IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Xiang Zhang
2023/6/29

# Outline

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

# Executive Summary

1.Our data collection and web scraping efforts resulted in a comprehensive dataset of SpaceX launches, including details about the launch sites, payloads, and outcomes.
2.Our EDA revealed several interesting trends and patterns, such as the relationship between payload mass and launch success, and the distribution of launches across different launch sites.
3.Our Folium visualizations provided geographical insights, such as the proximity of launch sites to the coast and their distances from cities and railways.
4.Our interactive dashboard provides a dynamic tool for users to explore the SpaceX launch data, with features allowing users to filter by launch site and payload range.
5.Our machine learning model provides a predictive tool for assessing the likelihood of future launch successes based on the features of the launch. The performance of this model was evaluated and found to be satisfactory for the given dataset.

# Introduction

- SpaceX, founded by Elon Musk, aims to make spaceflight a common reality

This presentation encapsulates the analysis and prediction of SpaceX Falcon 9 first stage landing success

Section 1

# Methodology

# Methodology

## Executive Summary

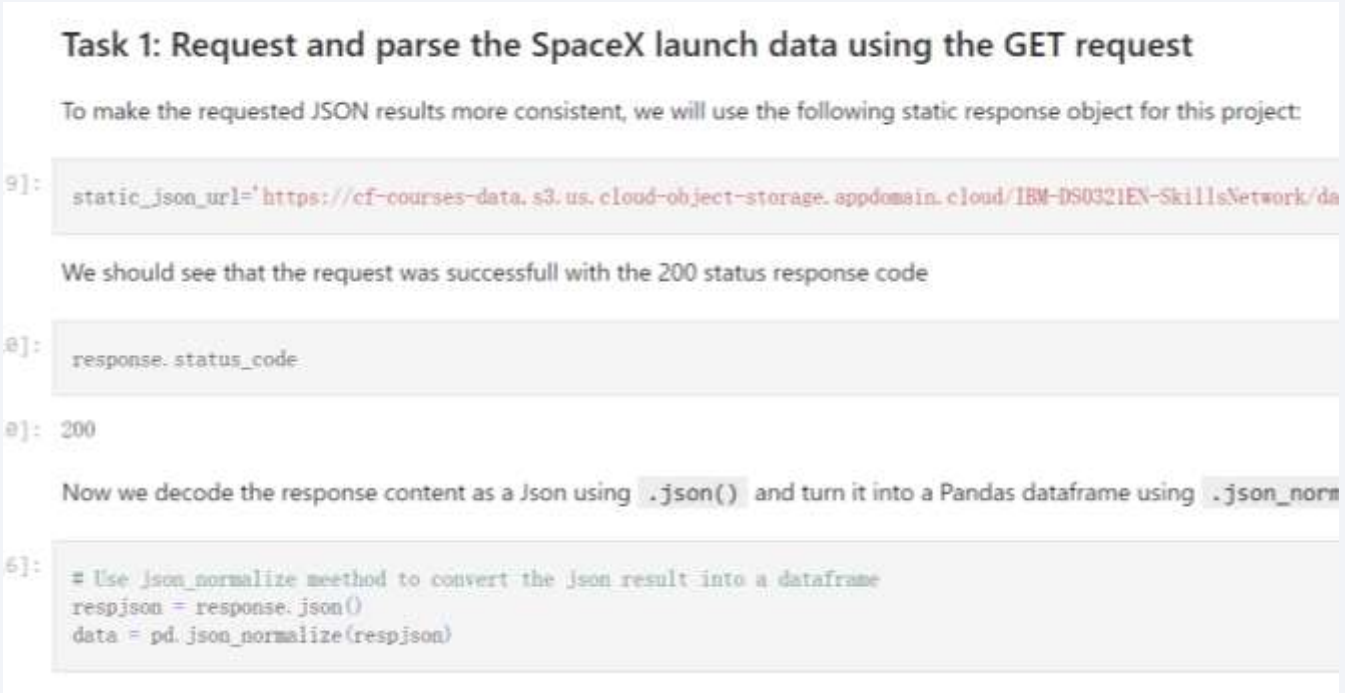
1. **Data Collection**: We utilized the SpaceX API to gather data about SpaceX launches. This process involved making requests to the API and parsing the returned JSON data to extract relevant information.
2. **Web Scraping**: We also scraped Falcon 9 and Falcon Heavy launch records from Wikipedia using Python libraries like BeautifulSoup and requests. This allowed us to gather additional data that was not available through the API.
3. **Data Wrangling**: After collecting the data, we performed data cleaning and preprocessing to ensure the data was in a suitable format for analysis. This involved handling missing values, converting data types, and structuring the data in a way that facilitated further analysis.
4. **Exploratory Data Analysis (EDA)**: We conducted EDA using SQL and Python libraries like Pandas and Matplotlib. This involved querying the data, creating visualizations, and identifying patterns and trends related to SpaceX launches.
5. **Interactive Visual Analytics**: We used the Folium library to create interactive maps that visualized the locations of SpaceX launch sites. This provided geographical context to our analysis and allowed us to explore potential relationships between launch site location and launch success.
6. **Interactive Dashboard Creation**: We built an interactive dashboard using Plotly Dash. This dashboard allows users to explore the SpaceX launch data in a user-friendly and interactive manner.
7. **Machine Learning Prediction**: We built a machine learning model to predict the success of SpaceX launches. This involved feature engineering, model training, and model evaluation steps.

# Data Collection

1. Making a GET request to the specified URL to retrieve the JSON data from the SpaceX API.
2. Checking the status code of the response to ensure it is successful (status code 200).
3. Decoding the response content as JSON using the **`.json()`** method.
4. Using the **`json_normalize()`** function to convert the JSON data into a Pandas dataframe.
5. Displaying the first 5 rows of the dataframe using the **`.head()`** method.

# Data Collection – SpaceX API

- The data was collected by making a GET request to the SpaceX API, parsing the JSON response, and converting it into a Pandas dataframe for further analysis.
- The link to the notebook : /1. Space-X Data Collection API.ipynb at main · XiangZhang-zx/Spaca-X-data-science (github.com)



**Task 1: Request and parse the SpaceX launch data using the GET request**

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
```

We should see that the request was successfull with the 200 status response code

```
8]: response.status_code
```

```
8]: 200
```

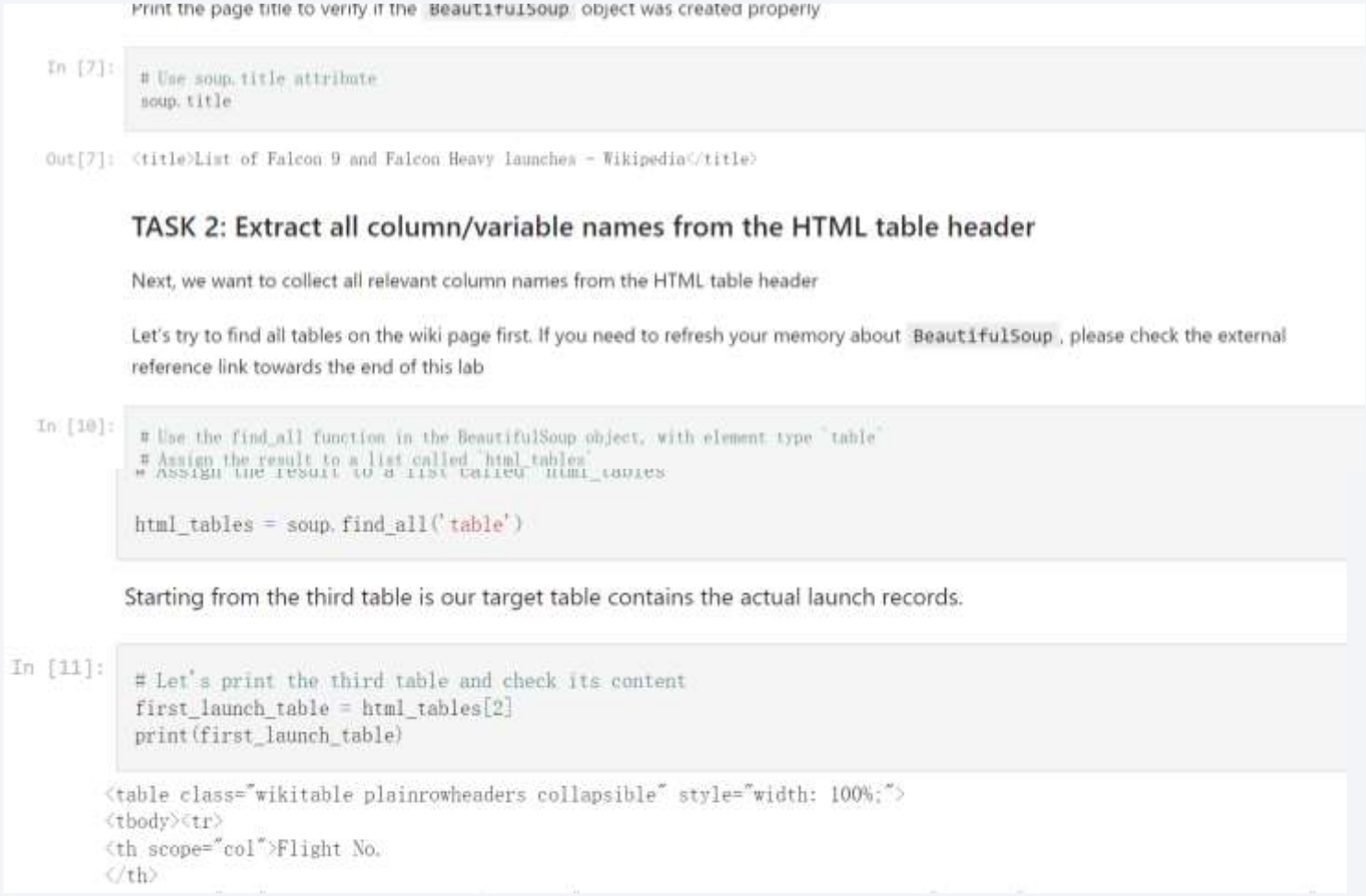
Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_norm`

