IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Kamalkishor Roj>
<27th January 2026>

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection with API
  - Data Collection by Web Scraping
  - Data Wrangling
  - EDA with SQL
  - EDA with Data Visualization
  - Interactive Visuals by Folium
  - ML Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive visuals
  - Predictive Results

Introduction

- Project background and context

  Space X advertises Falcon 9 rockets on its website with cost 62million dollars, other providers cost upwards 165 million dollars each, much of the saving is because SpaceX can reuse the first stage. Therefore if we determine if first stage will land, we can determine the cost of a launch. This info can be used if and alternative company wants to bid against SpaceX for a rocket launch. The goal of this project is to create a ML pipeline to predict if the first stage will land or not.

- Problems you want to find answers

  -what factors determine if the rocket will land successfully?

  -The interaction amongst various features that determine the success rate of a successful landing.

  -What operation conditions needs to be in place to ensure a successful landing program.

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 is https://github.com/kamalkishor-roj/IBM-Data-Science-Capstone-SpaceX-Project/blob/main/Data_collection_BY_API.ipynb

In [46]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [47]:
```
response = requests.get(spacex_url)
```

Check the content of the response

In [48]:
```
print(response.content)
```

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:

In [49]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_
```

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

In [50]:
```
response=requests.get(static_json_url)
```

In [51]:
```
response.status_code
```

Out[51]: 200

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

In [52]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

In [53]:
```
# Get the head of the dataframe
data.head()
```

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
- The link to notebook is https://github.com/kamalkishor-roj/IBM-Data-Science-Capstone-SpaceX-Project/blob/main/Data_collection_by_WebScraping.ipynb

In [20]:

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

headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
                  "AppleWebKit/537.36 (KHTML, like Gecko) "
                  "Chrome/91.0.4472.124 Safari/537.36"
}
```

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

In [28]:
```
# use requests.get() method with the provided static_url and headers
# assign the response to a object
response = requests.get(static_url, headers=headers)
```

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

In [29]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
```

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

In [30]:
```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

In [31]:
```
# 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')
```

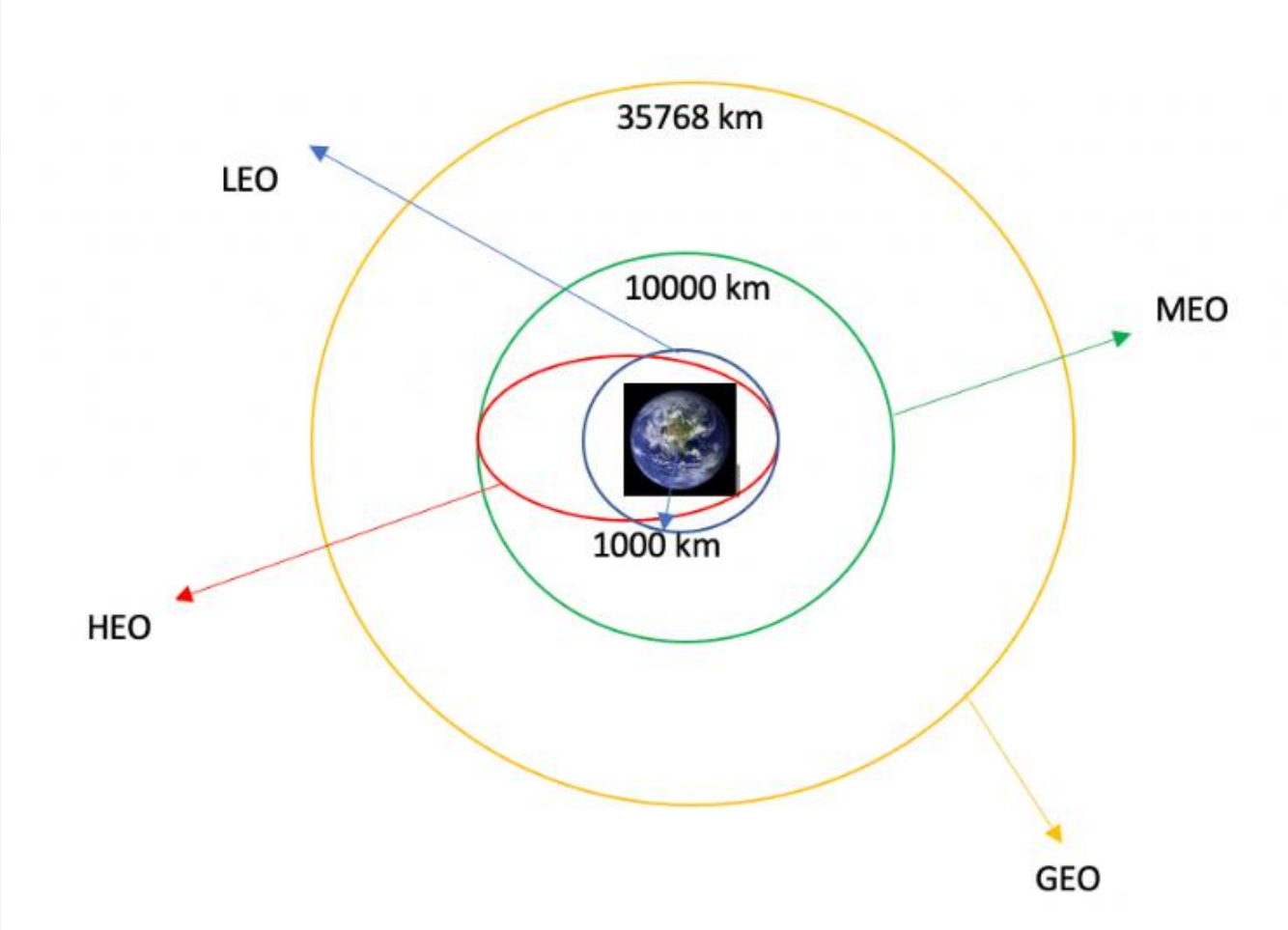
Starting from the third table is our target table contains the actual launch records.

In [32]:
```
# Check if tables were found
print(f"Number of tables found: {len(html_tables)}")
```

```
Number of tables found: 25
```

