
# Winning Space Race with Data Science

Jason Doan
Sept. 5, 2022

# Outline

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

# Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

# 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. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Find answers to the problems:
- Determine how the rocket will land successfully?
- Determine the success rate of a successful landing by using different techniques.
- Determine the operating conditions needed for a successful landing.

Section 1

# Methodology

# Methodology

## Executive Summary

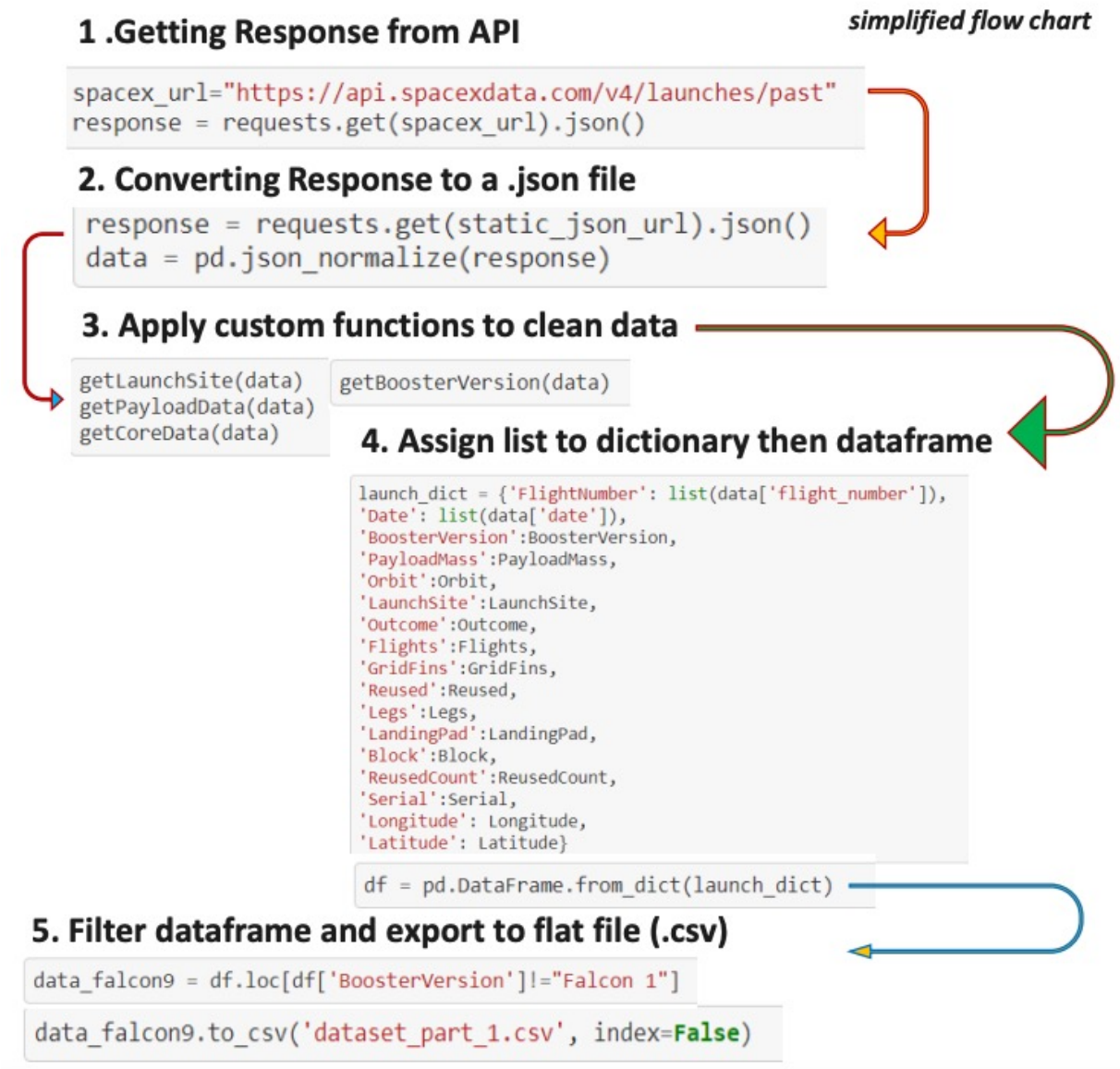
- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

# Data Collection

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

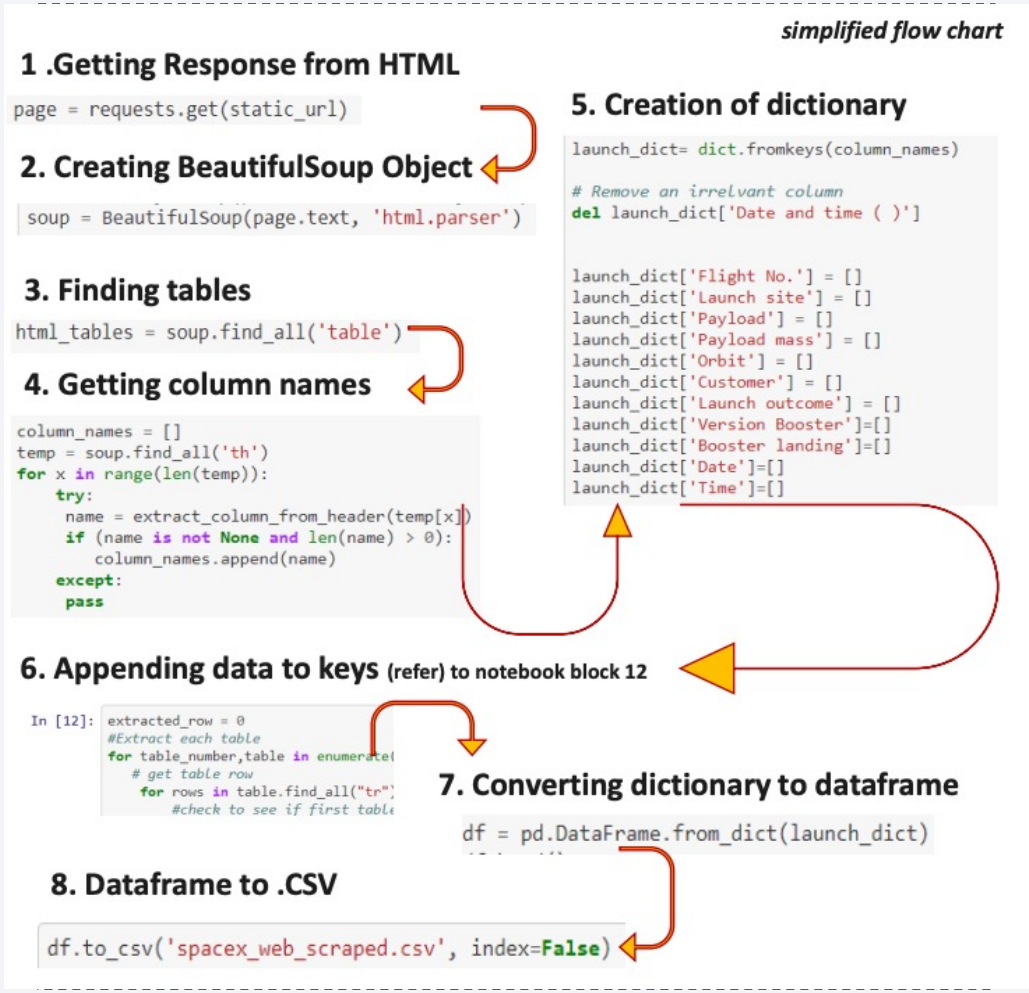
# Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- Link to the notebook: https://bit.ly/3KO79PW

