IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Essai Cardoza
Aug 16, 2023

Outline

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

Executive Summary

- ***Summary of methodologies***
  - Data collection through API
  - Data collection through 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 result***
  - ***Interactive analytics in screenshots***
  - ***Predictive analytics result***

INTRODUCTION

- ***Project background and context***
  - SpaceX promotes Falcon 9 rocket launches on its official site at a price of $62 million, in stark contrast to other suppliers whose charges surpass $165 million per launch. This substantial disparity in costs is largely due to SpaceX's pioneering ability to recycle the initial stage of the rocket. Consequently, establishing whether the first stage will achieve a successful landing becomes pivotal in estimating the overall launch expenses. This data could prove invaluable for competing firms aiming to challenge SpaceX in rocket launch bids. The core objective of this project is to construct a machine learning pipeline that can forecast the probability of a triumphant landing for the initial rocket stage.
- ***Problems you want to find answers***
  - What variables contribute to the success of a rocket landing?
  - The interplay between multiple factors influencing the likelihood of a prosperous landing achievement.
  - What specific operational criteria are necessary to establish for the realization of a successful landing initiative?

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

- **Step 1:** Data Collection Utilized a GET request to access the SpaceX API. Received response content in JSON format. Decoded JSON using .Json() function call.
- **Step 2**: Data Transformation Converted decoded JSON into a structured pandas data frame. Leveraged .Json normalize() for effective transformation.
- **Step 3**: Data Cleansing Checked for missing values within the data frame. Filled in missing values as necessary to ensure completeness.
- **Step 4:** Web Scraping from Wikipedia Employed web scraping techniques. Extracted Falcon 9 launch records from Wikipedia. Utilized Beautiful Soup for scraping.
- **Step 5:** Extracting Launch Records Focused on extracting launch records as an HTML table. Parsed and structured the extracted table.
- **Step 6:** Conversion to Pandas Data frame Converted parsed table into a PANDAS data frame. Created a structured format for analysis.
- **Step 7:** Objective and Benefit Objective: Facilitate future analysis of launch records. Benefit: Enhanced understanding and insights for analysis.

Data Collection – SpaceX API

- We employed a GET request to access the SpaceX API for data collection. Following data retrieval, we performed data cleaning, basic wrangling, and formatting to ensure its quality and suitability for analysis.
- https://github.com/Caressai/IBM-DATA-SCIENCE/blob/main/Data%20Collection%20API.ipynb

1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

In [12]:
```
# Use json_normalize method to convert the json result into a dataframe

# decode response content as json
static_json_df = res.json()
```