```
6]: # Use json_normalize meethod to convert the json result into a dataframe
    respjson = response.json()
    data = pd.json_normalize(respjson)
```

# Data Collection - Scraping

- Data was collected from the SpaceX API and Falcon 9 launch records on Wikipedia

- The link to the notebook : /2. Space-X Web scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia.ipynb at main · XiangZhang-zx/Spaca-X-data-science · GitHub



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

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

**TASK 2: Extract all column/variable names from the HTML table header**

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

In [10]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`

html_tables = soup.find_all('table')
```

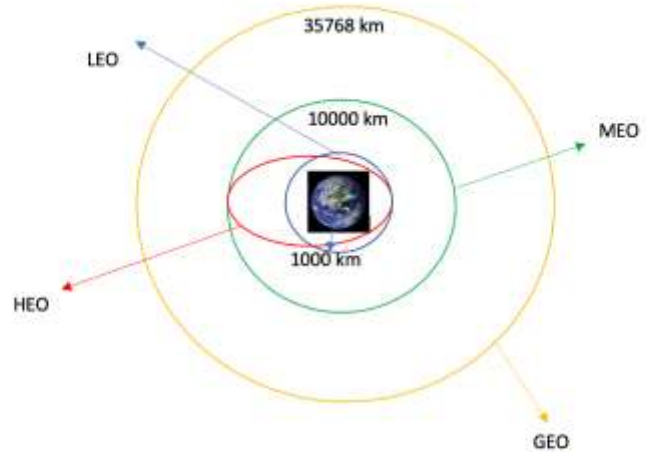
Starting from the third table is our target table contains the actual launch records.

In [11]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
```

# Data Wrangling

- we performed Exploratory Data Analysis (EDA) on the SpaceX dataset to find patterns and determine the training labels. We converted the outcomes of the booster landing into training labels, where 1 represents a successful landing and 0 represents an unsuccessful landing. The dataset was loaded and analyzed, and the number of launches on each site was calculated. We also determined the number and occurrence of each orbit type in the dataset.
- The link to the notebook :[Spaca-X-data-science/3. Space-X Data Wrangling spacex.ipynb at main · XiangZhang-zx/Spaca-X-data-science · GitHub]




**TASK 2: Calculate the number and occurrence of each orbit**

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

```
In [14]: # Apply value_counts on Orbit column
         occurence = df['Orbit'].value_counts()
         occurence
```

