IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Andrew Smith
03-02-2023

# Outline

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

# Executive Summary

- Summary of methodologies
  - Data Collection with an API and Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis(EDA) with SQL and Data Visualization
  - Interactive Visual Analytics with Folium and Dashboard
  - Machine Learning Predictions
- Summary of all results
  - The results use Exploratory Data Analysis, Interactive Analytics and Data Visualization, and Predictive Analysis to help a new space company, Space Y, determine rocket launch cost and whether first rockets can land successfully and be reused.

# Introduction

- The purpose of this project was to assume the role of a Data Scientist working for a start up rocket company looking to compete with Space X. In this project we were looking to determine the price of each rocket launch as well as predict whether the Space X Falcon 9's first stage rocket will land successfully as to be able to be reused by Space X.
- Problems we want answers for:
  - Was the landing successful and what are the main characteristics of a successful landing?
  - Can variables involved in successful rocket landing be used to determine the validity of Space X's $62 million cost?
  - What are the conditions that allow Space X to reuse the first stage of a rocket launch?

Section 1

# Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - Space X REST API
  - Web Scraping via Wikipedia
- Perform data wrangling
  - Data processed using one hot encoding for classification models
- 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

- Datasets were collected by using REST API and Web Scraping from Wikipedia.
- For the REST API, we first start by request rocket launch Data from the Space X API. We then decode the response content as JSON and load JSON result into a Pandas dataframe.
- For Web Scraping, we used BeautifulSoup to extract launch records as HTML tables. We then created a dataframe by parsing the launch HTML tables.

# Data Collection – SpaceX API

Request rocket launch for Space X API and convert response to JSON

Use json_normalize to convert result to dataframe

Cleanse dataframe and filter to only Falcon 9 launches

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)
```

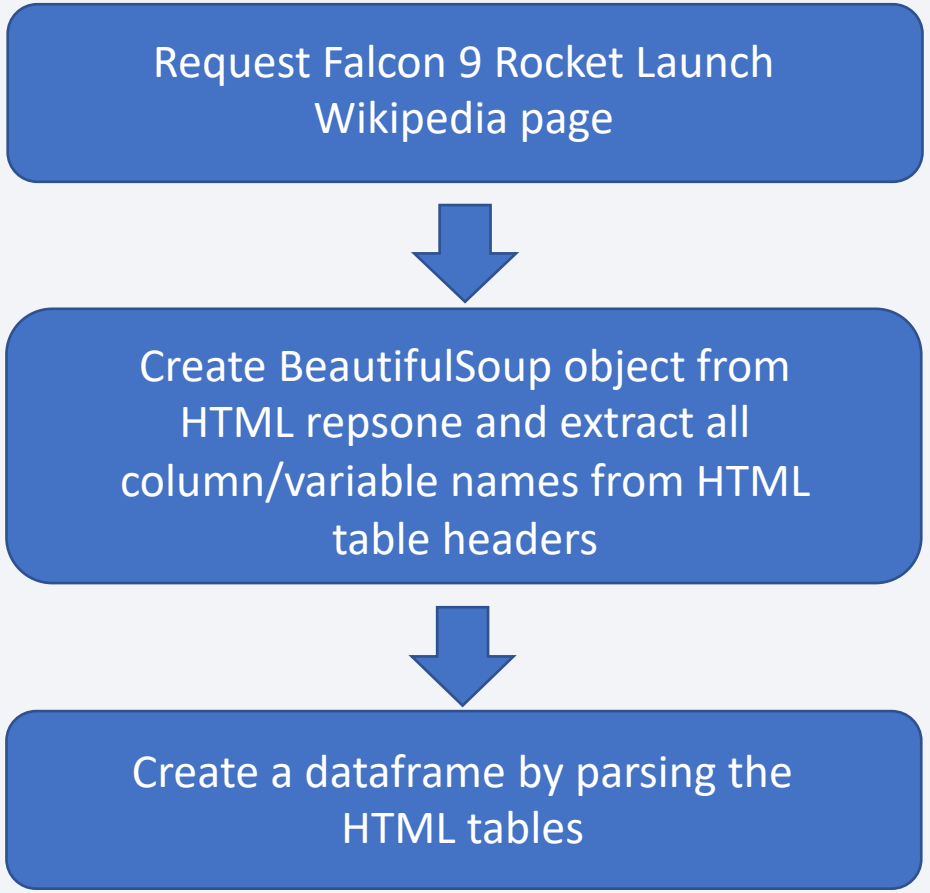
```
In [11]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())
```