In [33]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

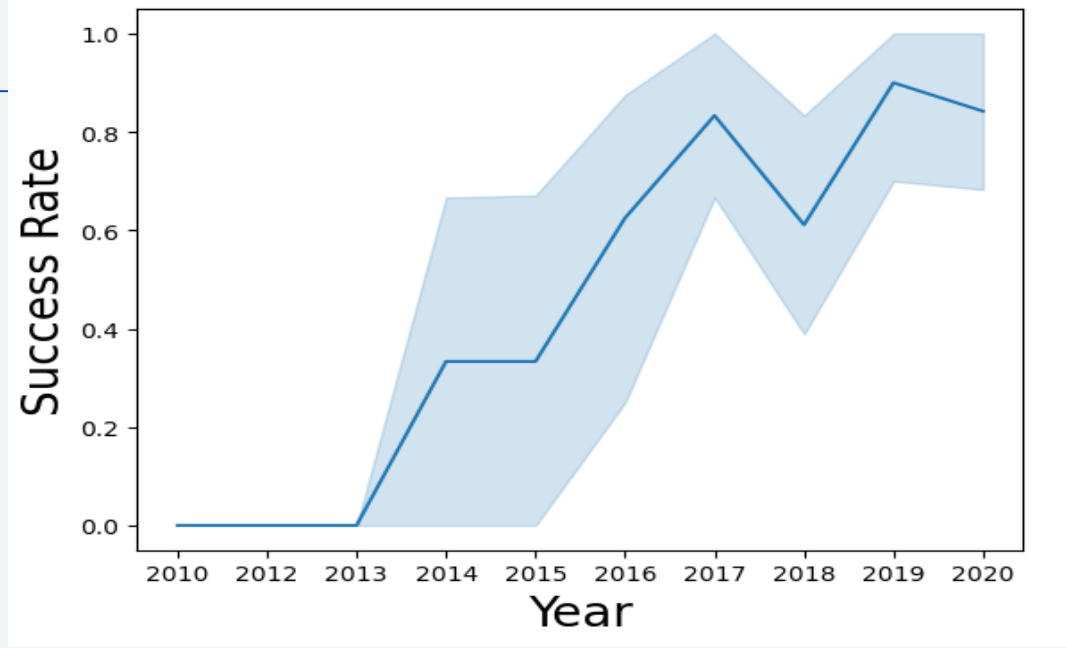
Data Wrangling

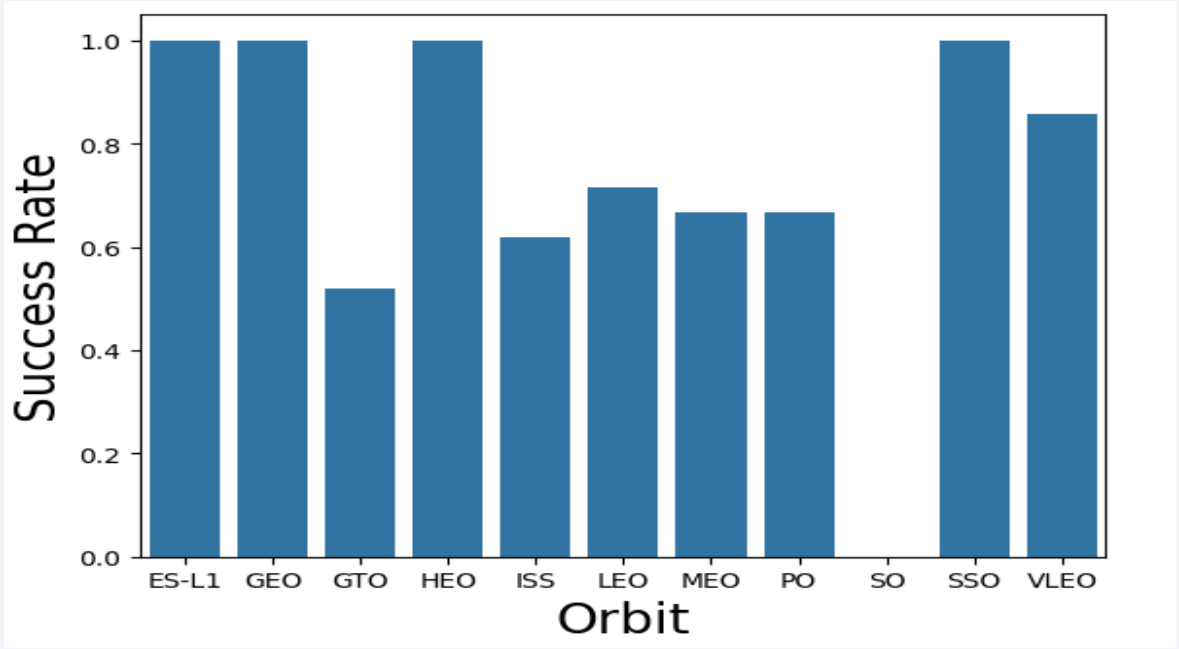


- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbit.
- We created landing outcome label from outcome column and exported the result to csv.
- The link to the notebook is https://github.com/kamalkishor-roj/IBM-Data-Science-Capstone-SpaceX-Project/blob/main/Data%20wrangling.ipynb

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





The link to the notebook is https://github.com/kamalkishor-roj/IBM-Data-Science-Capstone-SpaceX-Project/blob/main/EDA_Data_Visualization.ipynb

EDA with SQL

- We loaded the SpaceX dataset into PostgreSQL dataset 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.

- The link to notebook is https://github.com/kamalkishor-roj/IBM-Data-Science-Capstone-SpaceX-Project/blob/main/EDA_BY_SQL.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, add 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 features 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 distance between a launch site to its proximities. We answered some questions for instance:

  -Are launch site near railways, highways and coastline.

  -Do launch sites keep certain distance away from cities

Build a Dashboard with Plotly Dash

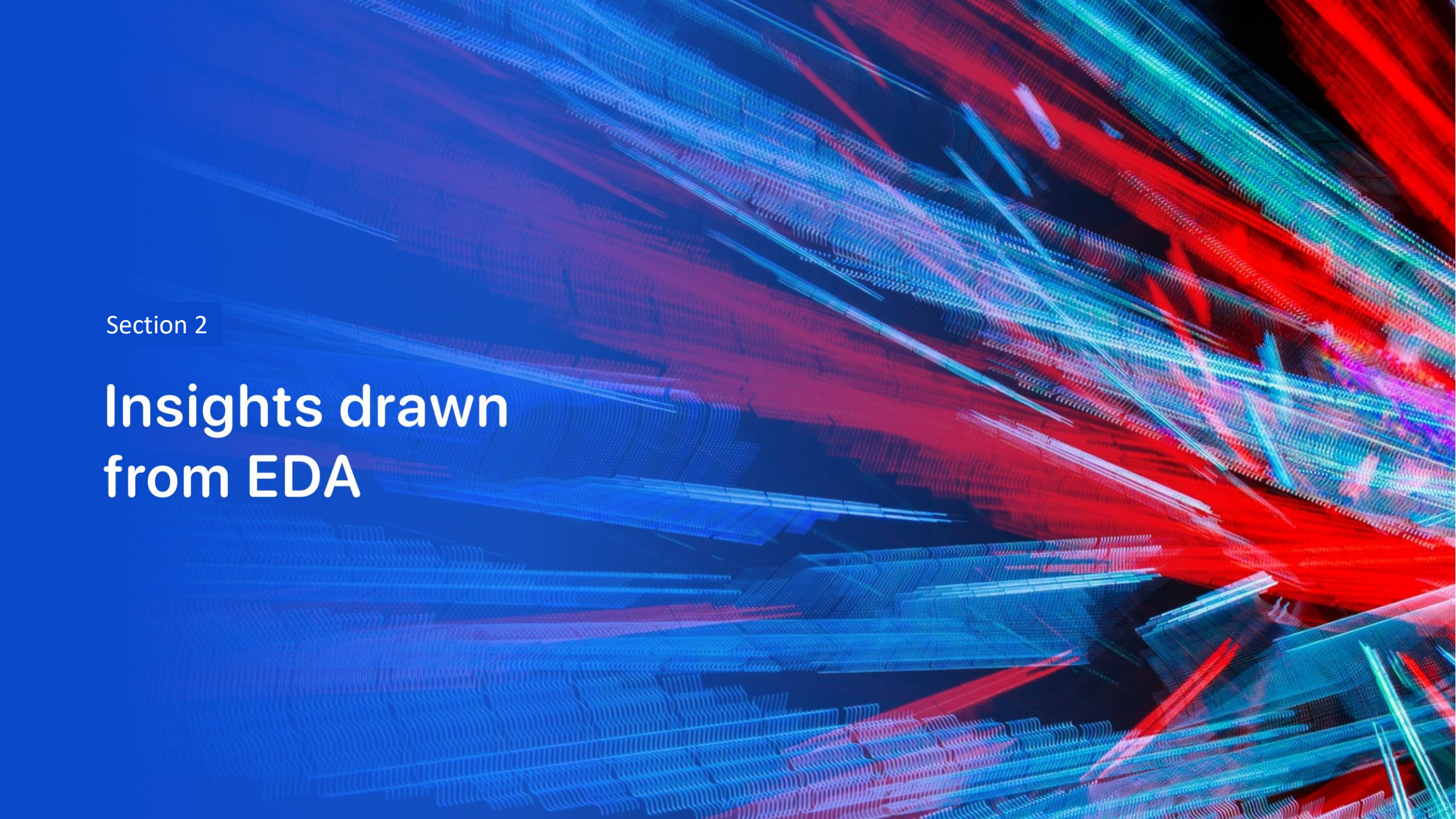
- We built an interactive dashboard with Plotly dash.
- We plotted pie charts showing the total launches by a certain sites.
- We plotted scatter graph showing the relationship with outcome and payload mass(KG) for the different booster versions.
- The link to the notebook is https://github.com/kamalkishor-roj/IBM-Data-Science-Capstone-SpaceX-Project/blob/main/spacex-dash-app.py

