
# Winning Space Race with Data Science

Arseniy
14/08/2023

# Outline

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

# Executive Summary

- Summary of methodologies
  - Data Collection API
  - Data Wrangling
  - EDA using SQL
  - EDA using Pandas and Matplotlib
  - Interactive Visual Analytics with Folium and Plotly Dash
  - Predictive analysis using Machine Learning
- Summary of all results
  - EDA results
  - Interactive analytics screenshots
  - Predictive analysis results

# Introduction

SpaceX, founded by Elon Musk in 2002, has revolutionized space technology with the ultimate goal of making it possible for humans to live on other planets. A significant advancement in their space journey is the Falcon 9 rocket, which is not only notable for its efficiency but also for its reusability. The cost-efficiency of the Falcon 9 is primarily due to SpaceX's ability to reuse its first stage, thereby cutting down on the expenses associated with manufacturing a new rocket for every launch. This technological innovation provides SpaceX a competitive edge over its rivals in the space industry. On SpaceX's website, the cost of a Falcon 9 rocket launch is pegged at 62 million dollars, a stark contrast to other providers who price their launches at a starting rate of 165 million dollars.

The ability to reuse the first stage of the Falcon 9 rocket is dependent on the successful landing of this component post-launch. If the first stage lands successfully, it can be refurbished and prepped for another mission. This reusability factor can significantly reduce costs and enhance the frequency of space missions. As a result, predicting the successful landing of the Falcon 9's first stage is of paramount importance, not only for SpaceX but also for any potential competitor or customer in the space market.

Problems to find answers:

- Predictability of Falcon 9's First Stage Landing
- Cost Implications of First Stage Landing
- Competitive Bidding Strategy

Section 1

# Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web Scraping
- Perform data wrangling
  - One-hot encoding
  - Data cleaning
- 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 collection serves as the foundation for informed decision-making, allowing for targeted information to be gathered, measured, and evaluated. For this project, data was primarily sourced via two methods: REST API and web scraping from Wikipedia.

For the **REST API** method:

1. Initiated with a 'GET' request to retrieve the data.
2. The returned response was then decoded from JSON format.
3. Leveraging the json_normalize() function, this data was transformed into a Pandas dataframe.
4. Following this, a thorough data cleaning process was executed, which involved identifying and addressing any missing values.

In the case of **web scraping**:

1. 'BeautifulSoup' was employed to extract launch records presented in HTML tables.
2. Subsequent to extraction, these tables were parsed and seamlessly converted into a Pandas dataframe to facilitate deeper analysis.

# Data Collection – SpaceX API

1. Initiated with a 'GET' request to retrieve the data.
2. The returned response was then decoded from JSON format.
3. Leveraging the json_normalize() function, this data was transformed into a Pandas dataframe.
4. Following this, a thorough data cleaning process was executed, which involved identifying and addressing any missing values.

Link: https://github.com/arn1305/applieddscapstone.git

```
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_json = response.json()
data = pd.json_normalize(data_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
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
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)]
```

# Data Collection - Scraping

1. 'BeautifulSoup' was employed to extract launch records presented in HTML tables.
2. Subsequent to extraction, these tables were parsed and seamlessly converted into a Pandas dataframe to facilitate deeper analysis.

Link:
https://github.com/arn1305/applieddscapstone.git

```
F9data = requests.get(static_url).text
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

soup = BeautifulSoup(F9data, 'html.parser')
```

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

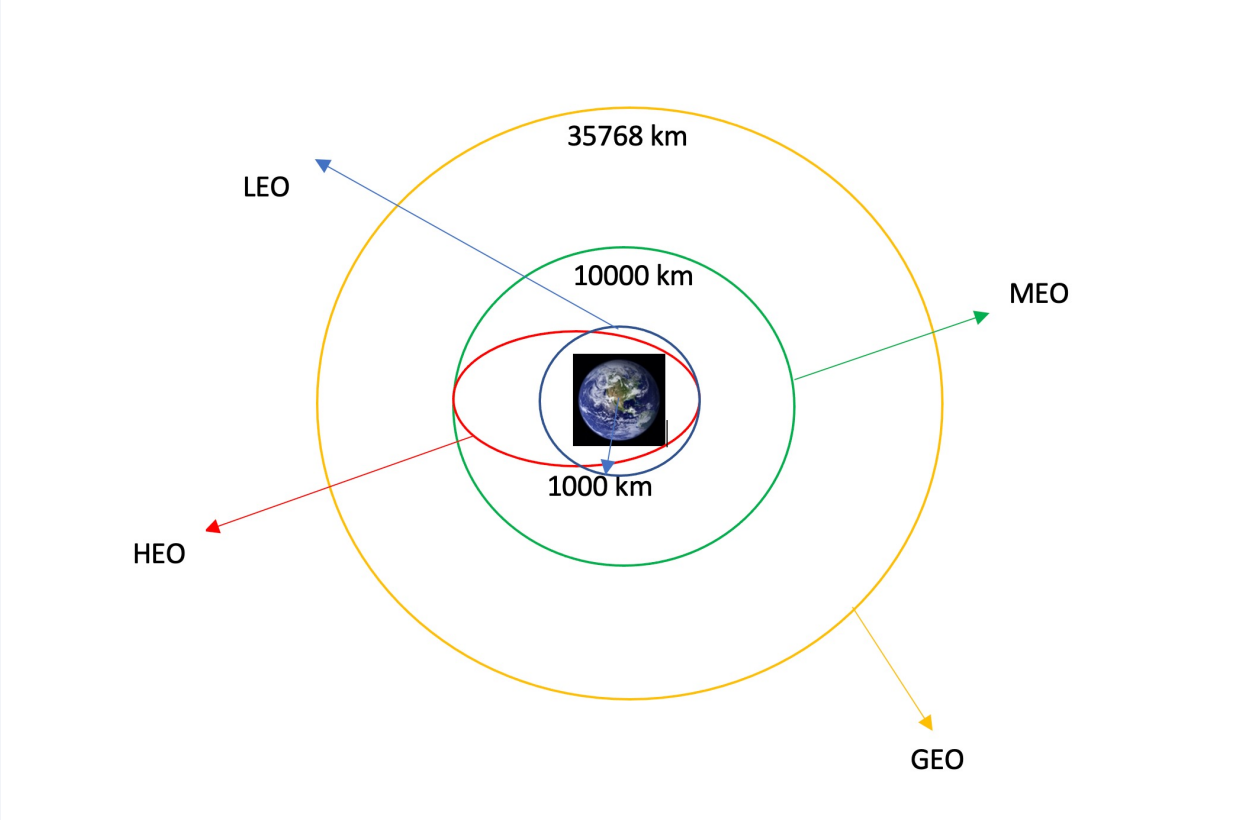
```
# Use soup.title attribute
soup.title
```

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

th_elements = first_launch_table.find_all('th')

for th in th_elements:
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

# Data Wrangling

Data wrangling involves cleaning, structuring, and enhancing raw data into a more digestible and usable format for subsequent analysis or modelling.

1. Calculate the number of launches on each site
2. Calculate the number and occurrence of each orbit
3. Calculate the number and occurrence of mission outcome per orbit type
4. Create a landing outcome label from Outcome column



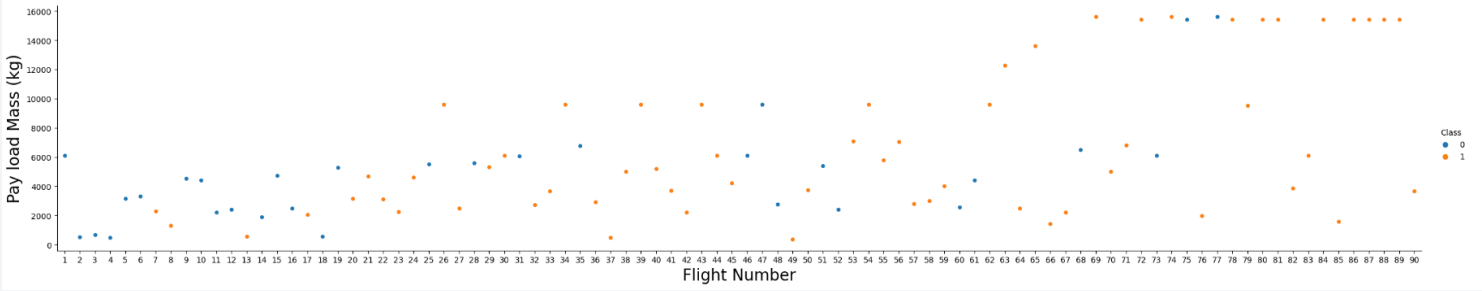

Link:
https://github.com/arn1305/applieddscapstone.git

# EDA with Data Visualization

Scatter plots were used to discover relationships between the following attributes:

- Payload Mass vs Flight Number
- Flight Number vs Launch Site
- Payload Mass vs Launch Site
- Flight Number vs Orbit Type
- Payload Mass vs Orbit Type

```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```




(Payload Mass vs Flight Number Example)

Scatter plots allow relationships to be easily visualized and allow us to see which factors influence the success of the launch the most.

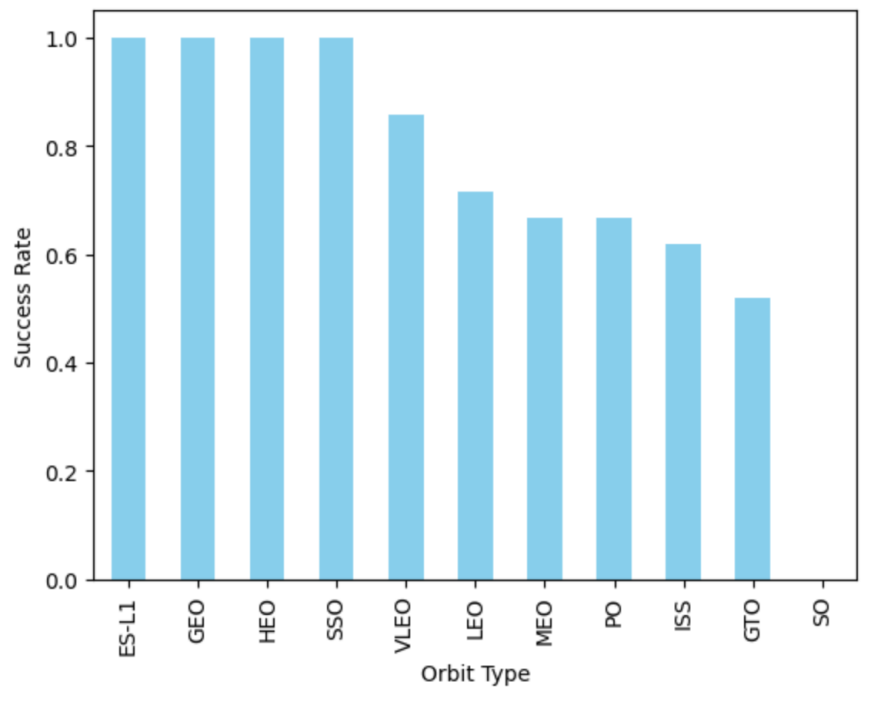
Link:
https://github.com/arn1305/applieddscapstone.git

# EDA with Data Visualization

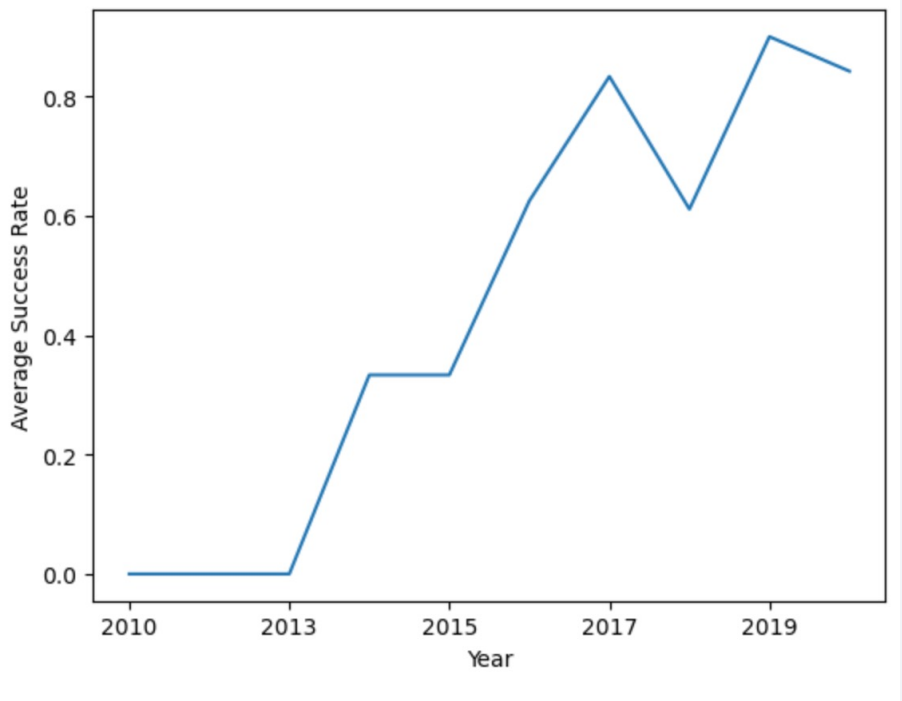
Other types of graphs were also used such as:

- Bar chart
- Line graph

By using various types of graphs, we have managed to explore the data in a holistic manner allowing us to draw valuable insights from the data.




Bar chart showing Success Rate vs Orbit Type relationship




Line graph showing Success Rate vs Year relationship

Link:
https://github.com/arn1305/applieddscapstone.git

# EDA with SQL

SQL allowed us to explore the dataset in more detail. The queries completed include:

- Names of the unique launch sites in the space mission
- 5 records where launch sites begin with the string 'CCA'
- Total payload mass carried by boosters launched by NASA (CRS)
- Average payload mass carried by booster version F9 v1.1
- Date when the first succesful landing outcome in ground pad was acheived.
- Names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Total number of successful and failure mission outcomes
- Names of the booster_versions which have carried the maximum payload mass.
- The records which display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
- Rank of 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.



Link:
https://github.com/arn1305/applieddscapstone.git

# Build an Interactive Map with Folium

We created a folium map with markers, circles, and lines to visualize the launch sites' locations, as well as the success/failure rate of launches. By adding these map objects, we aimed to identify patterns related to proximity to the Equator line and the coast.

We created markers for all launch records. If a launch was successful (class=1), we used a green marker and if a launch failed, we used a red marker (class=0).

We also marked down a point on the closest coastline using MousePosition and calculated the distance between the coastline point and the launch site.

```
from math import sin, cos, sqrt, atan2, radians

def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```



https://github.com/arn1305/applieddscapstone/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

# Build a Dashboard with Plotly Dash

We designed an intuitive interface using Plotly Dash, enabling users to interact with and manipulate the data according to their preferences. We illustrated pie charts that represent the total launches from specific locations. Following that, we showcased a scatter plot that depicts the correlation between Outcome and Payload Weight (Kg) across various booster models.

https://github.com/arn1305/applieddscapstone/blob/main/Untitled%203.py

# Predictive Analysis (Classification)

**Model Construction Steps**

**Dataset Initialization**:

Utilized NumPy and Pandas libraries to load the dataset.

**Data Preprocessing**:

Modifed the data and subsequently divided it into training and test segments.

**ML Strategy Selection**:

Determined the most suitable machine learning method.

**Parameter Tuning**:

Engaged GridSearchCV to set algorithmic parameters and integrated them into the dataset.

**Model Assessment Techniques**

**Accuracy Measurement**:

Evaluated each model's precision.

**Parameter Optimization**:

Extracted the refined hyperparameters associated with each algorithm.

**Visualization**:

Rendered the confusion matrices for clarity.

**Model Enhancement Techniques**

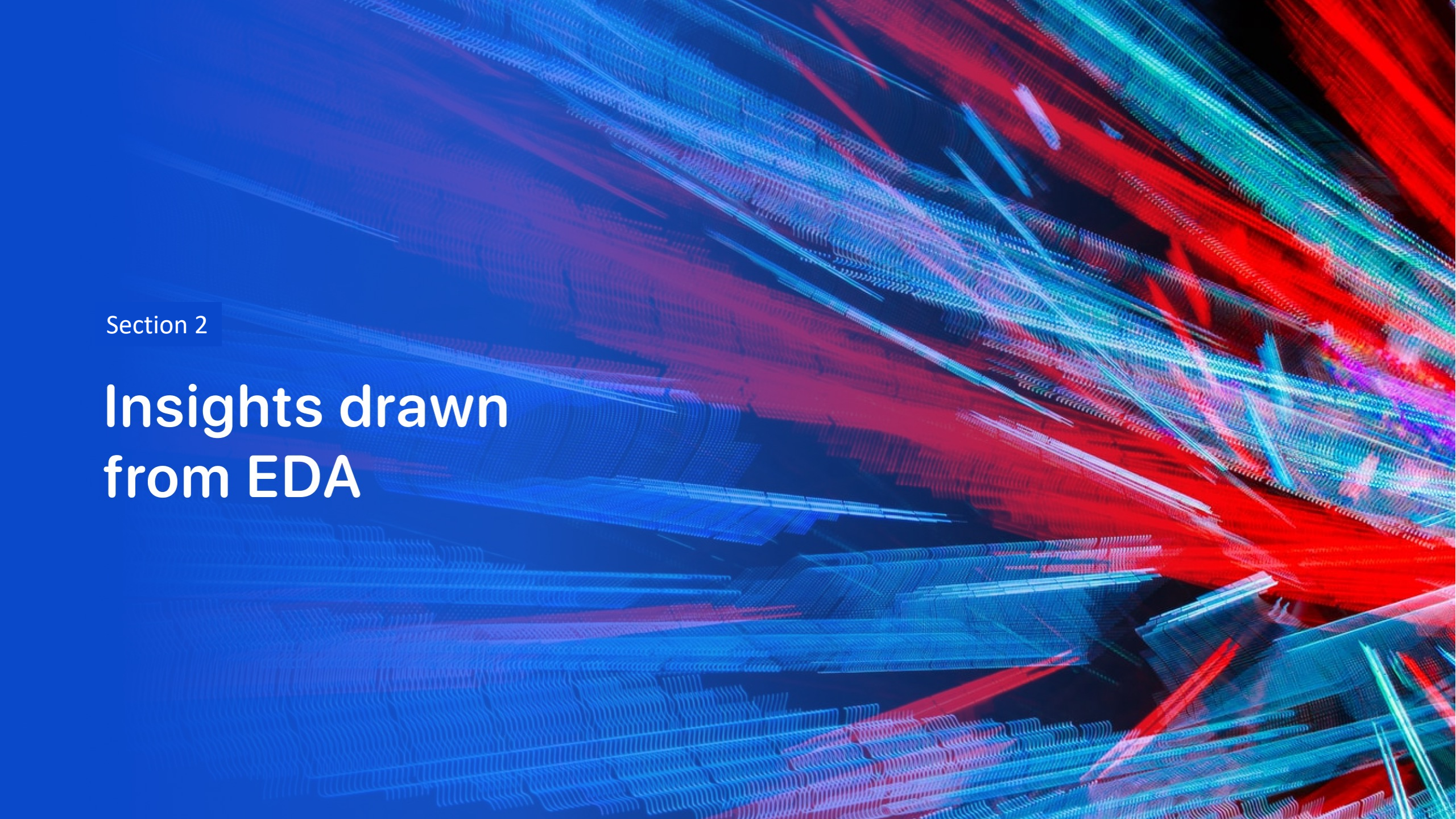
Refined the model's effectiveness, adopted feature engineering combined with algorithm tuning.