```
Out[14]: GTO    27
         ISS    21
         VLEO   14
```

```
In [12]: # Apply value_counts() on column LaunchSite
         launches = df['LaunchSite'].value_counts()
         launches
```

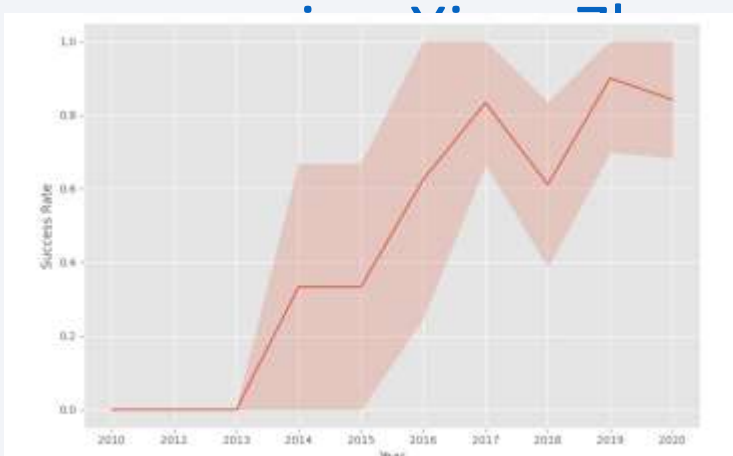
```
Out[12]: CCAFS SLC 40    55
         KSC LC 39A      22
         VAFB SLC 4E     13
         Name: LaunchSite, dtype: int64
```

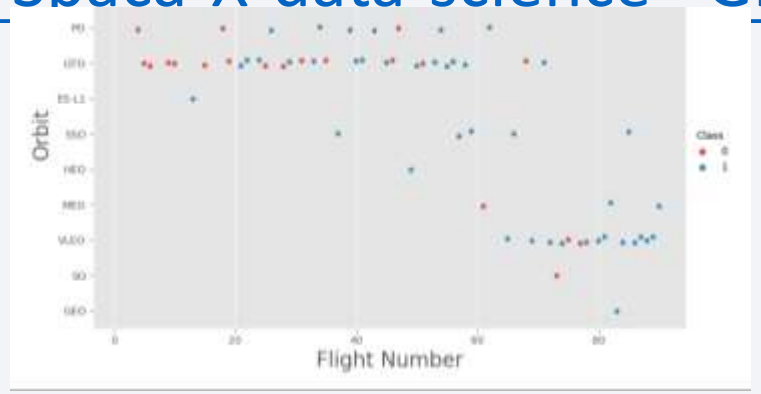
Each launch aims to an dedicated orbit, and here are some common orbit type

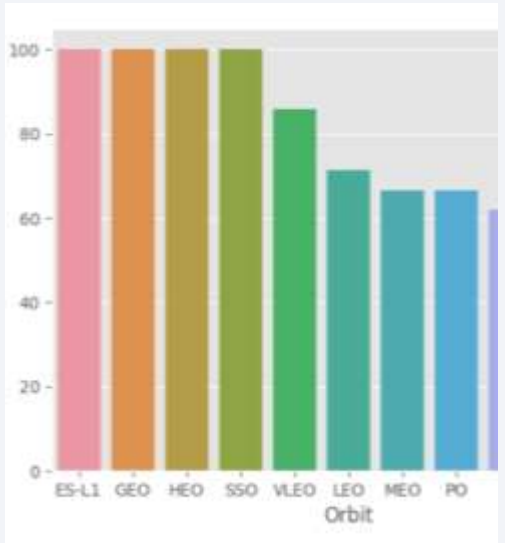
# EDA with Data Visualization

- **The scatter point charts (catplot) for FlightNumber vs LaunchSite and Payload vs LaunchSite were used to analyze the success rate of launches at different launch sites based on flight number and payload mass respectively. The bar chart showed the success rate of each orbit type, providing insights into which orbits had high or low success rates. Two more scatter point charts were plotted to visualize the relationship between flight number and orbit type, and payload mass and orbit type, revealing key patterns and correlations. Finally, a line chart was used to depict the trend of launch success over the years, showing an increasing success rate from 2013 to 2020.**

- The link to the notebook : Spaca-X-data-science/5. Space-X EDA DataViz Using Pandas and Matplotlib - SpaceX.ipynb at g-zx/Spaca-X-data-science · GitHub



# EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
- The link to the notebook : [Spaca-X-data-science/4. Space-X EDA Using SQL.ipynb at main · XiangZhang-zx/Spaca-X-data-science · GitHub]

**Task 1**

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

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

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

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

**Task 2**

Display 5 records where launch sites begin with the string '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 |
|---|---|---|---|---|---|---|---|---|---|
| 04-06-2010 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 08-12-2010 | 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) |
| 22-05-2012 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 08-10-2012 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 01-03-2013 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

**Task 3**

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

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

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

# Build an Interactive Map with Folium

**1. Content**:

1. In our project, we used the Folium library to create interactive maps for visualizing the locations of SpaceX launch sites.
2. We added markers to the map to represent each launch site. Each marker was placed at the latitude and longitude coordinates of the respective launch site. When clicked, these markers display the name of the launch site.
3. We also used lines to represent the distance between launch sites and their proximities such as the nearest coastline, city, railway, and highway. The distance was calculated using the Haversine formula, which determines the great-circle distance between two points on a sphere given their longitudes and latitudes.
4. We used different colors for the markers to represent the success (green) or failure (red) of launches.
5. We used marker clusters to group the markers for better visualization and understanding of the data.

**2. Why we added these objects**:

1. The markers provide a visual representation of where each launch site is located geographically. This can help us understand patterns in the data related to geographical location, such as whether certain locations are associated with higher or lower launch success rates.
2. The lines help us visualize the distances between the launch sites and their proximities. This can provide insights into the optimal location for building a launch site.
3. The color-coded markers help us to quickly identify the success rate of launches at each site.
4. The marker clusters simplify the map and make it easier to interpret when many markers share the same coordinates.

- The link to the notebook : Spaca-X-data-science/7. Build an Interactive Dashboard with Ploty Dash - spacex_dash_app.py at main · XiangZhang-zx/Spaca-X-data-science · GitHub

# Build a Dashboard with Plotly Dash

**1. Content**:

1. In our project, we used the Folium library to create interactive maps for visualizing the locations of SpaceX launch sites.
2. We added markers to the map to represent each launch site. Each marker was placed at the latitude and longitude coordinates of the respective launch site. When clicked, these markers display the name of the launch site.
3. We also used lines and circles to represent other geographical features relevant to our analysis. For example, we might use a line to represent the trajectory of a rocket, or a circle to represent the area within a certain distance of a launch site.
4. These map objects were not only visually informative but also interactive, providing a more engaging user experience.

**2. Why we added these objects**:

1. The markers provide a visual representation of where each launch site is located geographically. This can help us understand patterns in the data related to geographical location, such as whether certain locations are associated with higher or lower launch success rates.
2. The lines and circles can help us visualize the relationships between different geographical features and the launch sites. For example, if we were analyzing the impact of distance from a launch site on success rate, a circle representing a certain distance radius around a launch site could provide a useful visual aid.

- The link to the notebook : Spaca-X-data-science/7. Build an Interactive Dashboard with Ploty Dash - spacex_dash_app.py at main · XiangZhang-zx/Spaca-X-data-science (github.com)

# Predictive Analysis (Classification)

- The process began with data loading and preprocessing, including standardization and train-test split. A logistic regression model was then created and optimized using GridSearchCV for hyperparameter tuning. The model was trained, and the best parameters were identified based on validation data accuracy. The final model was evaluated using metrics like accuracy and F1 score, and iteratively improved through hyperparameter tuning. The best model was the one with the highest test data accuracy and F1 score.
- The link to the notebook : Spaca-X-data-science/8. SpaceX Machine Learning Prediction.ipynb at main · XiangZhang-zx/Spaca-X-data-science · GitHub

# Results

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

Section 2

# Insights drawn from EDA

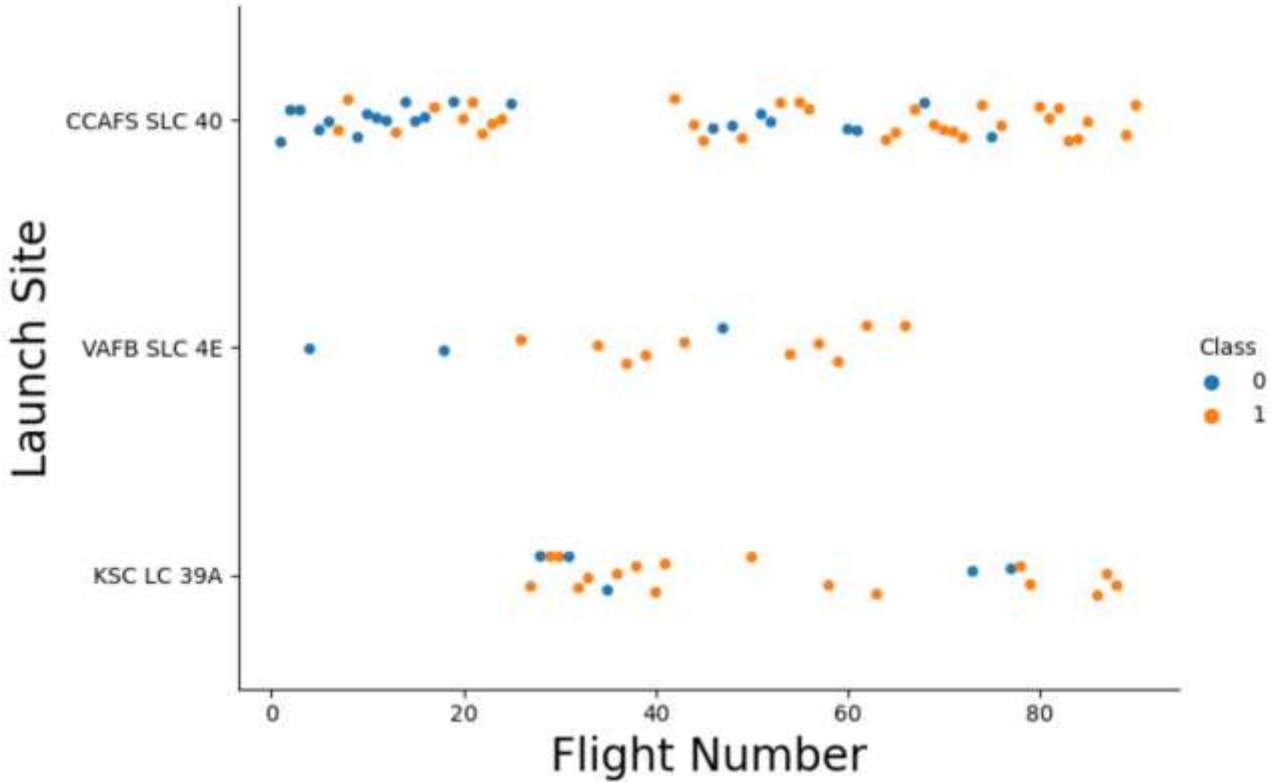
# Flight Number vs. Launch Site

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

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

In [ ]:

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 1.5 ,height=5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

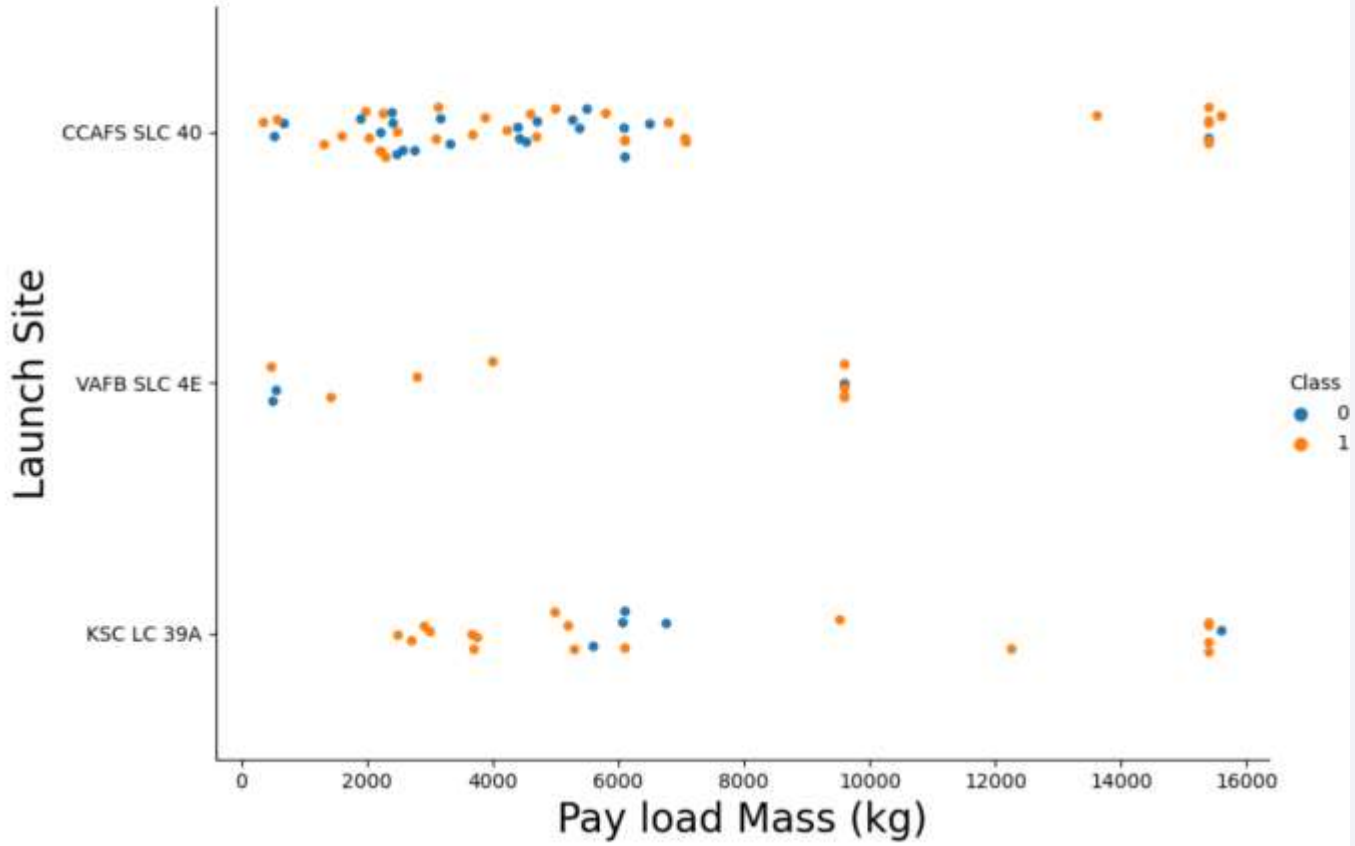


# Payload vs. Launch Site

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

We also want to observe if there is any relationship between launch sites and their payload mass.

In [21]:
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 1.5 ,height=6)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



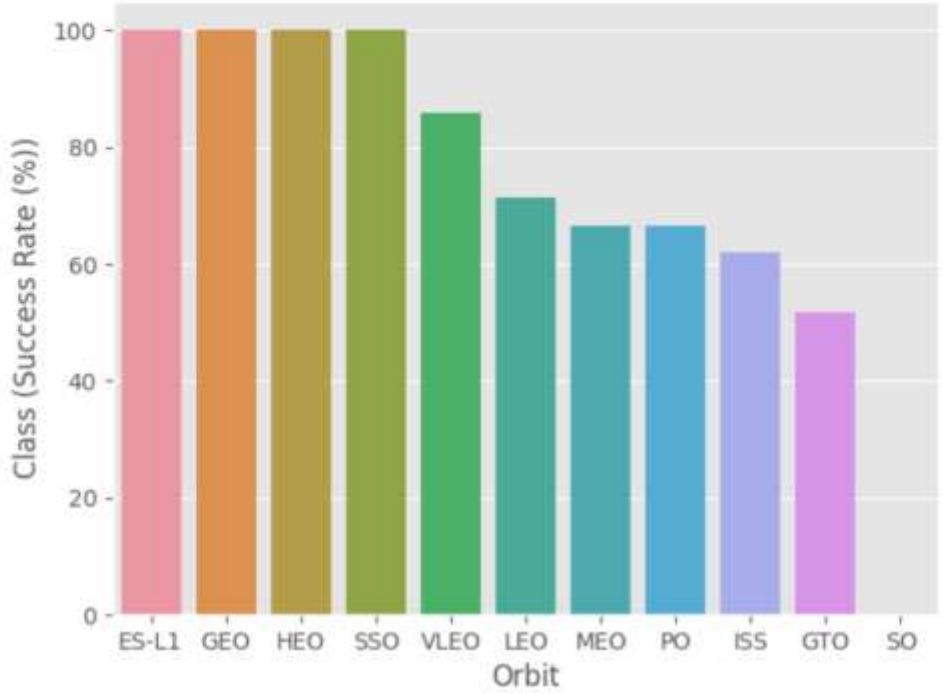
# Success Rate vs. Orbit Type

Let's create a `bar chart` for the sucess rate of each orbit

In [41]:

```
# HINT use groupby method on Orbit column and get the mean of Class column
sr_df = df.groupby('Orbit')['Class'].mean().reset_index().sort_values(by='Class', ascending=False)
sr_df['Class'] = sr_df['Class'] * 100


sns.barplot(data=sr_df, x='Orbit', y='Class')
plt.xlabel('Orbit')
plt.ylabel('Class (Success Rate (%))')
plt.show()
```

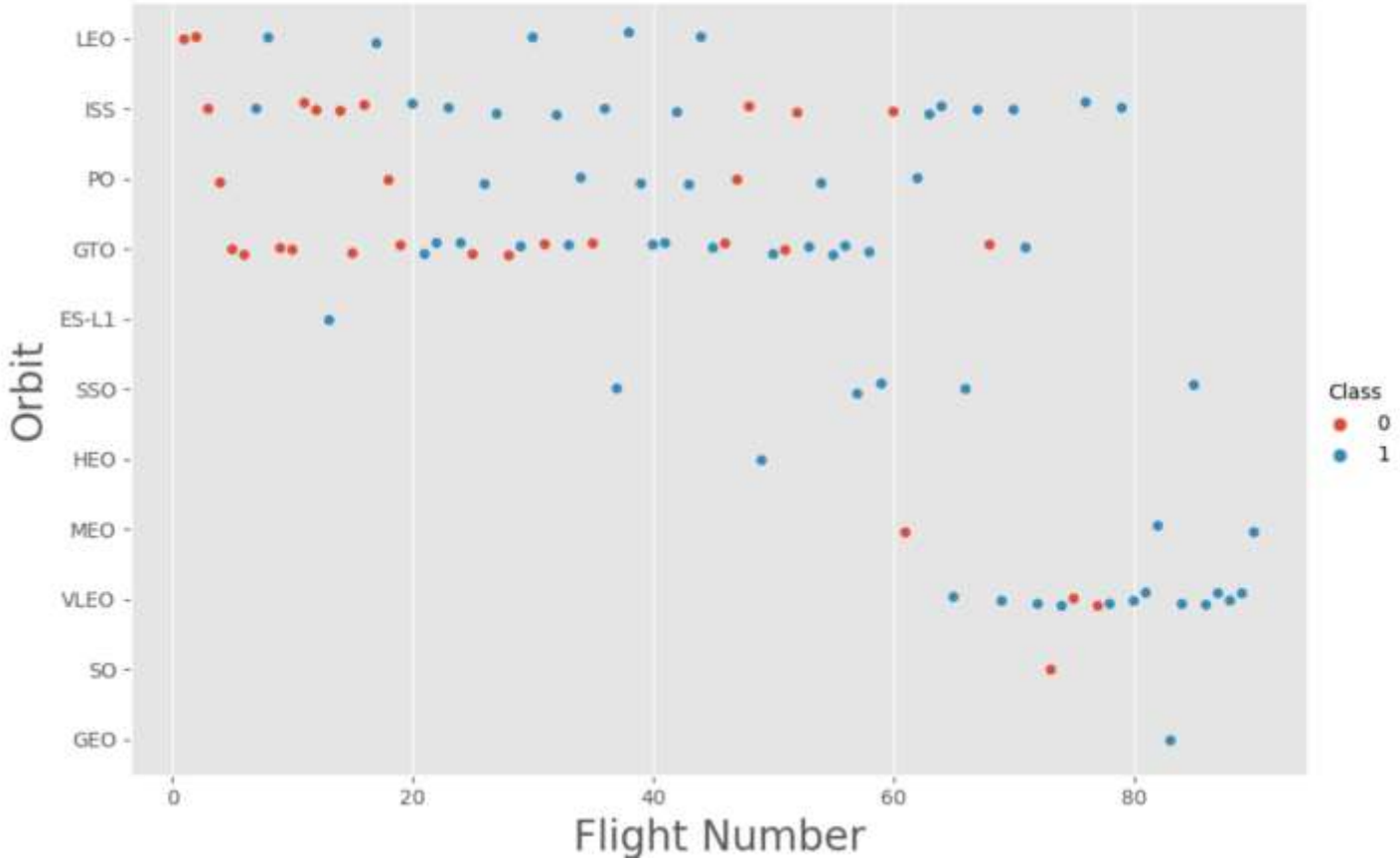


# Flight Number vs. Orbit Type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

```
6]: # Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
    sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 1.5 ,height=6)

