate.

Thank you!