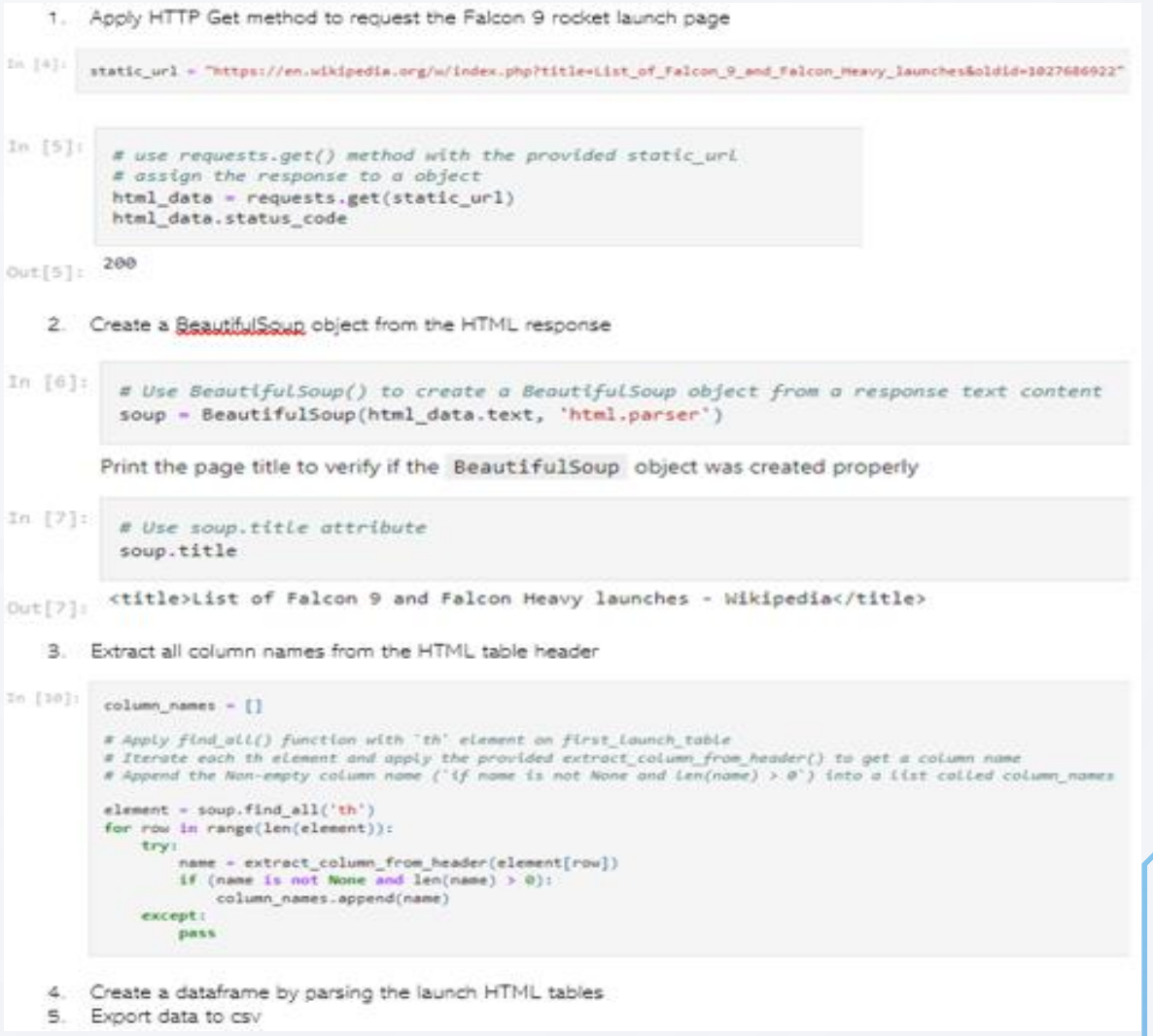
In [13]:
```
# apply json_normalize
data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

In [30]:
```
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

Data Collection - Scraping

- We applied web scrapping to web scrap Falcon 9 launch records with Beautiful Soup
- We parsed the table and converted it into a PANDAS data frame.
- https://github.com/Caressai/IBM-DATA-SCIENCE/blob/main/Data%20Collection%20API.ipynb