**Determination of the Best Model**

The model achieving the peak accuracy rate was recognized as the superior-performing model.

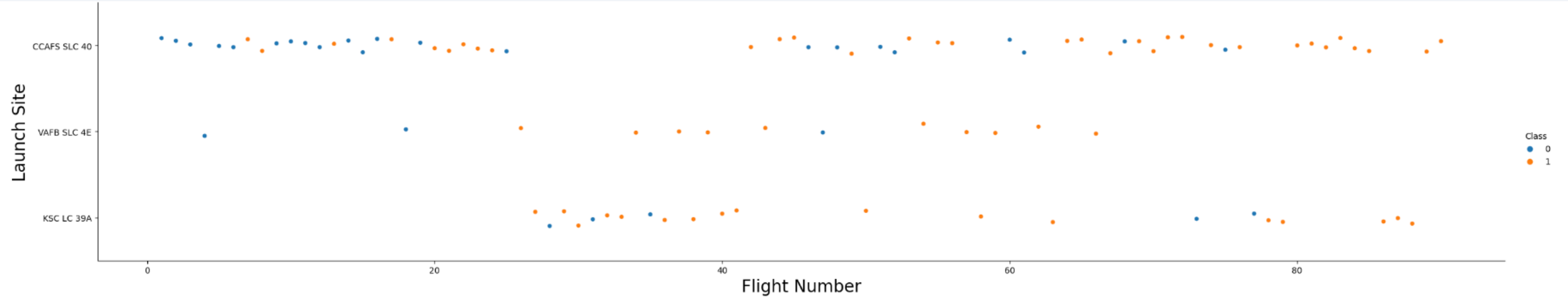
# Results

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


Section 2
Insights drawn from EDA

# Flight Number vs. Launch Site

```
### TASK 1: Visualize the relationship between Flight Number and Launch Site

sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

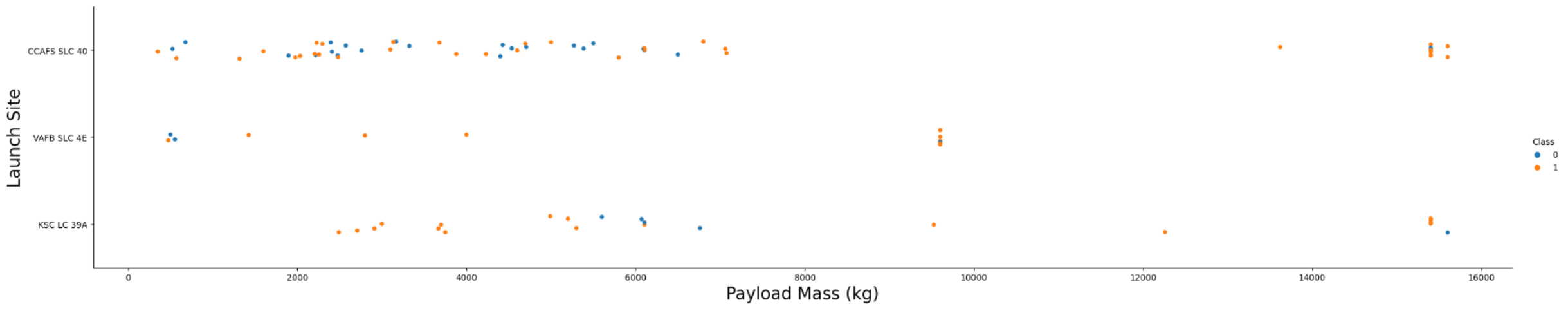





https://github.com/arn1305/applieddscapstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

# Payload vs. Launch Site

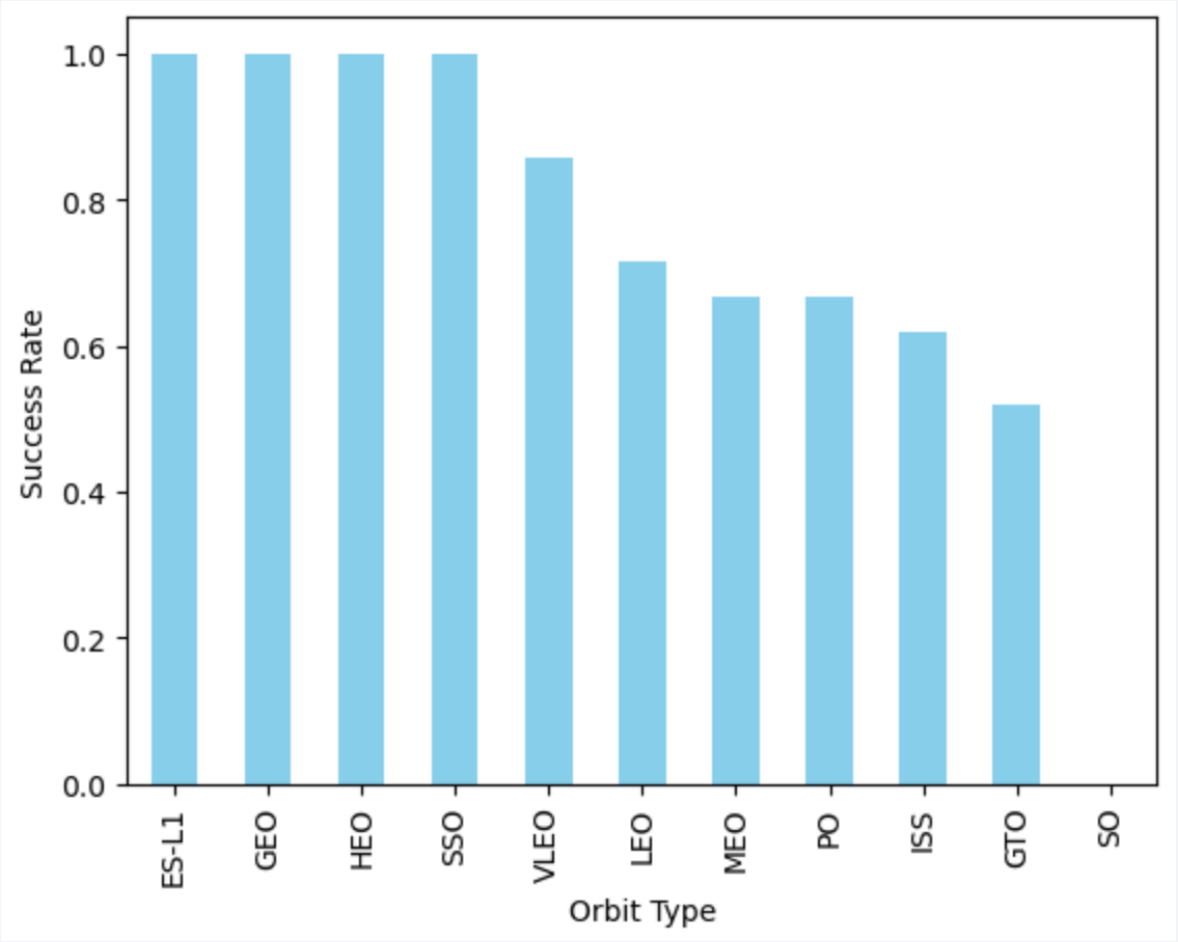
```
### TASK 2: Visualize the relationship between Payload and Launch Site

sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```






https://github.com/arn1305/applieddscapstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