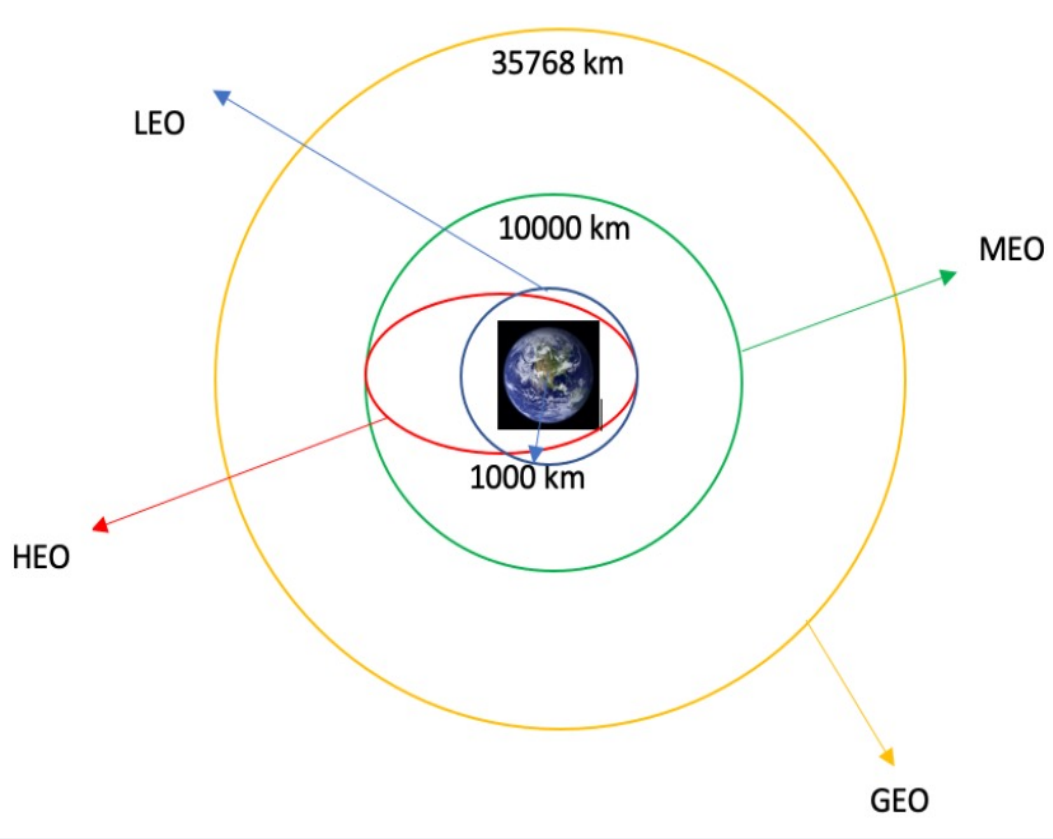


# Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- Link to the notebook: https://bit.ly/3KO79PW



# Data Wrangling

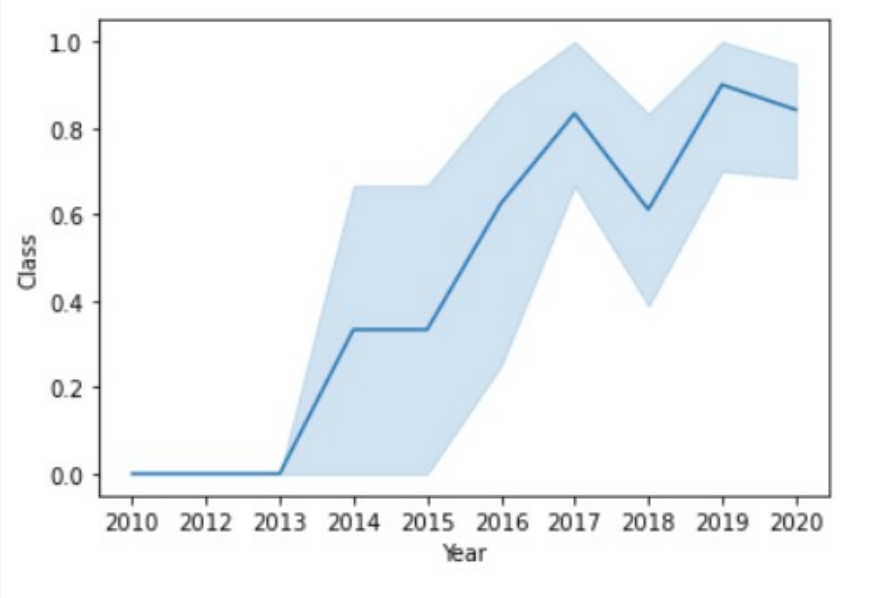


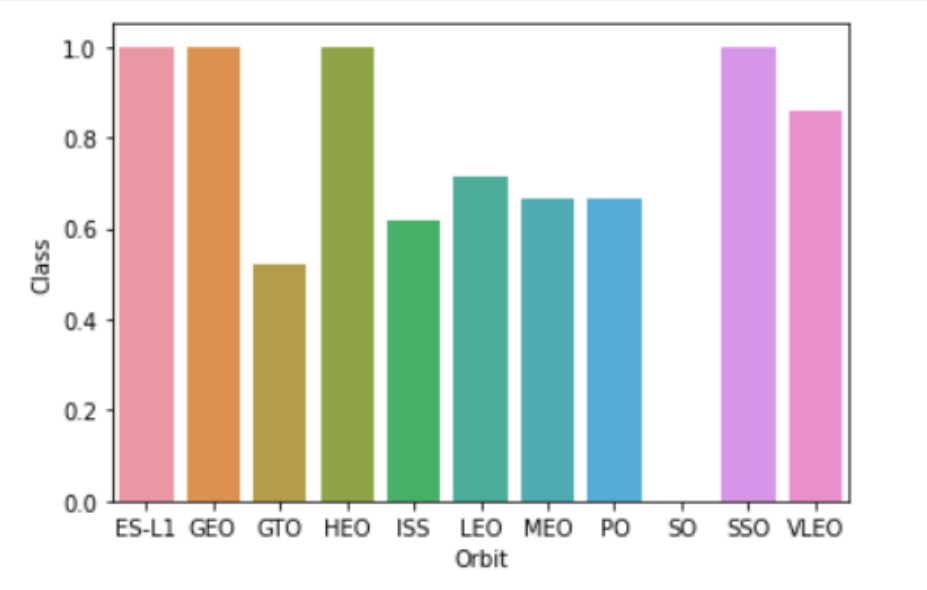

- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- Link to the notebook:

https://bit.ly/3KO79PW

# EDA with Data Visualization

- We explored the data by visualizing the relationship between 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.