```
In [13]: # Lets take a subset of our dataframe keeping only the features we want and the flight
         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
         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 ir
         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 d
         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)]
```

[Github source code]

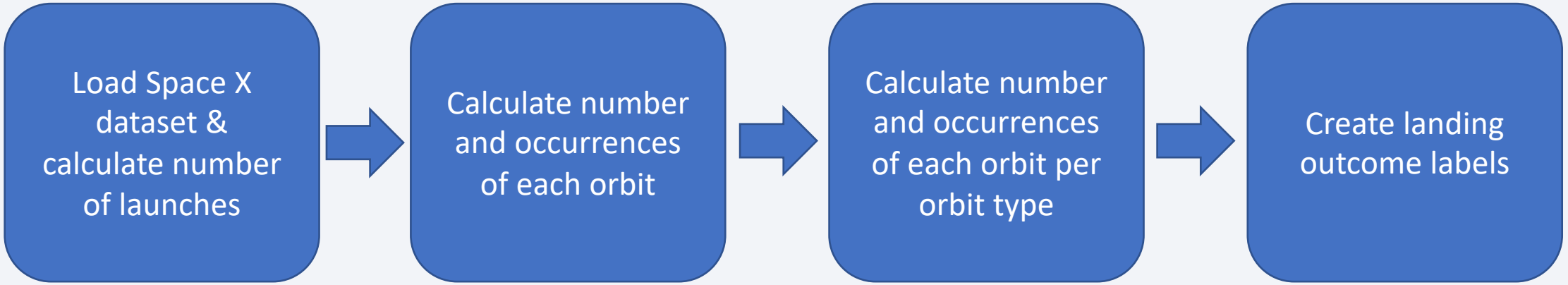
# Data Collection - Scraping

Github Source Code



# Data Wrangling

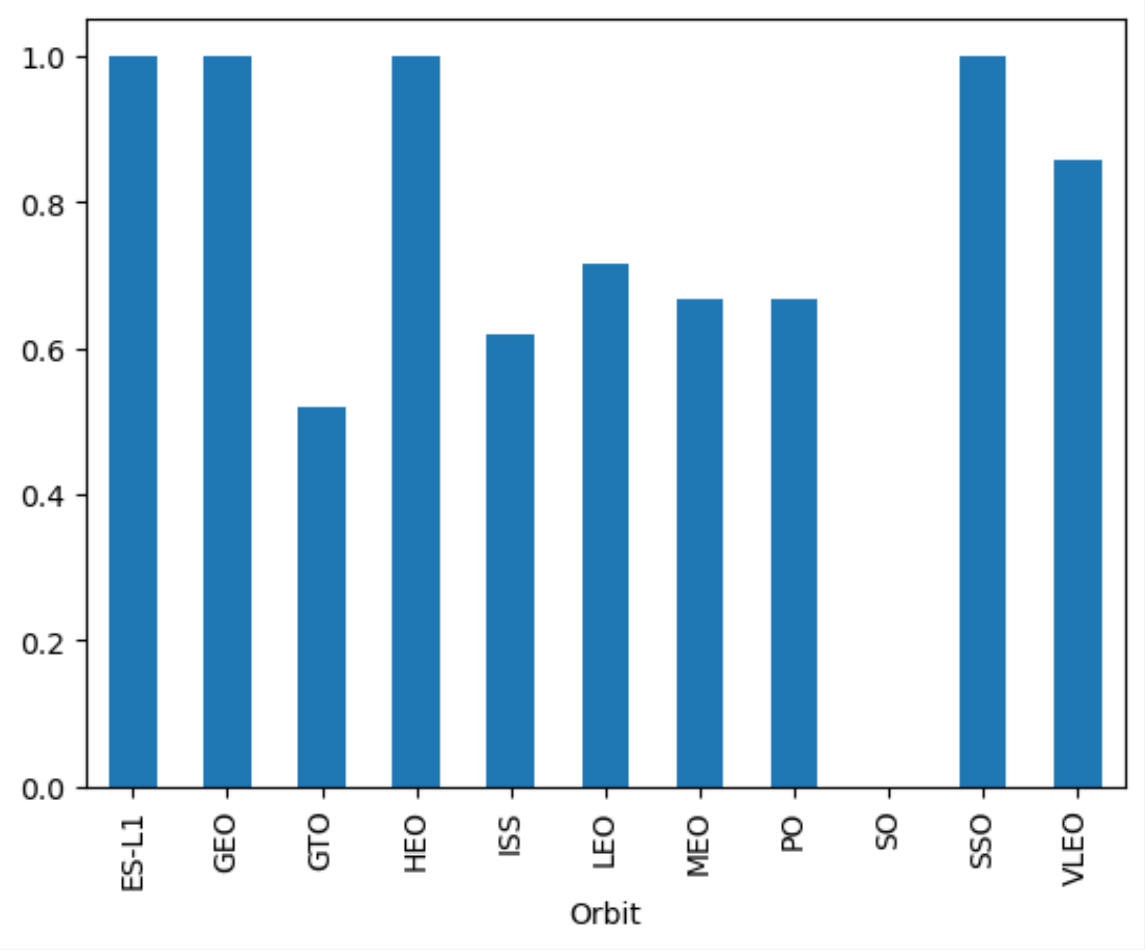
- We calculated the number of launches on each site, then calculated the number and occurrences of each orbit, and the number and occurrence of the mission outcome per orbit type.
- Then we created landing outcome labels from the Outcome column.




Github source code

# EDA with Data Visualization

- Scatter Plots
  - Flight number vs. Payload Mass
  - Flight number vs. Launch Site
  - Payload vs. Launch Site
  - Orbit vs. Flight Number
  - Payload vs. Orbit Type
  - Orbit vs Payload Mass
- Bar Graph
  - Success rate vs. Orbit
- Line Graph
  - Success rate vs. Year




[Github source code]

# EDA with SQL

- Displaying the names of the unique launch sites in the space mission
- 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 caried 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 failed landing_outcomes in drone ship, their booster versions, and launch site names in year 2015
- Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order

Github source code

# Build an Interactive Map with Folium

- Used the longitude and latitude at each launch site and added a circle marker around each launch site with a name label
- Used MarkerCluster() to assign the dataframe launch_outcomes(failure, success) to classes 0 and 1 with Red and Green markers
- Using Haversine's formula, we calculated the distance of launch sites to the closest coastlines, cities, highways, and railways.

Github Source Code

# Build a Dashboard with Plotly Dash

- When attempting to run the plotly dashboard in the skills lab, I received an error when setting up the project environment and was never able to open the .py file.

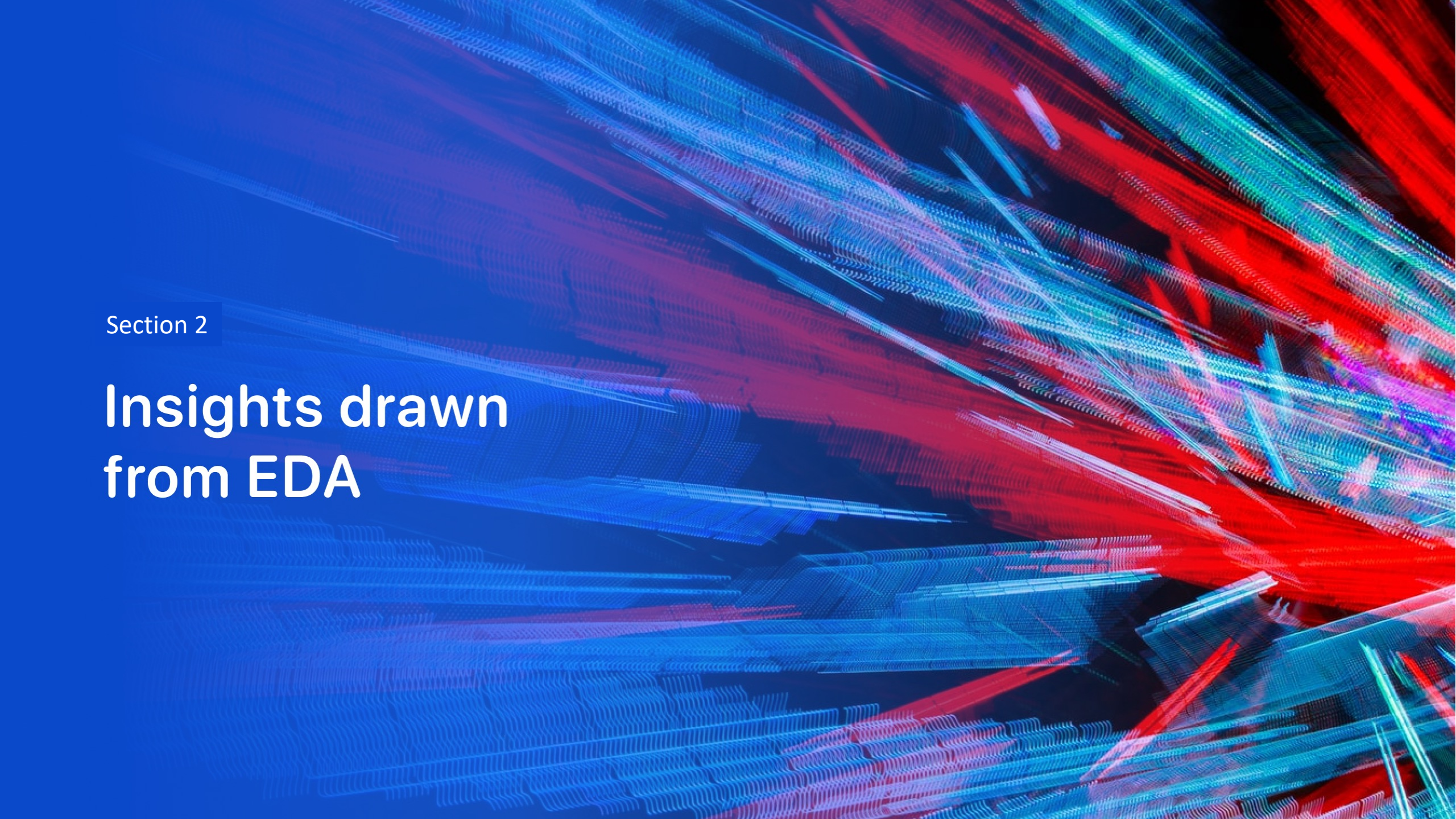
# Predictive Analysis (Classification)

- Data Preparation
  - Load dataset
  - Normalized data
  - Split data into training and test sets
- Model preparation
  - Selection of machine learning algorithms
  - Set parameters for each algorithm to GridSearchCV
  - Training GridSeachModel models with training dataset
- Model evaluation
  - Get tuned hyperparameters for each type of algorithm
  - Check the accuracy for each model
  - Plot confusion matrix
- Model Comparison
  - Comparison of models according to their accuracy