# Success Rate vs. Orbit Type



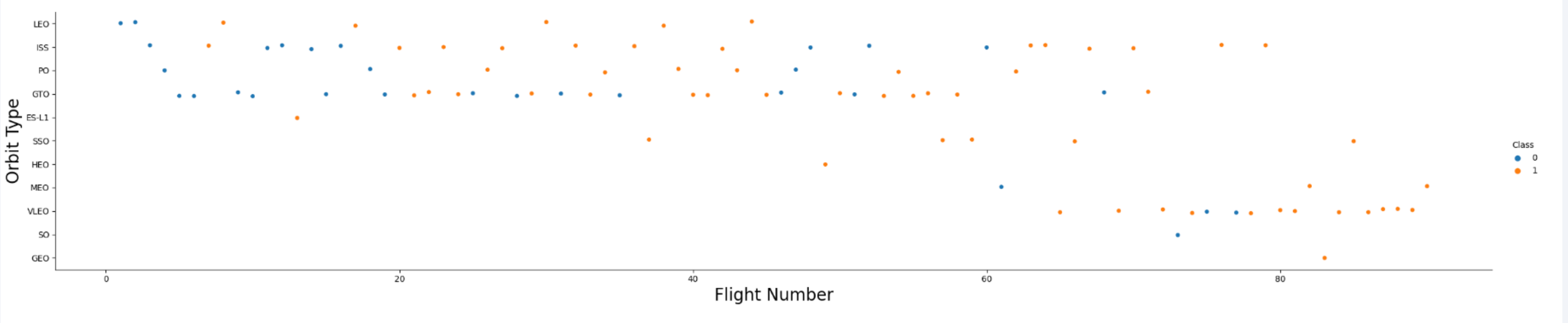

```
### TASK  3: Visualize the relationship between success rate of each orbit type

success_rate = df.groupby('Orbit')['Class'].mean()
success_rate.sort_values(ascending=False).plot(kind='bar', color='skyblue')
plt.ylabel('Success Rate')
plt.xlabel('Orbit Type')
plt.show()
```



https://github.com/arn1305/applieddscapstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

# Flight Number vs. Orbit Type

```
### TASK  4: Visualize the relationship between FlightNumber and Orbit type

sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit Type",fontsize=20)
plt.show()
```

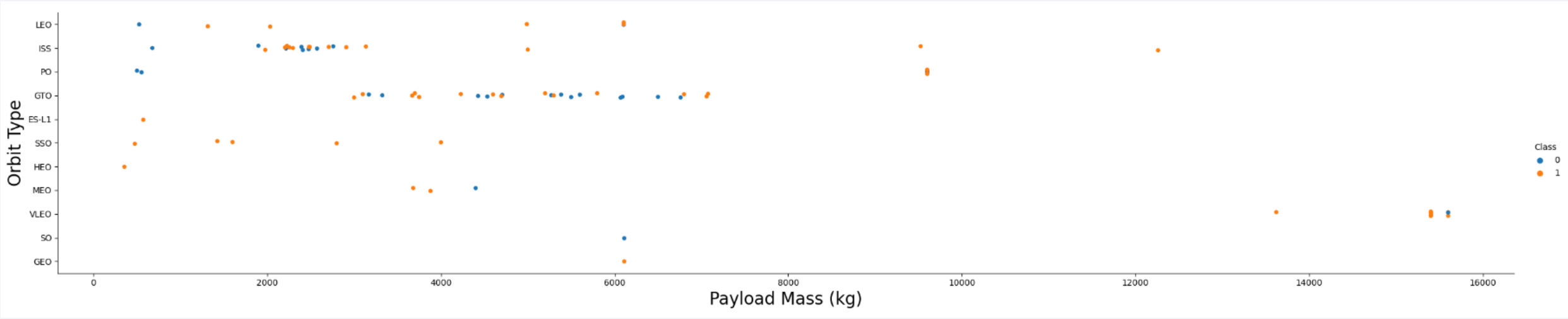





https://github.com/arn1305/applieddscapstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

# Payload vs. Orbit Type

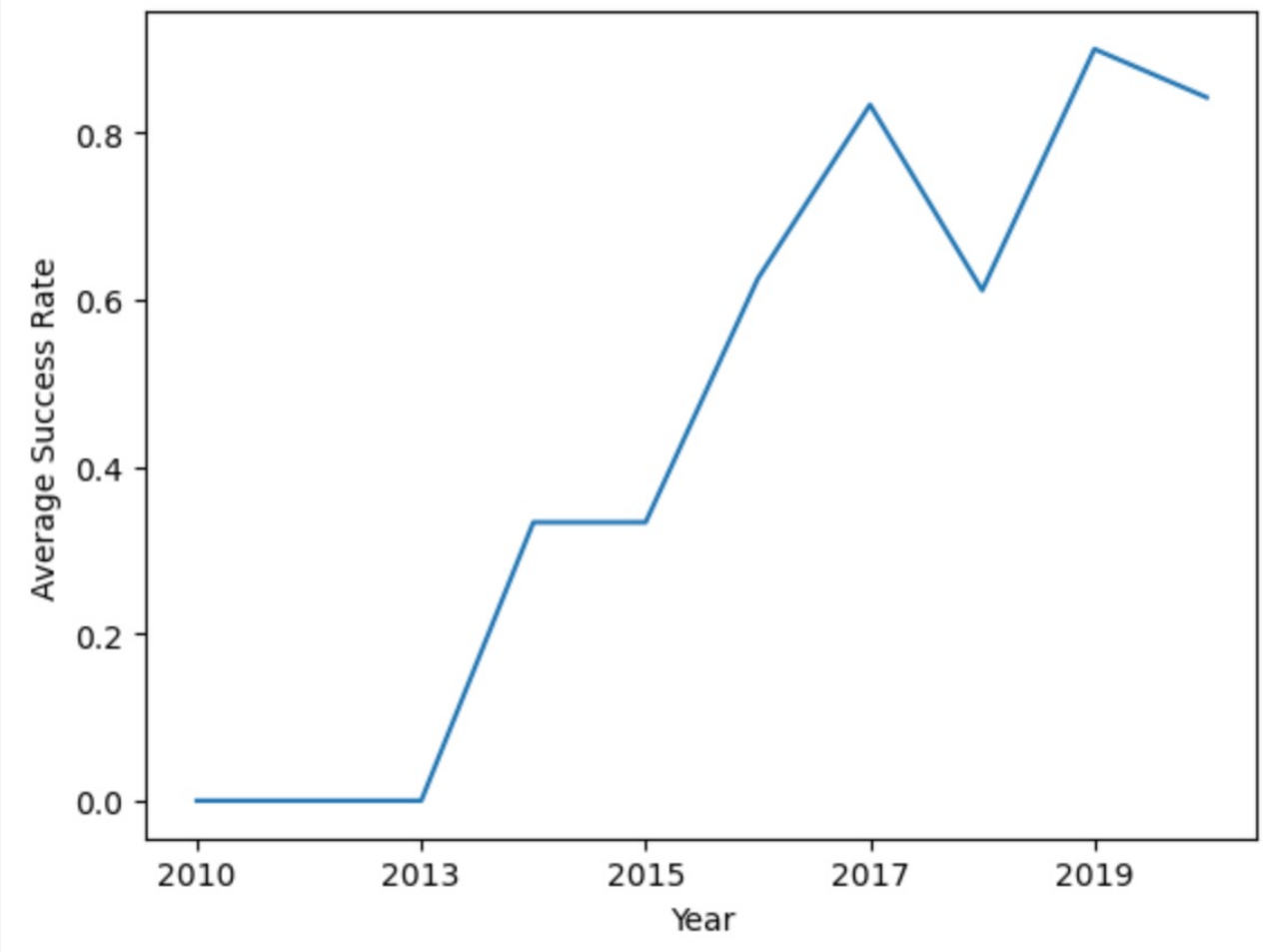
```
### TASK  5: Visualize the relationship between Payload and Orbit type

sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload Mass (kg)",fontsize=20)
plt.ylabel("Orbit Type",fontsize=20)
plt.show()
```






https://github.com/arn1305/applieddscapstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

# Launch Success Yearly Trend




```
### TASK  6: Visualize the launch success yearly trend
```

You can plot a line chart with x axis to be `Year` and y axis to be average success rate, to get the average launch success trend.