- Link to the notebook: https://bit.ly/3KO79PW

# EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- Link to the notebook: https://bit.ly/3KO79PW

# Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

# Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- Link to the notebook: https://bit.ly/3KO79PW

# Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- Link to the notebook: https://bit.ly/3KO79PW

# Results

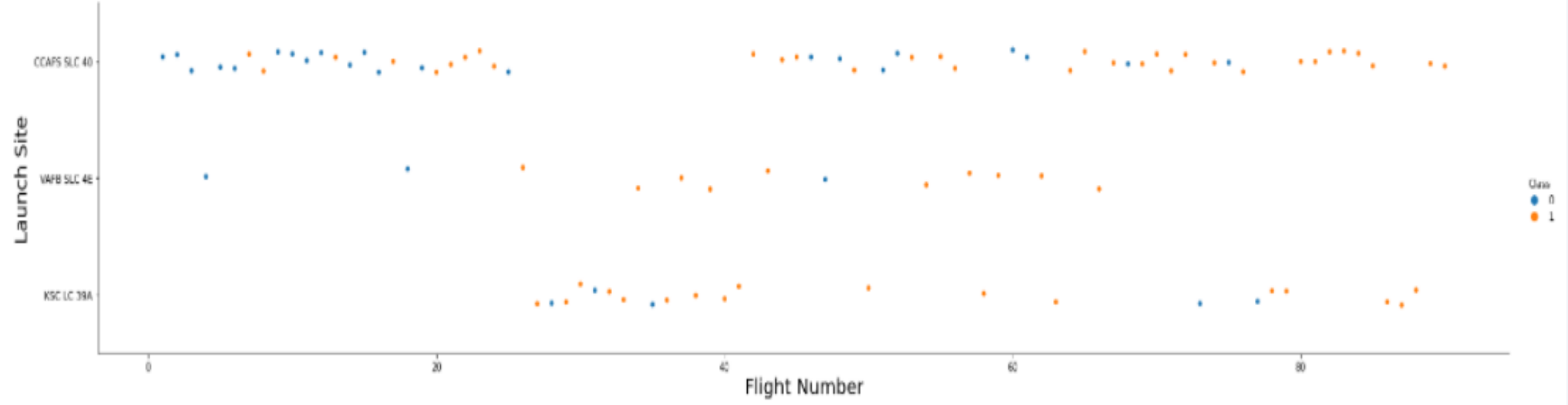
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

