# Winning Space Race with Data Science

Subagya Perera
2023-07-20

# Outline

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

# Executive Summary

- Summary of methodologies
  - Data Collection API and Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis
- Summary of all results
  - Exploratory Data Analysis and Dashboard
  - Predictive Analysis

# Introduction

- Project background and context

  Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. The goal is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers
  - Price of each launch ?
  - Factors for a successful landing & success rate ?
  - Geographical patterns about launch sites ?

Section 1

# Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - SpaceX API & Web scraping from Wikipedia.
- Perform data wrangling
  - Perform exploratory Data Analysis & determine Training Labels
  - One-hot encoding was applied to 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
  - How to build, tune, evaluate classification models

# Data Collection

- Data collection using get request to the SpaceX API
- Decode using .json() and turn it into a Pandas dataframe using .json_normalize()
- Cleaned the data (Check & fill in missing values )
- Web scraping from Wikipedia for Falcon 9 launch records (BeautifulSoup)
- Extract launch records (HTML table), parse the table & convert it to a pandas data frame

# Data Collection – SpaceX API

- *Get* request to collect data
- Cleaning & fill in missing values
- GitHub URL for data collection [can find here]

Get request for rocket launch data

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

```
[ ]: response = requests.get(spacex_url)
```

Convert .json file to pandas data frame

```
[ ]: # Use json_normalize meethod to convert the json result into a dataframe
     # Send GET request
     #response = requests.get(static_json_url)
     # Decode the response content as JSON
     data_json = response.json()

     # Convert JSON data into a Pandas DataFrame
     data = pd.json_normalize(data_json)
```

Data cleaning and dealing with the missing value

```
[ ]: # Calculate the mean value of PayloadMass column
     # Calculate the mean of the 'PayloadMass' column
     mean_payload_mass = data_falcon9['PayloadMass'].mean()

     # Replace the np.nan values with its mean value
     # Replace np.nan values with the calculated mean
     data_falcon9['PayloadMass'].replace(np.nan, mean_payload_mass, inplace=True)
```

# Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts
- Web scrapping to Falcon 9 launch records with BeautifulSoup
- convert to a pandas dataframe
- Notebook is available [here]

Use requests.get() method with the provided static_url

```
[1]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

```
[ ]: # use requests.get() method with the provided static_url
     # assign the response to a object
     response = requests.get(static_url)
     html_data.status_code
```

Create a BeautifulSoup object from the HTML response

```
[ ]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
     soup = BeautifulSoup(html_content, "html.parser")
```

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

```
[ ]: # Use soup.title attribute
     # Print the page title
     print(soup.title.string)
```

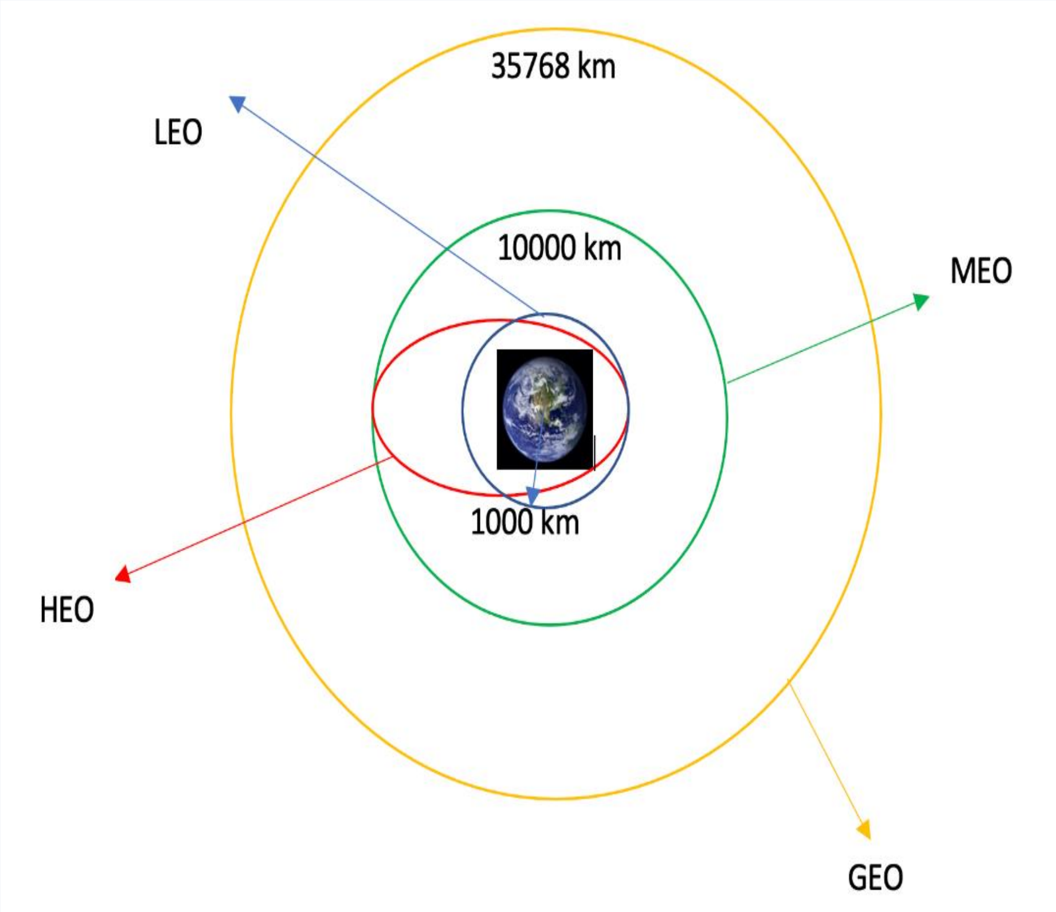
Extract column name one by one

```
[ ]: 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
     # Apply find_all() function with `th` element on the target_table

     # Apply find_all() function with 'th' element on the target_table
     th_elements = target_table.find_all('th')

     # Iterate through each th element and apply the extract_column_from_header() function
     for th in th_elements:
         name = th.get_text(strip=True)  # Get the text content of the th element