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed data, split data into training and testing dataset.
- We built different ML model and tuned different hyperparatmeter using GridSearchCv.
- We used accuracy as the metric for our model, improved the model using features engineering and algorithm tunning.
- We found the best performing classification model.
- The link to the notebook is https://github.com/kamalkishor-roj/IBM-Data-Science-Capstone-SpaceX-Project/blob/main/Machine_Learning_Prediction.ipynb

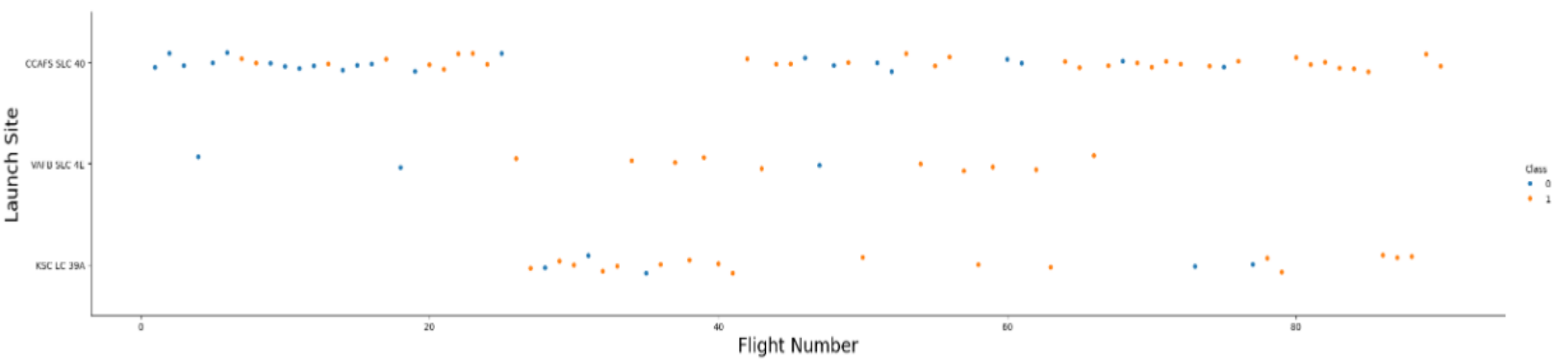
Results

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


Section 2
Insights drawn from EDA

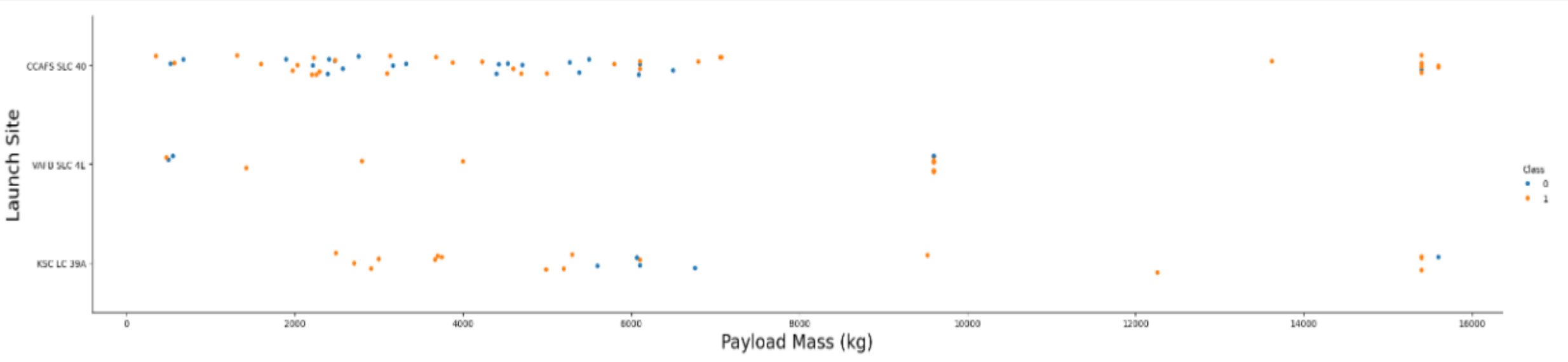
Flight Number vs. Launch Site

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



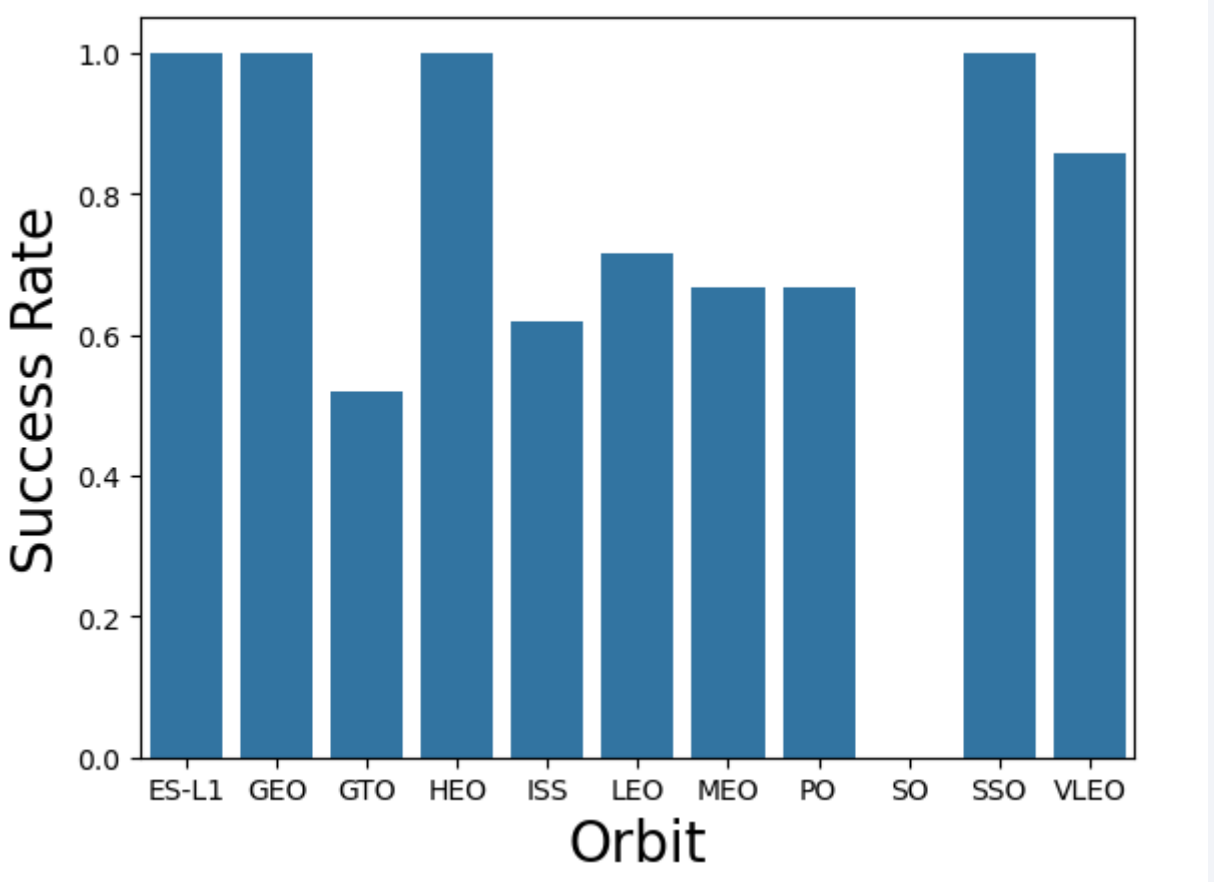
Payload vs. Launch Site

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



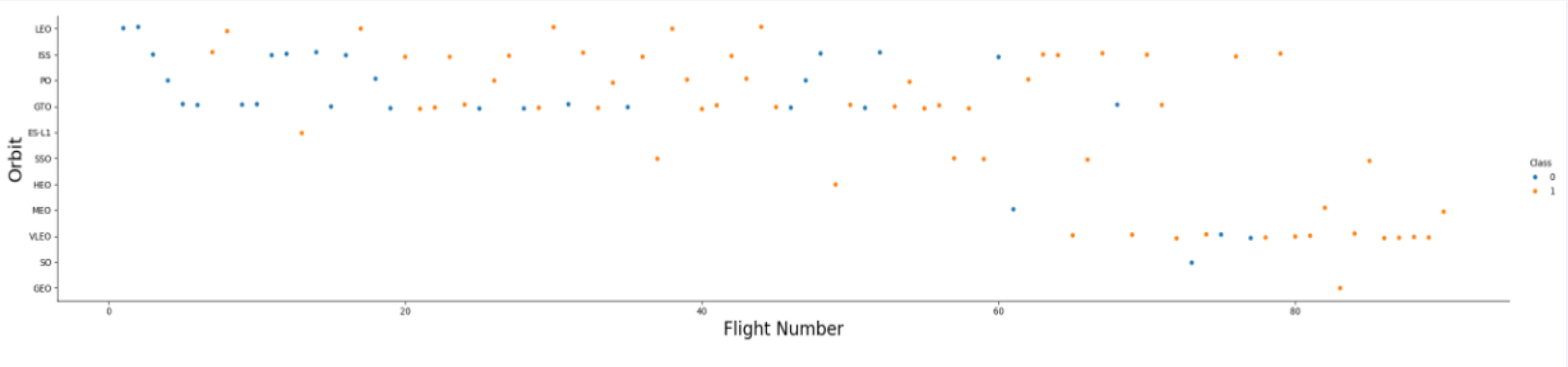
Success Rate vs. Orbit Type

- From the plot, we can see that ES-I1, GEO, HEO, SSO, VLEO had the most success rate.