    plt.xlabel("Flight Number",fontsize=20)
    plt.ylabel("Orbit",fontsize=20)
    plt.show()
```

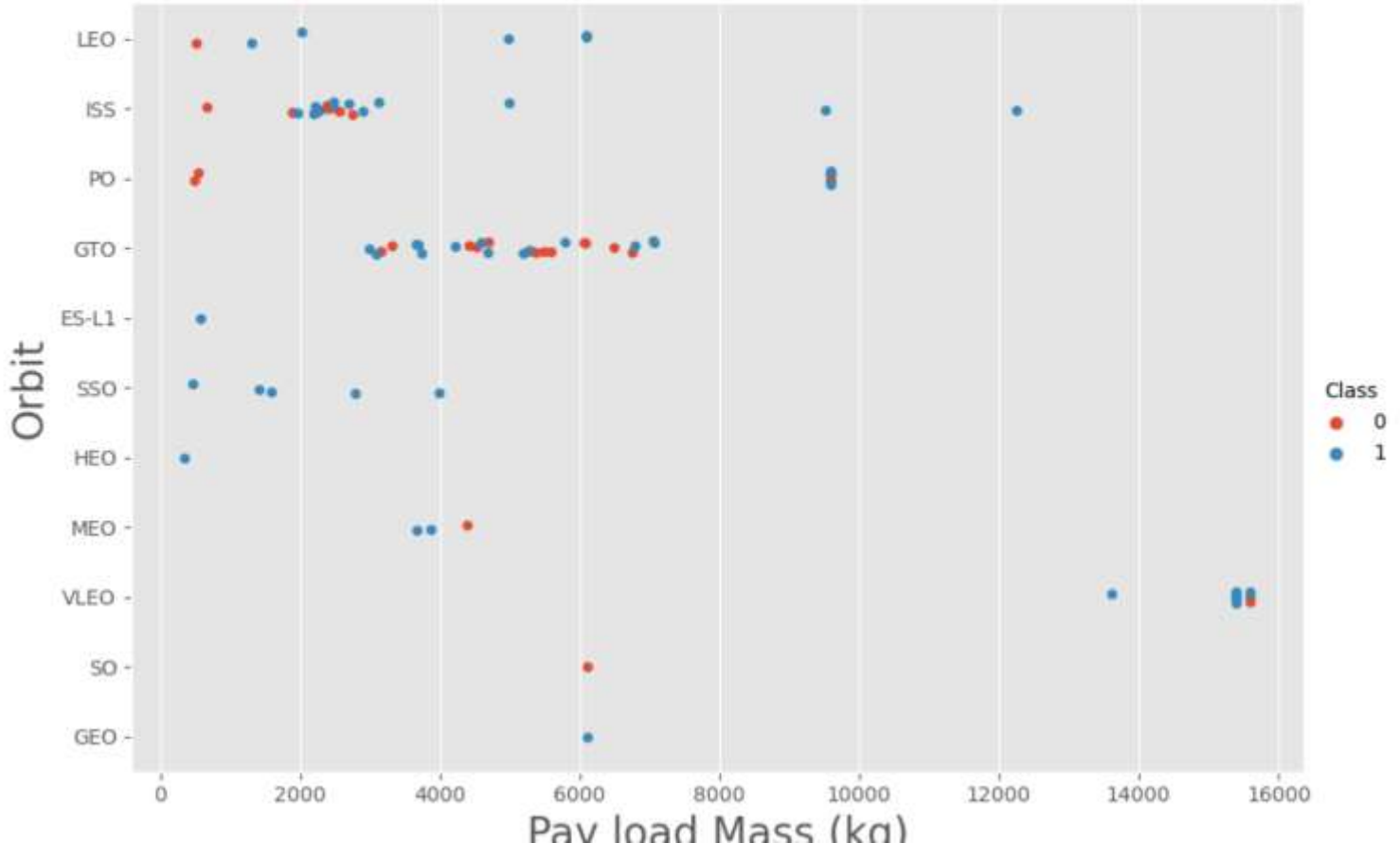


# Payload vs. Orbit Type

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

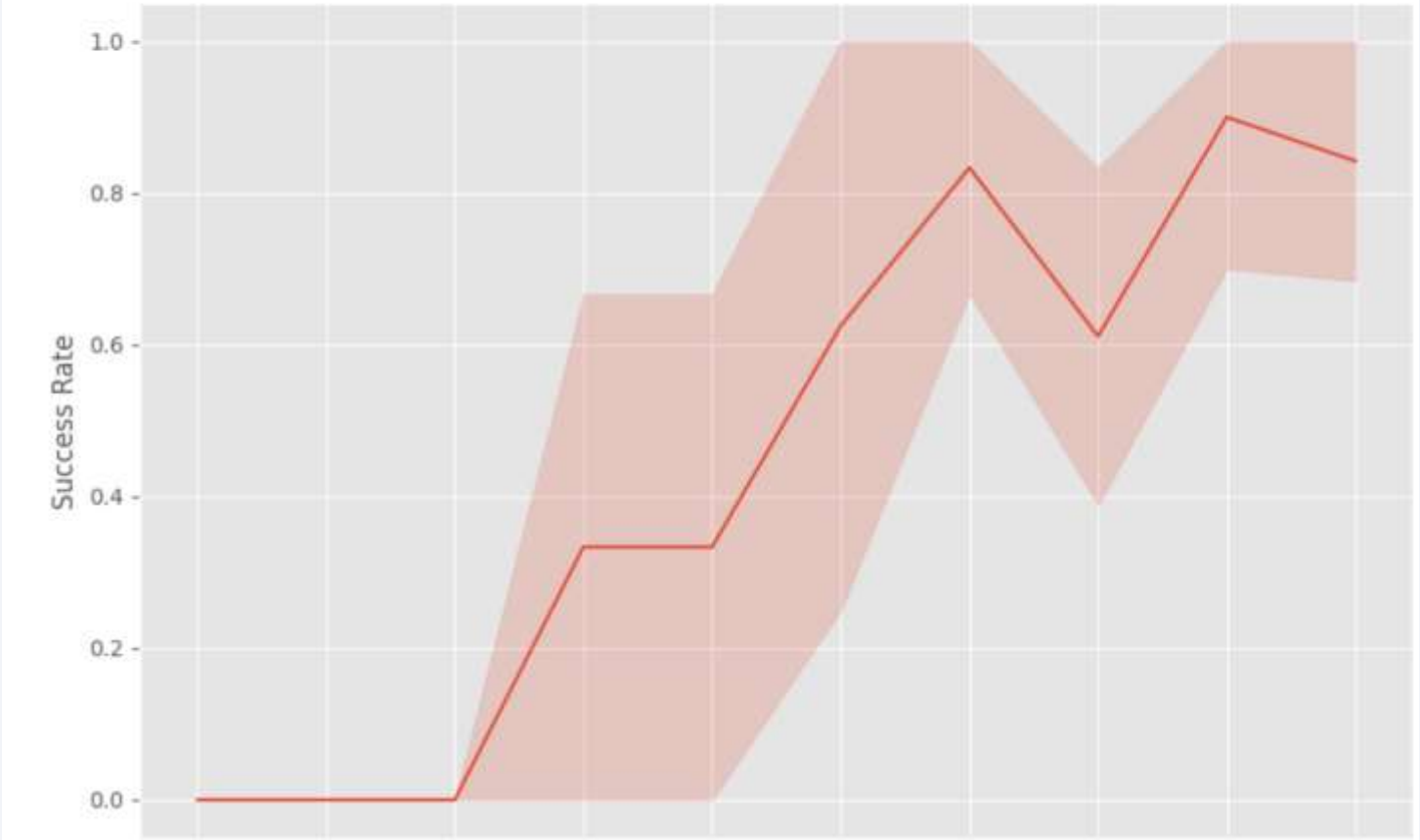
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 1.5 ,height=6)

plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



# Launch Success Yearly Trend

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
sns.lineplot(data=df, x="Date", y="Class")
plt.xlabel("Year")
plt.ylabel("Success Rate")
plt.show()
```



# All Launch Site Names

```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

| | Launch Site | Lat | Long |
|---|---|---|---|
| **0** | CCAFS LC-40 | 28.562302 | -80.577356 |
| **1** | CCAFS SLC-40 | 28.563197 | -80.576820 |
| **2** | KSC LC-39A | 28.573255 | -80.646895 |
| **3** | VAFB SLC-4E | 34.632834 | -120.610745 |

# Launch Site Names Begin with 'CCA'

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing _Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 04-06-2010 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 08-12-2010 | 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) |
| 22-05-2012 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 08-10-2012 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 01-03-2013 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

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

# Total Payload Mass

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

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

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

# Average Payload Mass by F9 v1.1

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

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

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

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

# First Successful Ground Landing Date

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

*Hint:Use min function*

In [21]:
```
%sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (ground pad)";
```

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

Out[21]:

| MIN(DATE) |
|---|
| 01-05-2017 |

# 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 [26]:
```
# %sql SELECT * FROM 'SPACEXTBL'
```

In [27]:
```
%sql SELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing _Outcome" = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4
```

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

Out[27]:

| Booster_Version | Payload |
|---|---|
| F9 FT B1022 | JCSAT-14 |
| F9 FT B1026 | JCSAT-16 |
| F9 FT B1021.2 | SES-10 |
| F9 FT B1031.2 | SES-11 / EchoStar 105 |

# Total Number of Successful and Failure Mission Outcomes

## Task 7

List the total number of successful and failure mission outcomes

In [28]:
```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

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

Out[28]:

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

# Boosters Carried Maximum Payload

## Task 8

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

In [30]:
```
%sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_")
```

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

Out[30]:

| Booster_Version | Payload | PAYLOAD_MASS__KG_ |
|---|---|---|
| F9 B5 B1048.4 | Starlink 1 v1.0, SpaceX CRS-19 | 15600 |
| F9 B5 B1049.4 | Starlink 2 v1.0, Crew Dragon in-flight abort test | 15600 |
| F9 B5 B1051.3 | Starlink 3 v1.0, Starlink 4 v1.0 | 15600 |
| F9 B5 B1056.4 | Starlink 4 v1.0, SpaceX CRS-20 | 15600 |
| F9 B5 B1048.5 | Starlink 5 v1.0, Starlink 6 v1.0 | 15600 |
| F9 B5 B1051.4 | Starlink 6 v1.0, Crew Dragon Demo-2 | 15600 |
| F9 B5 B1049.5 | Starlink 7 v1.0, Starlink 8 v1.0 | 15600 |
| F9 B5 B1060.2 | Starlink 11 v1.0, Starlink 12 v1.0 | 15600 |
| F9 B5 B1058.3 | Starlink 12 v1.0, Starlink 13 v1.0 | 15600 |
| F9 B5 B1051.6 | Starlink 13 v1.0, Starlink 14 v1.0 | 15600 |
| F9 B5 B1060.3 | Starlink 14 v1.0, GPS III-04 | 15600 |
| F9 B5 B1049.7 | Starlink 15 v1.0, SpaceX CRS-21 | 15600 |

# 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 month 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.**

In [68]:
```
%sql SELECT substr(Date,7,4), substr(Date, 4, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Outcome",
```

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

Out[68]:

| substr(Date,7,4) | substr(Date, 4, 2) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|
| 2015 | 01 | F9 v1.1 B1012 | CCAFS LC-40 | SpaceX CRS-5 | 2395 | Success | Failure (drone ship |
| 2015 | 04 | F9 v1.1 B1015 | CCAFS LC-40 | SpaceX CRS-6 | 1898 | Success | Failure (drone ship |

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

## Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

In [74]:
```
%sql SELECT * FROM SPACEXTBL WHERE "Landing _Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017') ORDER BY Date DES
```

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

Out[74]:

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing _Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 19-02-2017 | 14:39:00 | F9 FT B1031.1 | KSC LC-39A | SpaceX CRS-10 | 2490 | LEO (ISS) | NASA (CRS) | Success | Success (ground pad) |
| 18-10-2020 | 12:25:57 | F9 B5 B1051.6 | KSC LC-39A | Starlink 13 v1.0, Starlink 14 v1.0 | 15600 | LEO | SpaceX | Success | Success |
| 18-08-2020 | 14:31:00 | F9 B5 B1049.6 | CCAFS SLC-40 | Starlink 10 v1.0, SkySat-19, -20, -21, SAOCOM 1B | 15440 | LEO | SpaceX, Planet Labs, PlanetIQ | Success | Success |
| 18-07-2016 | 04:45:00 | F9 FT B1025.1 | CCAFS LC-40 | SpaceX CRS-9 | 2257 | LEO (ISS) | NASA (CRS) | Success | Success (ground pad) |
| 18-04-2018 | 22:51:00 | F9 B4 B1045.1 | CCAFS SLC-40 | Transiting Exoplanet Survey Satellite (TESS) | 362 | HEO | NASA (LSP) | Success | Success (drone ship) |
| 17-12-2019 | 00:10:00 | F9 B5 B1056.3 | CCAFS SLC-40 | JCSat-18 / Kacific 1, Starlink 2 v1.0 | 6956 | GTO | Sky Perfect JSAT, Kacific 1 | Success | Success |
| 16-11-2020 | 00:27:00 | F9 B5B1061.1 | KSC LC-39A | Crew-1, Sentinel-6 Michael | 12500 | LEO (ISS) | NASA (CCP) | Success | Success |