  - The model with the best accuracy score will be chosen as best performing model.

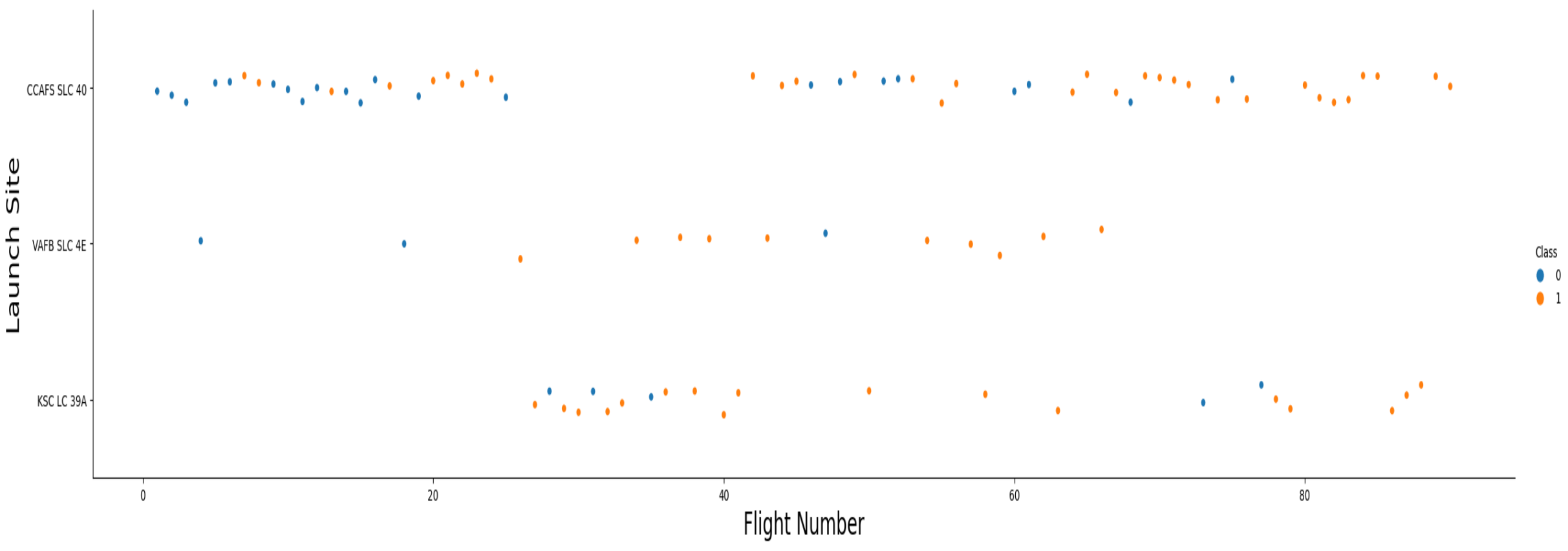
Github Source Code

# Results

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

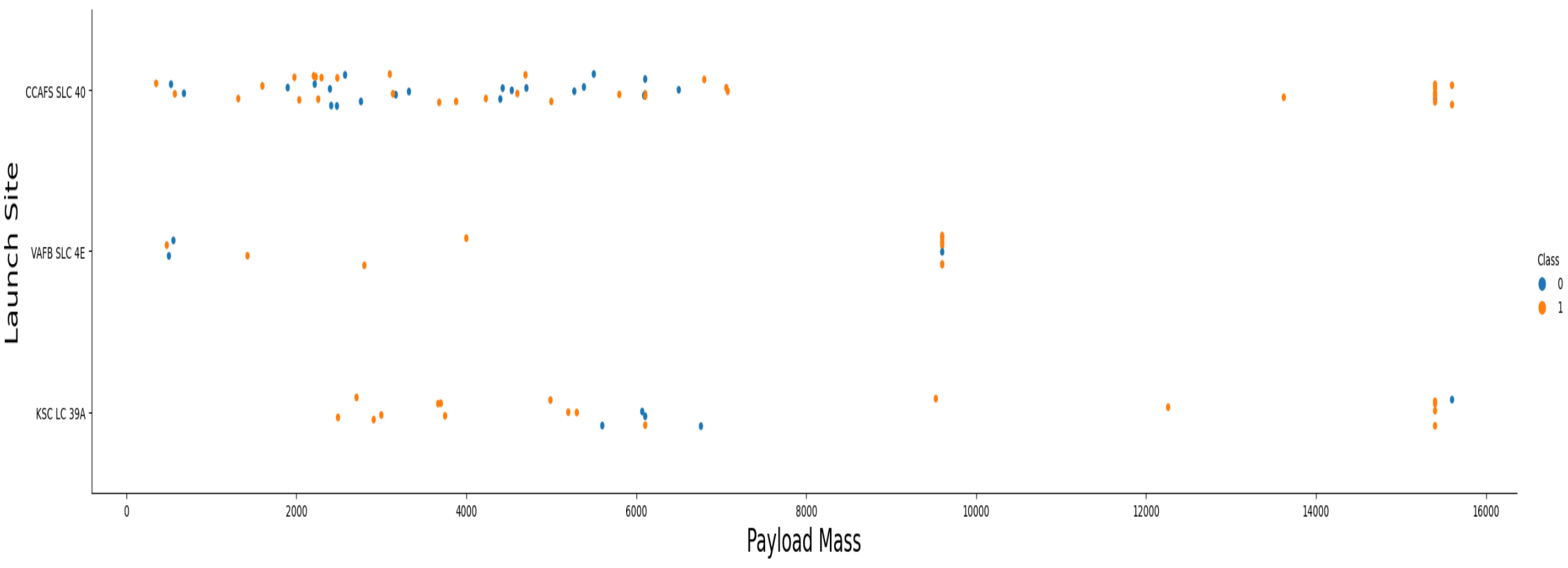

Section 2
Insights drawn from EDA

# Flight Number vs. Launch Site




This scatter plot shows that as flight numbers increase, the success rate also increases.

# Payload vs. Launch Site

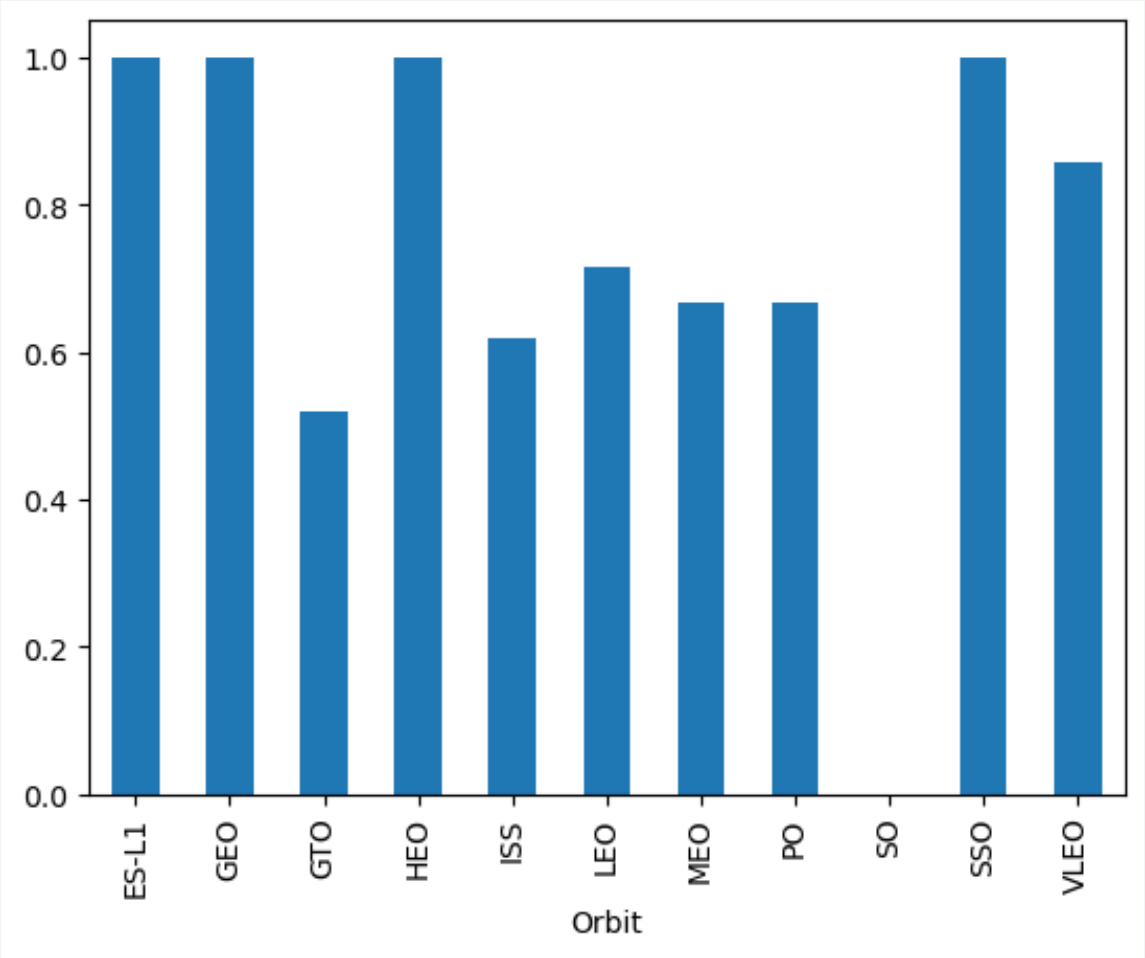



The scatter pot shows that heavier payloads above 9000 kg have a higher success rate.
VAFB-SLC launchsite has no rockets with a payload mass above 10000 kg.

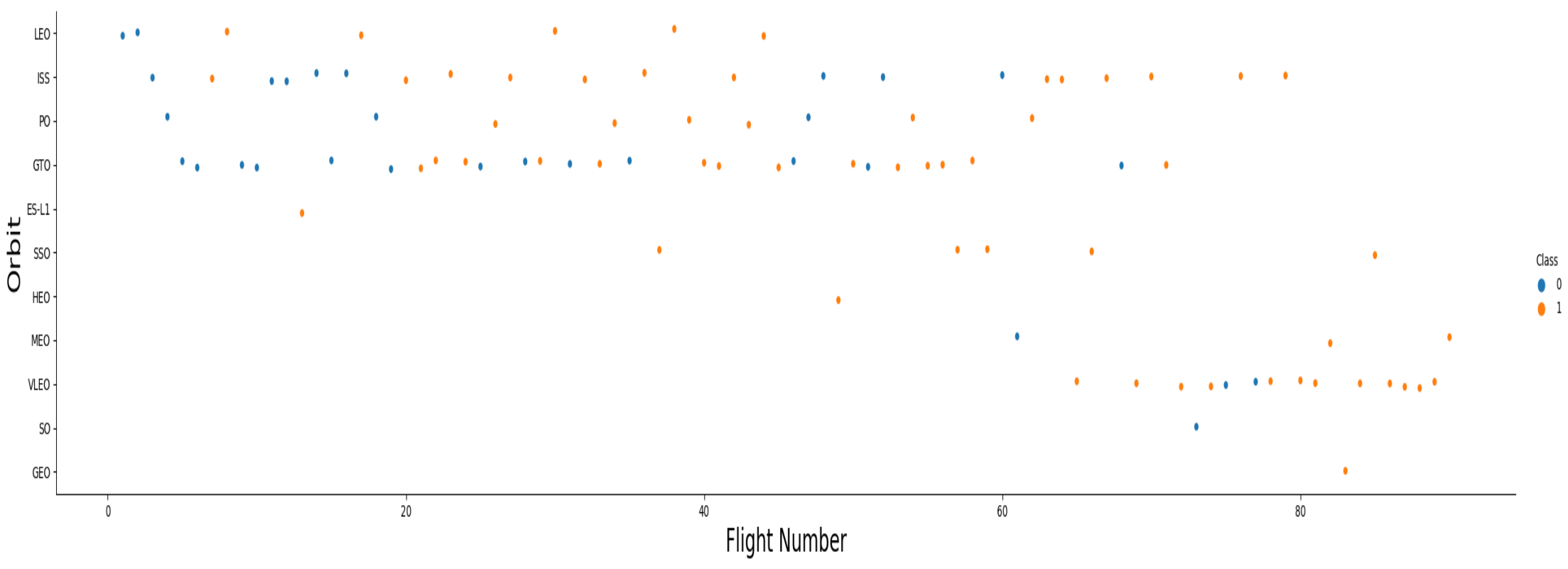
# Success Rate vs. Orbit Type

- This bar chart shows the success rate of different orbit types.

- We not that Orbit type ES-L1, GEO, HEO, and SSO have the greatest success rate.

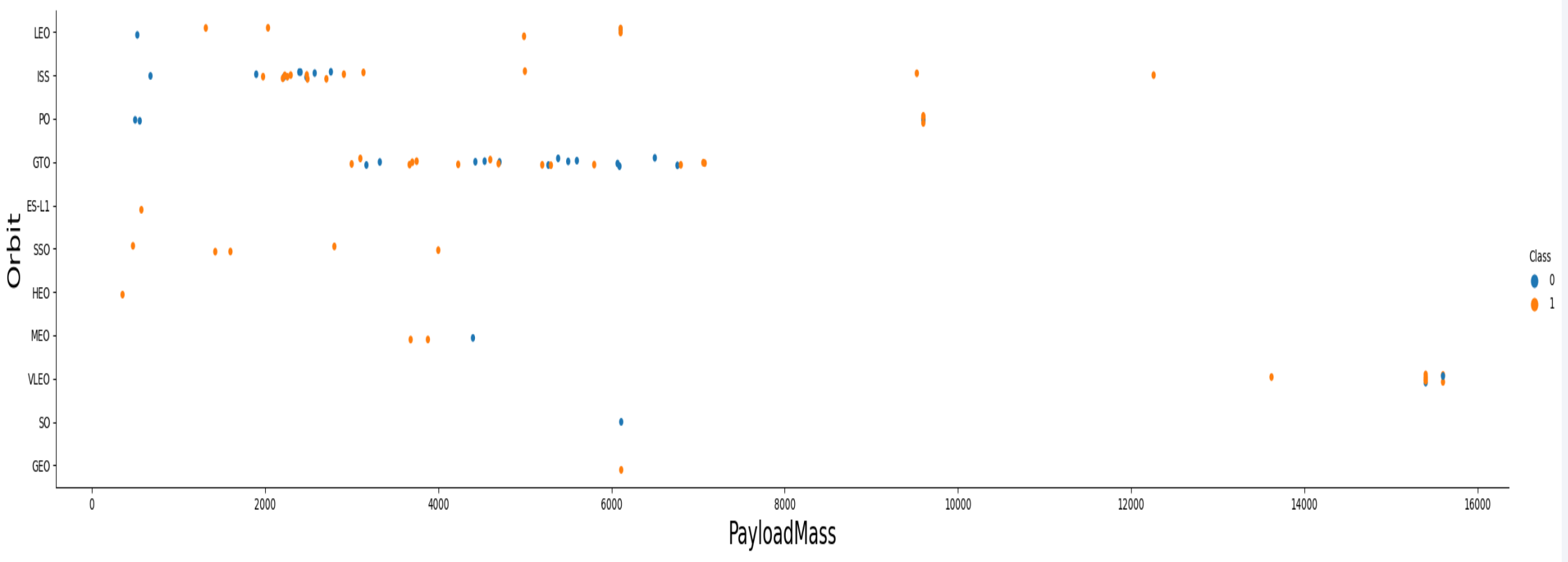


# Flight Number vs. Orbit Type




Success rate improved over time for all orbits.

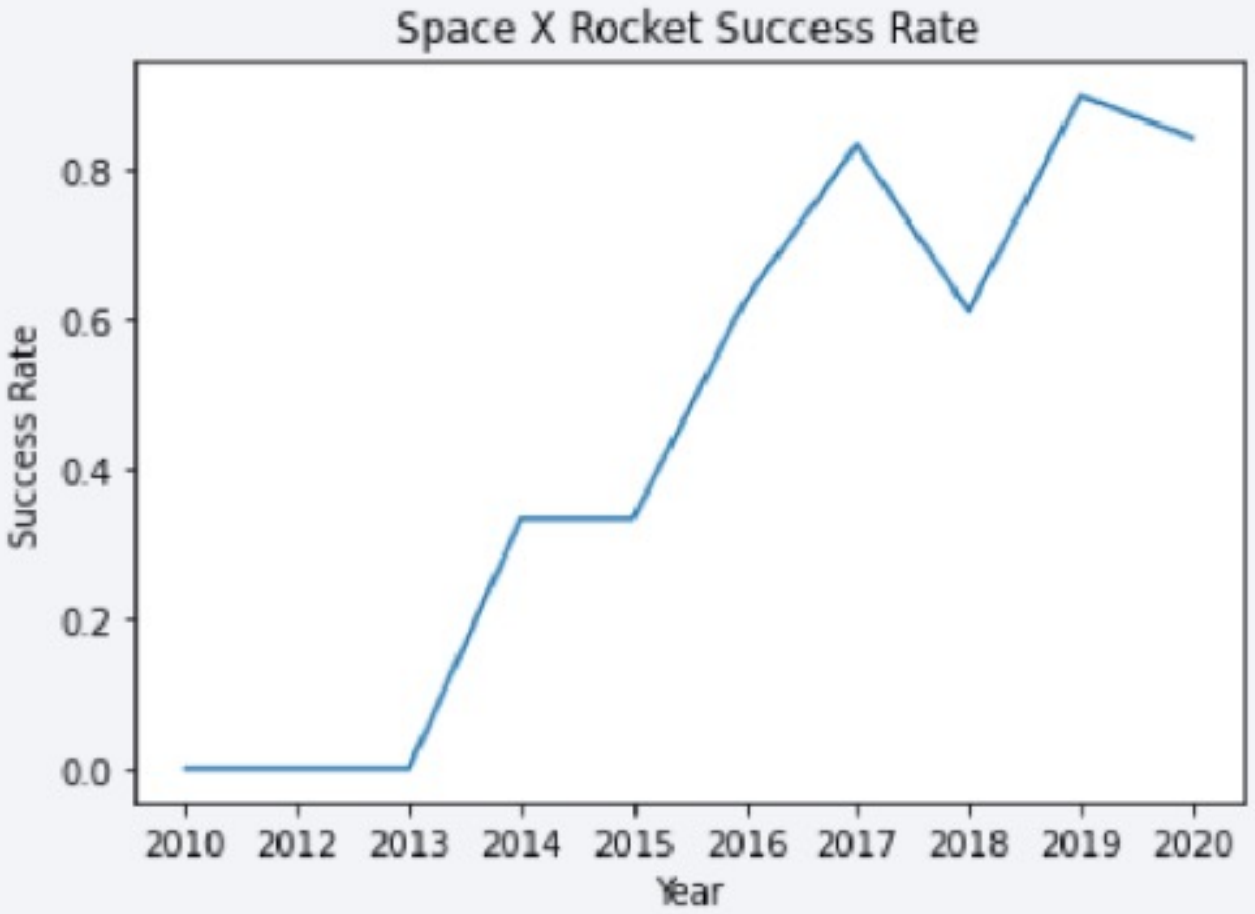
# Payload vs. Orbit Type




Heavier payloads have greater success in certain orbits such as LEO.

# Launch Success Yearly Trend

- After a period of adjustments from 2010 to 2013, the success rate increased every year from 2013 through 2020.



# All Launch Site Names

- The query uses DISTINCT to pull unique Launch Sites.

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;
```