         # Append the non-empty column name to the column_names list
         if name is not None and len(name) > 0:
             column_names.append(name)
```

Create a data frame by parsing the launch HTML tables

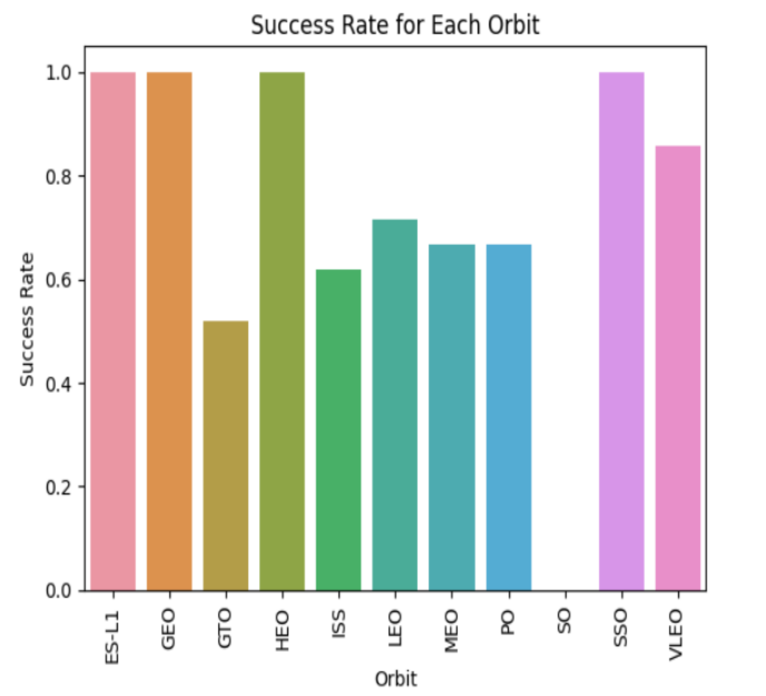
Export data to a CSV

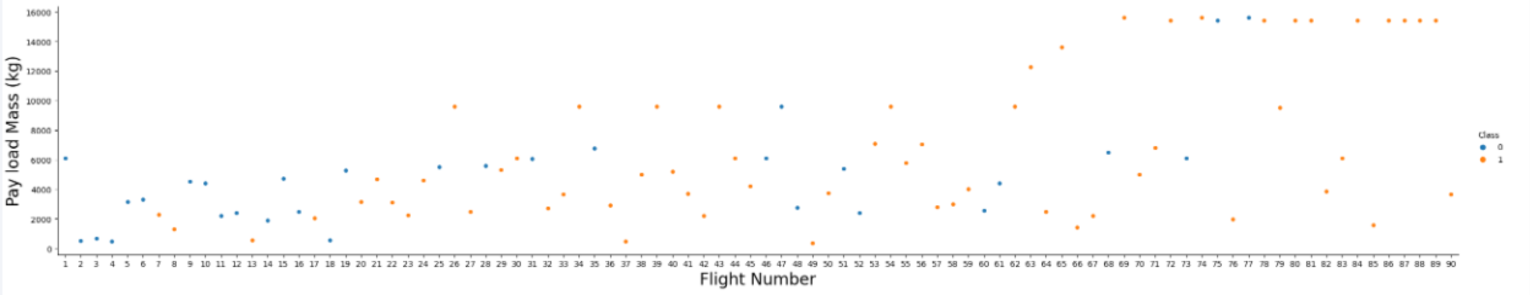
# Data Wrangling

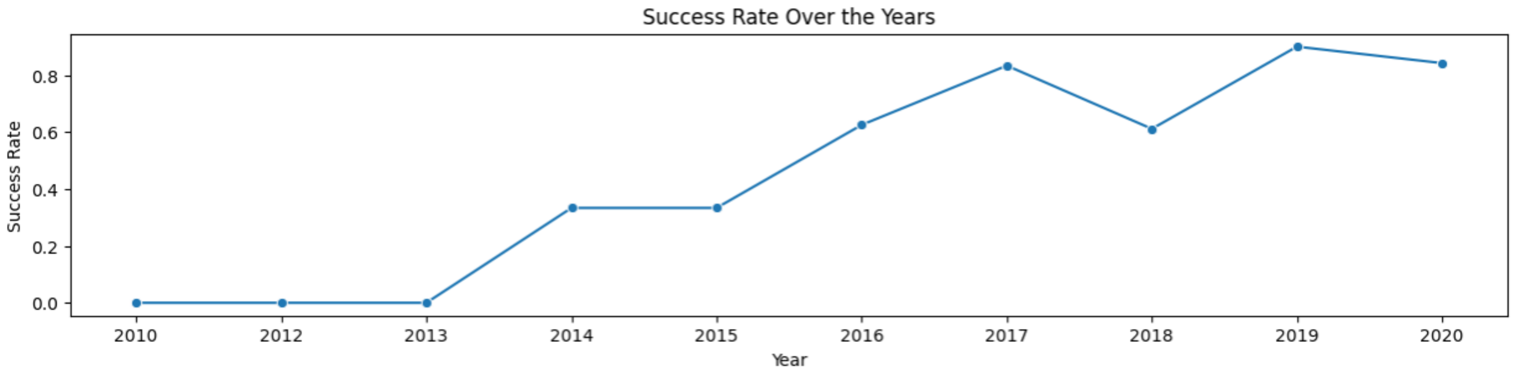



- Describe how data were processed
- Exploratory data analysis to determine the training labels
- Calculated
  - The number of launches at each site
  - The number and occurrence of each orbits
  - The number and occurrence of mission outcome per orbit type
- Create a landing outcome label from Outcome column
- Notebook is available [here]

# EDA with Data Visualization




- Explored the data by visualizing the relationship between various factors.
  - flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend
- Notebook is available [here]

# EDA with SQL

- Applied EDA with SQL to get insight from the data
  - Unique launch sites in the space mission
  - 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
  - Booster_versions which have carried the maximum payload mass
  - Rank the count of landing outcomes or Success
- Notebook is available [here]

# Build an Interactive Map with Folium

- Added map objects such as markers (each launch result with colors based on the class value), circles (highlighted circle area with a text label on a specific coordinate), lines (distance between launch site and a point) to mark the success or failure of launches for each site on the folium map.
- Launch outcomes (failure or success) to class 0 (red) and 1 (green).i.e., 0 for failure, and 1 for success
- Identified which launch sites have relatively high success rate (color-labeled marker clusters)
- Calculated the distances between a launch site to its proximities and answered some questions
- Notebook is available [here]

# Build a Dashboard with Plotly Dash

- Built an interactive dashboard with Plotly dash
- Plotted pie charts showing the total launches by a certain sites
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- Notebook is available here

# Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- Built different machine learning models and tune different hyperparameters using GridSearchCV
- Used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning
- Found the best performing classification model
- Notebook is available here

# Results

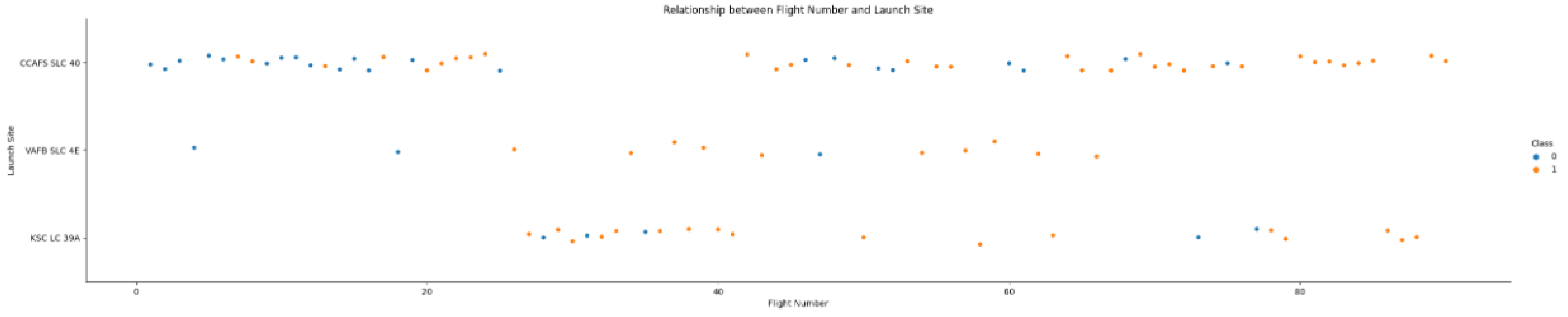
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

# Insights drawn from EDA

# Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.

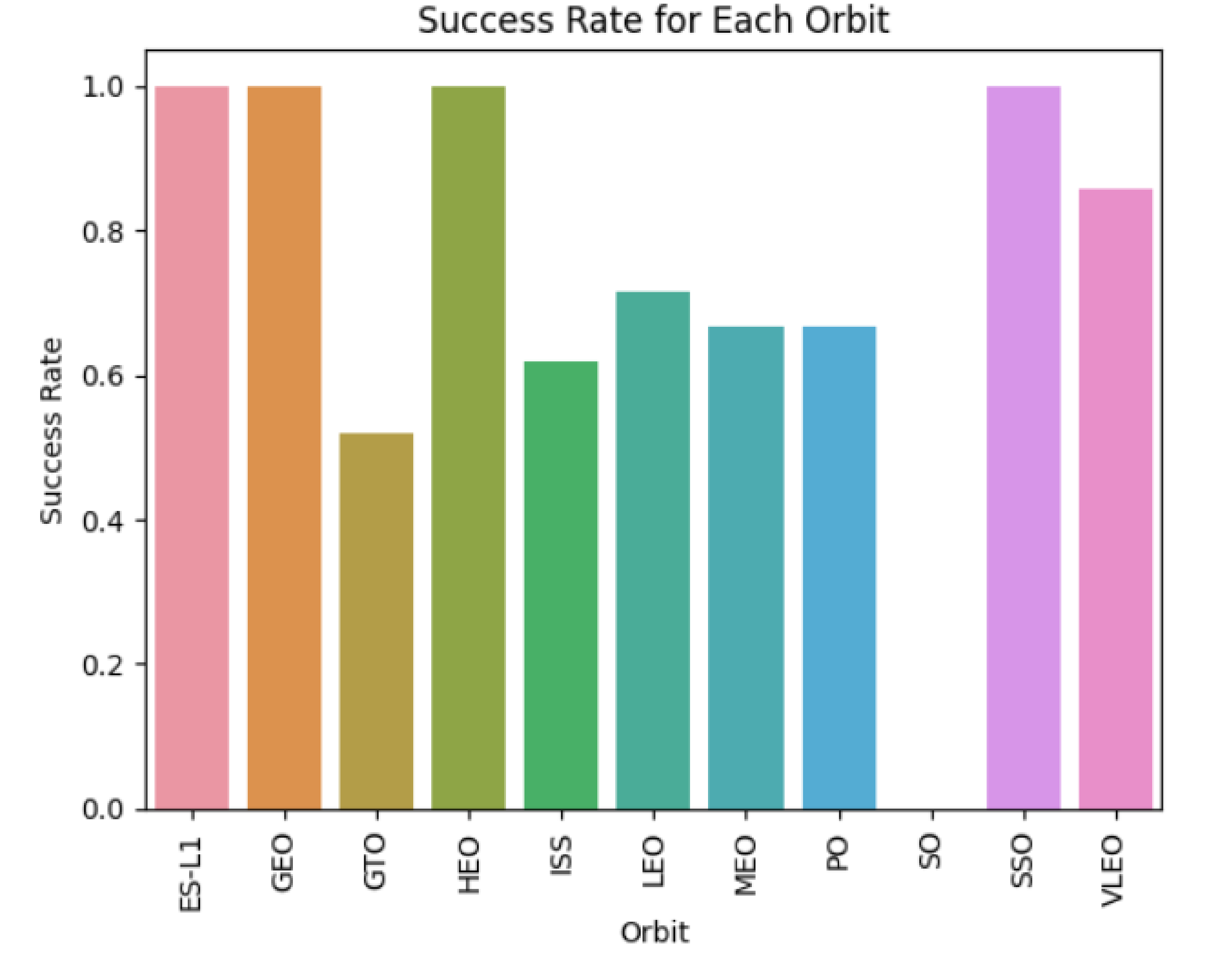


# Payload vs. Launch Site

- The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate  for the rocket

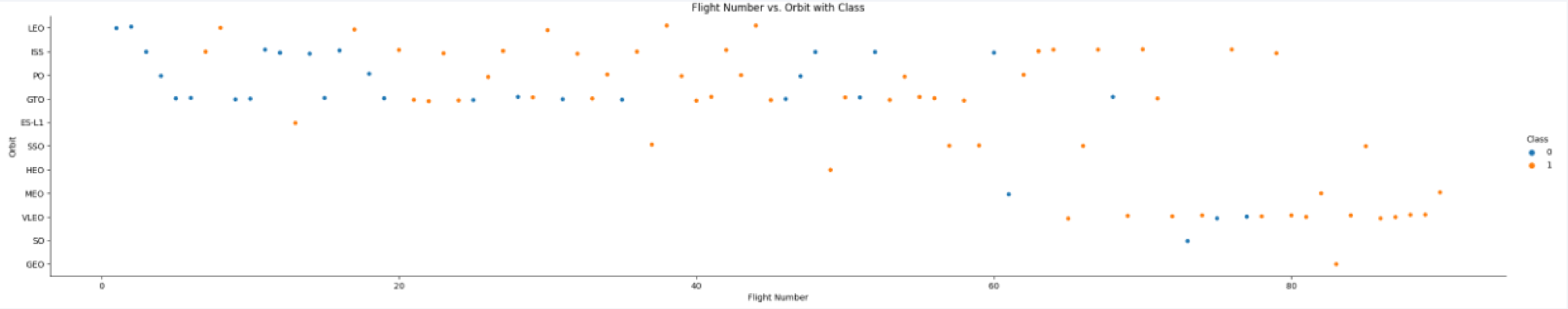
# Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



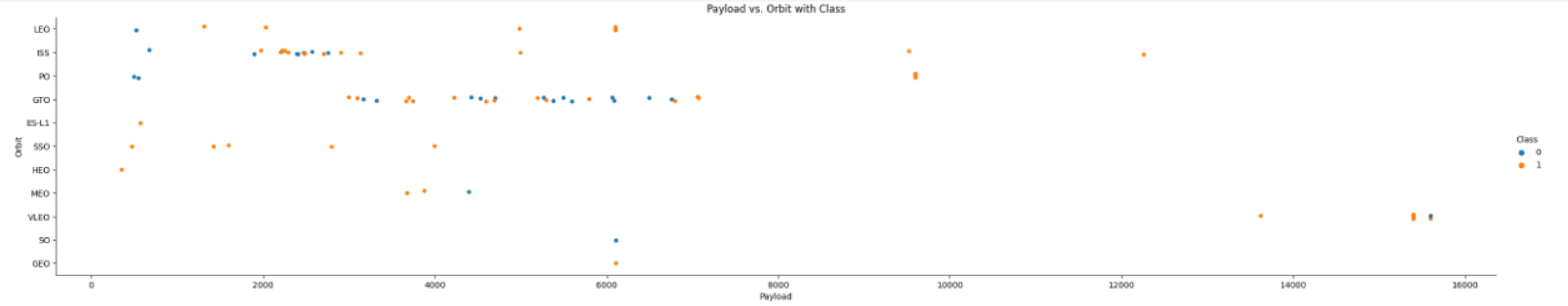
# Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



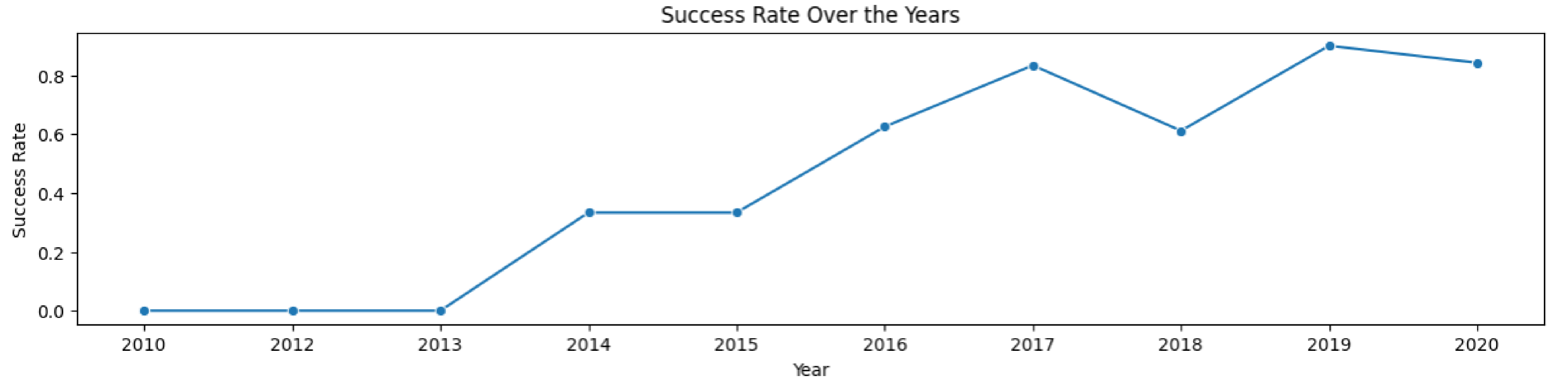
# Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



# Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



# All Launch Site Names

- Used the key word DISTINCT to show only unique launch sites from the SpaceX data.

## Task 1

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

In [7]:
```
%%sql
SELECT DISTINCT "Launch_Site" FROM SPACEXTBL
```

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

Out[7]:

| Launch_Site |
|---|
| CCAFS LC-40 |
| VAFB SLC-4E |
| KSC LC-39A |
| CCAFS SLC-40 |
| None |

# Launch Site Names Begin with 'CCA'

- Used the query below to display 5 records where launch sites begin with `CCA`

## Task 2

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

In [27]:
```
%%sql
SELECT * FROM SPACEXTBL WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5
```

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

Out[27]:

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outc |
|---|---|---|---|---|---|---|---|---|---|
| 06/04/2010 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0.0 | LEO | SpaceX | Success | Failure (paracl |
| 12/08/2010 | 15:43:00 | F9 v1.0 B0004 | CCAFS LC-40 | Dragon demo flight C1, two CubeSats, barrel of Brouere cheese | 0.0 | LEO (ISS) | NASA (COTS) NRO | Success | Failure (paracl |
| 22/05/2012 | 7:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525.0 | LEO (ISS) | NASA (COTS) | Success | No att |
| 10/08/2012 | 0:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500.0 | LEO (ISS) | NASA (CRS) | Success | No att |
| 03/01/2013 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677.0 | LEO (ISS) | NASA (CRS) | Success | No att |

# Total Payload Mass

- Calculated the total payload carried by boosters from NASA as 45596 using the query below

## Task 3

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

In [9]:
```
%sql SELECT SUM("PAYLOAD_MASS__KG_") AS TotalPayloadMass FROM SPACEXTBL WHERE "Customer" LIKE '%NASA (CRS)%'
```

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

Out[9]:

| TotalPayloadMass |
|---|
| 48213.0 |

# Average Payload Mass by F9 v1.1

- Calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

## Task 4

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

In [10]:

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

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

Out[10]:

| AveragePayloadMass |
| --- |
| 2928.4 |

# First Successful Ground Landing Date

- Observed that the dates of the first successful landing outcome on ground pad was 01/08/2018

### Task 5

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

*Hint:Use min function*