# Insights drawn from EDA

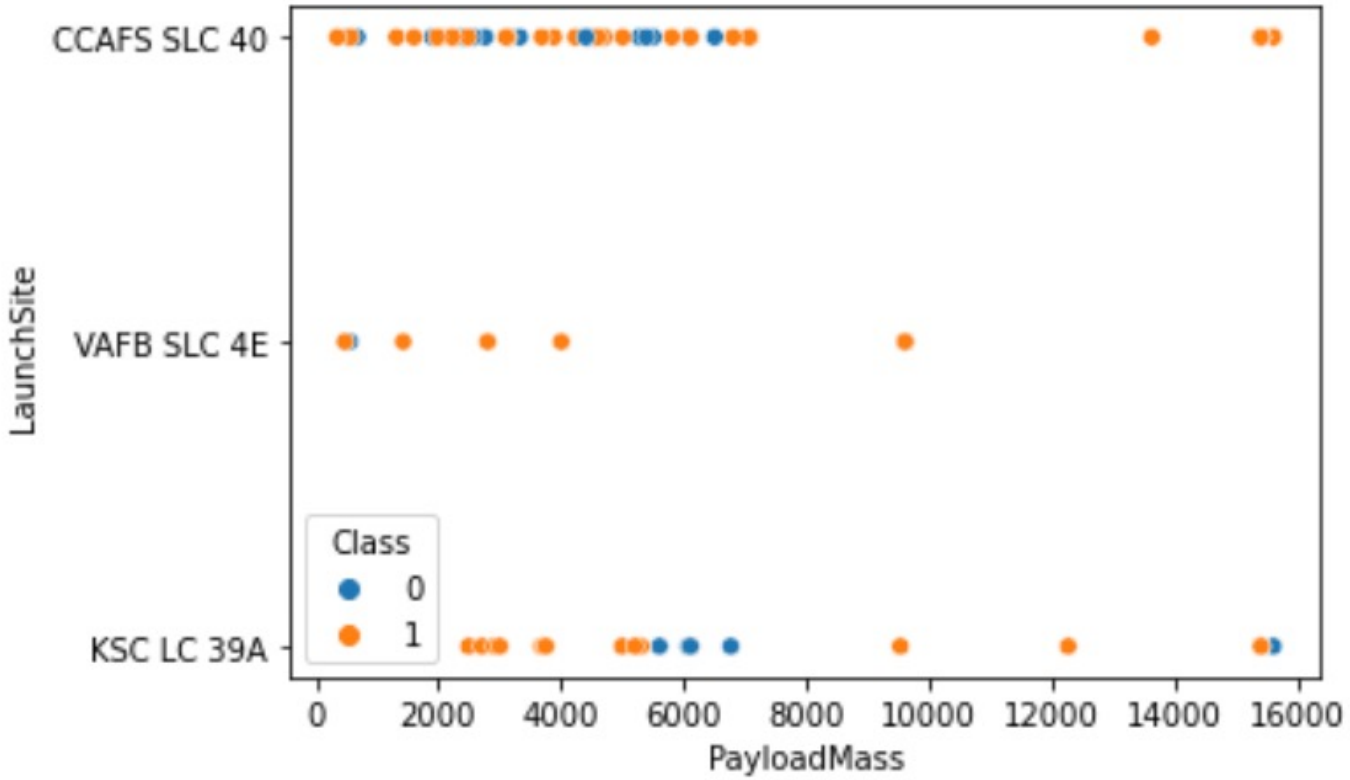
# Flight Number vs. Launch Site

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



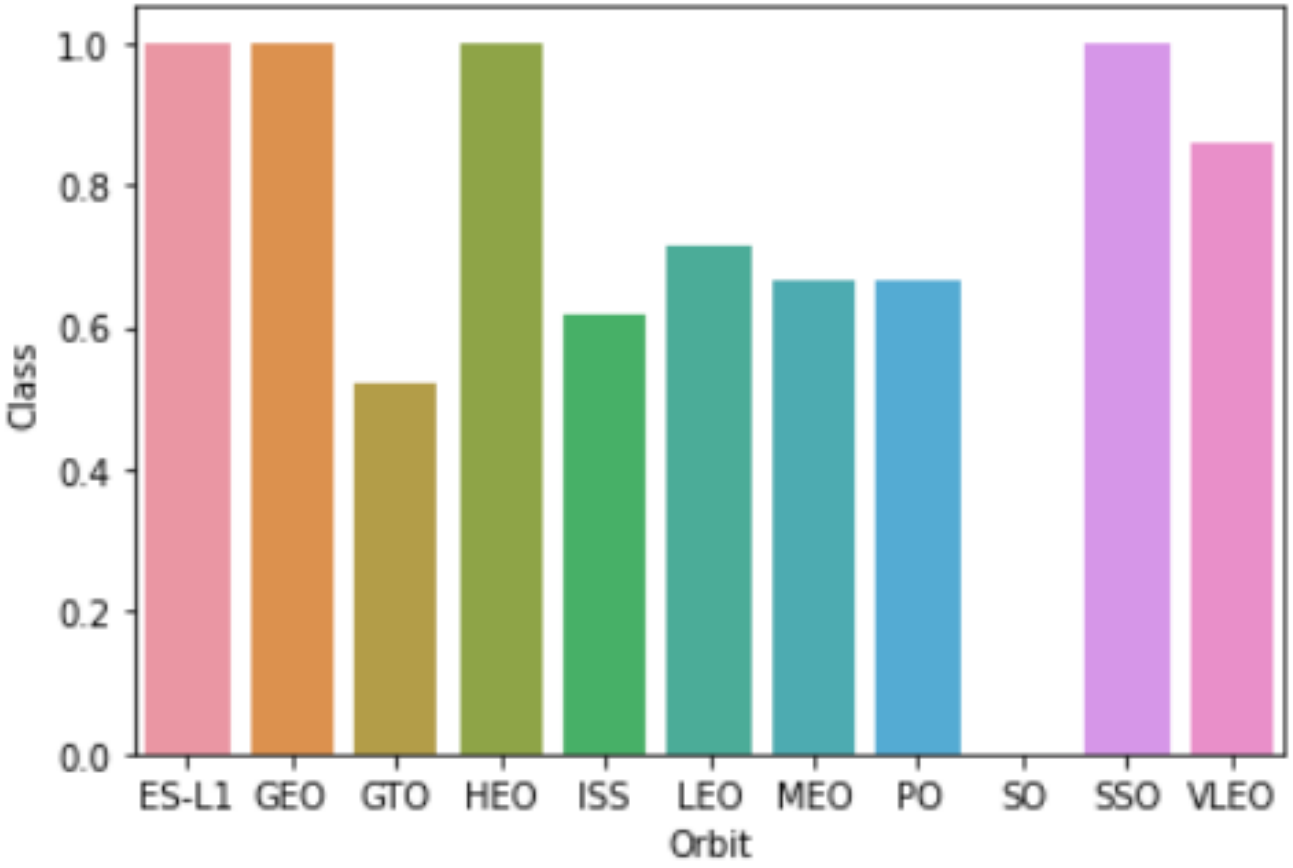
# Payload vs. Launch Site

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



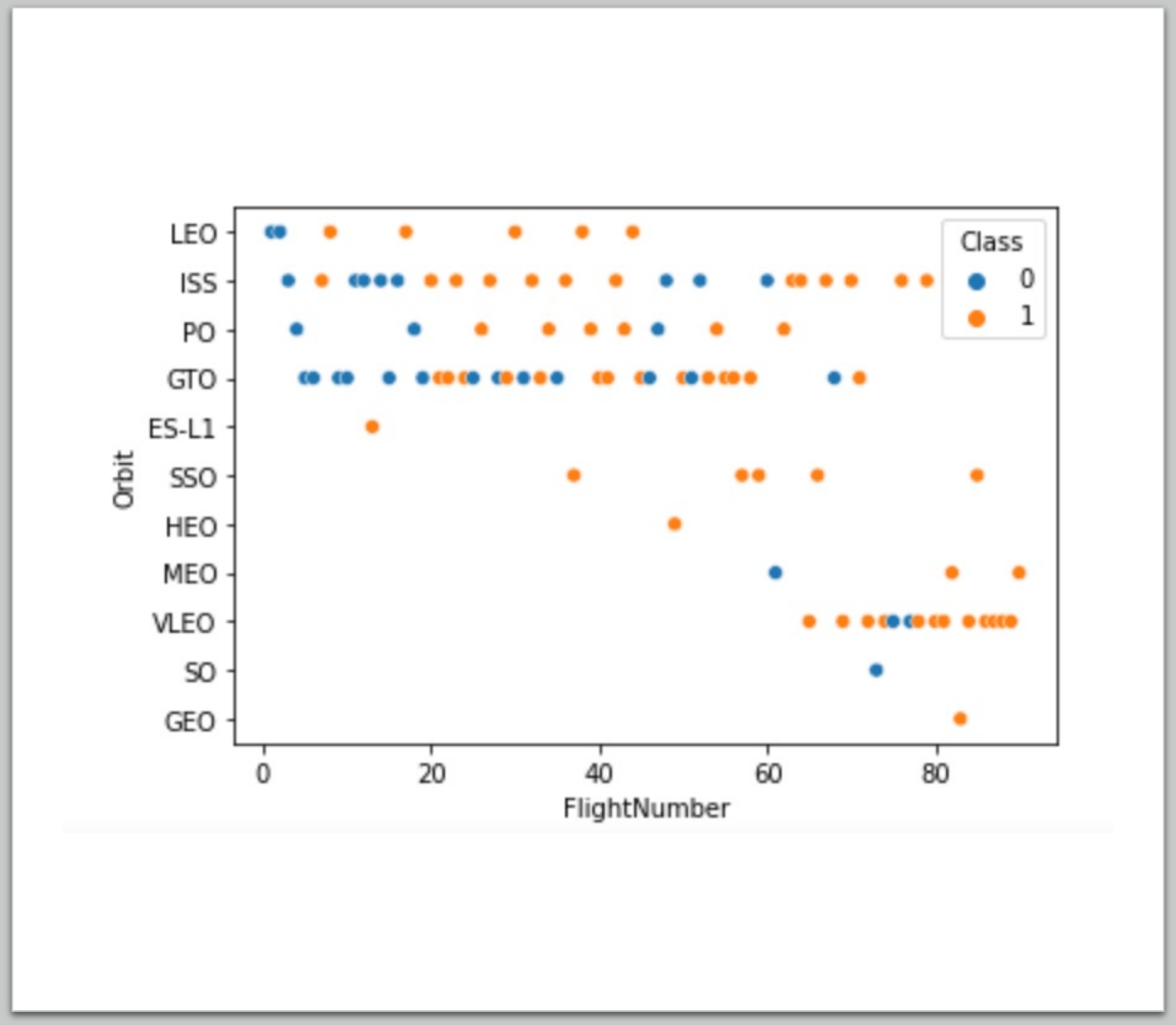
# Success Rate vs. Orbit Type

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



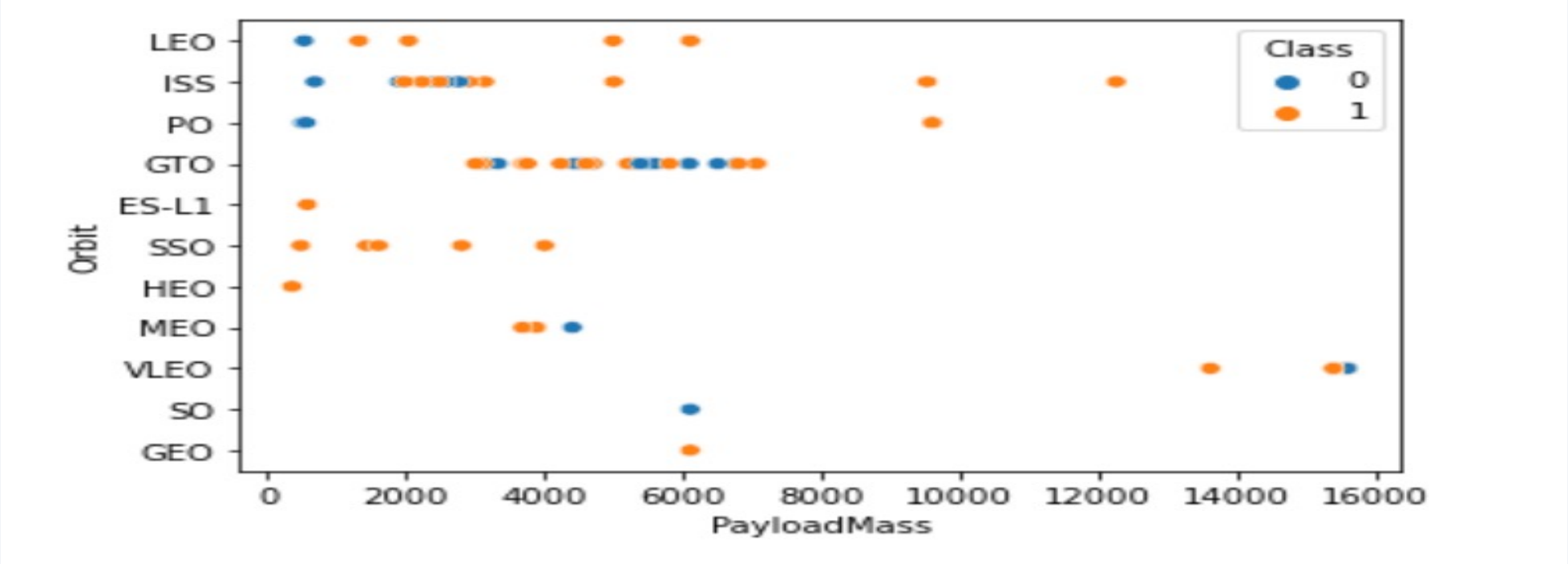
# Flight Number vs. Orbit Type

- We see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



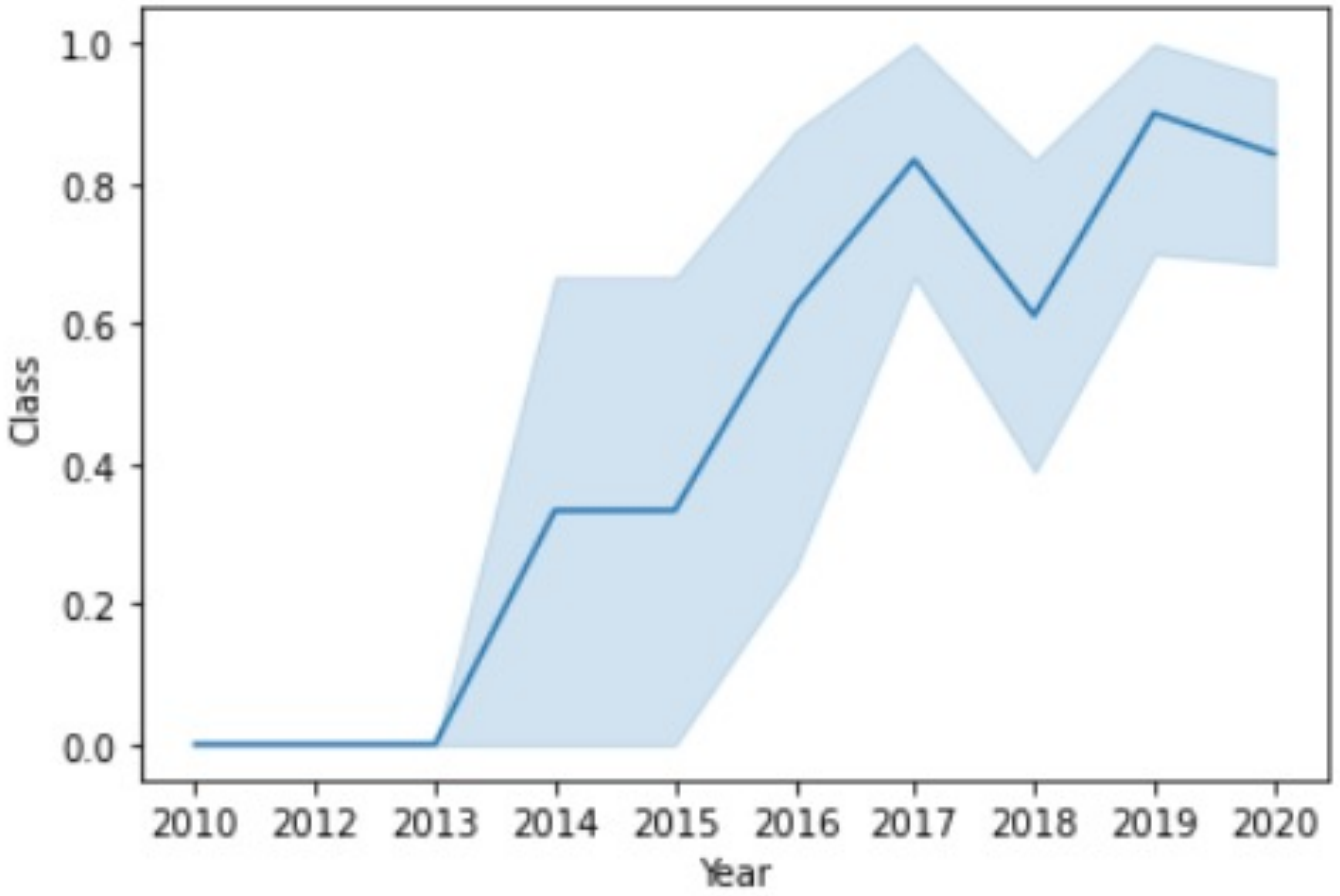
# Payload vs. Orbit Type

- As we observe, with heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.



# Launch Success Yearly Trend

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



# All Launch Site Names

- We used the key word **Unique** to show only unique launch sites from the SpaceX data.

```
In [13]: %sql select Unique(LAUNCH_SITE) from SPACEXTBL;
 * ibm_db_sa://skb43131:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:3245
9/bludb
   sqlite:///my_data1.db
Done.
```

Out[13]:

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

# Launch Site Names Begin with 'CCA'

```
In [14]: %sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
 * ibm_db_sa://skb43131:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
   sqlite:///my_data1.db
Done.
```

Out[14]:

| launch_site |
|---|
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |

- We used the query above to display 5 records where launch sites begin with `CCA`

# Total Payload Mass

- We calculated the total payload carried by boosters from NASA is 45596 using the query below

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

In [34]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```

```
 * ibm_db_sa://skb43131:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:3245