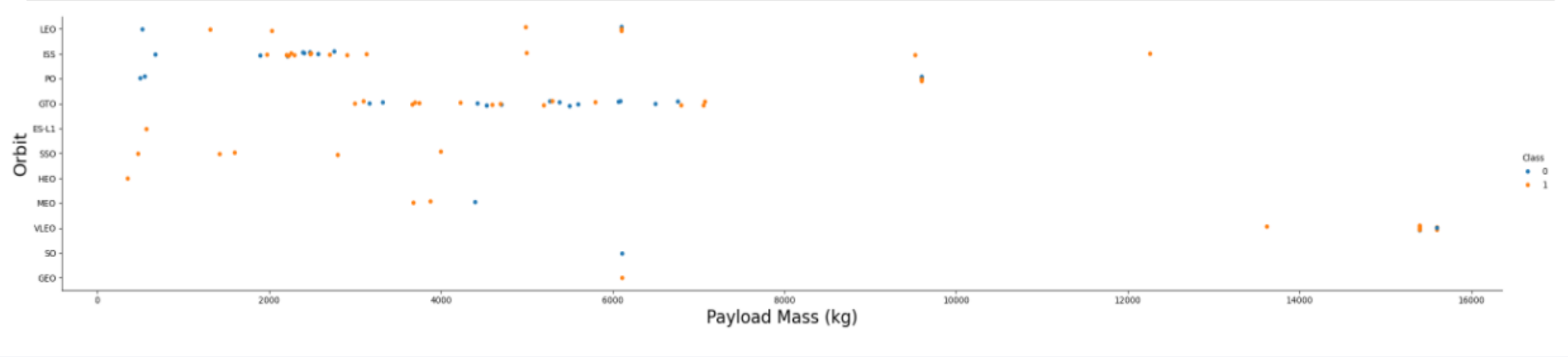
Flight Number vs. Orbit Type

- 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.



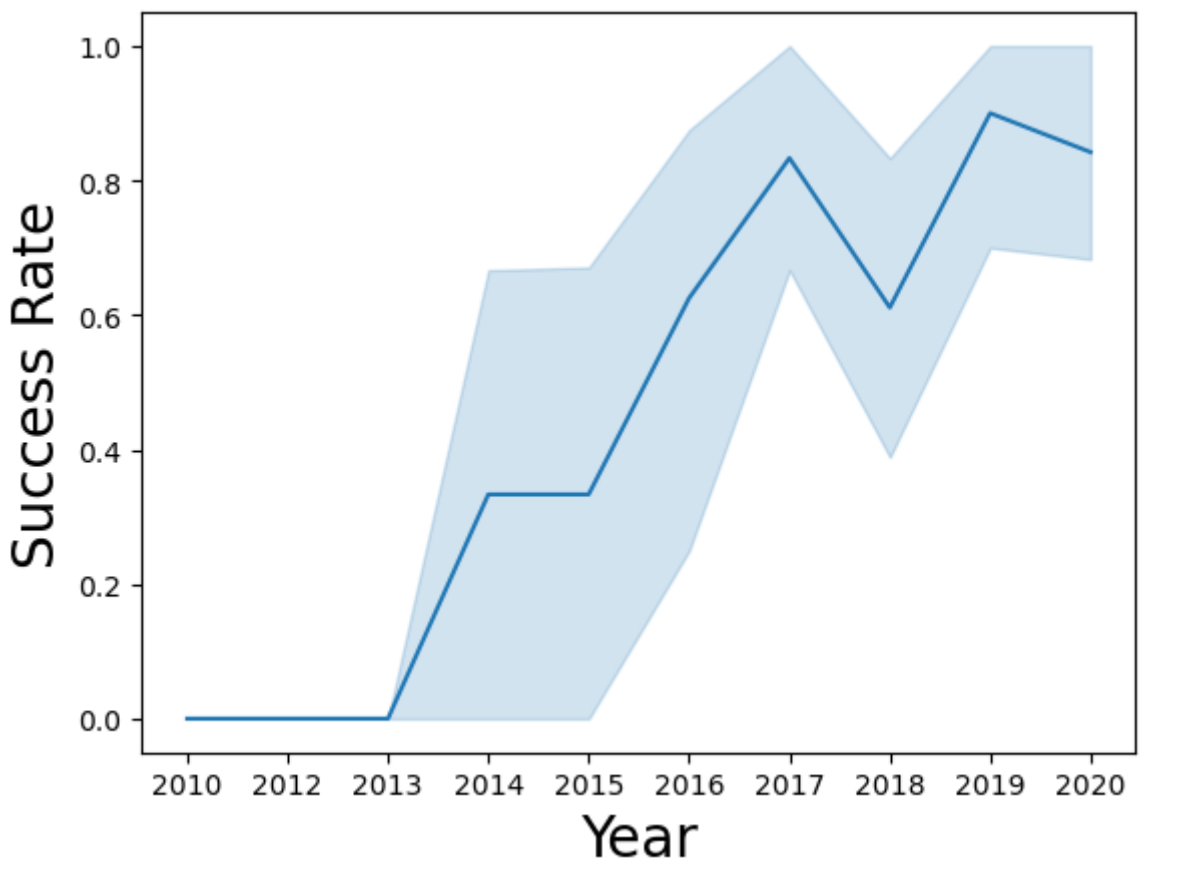
Payload vs. Orbit Type

- We can observe that with heavy payload, 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

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

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

In [31]:
```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

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

Out[31]:

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

Launch Site Names Begin with 'CCA'

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

In [32]:
```
%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Out[32]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	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)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- 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 as 45596 using the query below.