```
[16]: %sql SELECT MIN("Date") AS FirstSuccessfulGroundPadLandingDate FROM SPACEXTBL WHERE "Landing_Outcome" = 'Success (ground pad)'
```

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

[16]:

| FirstSuccessfulGroundPadLandingDate |
|---|
| 01/08/2018 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

- Used the WHERE clause to filter for boosters which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 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

[19]:
```
%%sql
SELECT "Booster_Version" FROM SPACEXTBL WHERE "Landing_Outcome" = 'Success (drone ship)'
       AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000
```

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

[19]:

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

# Total Number of Successful and Failure Mission Outcomes

- Used wildcard like ‘%’ to filter for WHERE MissionOutcome was a success or a failure.

## Task 7

List the total number of successful and failure mission outcomes

[20]:
```
%sql SELECT "Mission_Outcome", COUNT(*) AS TotalCount FROM SPACEXTBL GROUP BY TRIM("Mission_Outcome")
```

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

[20]:

| Mission_Outcome | TotalCount |
|---|---|
| None | 898 |
| Failure (in flight) | 1 |
| Success | 99 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

- Determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.

### Task 8

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

In [14]:
```
%sql SELECT "Booster_Version" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL)
```

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

Out[14]:

| 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

- We used a combinations of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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.**

In [16]:

```
%%sql SELECT
    CASE
        WHEN substr("Date", 4, 2) = '01' THEN 'January'
        WHEN substr("Date", 4, 2) = '02' THEN 'February'
        WHEN substr("Date", 4, 2) = '03' THEN 'March'
        WHEN substr("Date", 4, 2) = '04' THEN 'April'
        WHEN substr("Date", 4, 2) = '05' THEN 'May'
        WHEN substr("Date", 4, 2) = '06' THEN 'June'
        WHEN substr("Date", 4, 2) = '07' THEN 'July'
        WHEN substr("Date", 4, 2) = '08' THEN 'August'
        WHEN substr("Date", 4, 2) = '09' THEN 'September'
        WHEN substr("Date", 4, 2) = '10' THEN 'October'
        WHEN substr("Date", 4, 2) = '11' THEN 'November'
        WHEN substr("Date", 4, 2) = '12' THEN 'December'
        ELSE 'Invalid Month'
    END AS MonthName,
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTBL
WHERE substr("Date", 7, 4) = '2015'
    AND "Landing_Outcome" LIKE '%Failure (drone ship)%'
```

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

Out[16]:

| MonthName | Landing_Outcome | Booster_Version | Launch_Site |
|---|---|---|---|
| October | Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 |
| April | Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 |

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

- Selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.
- Applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.

### 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"
FROM SPACEXTBL
WHERE "Date" BETWEEN '04-06-2010' AND '20-03-2017'
GROUP BY TRIM("Landing_Outcome")
ORDER BY COUNT(*) DESC
```