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

# Launch Site Names Begin with 'CCA'

- Query finds 5 records where launch sites begin with `CCA`

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

| DATE | time__utc_ | booster_version | launch_site | payload | payload_mass__kg_ | orbit | customer | mission_outcome | landing__outcome |
|---|---|---|---|---|---|---|---|---|---|
| 2010-04-06 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 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 (parachute) |
| 2012-08-10 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 2013-01-03 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 2013-03-12 | 22:41:00 | F9 v1.1 | CCAFS LC-40 | SES-8 | 3170 | GTO | SES | Success | No attempt |

# Total Payload Mass

- Calculate the total payload carried by boosters from NASA

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL WHERE CUSTOMER LIKE '%NASA (CRS)%';
```

| total_payload_mass |
| --- |
| 24624 |

# Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD_MASS FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
```

| avg_payload_mass |
| --- |
| 3676 |

# First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

```
%sql SELECT MIN(DATE) AS SUCCESSFUL_LANDING FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

| successful_landing |
| --- |
| 2017-01-05 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
%sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 AND LANDING__OUTCOME = 'Success (drone s
```

| booster_version |
| --- |
| F9 FT B1031.2 |
| F9 FT B1022 |

# Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(*) AS TOTAL FROM SPACEXTBL GROUP BY MISSION_OUTCOME ORDER BY MISSION_OUTCOME;
```

| mission_outcome | total |
|---|---|
| Success | 44 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

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

```
%sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL) ORDER BY BOOS
```

| booster_version |
| --- |
| F9 B5 B1048.4 |
| F9 B5 B1049.4 |
| F9 B5 B1049.5 |
| F9 B5 B1058.3 |
| F9 B5 B1060.2 |

# 2015 Launch Records

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

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND DATE_PART('Year', DATE) = 201
```