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

```
In [33]: %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)';
```

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

Out[33]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- We 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 [34]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1';
 * sqlite:///my_data1.db
Done.
Out[34]: AVG(PAYLOAD_MASS__KG_)
         2928.4
```

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015.

Task 5

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

Hint:Use min function

In [35]:
```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)';
```

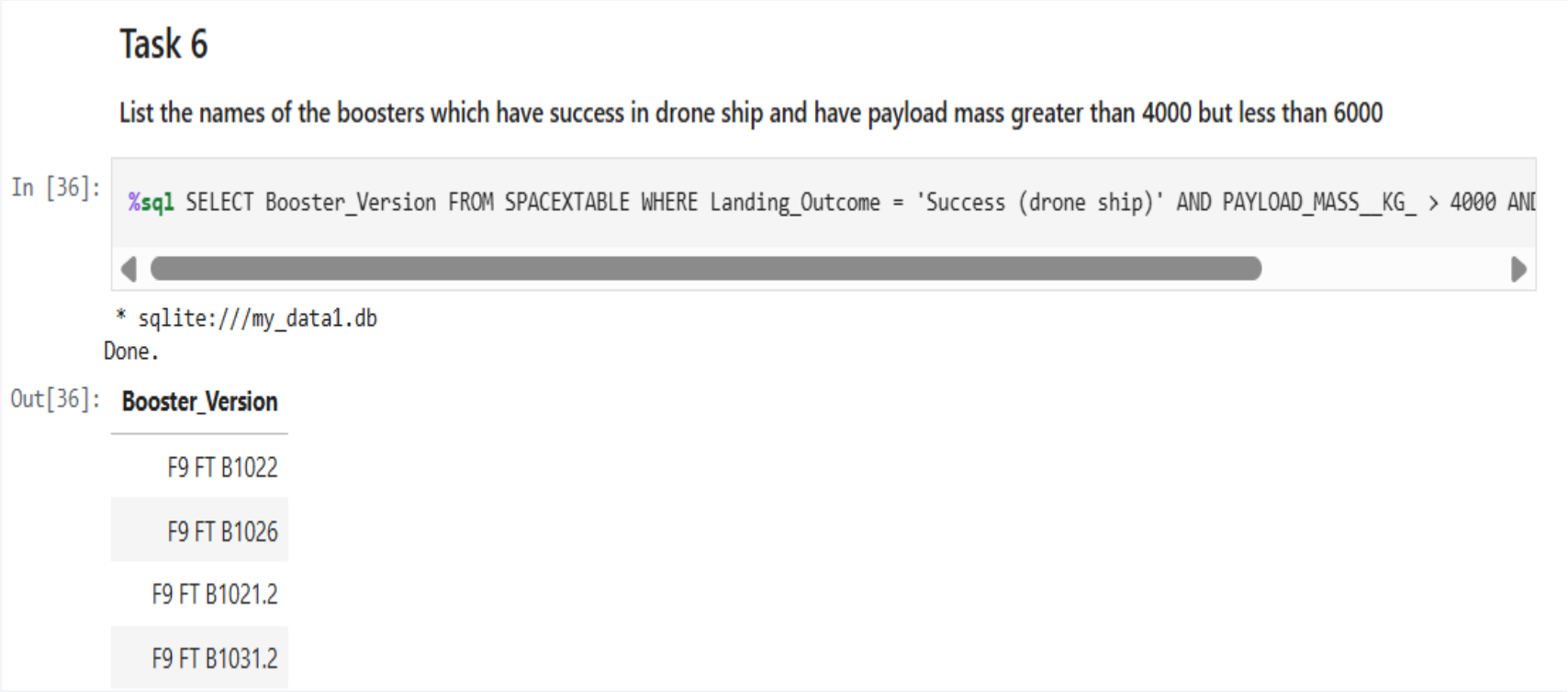
```
 * sqlite:///my_data1.db
Done.
```

Out[35]:

MIN(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the WHERE clause to filter for booster which have successfully landed on drone ship 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

In [36]:
```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND
```

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

Out[36]:

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 used wildcard like '%' to filter for WHERE Mission Outcome was a success or a failure.

List the total number of successful and failure mission outcomes

In [37]:
```