The function will help you get the year from the date:

```
# A function to Extract years from the date
year=[]
def Extract_year():
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
Extract_year()
df['Date'] = year
df.head()
```

| | FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome | Flights | GridFins | Reused | Legs | LandingPad |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2010 | Falcon 9 | 6104.959412 | LEO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN |
| **1** | 2 | 2012 | Falcon 9 | 525.000000 | LEO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN |
| **2** | 3 | 2013 | Falcon 9 | 677.000000 | ISS | CCAFS SLC 40 | None None | 1 | False | False | False | NaN |
| **3** | 4 | 2013 | Falcon 9 | 500.000000 | PO | VAFB SLC 4E | False Ocean | 1 | False | False | False | NaN |
| **4** | 5 | 2013 | Falcon 9 | 3170.000000 | GTO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN |

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate

yearly_success_rate = df.groupby('Date')['Class'].mean()
yearly_success_rate.plot(kind='line', linestyle='-')
plt.ylabel('Average Success Rate')
plt.xlabel('Year')
plt.show()
```



https://github.com/arn1305/applieddscapstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

# All Launch Site Names

## Task 1

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

In [8]:
```
%sql select distinct(LAUNCH_SITE) from SPACEXTABLE
```

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

Out[8]:

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

# Launch Site Names Begin with 'CCA'

## Task 2

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

In [9]:
```
%sql select * from SPACEXTABLE where LAUNCH_SITE like 'CCA%' limit 5
```

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

Out[9]:

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_ |
|---|---|---|---|---|---|---|---|---|---|
| 2010-04-06 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (I |
| 2010-08-12 | 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 (I |
| 2012-05-22 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | N |
| 2012-08-10 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | N |
| 2013-01-03 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | N |

# Total Payload Mass

## Task 3

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

In [10]:
```
%sql select SUM(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer = 'NASA (CRS)'
```

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

Out[10]:

| SUM(PAYLOAD_MASS__KG_) |
|---|
| 45596 |

# Average Payload Mass by F9 v1.1

## Task 4

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

In [11]:
```
%sql select AVG(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version = 'F9 v1.1'
```

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

Out[11]:

| AVG(PAYLOAD_MASS__KG_) |
|---|
| 2928.4 |

# First Successful Ground Landing Date

## Task 5

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

*Hint:Use min function*

In [12]:
```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome = 'Success (ground pad)'
```

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

Out[12]:

| min(Date) |
|---|
| 2015-12-22 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

## Task 6

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

In [13]:
```
%sql select Booster_Version from SPACEXTABLE where Landing_Outcome = 'Success (drone ship)' and 4000<PAYLOAD_MASS
```

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

Out[13]:

| Booster_Version |
|---|
| F9 FT B1021.1 |
| F9 FT B1022 |
| F9 FT B1023.1 |
| F9 FT B1026 |
| F9 FT B1029.1 |
| F9 FT B1021.2 |
| F9 FT B1029.2 |
| F9 FT B1036.1 |
| F9 FT B1038.1 |
| F9 B4 B1041.1 |
| F9 FT B1031.2 |
| F9 B4 B1042.1 |
| F9 B4 B1045.1 |
| F9 B5 B1046.1 |

# Total Number of Successful and Failure Mission Outcomes

## Task 7

List the total number of successful and failure mission outcomes

In [17]:
```
%sql select count(Mission_Outcome) from SPACEXTABLE where Mission_Outcome like'Success%' or Mission_Outcome like'
```

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

Out[17]:

| count(Mission_Outcome) |
|---|
| 101 |

# Boosters Carried Maximum Payload

## Task 8

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

In [21]:
```
%sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ in (select max(PAYLOAD_MASS__KG_) from SPACE
```

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

Out[21]:

| 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

## Task 9

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

**Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.**