9/bludb
   sqlite:///my_data1.db
Done.
```

Out[34]:

| total_payload_mass |
|---|
| 45596 |

# Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 is 2928 using the query below.

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

In [35]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

```
 * ibm_db_sa://skb43131:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:3245
9/bludb
   sqlite:///my_data1.db
Done.
```

Out[35]:

| average_payload_mass |
|---|
| 2928 |

# First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was December 22,2015 using query below.

In [36]:
```
%sql SELECT MIN(DATE) AS FIRST_SUCCESSFUL_GROUND_LANDING FROM SPACEXTBL \
    WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

```
 * ibm_db_sa://skb43131:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:3245
9/bludb
   sqlite:///my_data1.db
Done.
```

Out[36]:

| first_successful_ground_landing |
|---|
| 2015-12-22 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

- We 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 using query below.

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

In [18]:
```
%sql select BOOSTER_VERSION from SPACEXTBL where LANDING__OUTCOME='Success (drone ship)'
and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

```
 * ibm_db_sa://skb43131:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:3245
9/bludb
   sqlite:///my_data1.db
Done.
```

Out[18]:

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

# Total Number of Successful and Failure Mission Outcomes

- We use the following query below to list the total number of successful and failure mission outcomes.

```
In [33]: %sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
 * ibm_db_sa://skb43131:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
   sqlite:///my_data1.db
Done.
```