1. Apply HTTP Get method to request the Falcon 9 rocket launch page

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

Out[5]:
```
200
```

2. Create a BeautifulSoup object from the HTML response

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

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

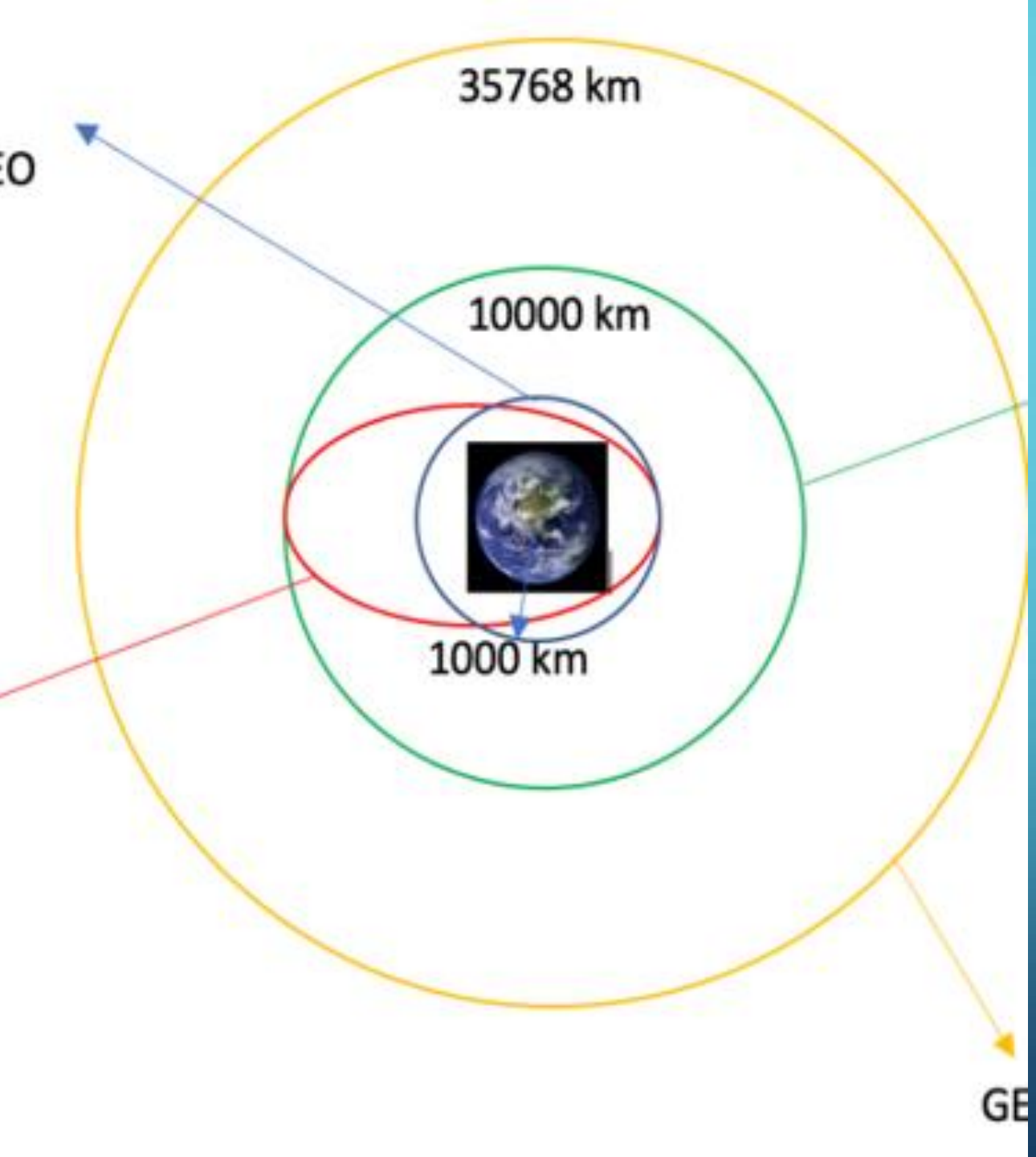
In [7]:
```
# Use soup.title attribute
soup.title
```