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

Out[28]:

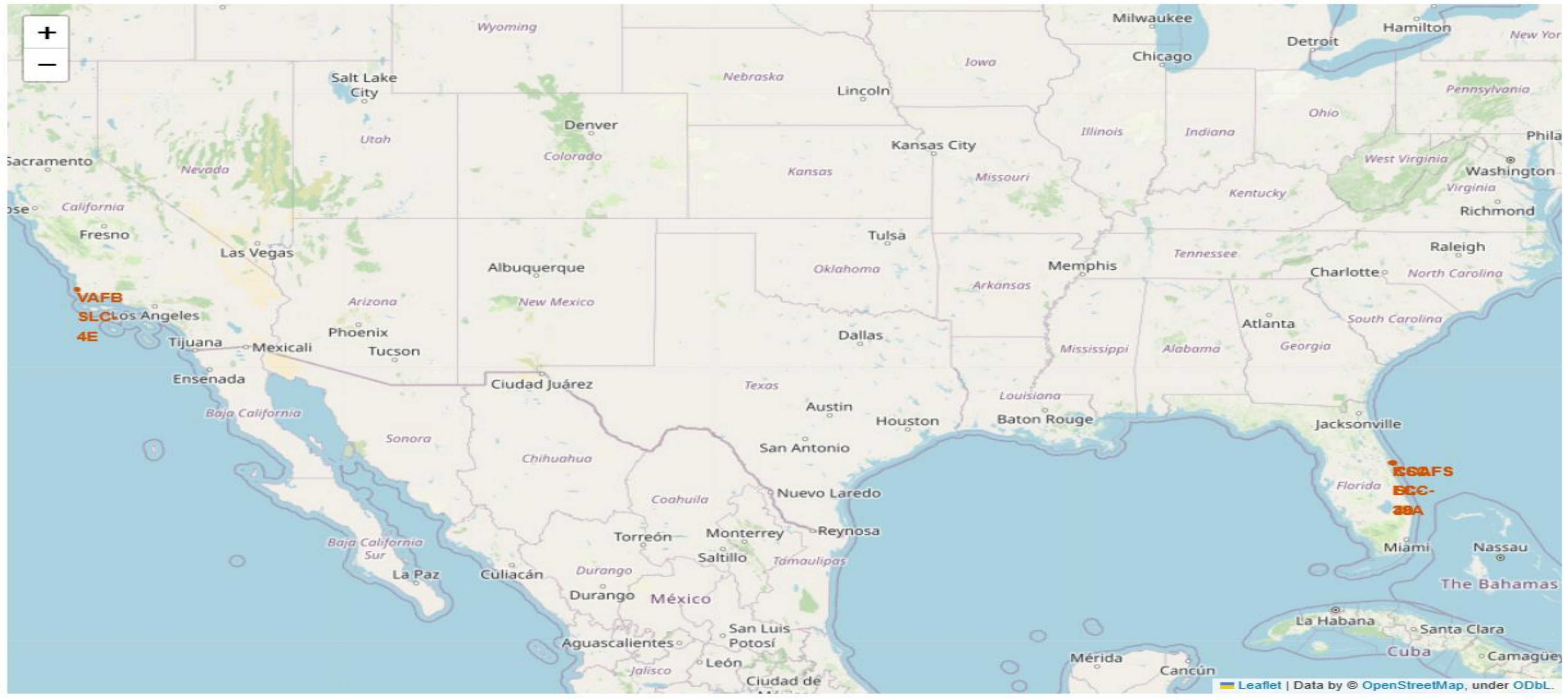
| Landing_Outcome | Count |
|---|---|
| Success | 20 |
| No attempt | 11 |
| Success (drone ship) | 8 |
| Success (ground pad) | 7 |
| Failure (drone ship) | 3 |
| Failure | 3 |
| Failure (parachute) | 2 |
| Controlled (ocean) | 2 |