Out[33]:

| mission_outcome | total_number |
|---|---|
| Failure (in flight) | 1 |
| Success | 99 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function using query below.

```
In [25]: %sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_)
         FROM SPACEXTBL)
```

```
 * ibm_db_sa://skb43131:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:3245
9/bludb
   sqlite:///my_data1.db
Done.
```

Out[25]:

| boosterversion |
|---|
| 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 the following query to conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

In [32]:
```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL \
    WHERE (LANDING__OUTCOME = 'Failure (drone ship)') AND (EXTRACT(YEAR FROM DATE) = '2015');
```

```
 * ibm_db_sa://skb43131:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:3245
9/bludb
   sqlite:///my_data1.db
Done.
```

Out[32]:

| booster_version | launch_site |
|---|---|
| F9 v1.1 B1012 | CCAFS LC-40 |
| F9 v1.1 B1015 | CCAFS LC-40 |

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

- We used 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 2017-03-20.
- We used the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

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

In [31]:
```
%sql SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL \
    WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
    GROUP BY LANDING__OUTCOME \
    ORDER BY TOTAL_NUMBER DESC;
```

```
 * ibm_db_sa://skb43131:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:3245
9/bludb
   sqlite:///my_data1.db
Done.
```

Out[31]:

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

Section 4

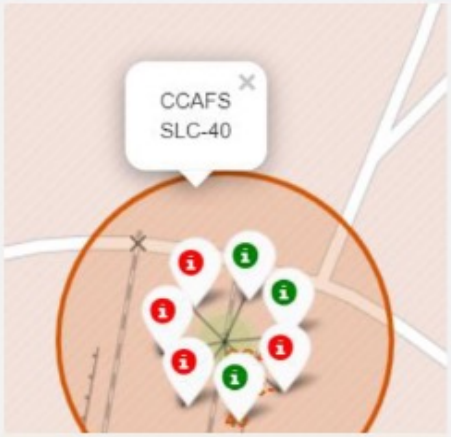
# Launch Sites Proximities Analysis

# All launch sites global map markers

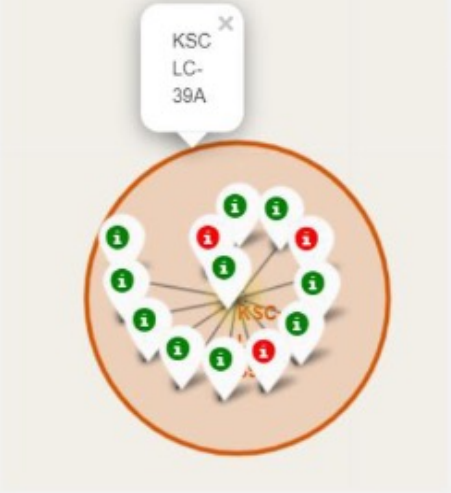
Out[13]:


According to figure, Space X launch sites are located on the coast of California and Florida states in the US.