```
%sql select substr('Date',4,2) as month, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE where sub
```

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

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

## Task 10

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

In [28]:
```
%sql select Landing_Outcome, count(*) as count_outcomes from SPACEXTABLE where Date between '2010-06-04' and '201
```

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

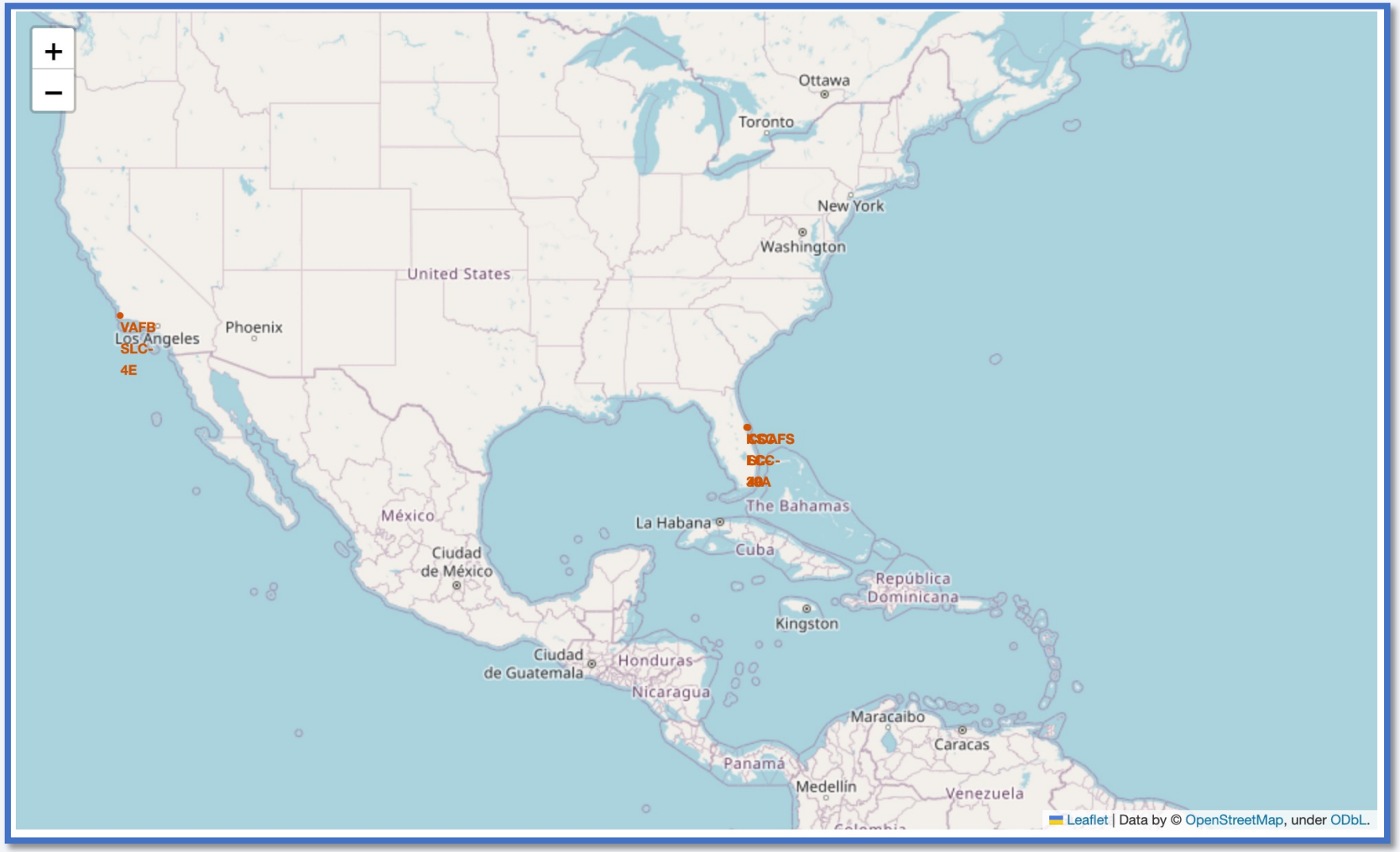
Out[28]:

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

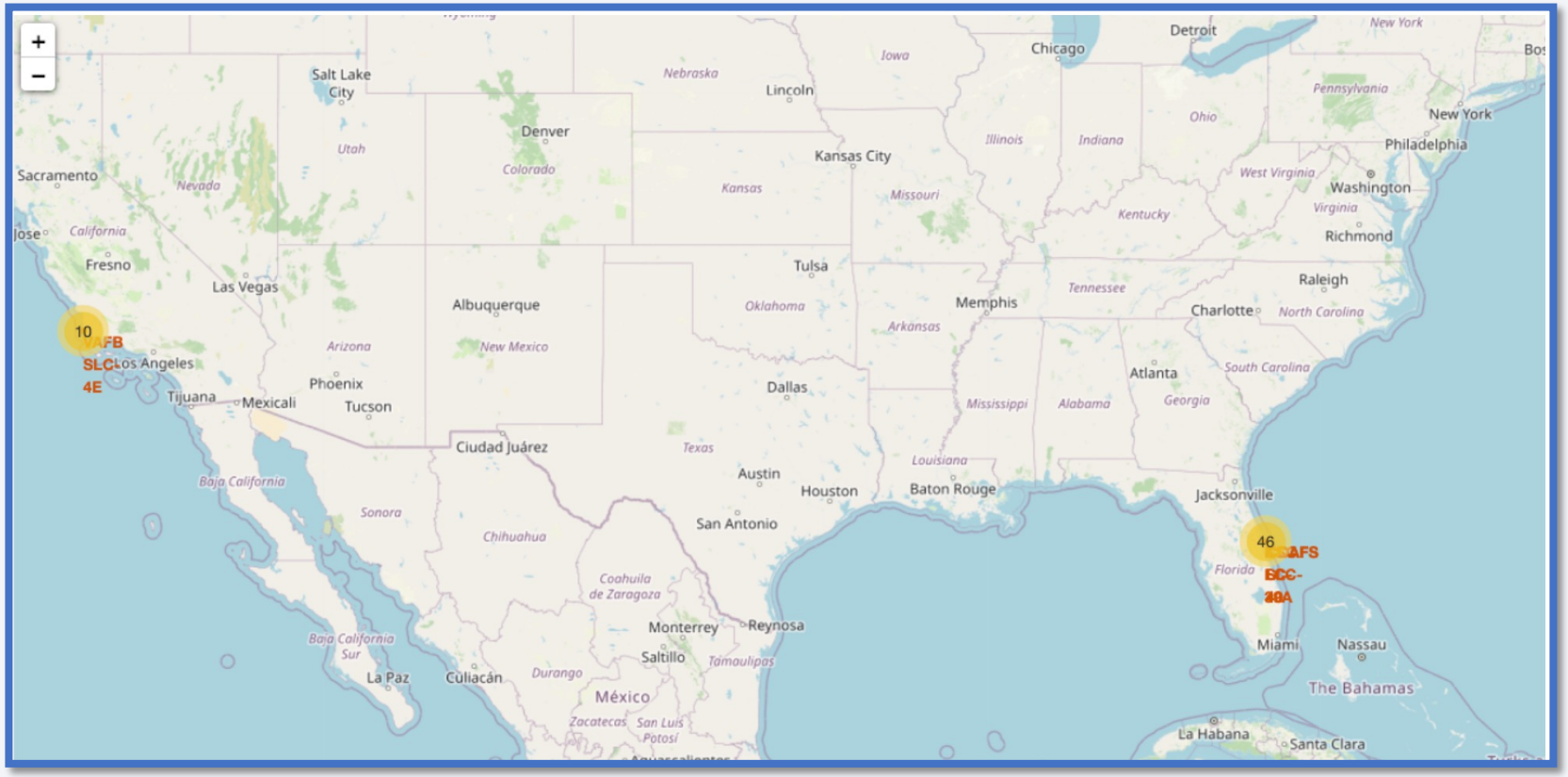