Section 3

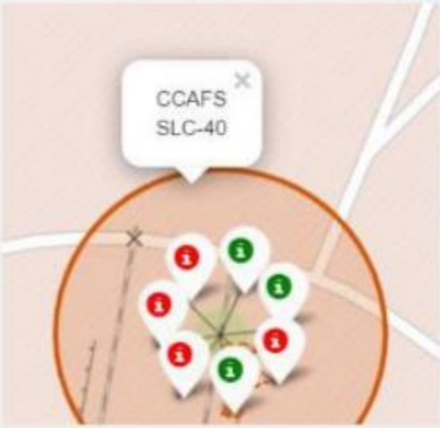
# Launch Sites Proximities Analysis

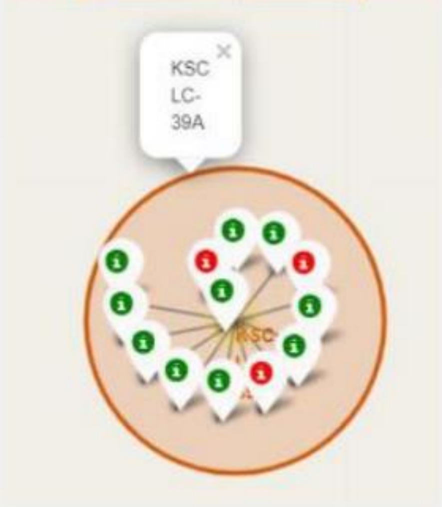
# launch sites global map markers

- SpaceX launch sites are in the US coast Florida and California



# Markers showing launch sites with color labels





***Florida Launch Sites***

*Green Marker shows successful Launches and Red Marker shows Failures*


***California Launch Site***

37

# Launch Site distance to landmarks

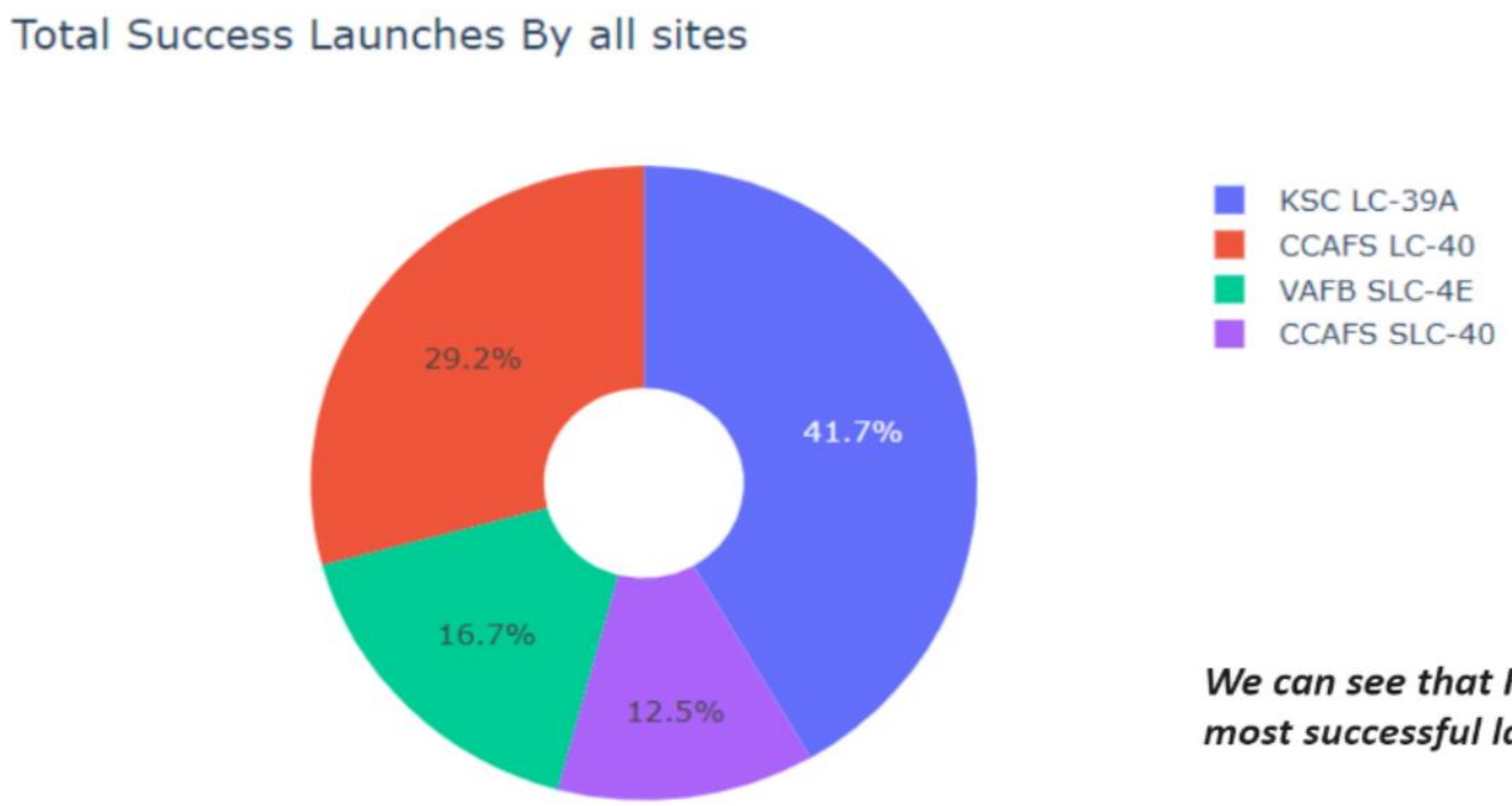



- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

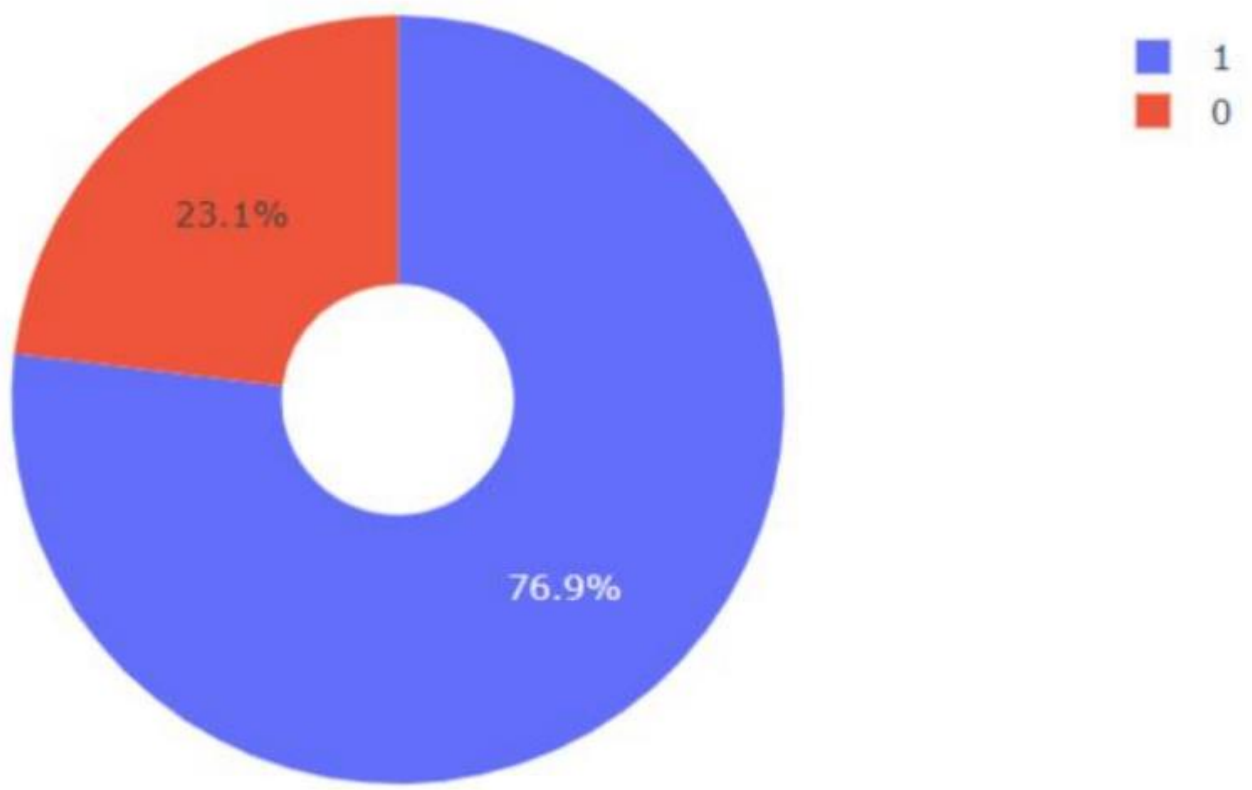
# Build a Dashboard with Plotly Dash

# Pie chart showing the success percentage achieved by each launch site



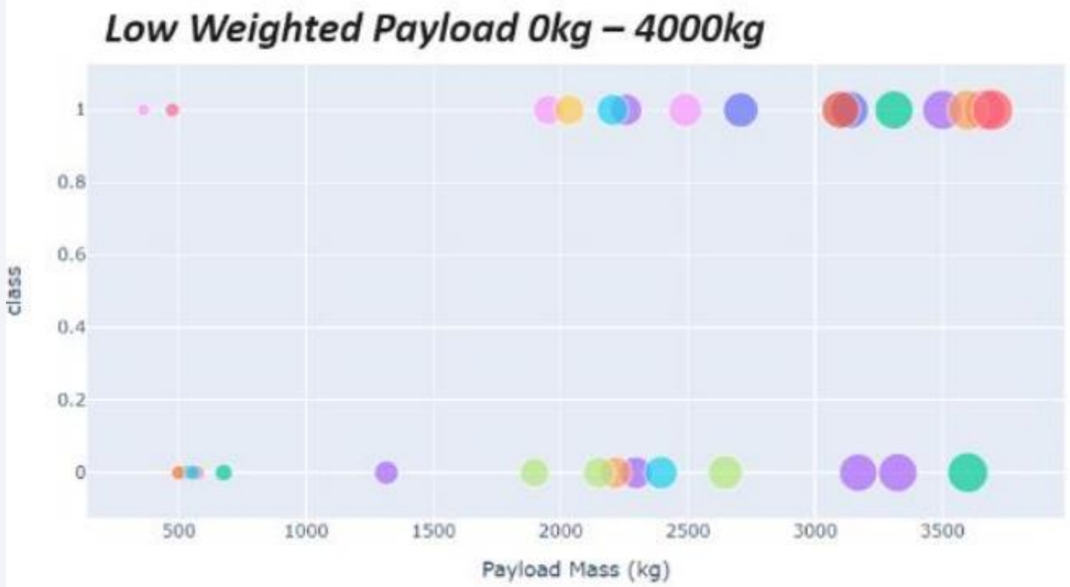

***We can see that KSC LC-39A had the most successful launches from all the sites***

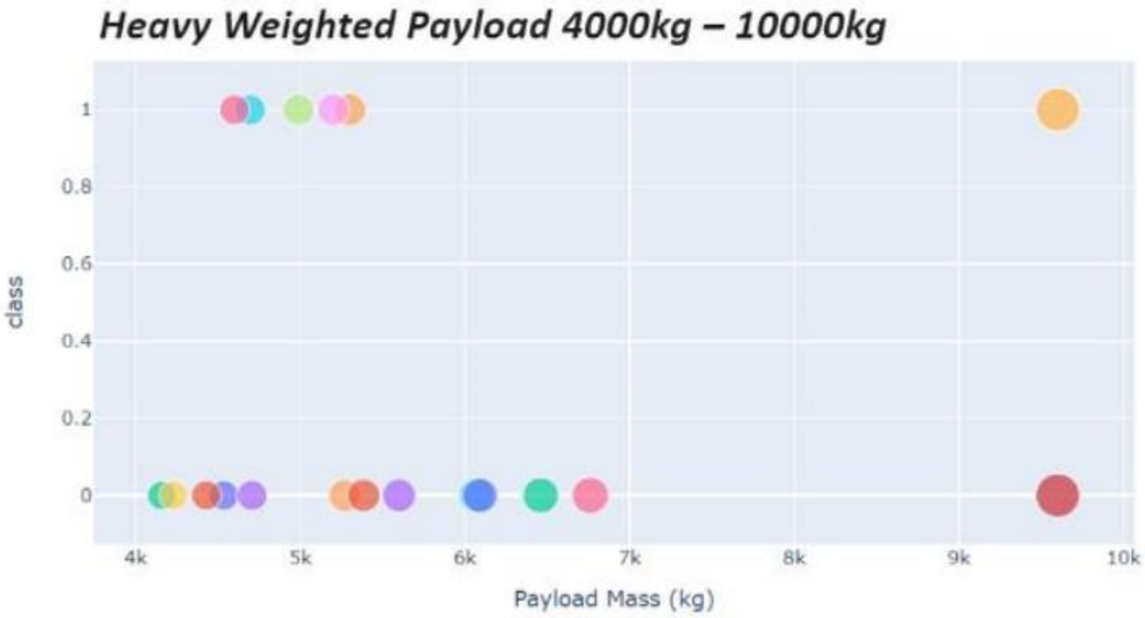
# Pie chart showing the Launch site with the highest launch success ratio




*KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate*

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




*We can see the success rates for low weighted payloads is higher than the heavy weighted payloads*

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

- The Logistic Regression is the model with the highest classification accuracy

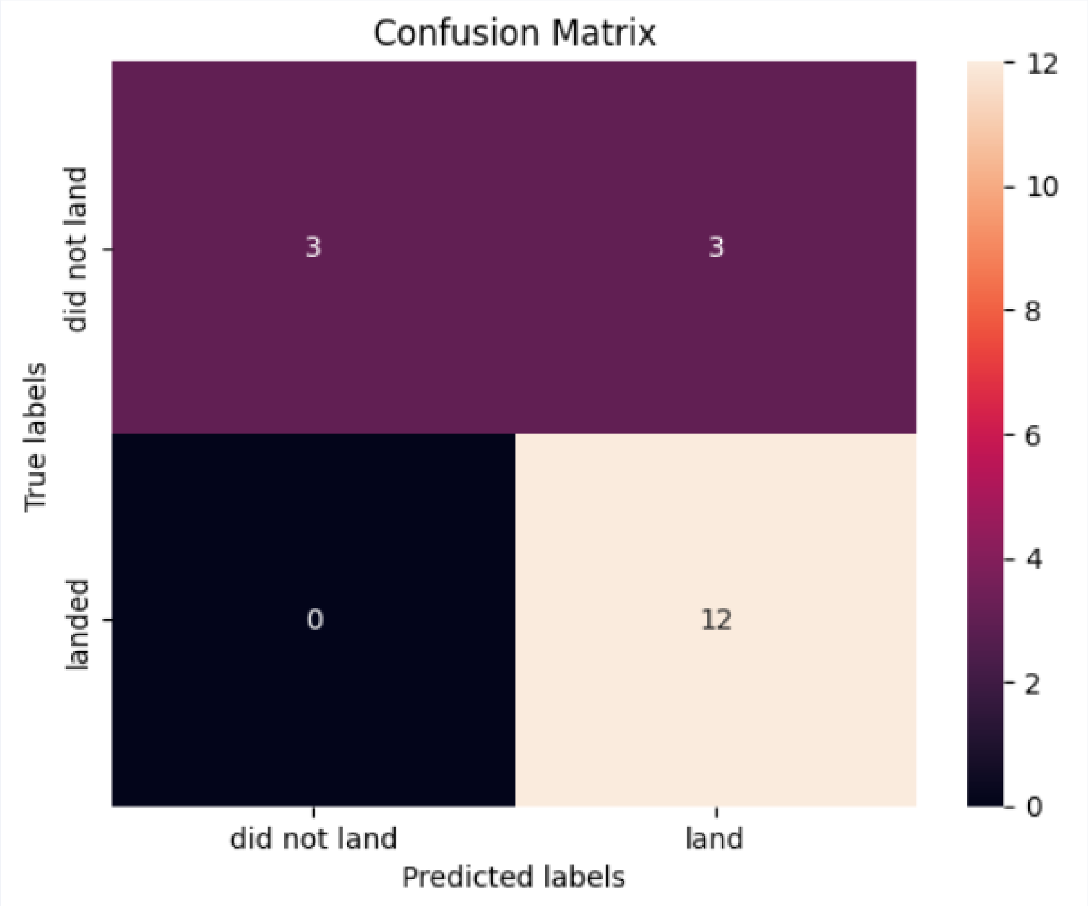
Find the method performs best:

[36]:
```
accuracy_scores = {
    'Logistic Regression': logreg_cv.score(X_test, Y_test),
    'Support Vector Machine': svm_cv.score(X_test, Y_test),
    'Decision Tree': tree_cv.score(X_test, Y_test),
    'K Nearest Neighbors': knn_cv.score(X_test, Y_test)
}

#The key=dict.get argument specifies that the comparison should be based on the values returned by dict.get(key).
best_method = max(accuracy_scores, key=accuracy_scores.get) #max_key = max(dict, key=dict.get)
best_accuracy = accuracy_scores[best_method] #max_value = dict[max_key]

print("Best-performing method:", best_method)
print("Accuracy:", best_accuracy)
```

```
Best-performing method: Logistic Regression
Accuracy: 0.8333333333333334
```

# Confusion Matrix

- The confusion matrix for the Logistic Regression shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



# Conclusions

- Can conclude that:
  - The larger the flight amount at a launch site, the greater the success rate at a launch site.
  - Launch success rate started to increase in 2013 till 2020.
  - Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
  - KSC LC-39A had the most successful launches of any sites.
  - The Logistic Regression is the best machine learning algorithm for this task.

# Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!