# Markers showing launch sites with color labels







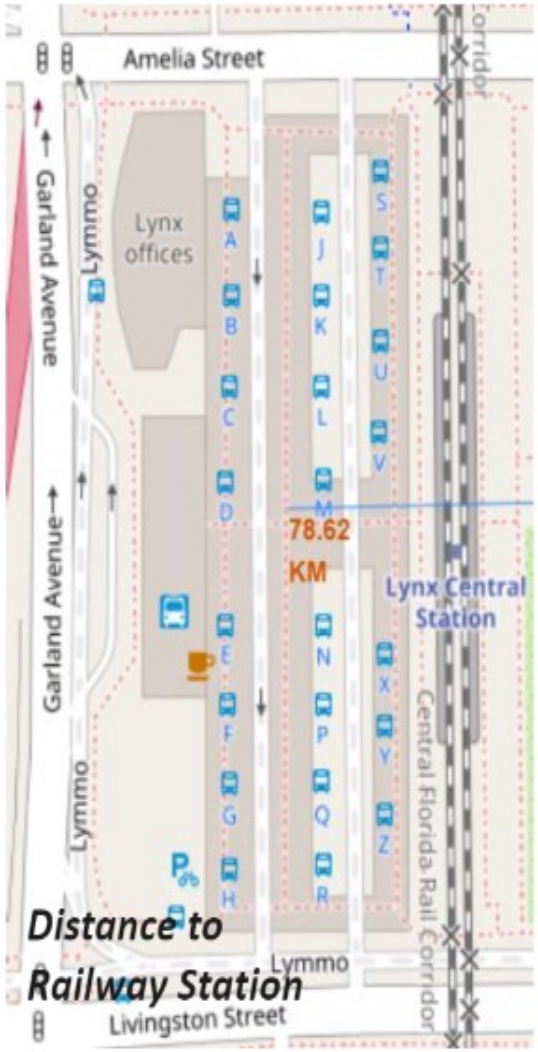

***Florida Launch Sites***

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


***California Launch Site***

37

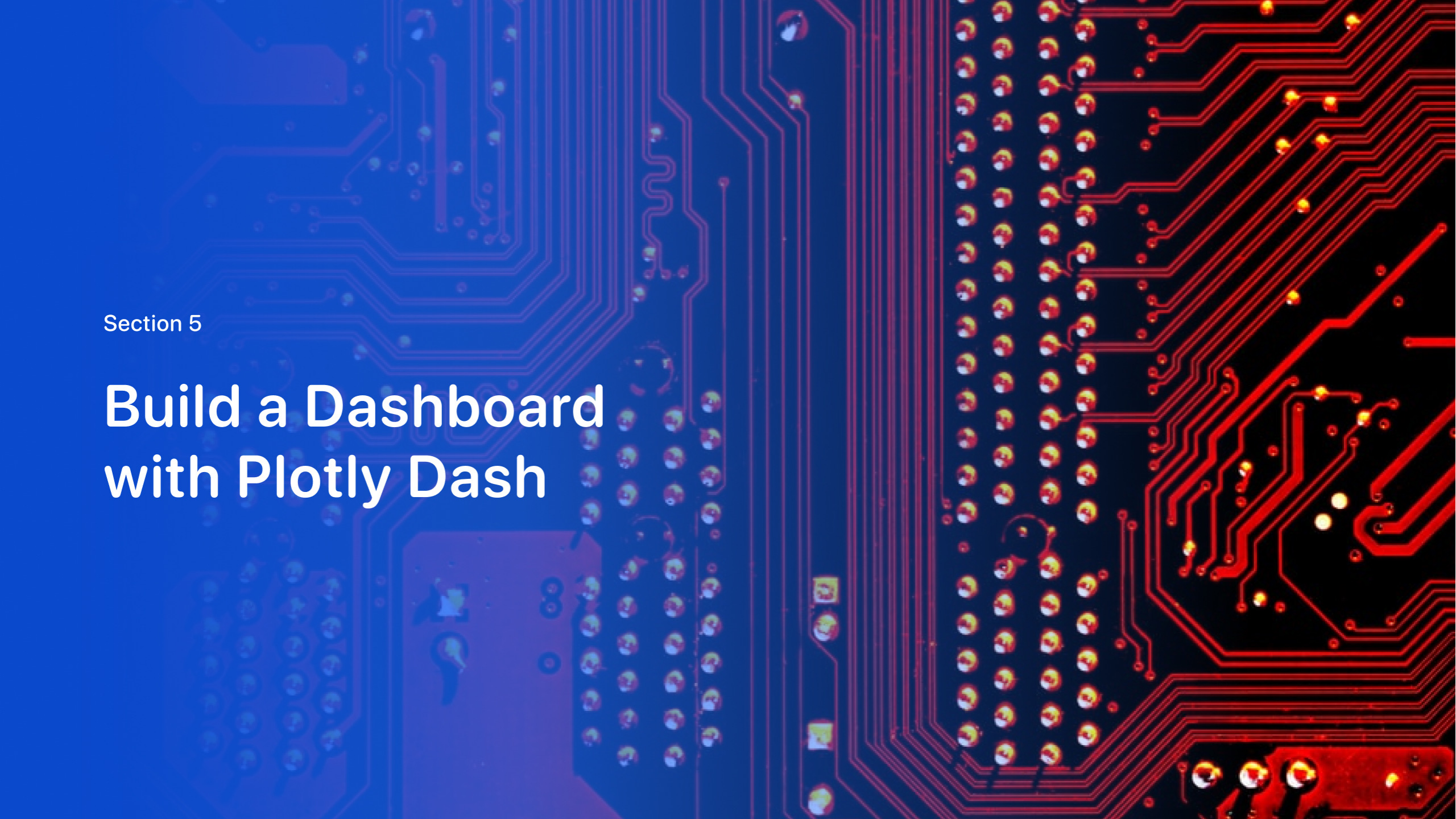
# Launch Site distance to landmarks







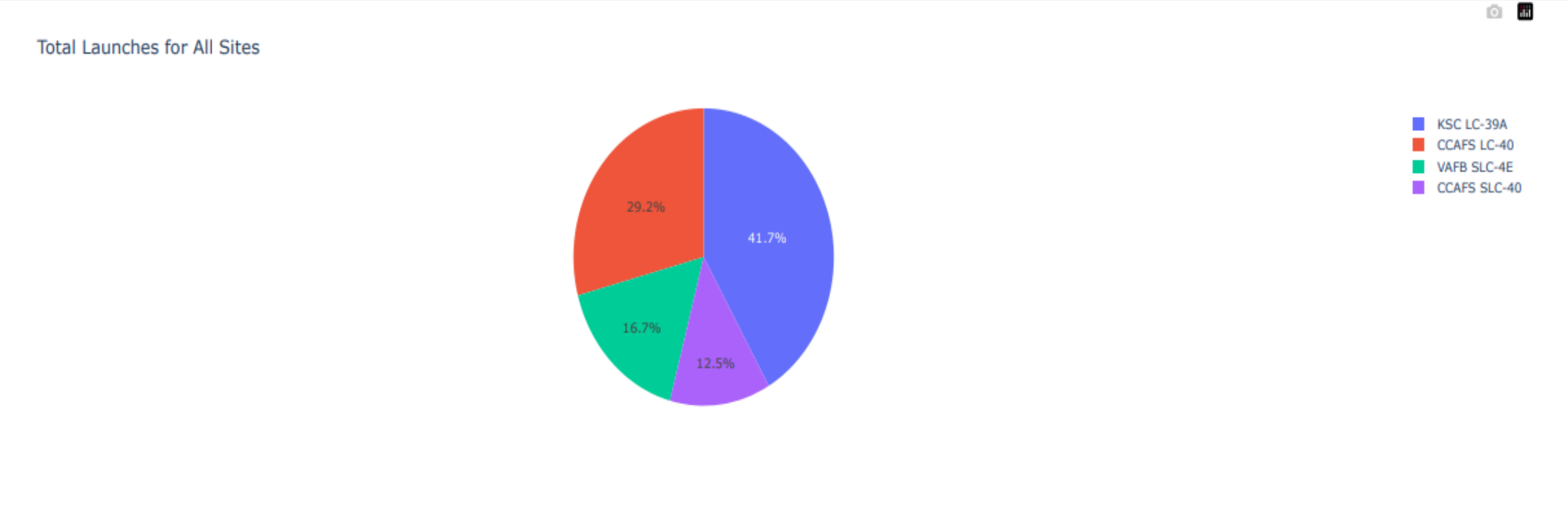

- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes



Section 5

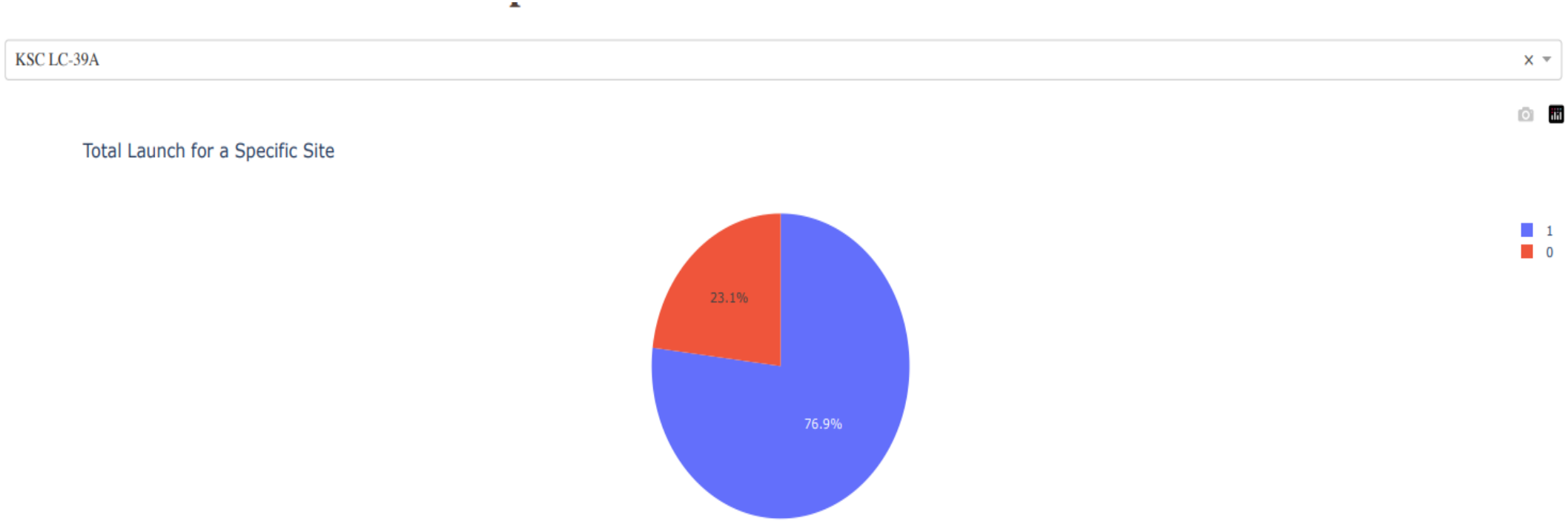
# Build a Dashboard with Plotly Dash

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



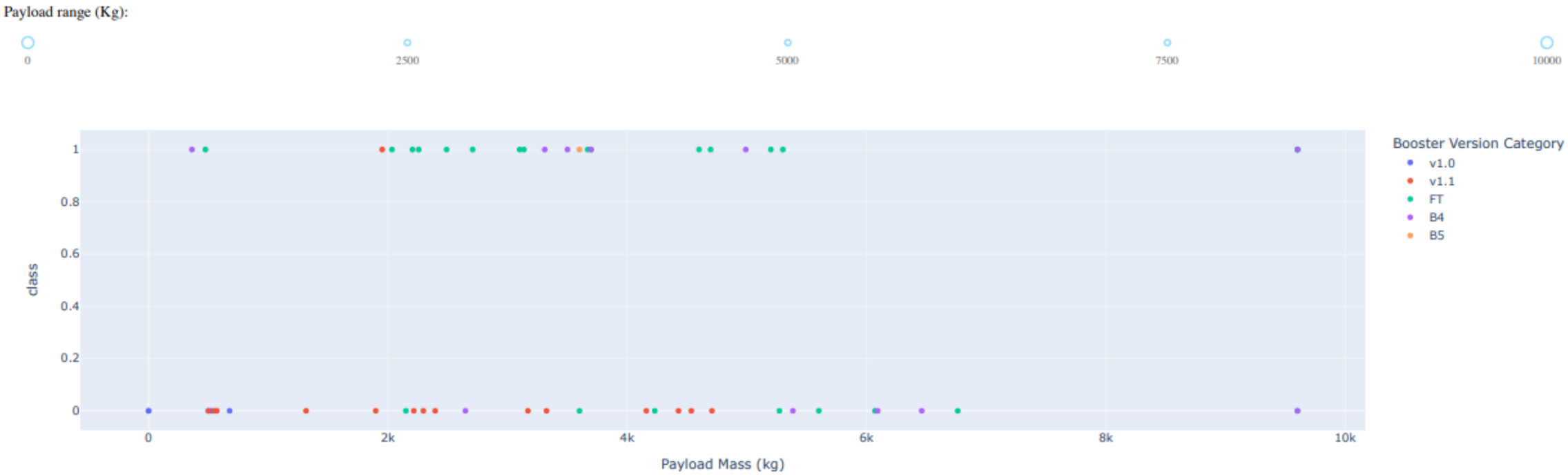

According to pie-chart, KSC LC-39A launch's site has the most successful launch.

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




KSC LC-39A has 76.9% successful launch rate vs 23.1% failure rate.

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



Section 6

# Predictive Analysis (Classification)

# Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

In [45]:
```
algorithms = ['Logistic Regression', 'Support Vector Machine', 'Decision Tree', 'K Nearest Neighbours']

scores = [lr_score, svm_score, tree_score, knn_score]

best_scores = [lr_best_score, svm_best_score, tree_best_score, knn_best_score]