| booster_version | launch_site |
|---|---|
| F9 v1.1 B1012 | CCAFS LC-40 |

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

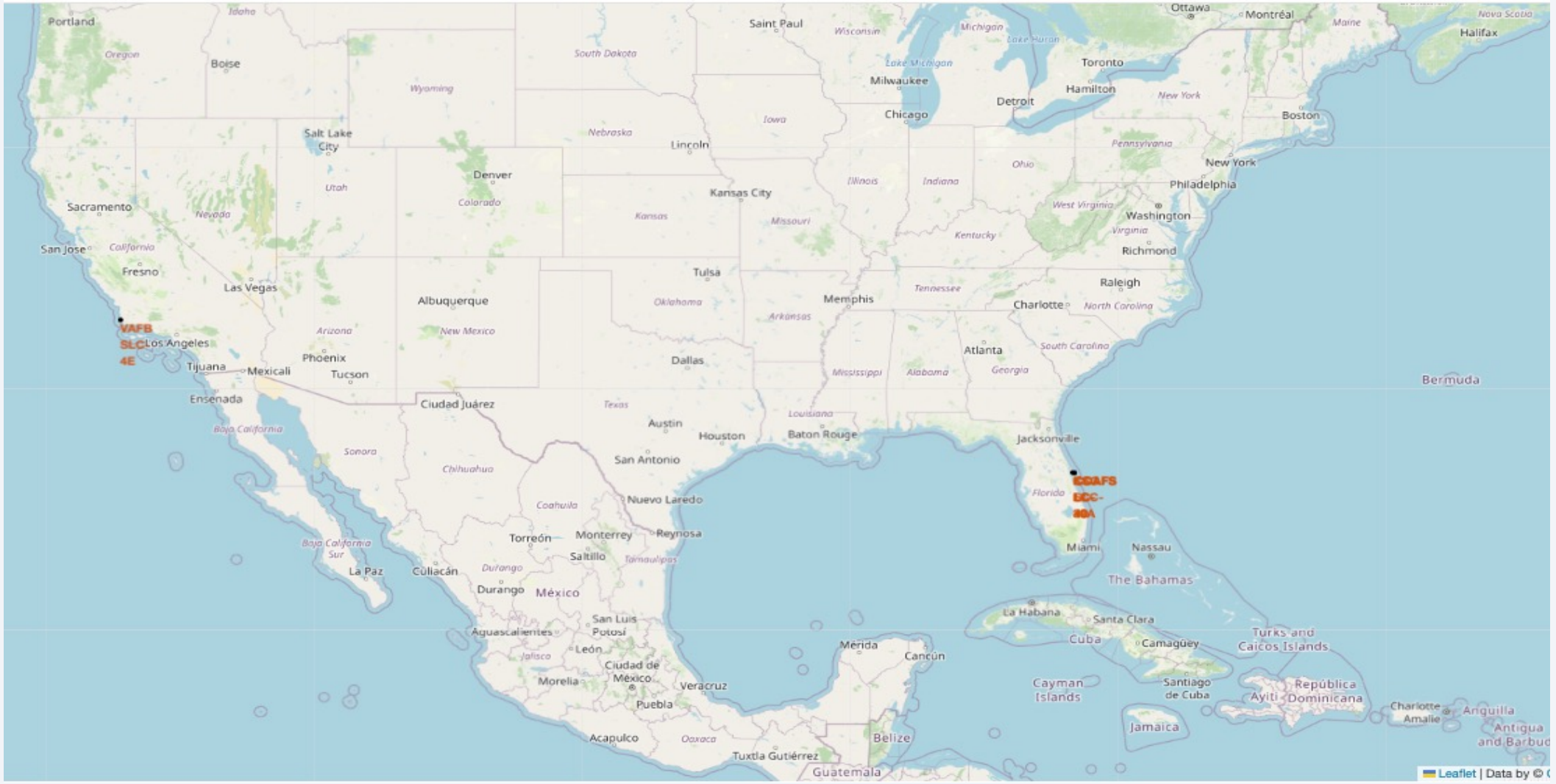
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
%sql SELECT LANDING__OUTCOME, COUNT(*) AS TOTAL FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING__OUTCOME
```

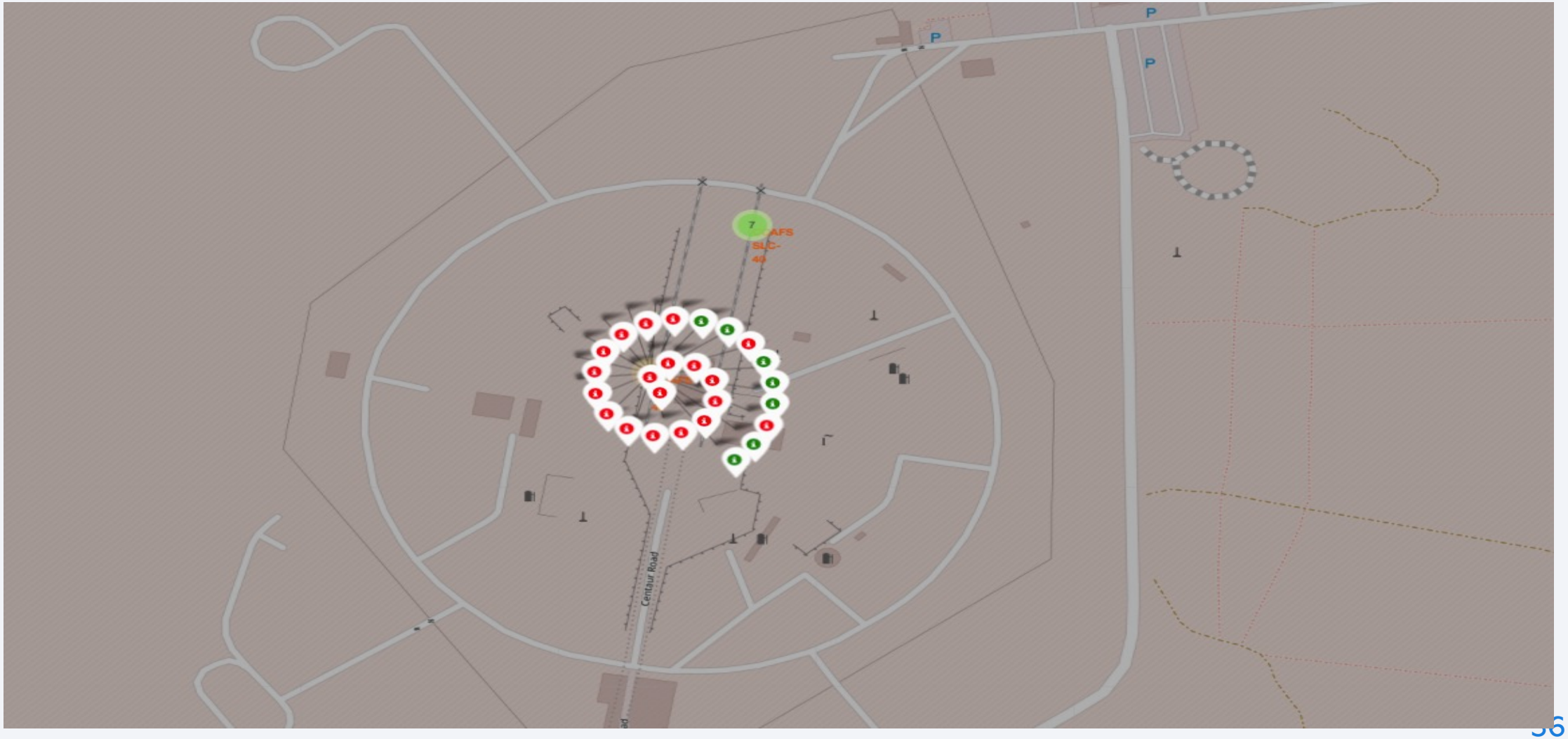
| landing__outcome | total |
|---|---|
| No attempt | 7 |
| Failure (drone ship) | 2 |
| Success (drone ship) | 2 |
| Success (ground pad) | 2 |
| Controlled (ocean) | 1 |
| Failure (parachute) | 1 |


Section 3
Launch Sites
Proximities Analysis

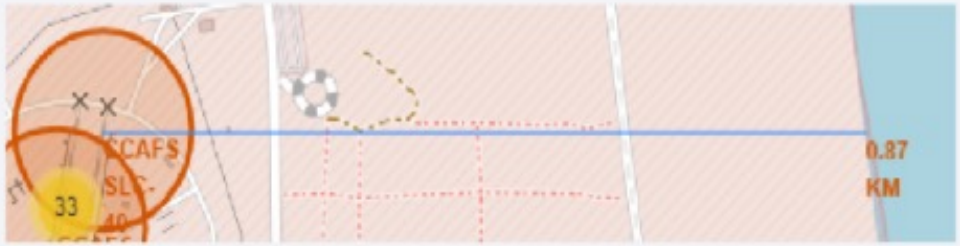
# Folium Map – All Launch Sites

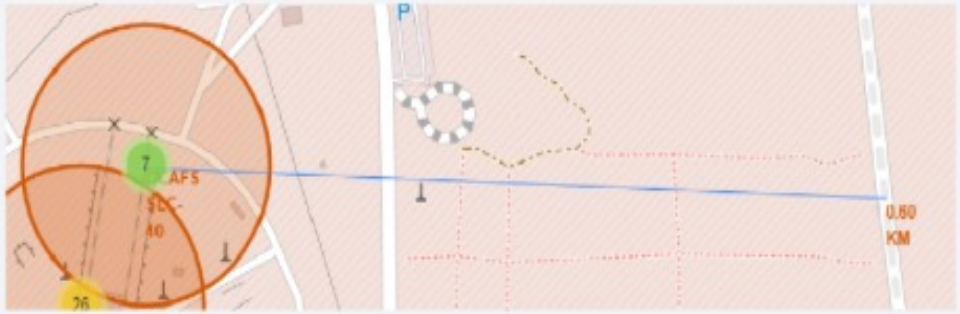


# Folium Map – Color Markers

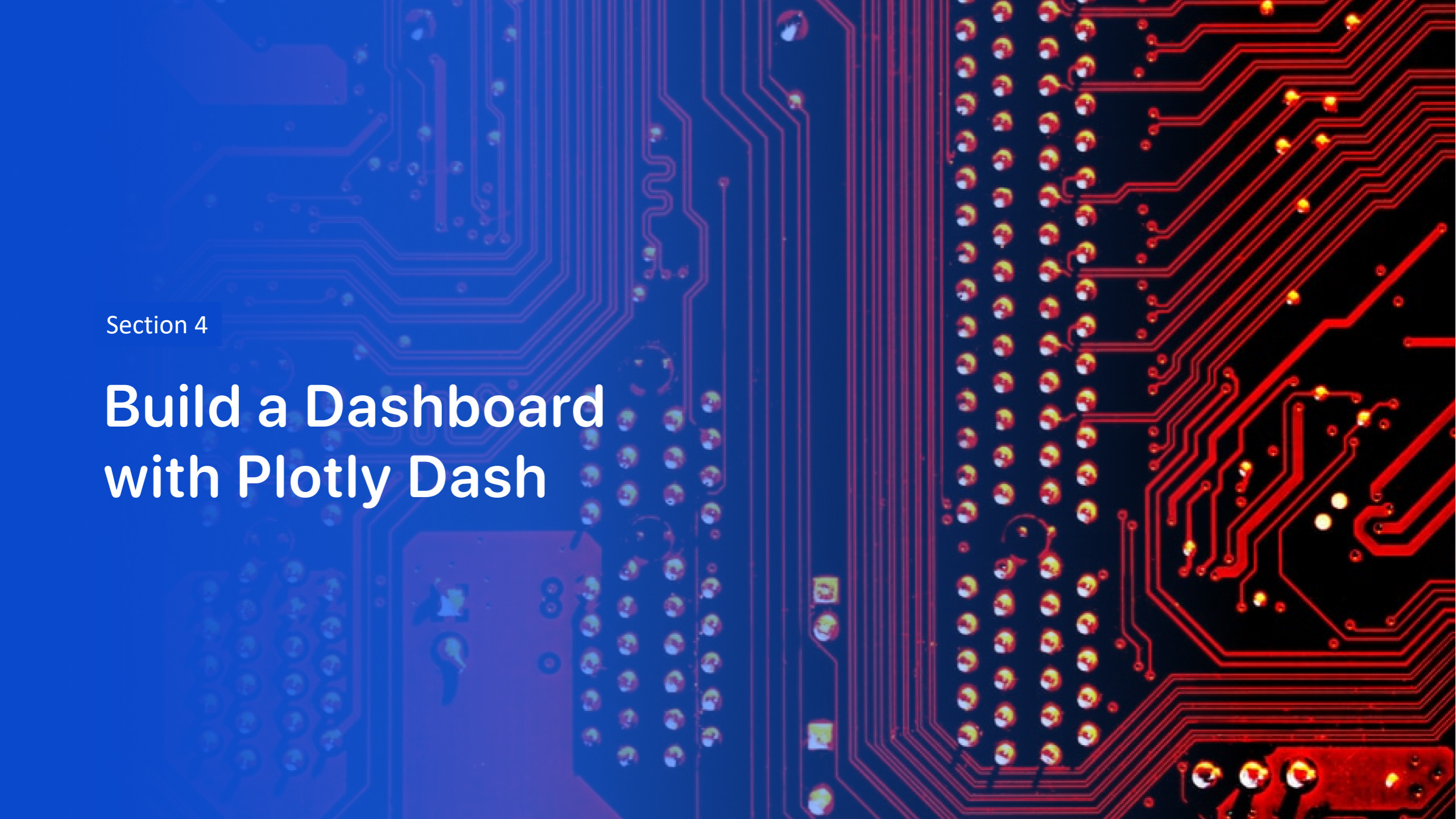