%sql SELECT Landing_Outcome, COUNT(Landing_Outcome) FROM SPACEXTABLE GROUP BY Landing_Outcome;
```

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

Out[37]:

Landing_Outcome	COUNT(Landing_Outcome)
Controlled (ocean)	5
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	21
No attempt	1
Precluded (drone ship)	1
Success	38
Success (drone ship)	14
Success (ground pad)	9
Uncontrolled (ocean)	2

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

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

In [38]:
```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
```

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

Out[38]:

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 combination of the WHERE clause, LIKE, AND and BETWEEN conditions to filter for failed landing outcome in drone ship, their booster versions, and launch site names for year 2015.

In [39]:
```
%sql SELECT substr(Date,6,2) as Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr(Date,1,4)
```

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

Out[39]:

Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- We 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.
- We applied the GROUP BY cluse to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.

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 [40]:
```
%sql SELECT Landing_Outcome, COUNT(Landing_Outcome) as OutcomeCount FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '20
```

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

Out[40]:

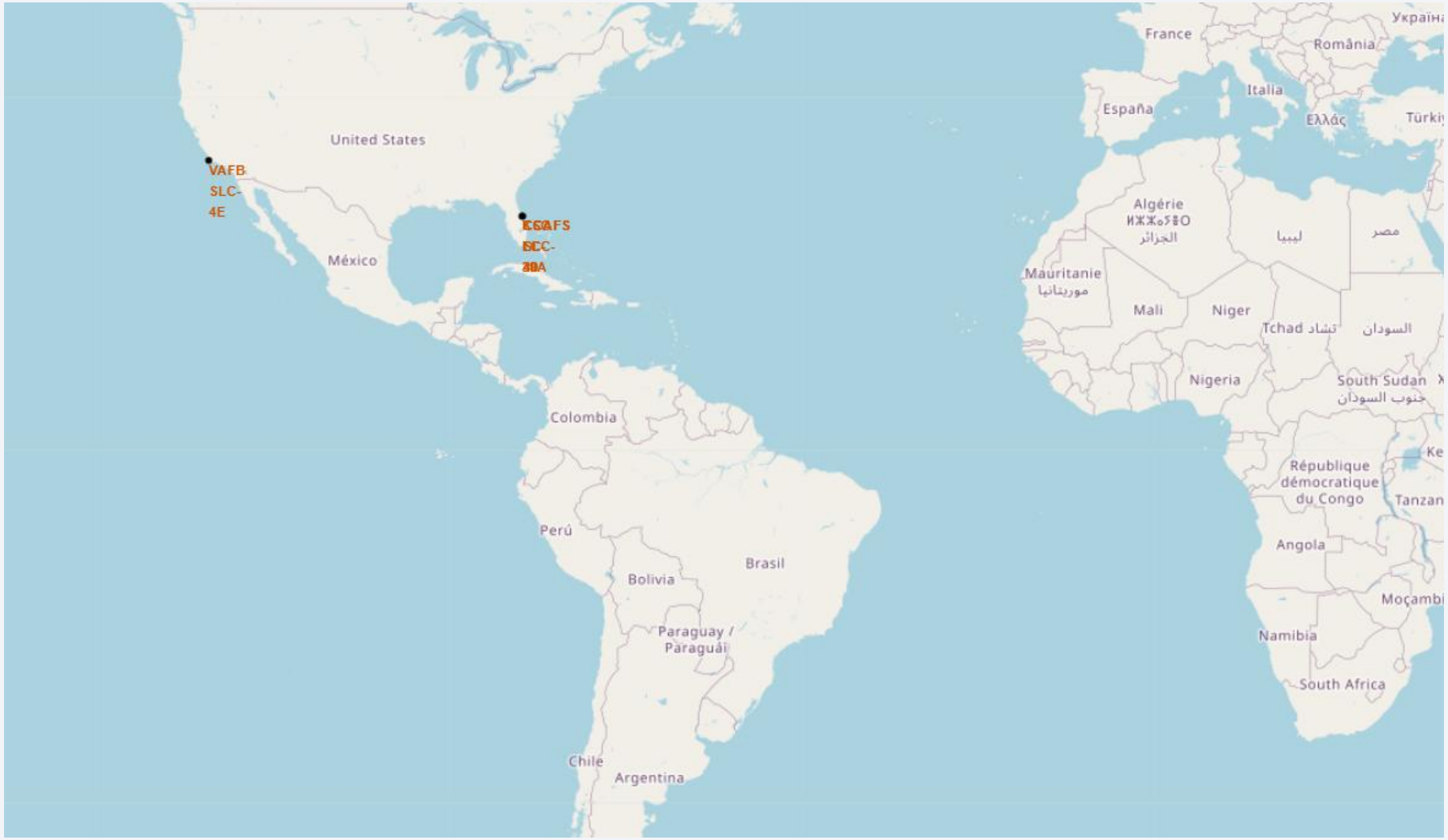
Landing_Outcome	OutcomeCount
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

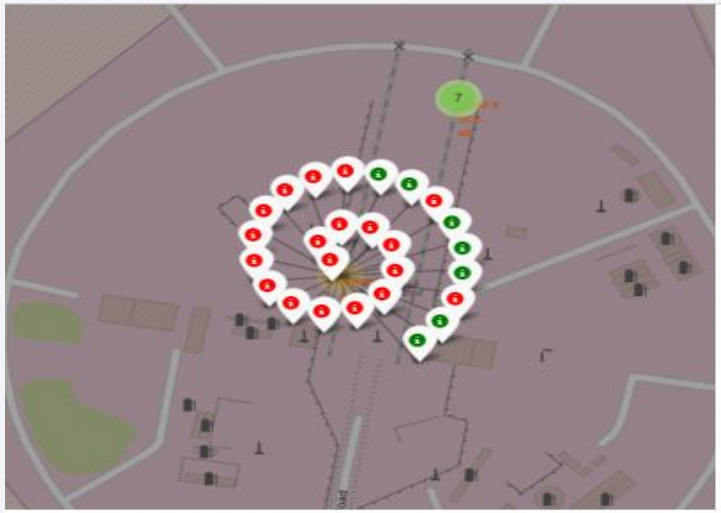
Launch Sites Proximities Analysis

ALL launch sites global map Markers

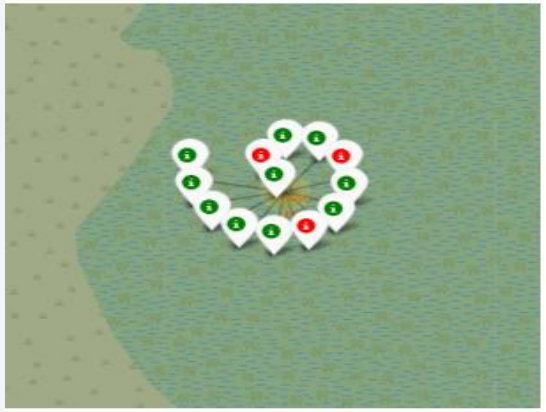
- We can see that Launch sites of SpaceX in USA are in Florida and California



Markers showing launch sites with color labels



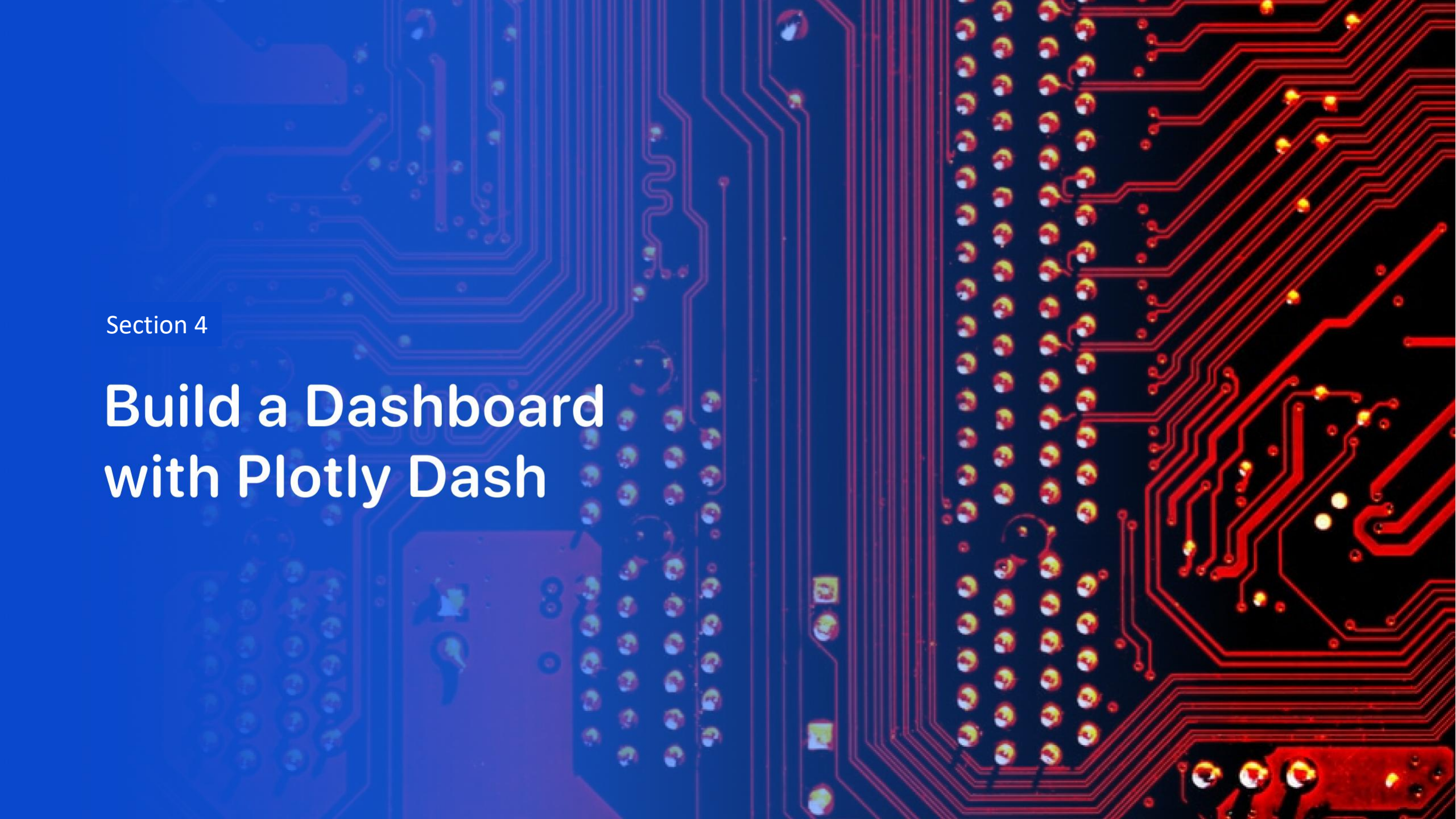
Florida Launch sites



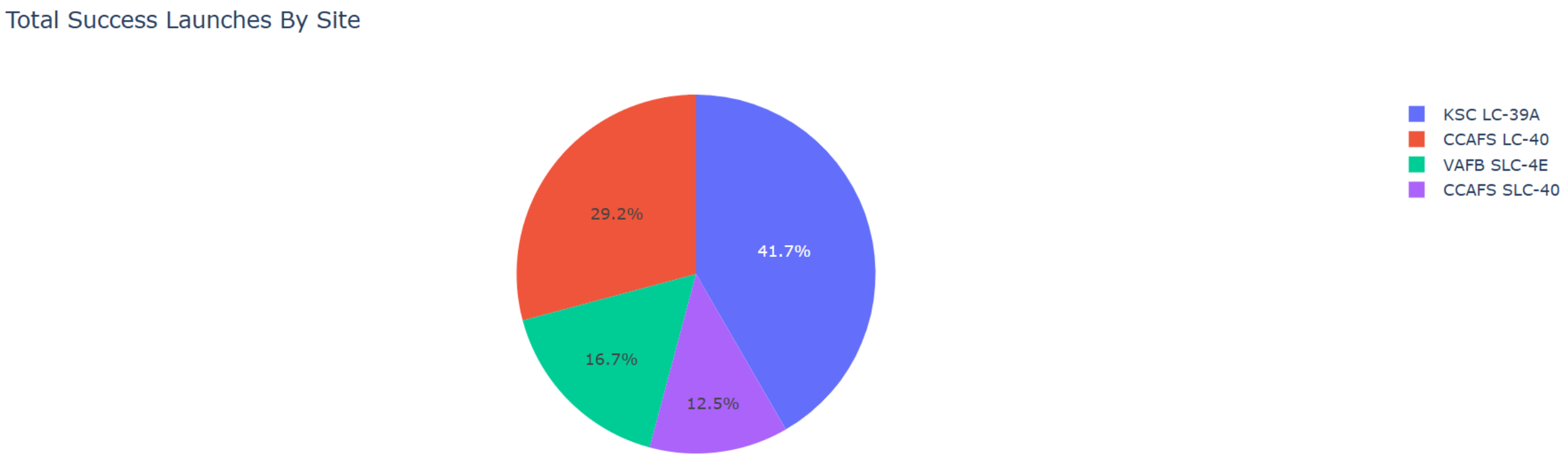
California Launch sites

Green marker shows successful Launch and **red** marker shows failed Launch

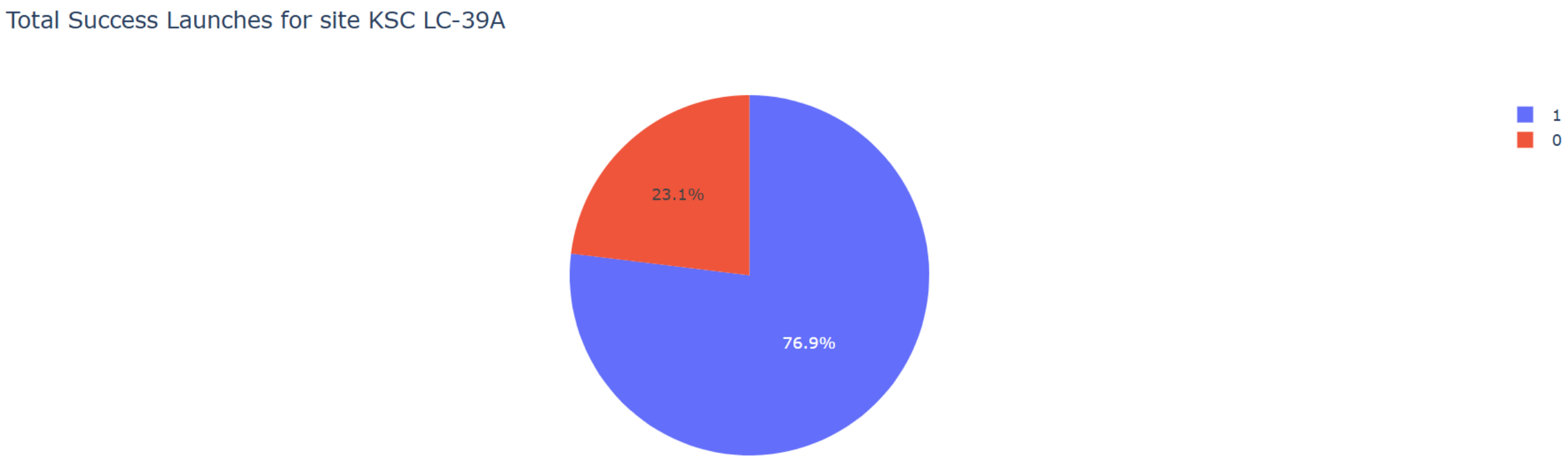
Launch site distance to landmarks


Section 4
Build a Dashboard with Plotly Dash

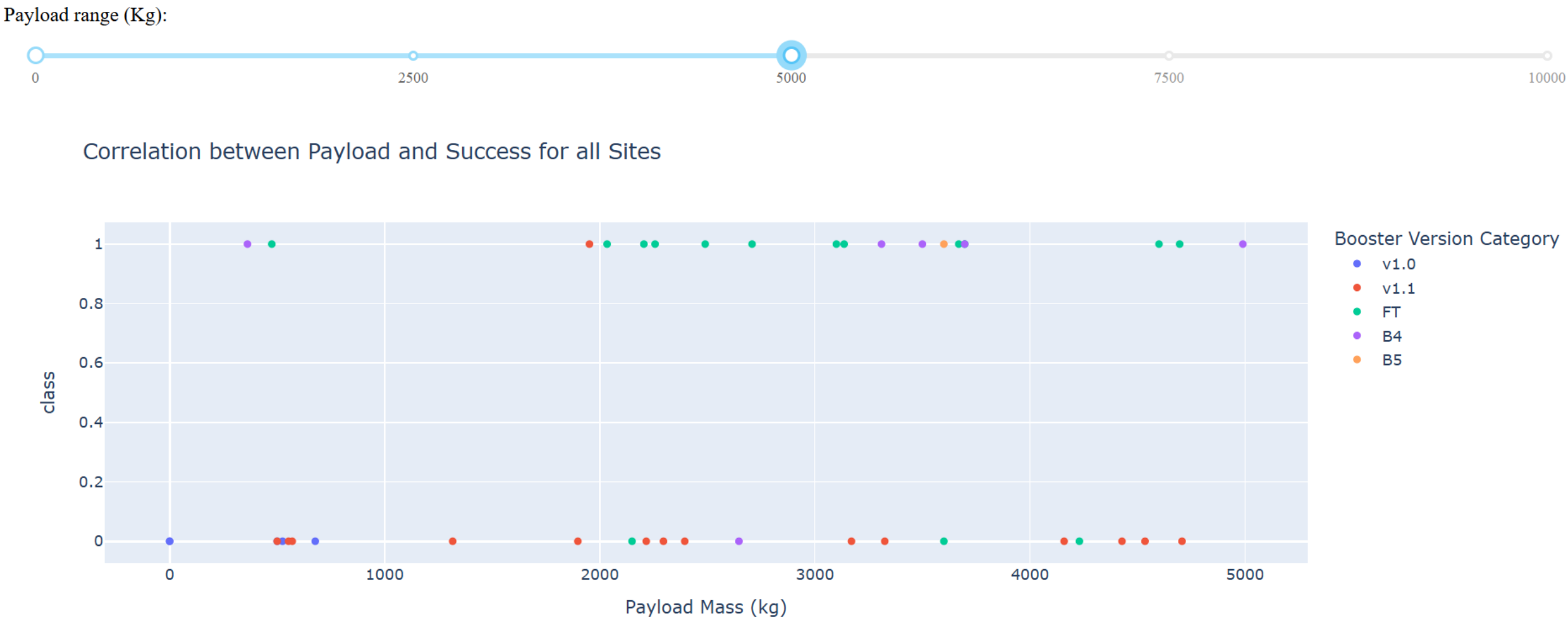
Pie chart showing the success percentage achieved by each launch site



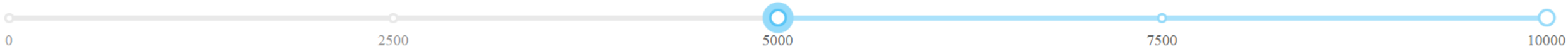
Pie chart showing the launch site with the highest launch success ration

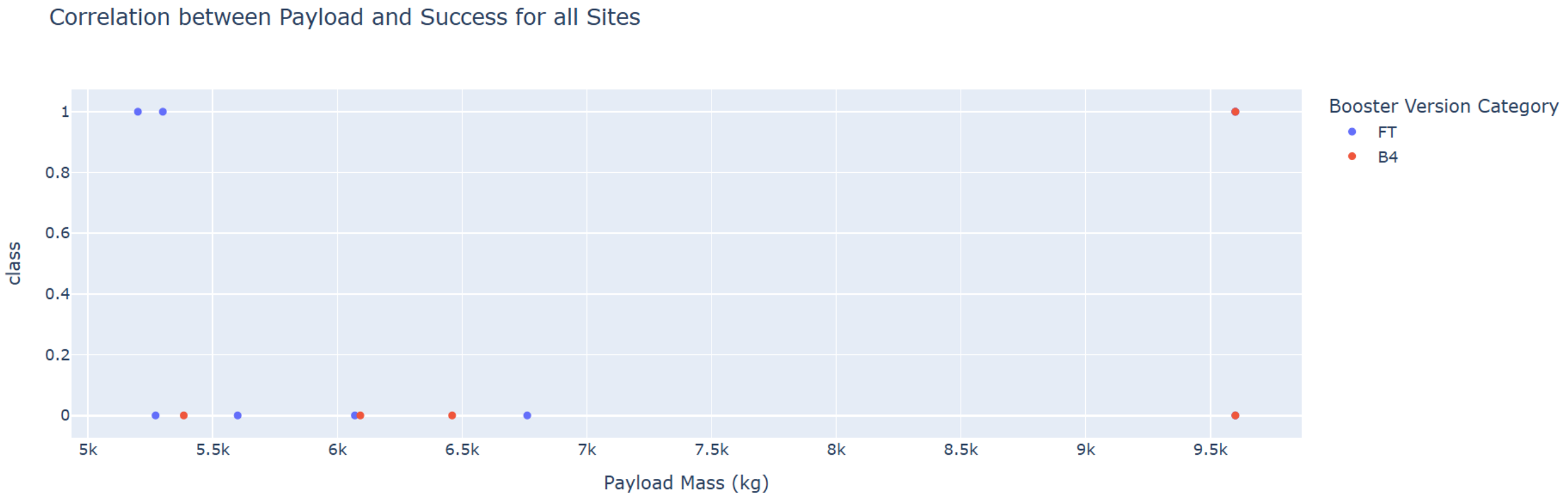


Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



Payload range (Kg):





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

Section 5

Predictive Analysis (Classification)

Classification Accuracy