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

3. Extract all column names from the HTML table header

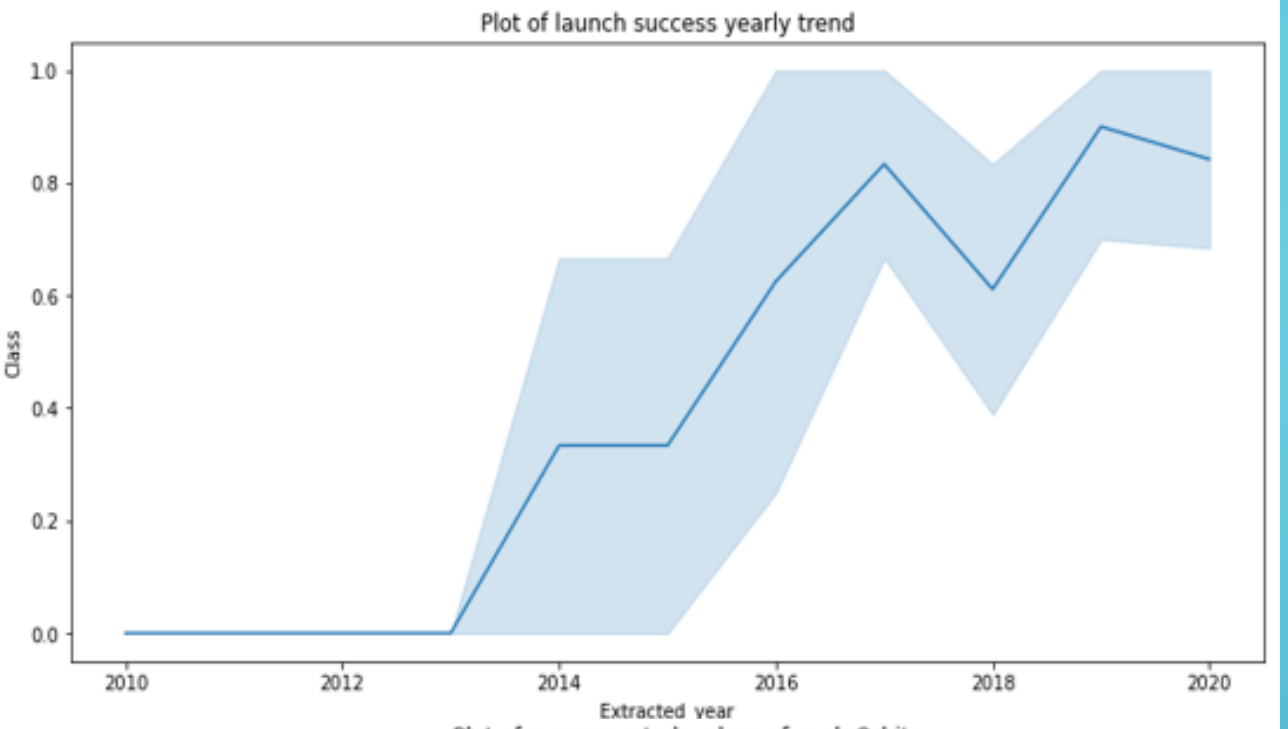
In [10]:
```
column_names = []

# 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

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

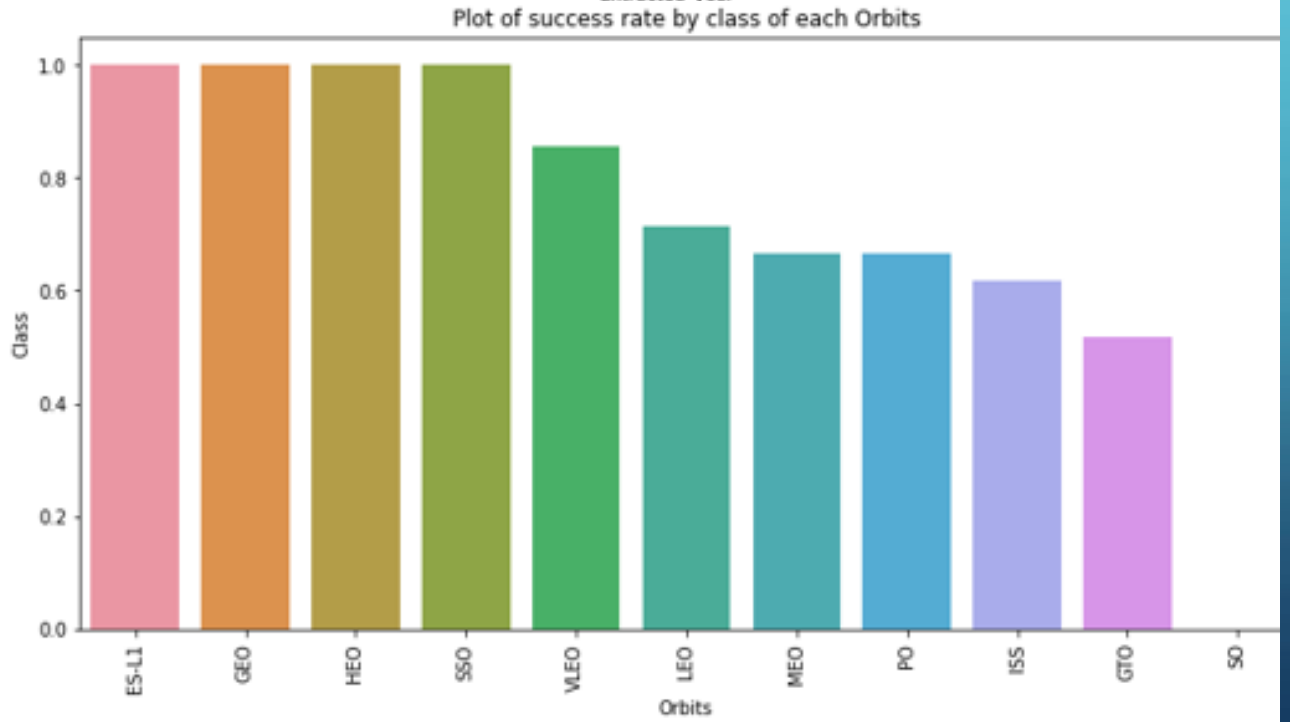
4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv



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.
- https://github.com/Caressai/IBM-DATA-SCIENCE/blob/main/Data%20Wrangling.ipynb





EDA WITH DATA VISUALIZATION

We delved into the dataset through visualization, examining various relationships. This encompassed visualizing correlations between flight number and launch site, payload and launch site, success rates across different orbit types, flight number and orbit type, as well as capturing the yearly trend in launch success.

- https://github.com/Caressai/IBM-DATA-SCIENCE/blob/main/EDA%20with%20Data%20Visualization.ipynb

EDA WITH SQL

We loaded the SpaceX dataset into a PostgreSQL database.

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.

- https://github.com/Caressai/IBM-DATA-SCIENCE/blob/main/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

I plotted each launch site on the map and integrated map elements, such as markers, circles, and lines, to signify the outcomes (success or failure) of launches at each site using the Folium library.