# Folium Map – Proximity to Landmarks






Section 4
Build a Dashboard with Plotly Dash

# <Dashboard Screenshot 1>

- Plotly project gave an error multiple times when attempting to update the python packaging.

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

Find the method performs best:

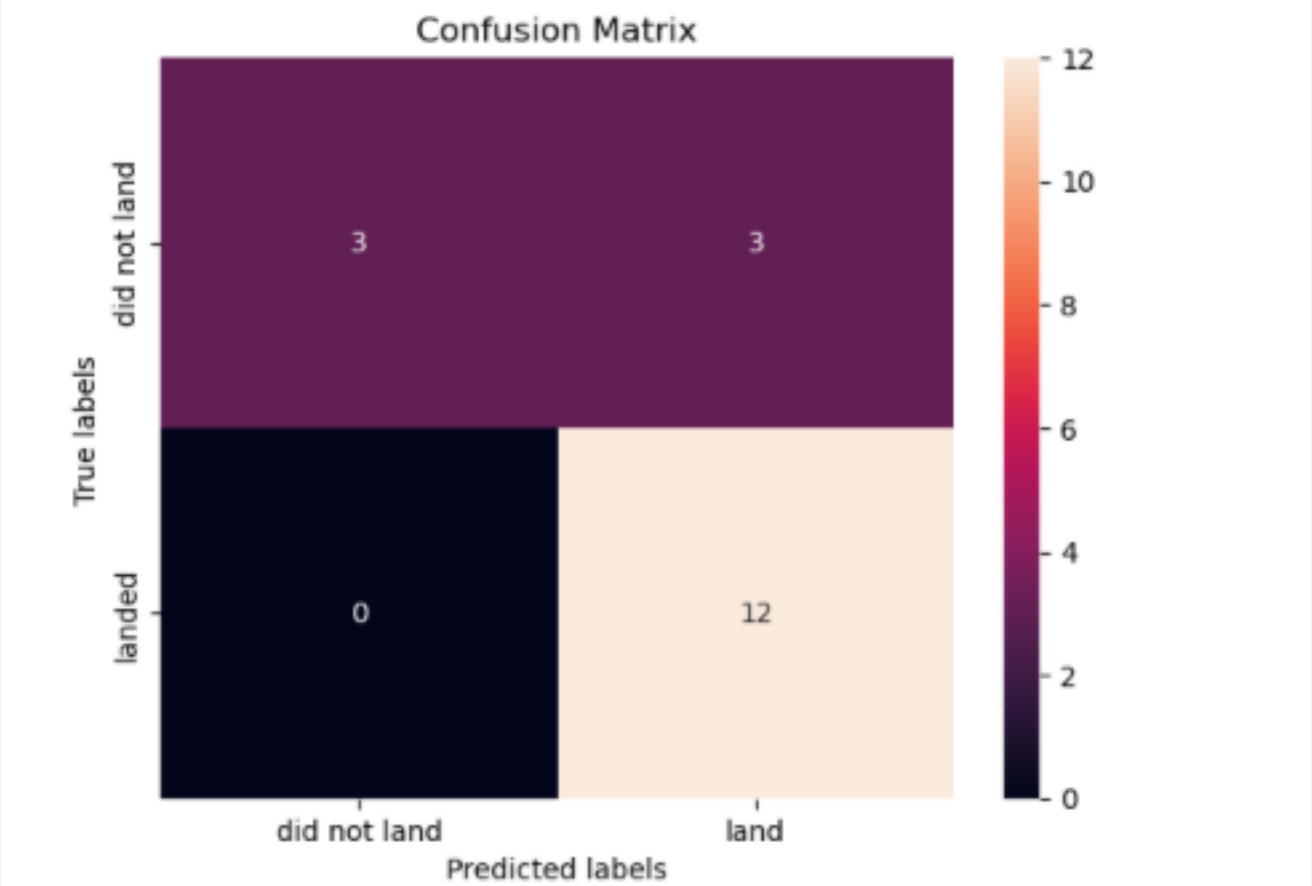
```
print("Model\t\tAccuracy\tTestAccuracy")#,logreg_cv.best_score_)
print("LogReg\t\t{}\t\t{}".format((logreg_cv.best_score_).round(5), logreg_cv.score(X_test, Y_test).round(5)))
print("SVM\t\t{}\t\t{}".format((svm_cv.best_score_).round(5), svm_cv.score(X_test, Y_test).round(5)))
print("Tree\t\t{}\t\t{}".format((tree_cv.best_score_).round(5), tree_cv.score(X_test, Y_test).round(5)))
print("KNN\t\t{}\t\t{}".format((knn_cv.best_score_).round(5), knn_cv.score(X_test, Y_test).round(5)))