Section 3
Launch Sites
Proximities Analysis

# Launch Sites

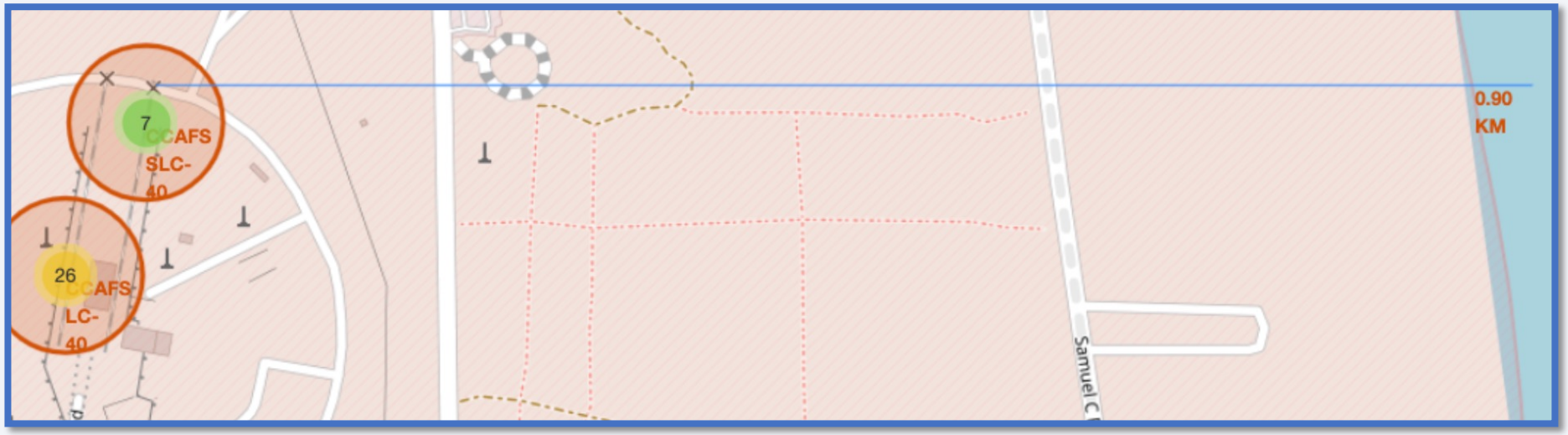
From the map, we can observe that all the launch sites are located in USA (in Florida and California)

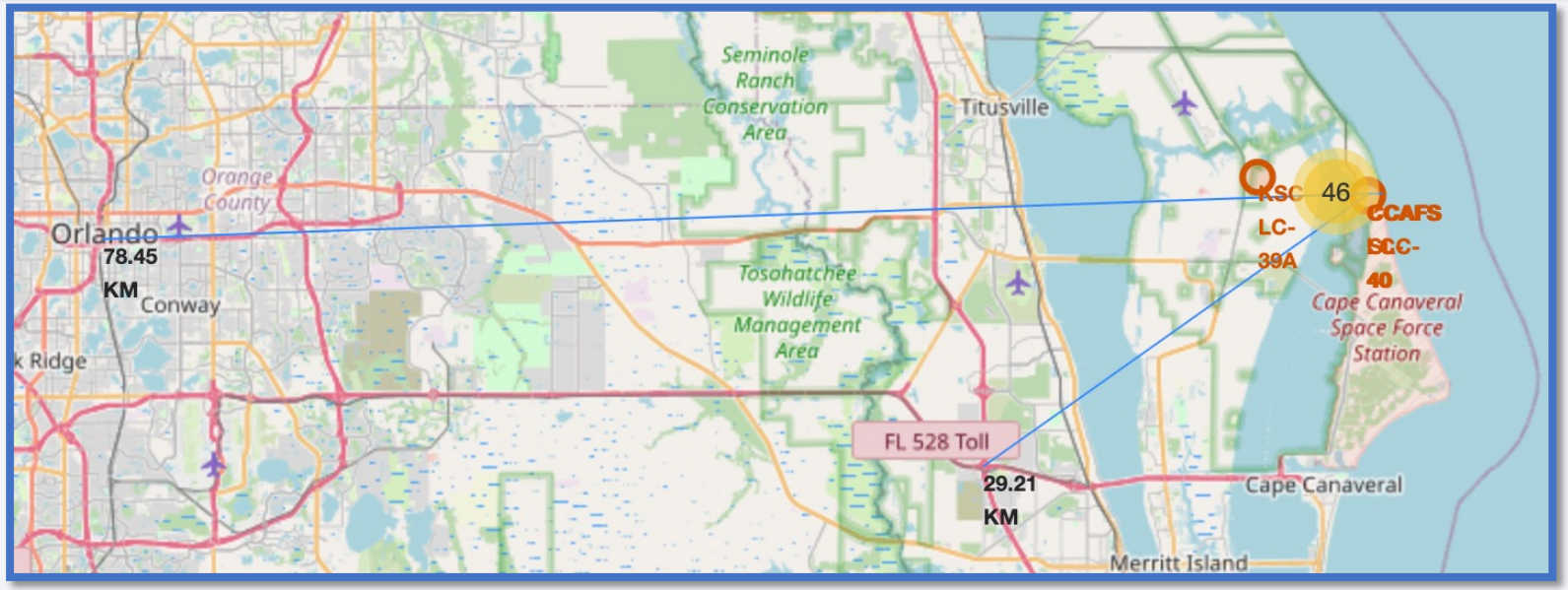


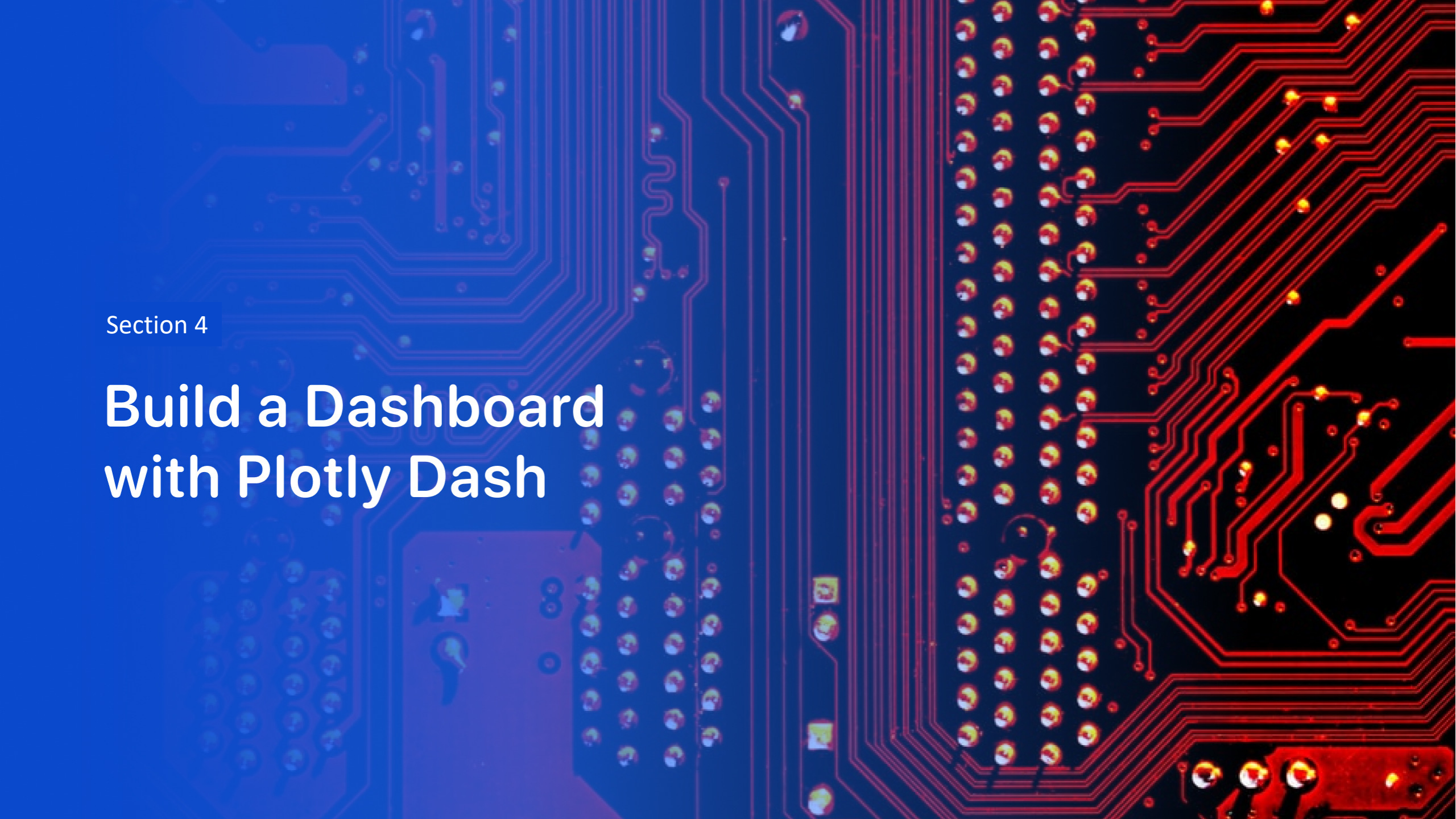
# Launch Sites with Color Labels and Markers



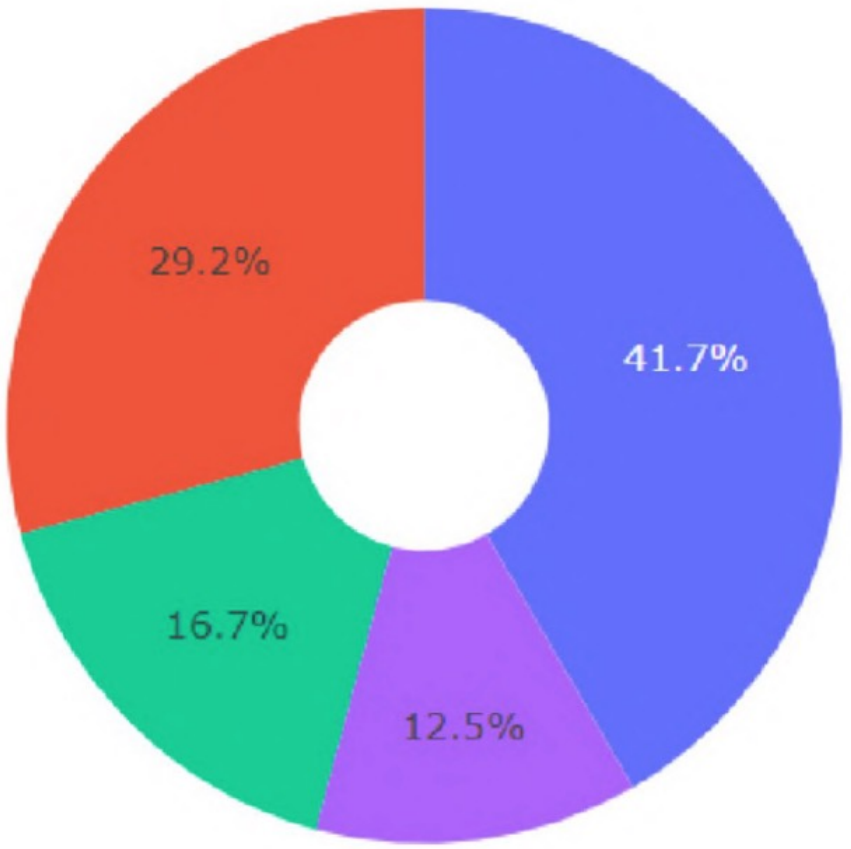
# Launch Site Distances to Key Places

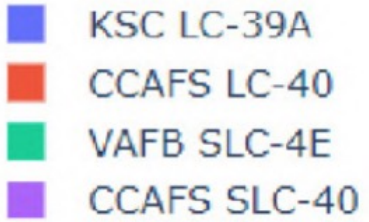


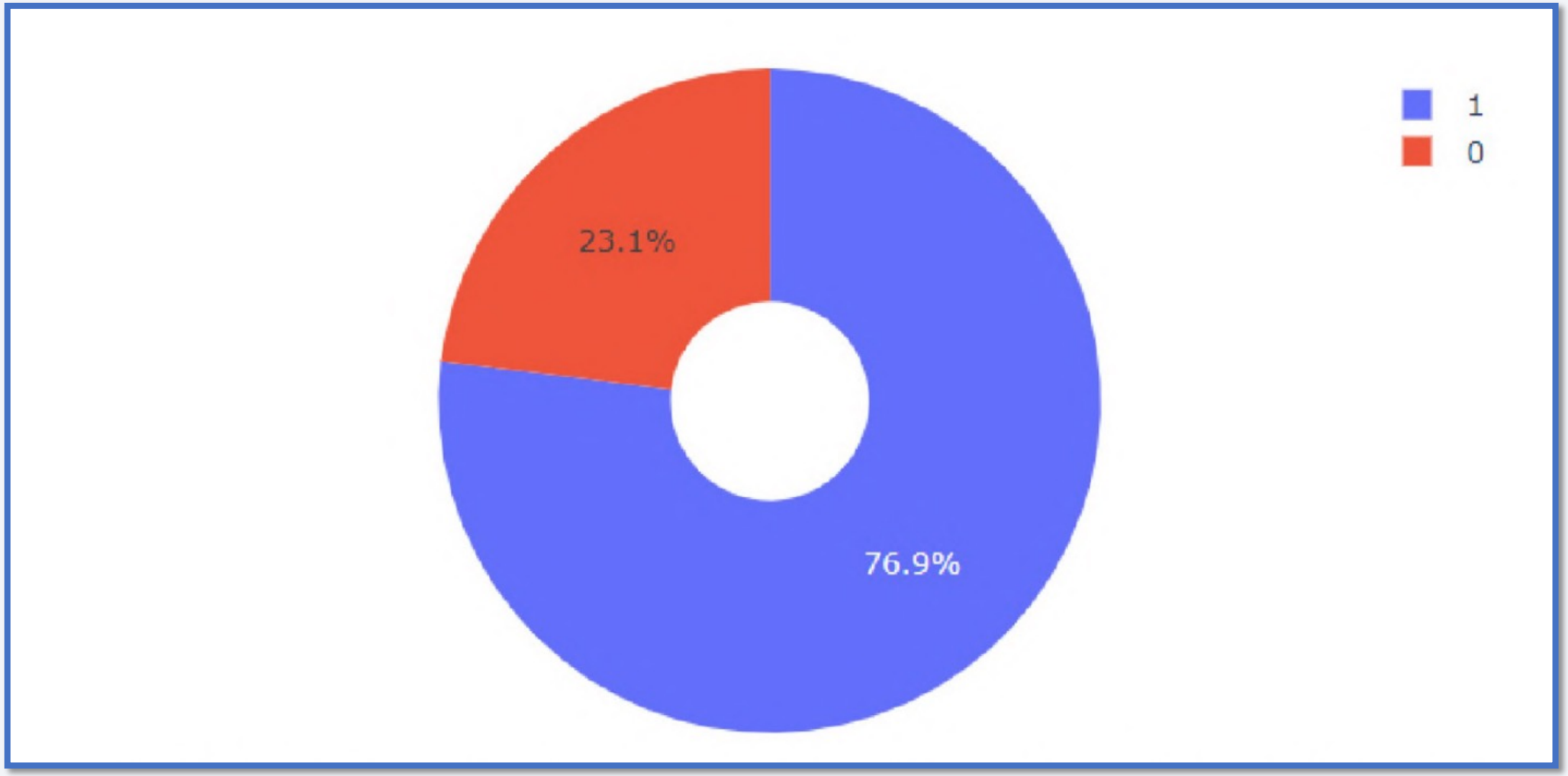

Section 4
Build a Dashboard with Plotly Dash

# Launch Site Success Percentage

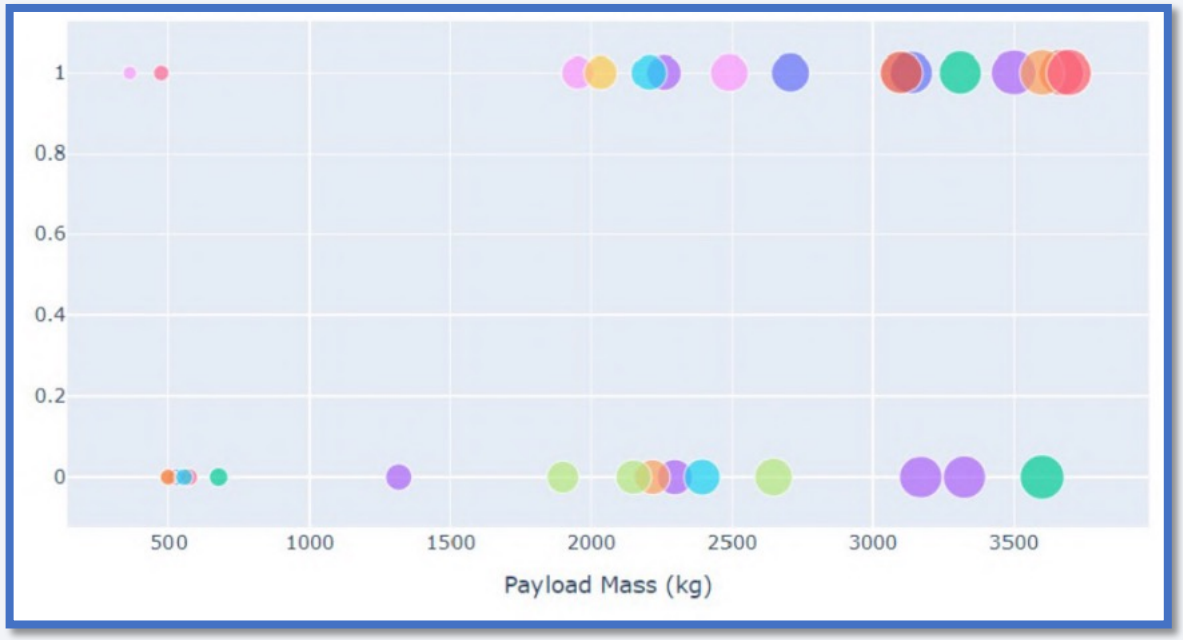