column_names = ['Algorithm', 'Accuracy Score', 'Best Score']
```

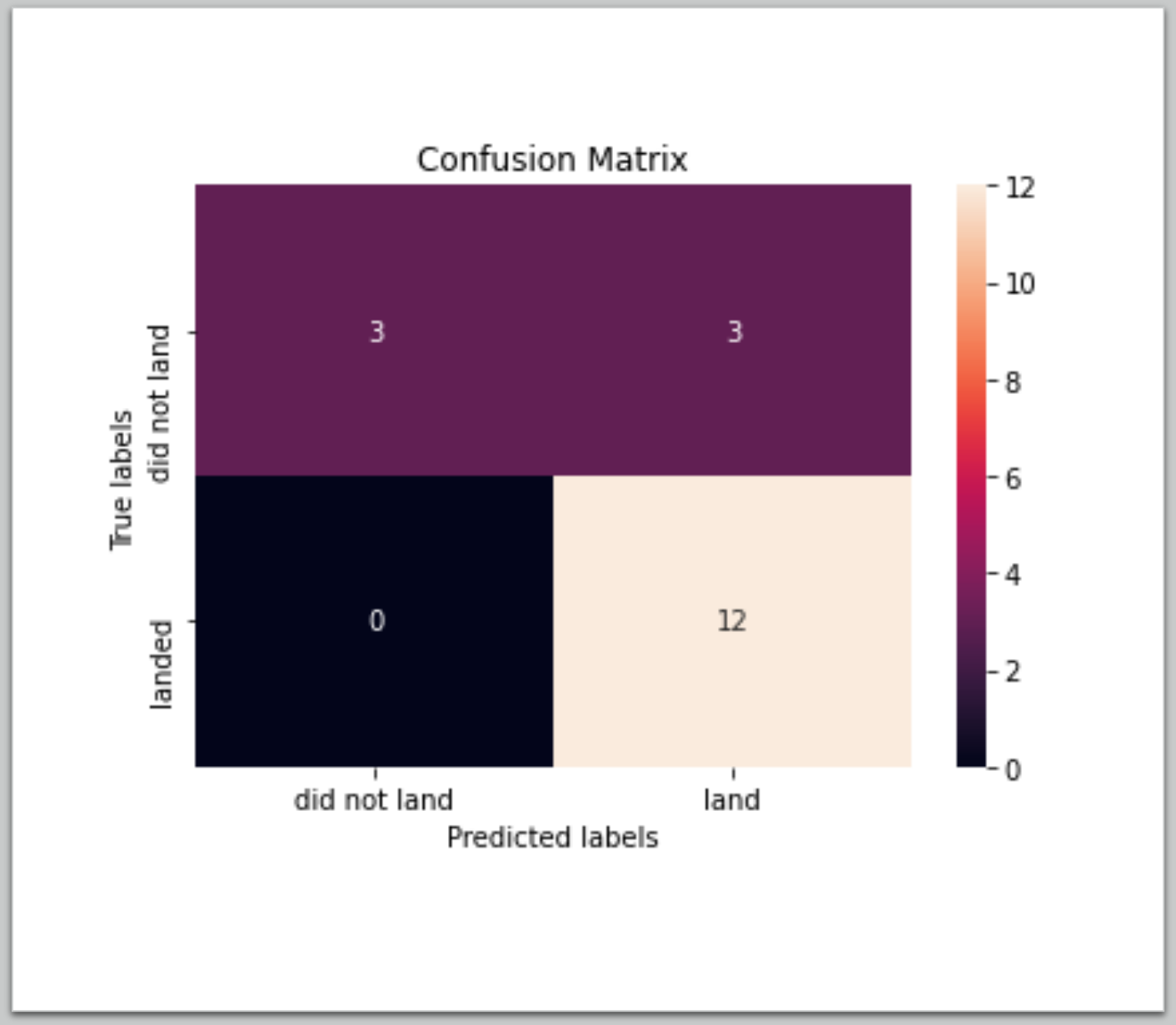
In [46]:
```
df = pd.DataFrame(list(zip(algorithms, scores, best_scores)),columns = column_names)
df
```

Out[46]:

| | Algorithm | Accuracy Score | Best Score |
|---|---|---|---|
| 0 | Logistic Regression | 0.833333 | 0.846429 |
| 1 | Support Vector Machine | 0.833333 | 0.848214 |
| 2 | Decision Tree | 0.944444 | 0.876786 |
| 3 | K Nearest Neighbours | 0.833333 | 0.889286 |

# Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives.



# Conclusions

- The success of a mission can be explained by several factors such as the launch site, the orbit and especially the number of previous launches. Indeed, we can assume that there has been a gain in knowledge between launches that allowed to go from a launch failure to success.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- Launch success rate started to increase from 2013 till 2020.
- KSC LC-39A had the most successful launches out of all the sites.
- We choose the Decision Tree Algorithm as the best model even if the test accuracy between all the models used is identical. We choose Decision Tree Algorithm because it has a better train accuracy.

# Thank you!