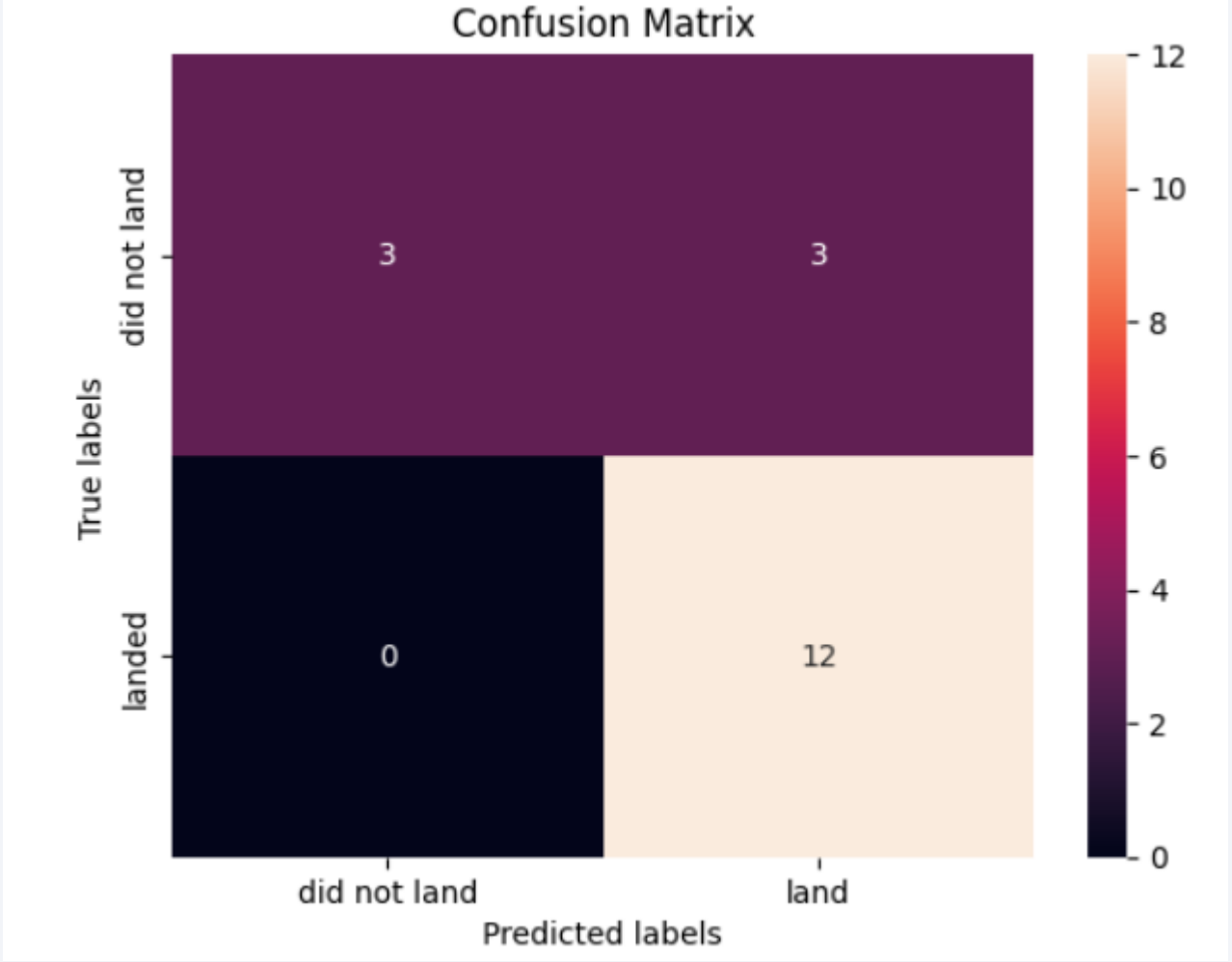
- The Logistic Regression classifier is the model with the highest classification accuracy

[32]:
```
# Create a dictionary of the models and their test accuracies
model_scores = {
    "Logistic Regression": logreg_cv.score(X_test, Y_test),
    "SVM": svm_cv.score(X_test, Y_test),
    "Decision Tree": tree_cv.score(X_test, Y_test),
    "KNN": knn_cv.score(X_test, Y_test)
}

# Print the scores
for model, score in model_scores.items():
    print(f"{model}: {score}")

# Find the best model
best_model = max(model_scores, key=model_scores.get)
print(f"\nThe best performing model is: {best_model} with accuracy {model_scores[best_model]}")
```

```
Logistic Regression: 0.8333333333333334
SVM: 0.8333333333333334
Decision Tree: 0.7222222222222222
KNN: 0.8333333333333334

The best performing model is: Logistic Regression with accuracy 0.8333333333333334
```

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 .i.e. unsuccessful landing marked as successful landing by the classifier.



Conclusions

We can conclude that:

- The larger the flight amount at a launch sit, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbit ES-L1, GEO, HEO, SSO, VLEO, had the most success rate.
- KSC LC-39A had the most successful launch of any sites.
- The Logistic Regression classifier is the best ML algorithm for this task

Thank you!