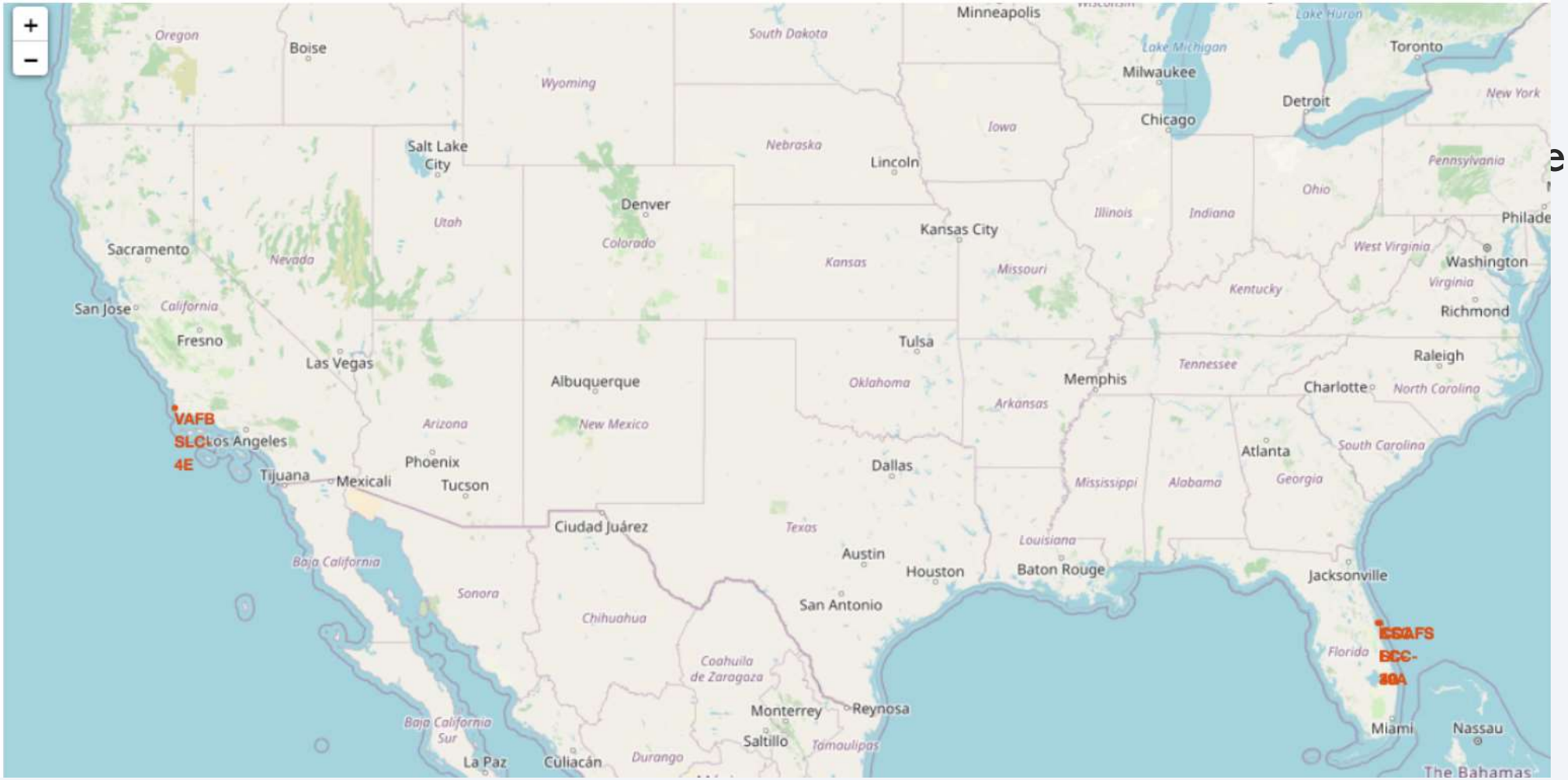
ound
g order

Section 3

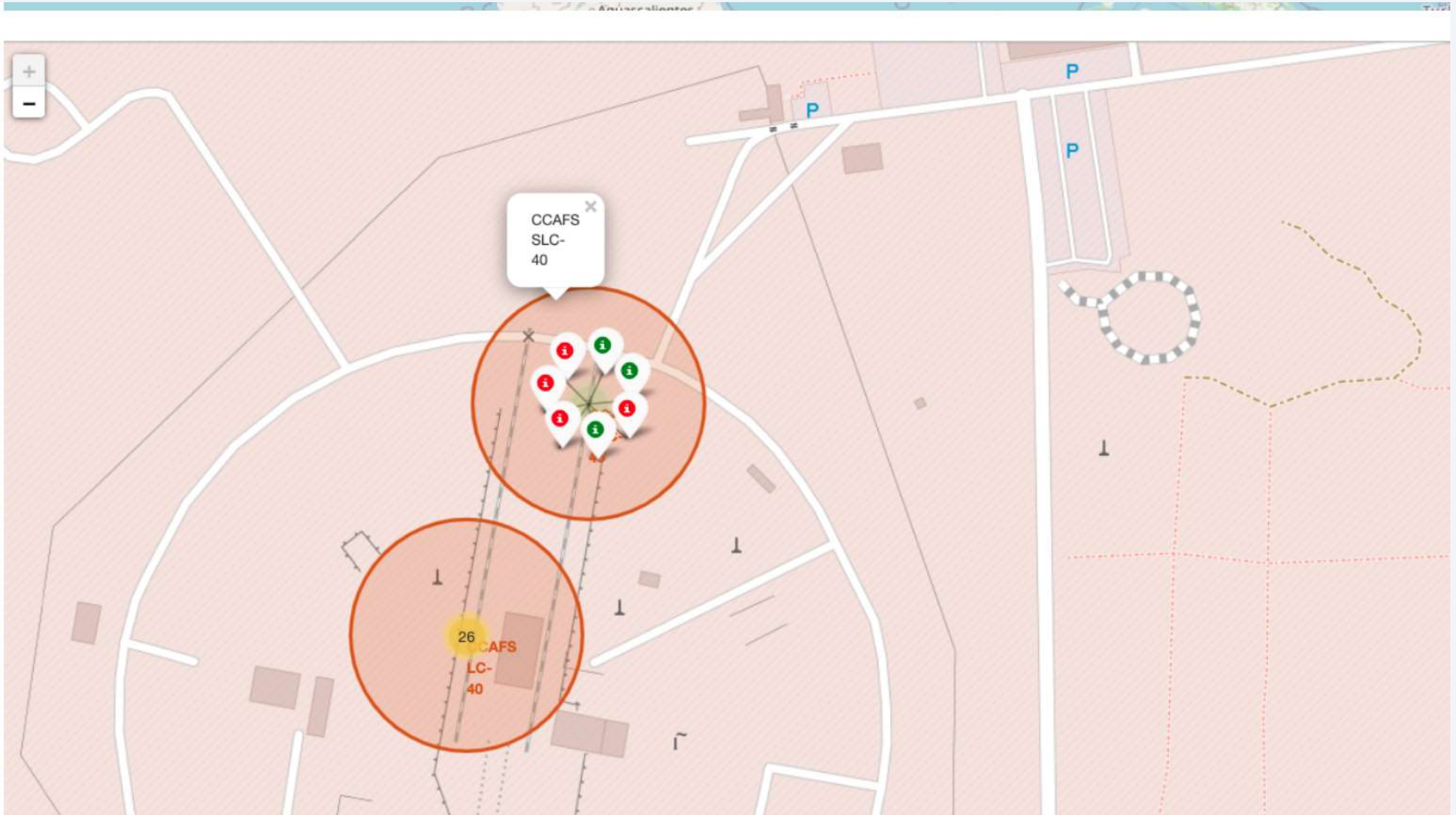
# Launch Sites Proximities Analysis

# generated map

The generated map with marked launch sites should look similar to the following:

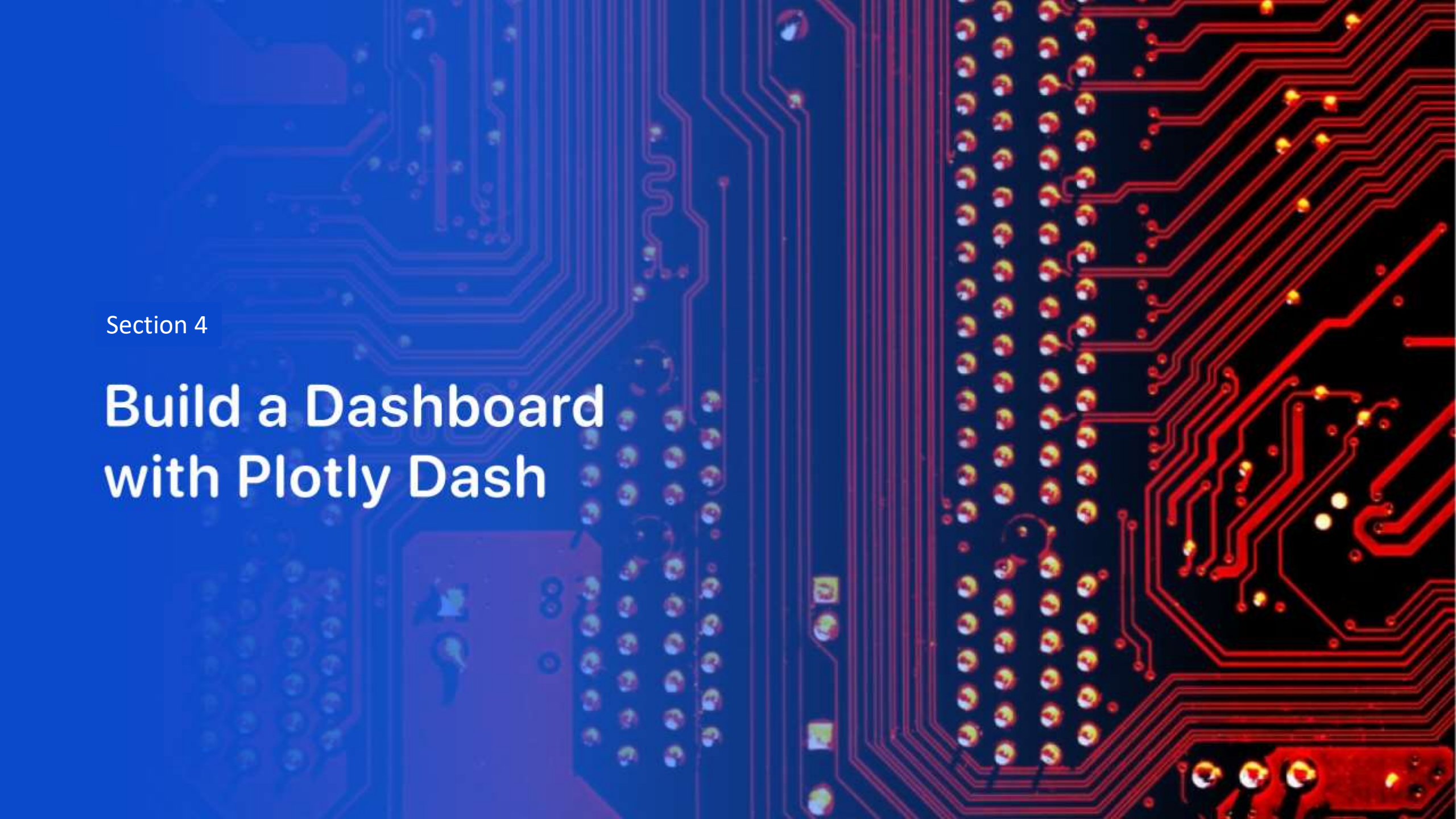


# color-labeled marker

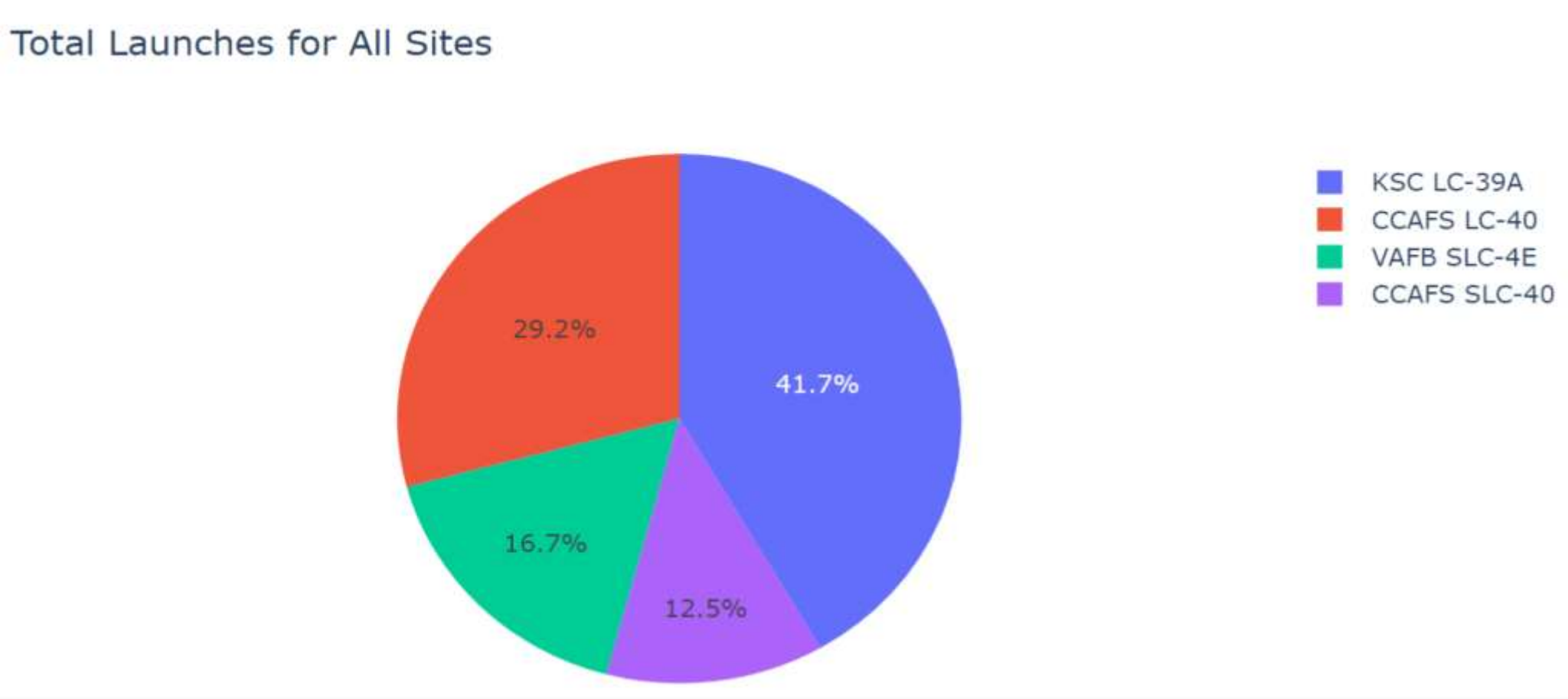


# updated map with distance line

Your updated map with distance line should look like the following screenshot:

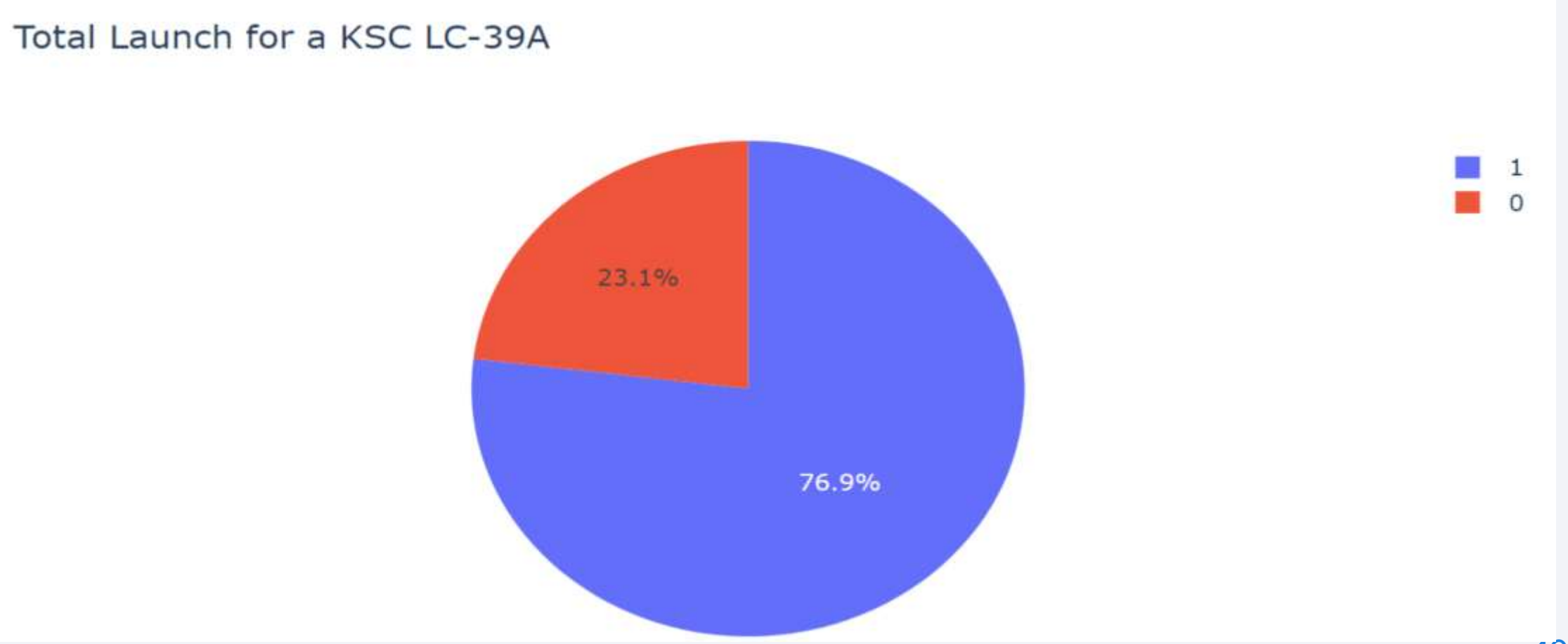

Section 4
Build a Dashboard with Plotly Dash

# Pie chart showing the success percentage



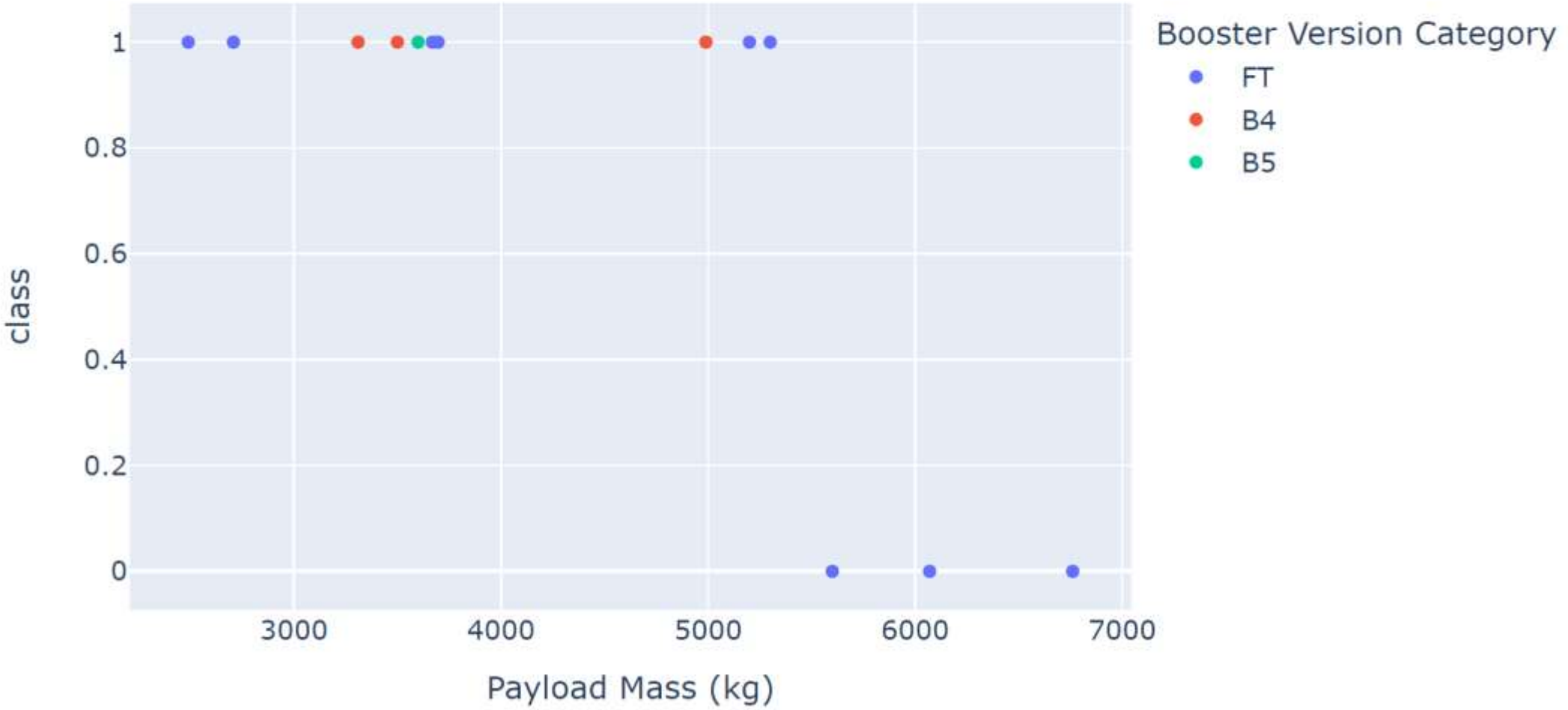

KSC LC-39A had the most successful launches

# Pie chart showing the Launch site with the launch success




archieved a 76.9% success rate with 23.1% failure rate

# Scatter plot of Payload vs Launch Outcome




the success rates for 3000-4000 kg payloads is higher

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

| | 0 |
|---|---|
| Method | Test Data Accuracy |
| Logistic_Reg | 0.833333 |
| SVM | 0.833333 |
| Decision Tree | 0.833333 |
| KNN | 0.833333 |

*All the methods perform equally on the test data: i.e. They all have the same accuracy of 0.833333 on the test Data*

# Confusion Matrix

The confusion matrix for each of the models (Logistic Regression, SVM, Decision Tree, KNN) in the project is a 2x2 matrix due to binary classification (landed or not landed). The four outcomes are:

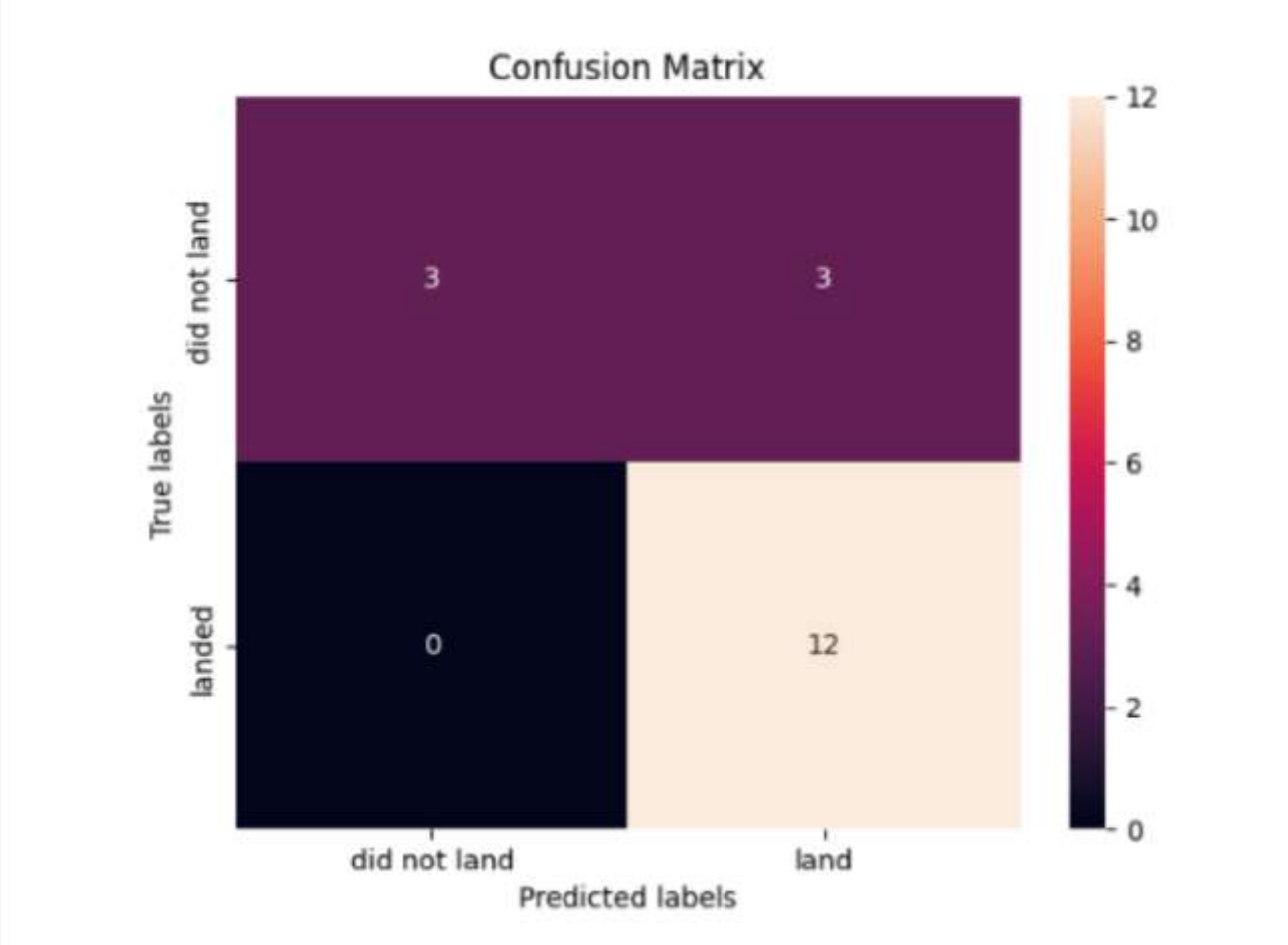
•True Positives (TP): The model correctly predicted the rocket would land.

•True Negatives (TN): The model correctly predicted the rocket would not land.

•False Positives (FP): The model incorrectly predicted the rocket would land.

•False Negatives (FN): The model incorrectly predicted the rocket would not land.

The accuracy of each model can be calculated using the formula (TP+TN)/(TP+TN+FP+FN). The confusion matrix also allows calculation of other metrics like precision, recall, and F1-score.



# Conclusions

- The project successfully analyzed and predicted SpaceX Falcon 9 first stage landing success
- the success rates for 3000-4000 kg payloads is higher
- KSC archieved a 76.9% success rate with 23.1% failure rate
- KSC LC-39A had the most successful launches

# Thank you!