comparison = {}

comparison['LogReg'] = {'Accuracy': logreg_cv.best_score_.round(5), 'TestAccuracy': logreg_cv.score(X_test, Y_test).round(5)}
comparison['SVM'] = {'Accuracy': svm_cv.best_score_.round(5), 'TestAccuracy': svm_cv.score(X_test, Y_test).round(5)}
comparison['Tree'] = {'Accuracy': tree_cv.best_score_.round(5), 'TestAccuracy': tree_cv.score(X_test, Y_test).round(5)}
comparison['KNN'] = {'Accuracy': knn_cv.best_score_.round(5), 'TestAccuracy': knn_cv.score(X_test, Y_test).round(5)}
```

```
Model           Accuracy        TestAccuracy
LogReg          0.84643         0.83333
SVM             0.84821         0.83333
Tree            0.8875          0.94444
KNN             0.84821         0.83333
```

- The model that had the highest classification accuracy was the Decision Tree Algorithm.

# Confusion Matrix

- The confusion matrix for the Decision Tree classifier was the best. The major problem were the false positive results.




TASK 10

# Conclusions

- The Decision Tree was the best algorithm for this dataset
- The orbit with the best success rates are GEO, HEO, SSO, ES-L1.
- Launches with a payload above 9000 KG have a greater chance of being successful.

# 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!