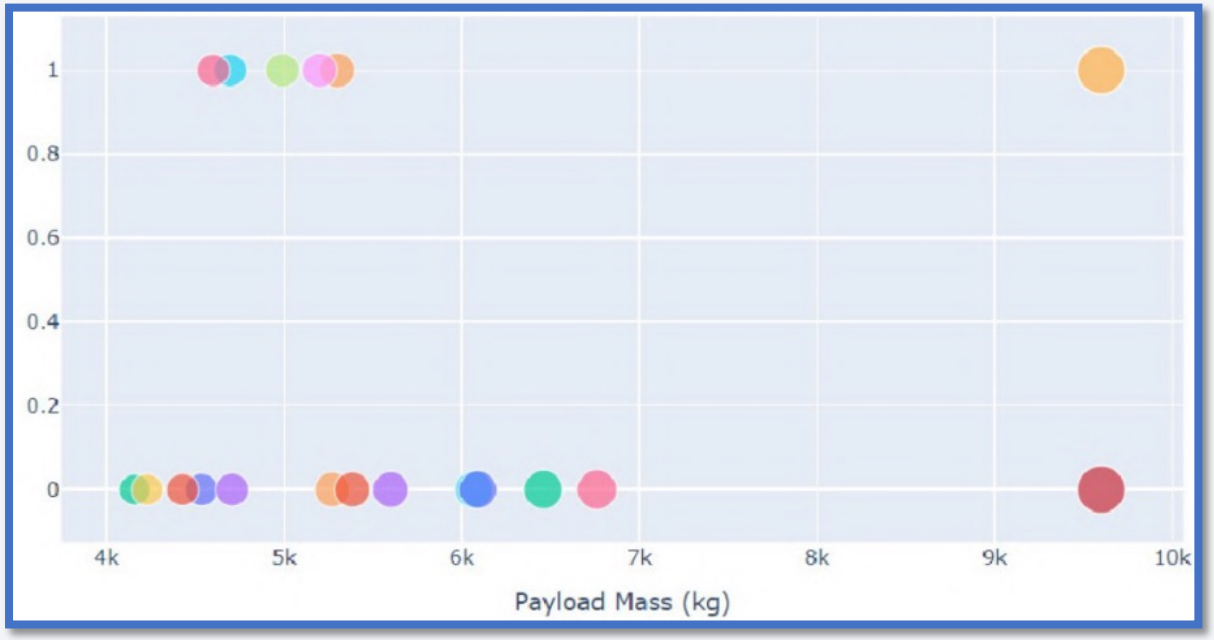



# Launch Site with Highest Launch Success Ratio



# Payload vs Launch Outcome Scatter Plots




Lighter payloads experienced more launch outcomes which were a success.

Section 5

# Predictive Analysis (Classification)

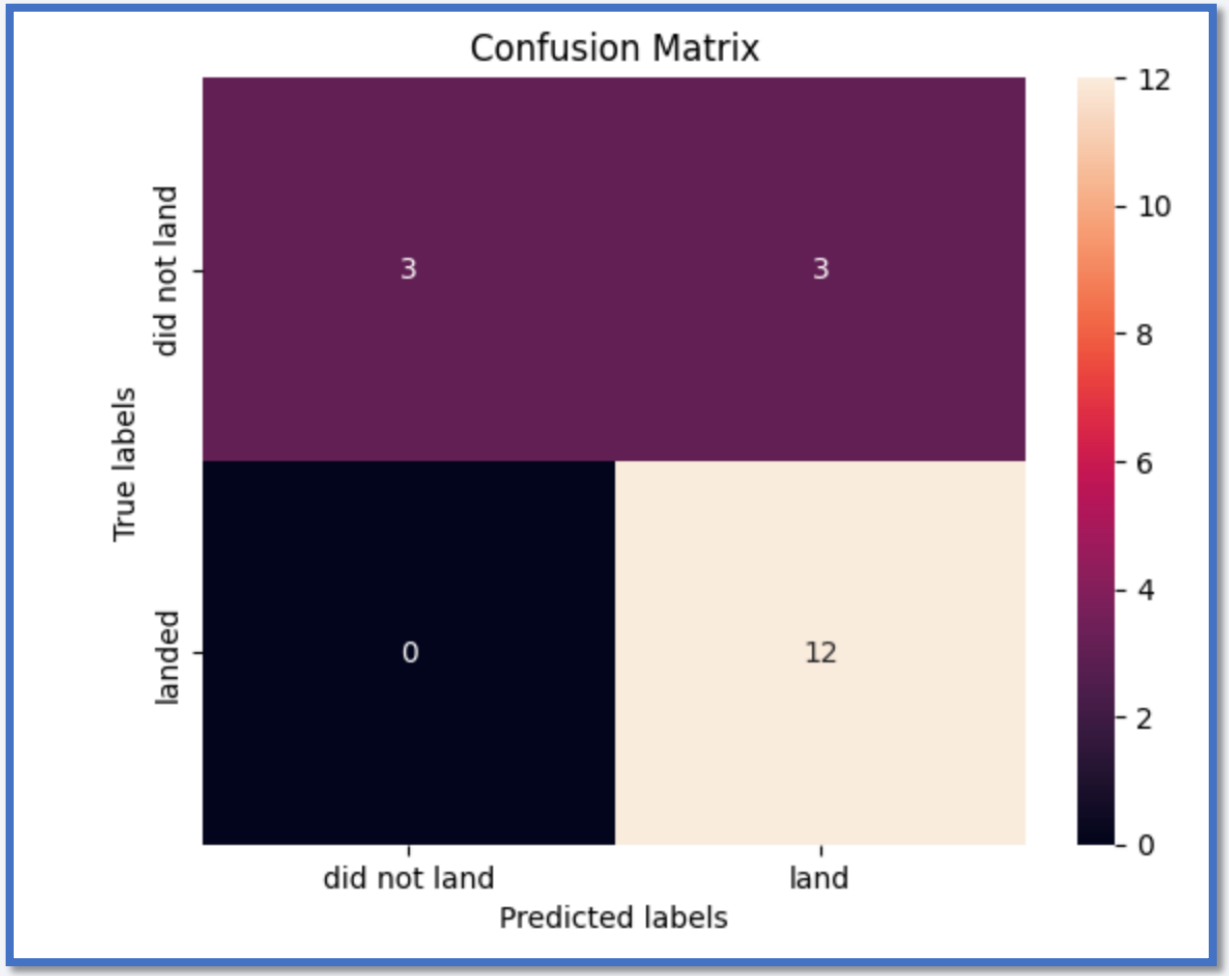
# Classification Accuracy

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.9017857142857142
Best Params is : {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_sampl
es_split': 10, 'splitter': 'random'}
```

# Confusion Matrix

The classifier had no false negatives and low number of false positives.



# Thank you!