Attributed launch outcomes to class labels: 0 for failure and 1 for success. By employing marker clusters with distinctive colors, then we discerned launch sites exhibiting comparatively higher success rates. Our analysis further extended to computing the distances between launch sites and their neighboring areas.

Then addressed pertinent inquiries, including Proximity to railways, highways, and coastlines. Adequate distance from urban areas. These steps provided valuable insights into spatial relationships and geographical considerations associated with launch sites.

Build a Dashboard with Plotly Dash

- I Built an interactive dashboard with Plotly dash
- Then Plotted pie charts showing the total launches by a certain sites
- Finally plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- https://github.com/Caressai/IBM-DATA-SCIENCE/blob/main/Dashboard%20plotly.py

Predictive Analysis (Classification)

- I loaded the data using NumPy and pandas, transformed the data, split our data into training and testing.
- then built different machine learning models and tune different hyperparameters using GridSearchCV.
- Finally used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- https://github.com/Caressai/IBM-DATA-SCIENCE/blob/main/Machine%20Learning%20Prediction.ipynb

Results

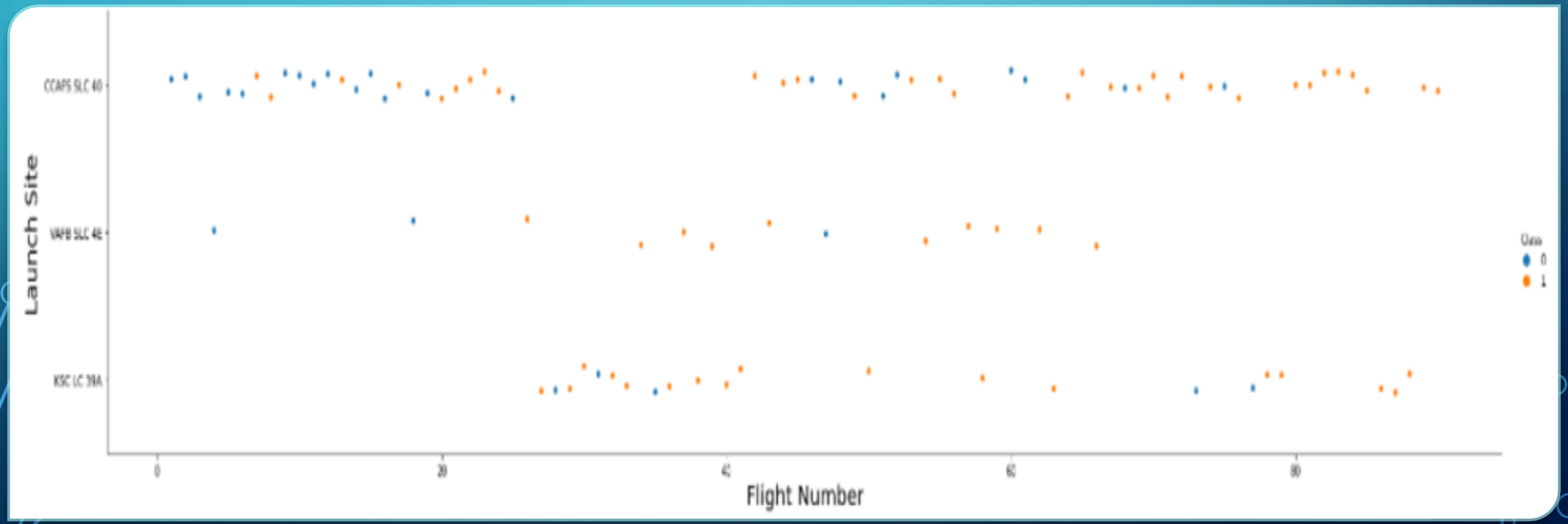
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

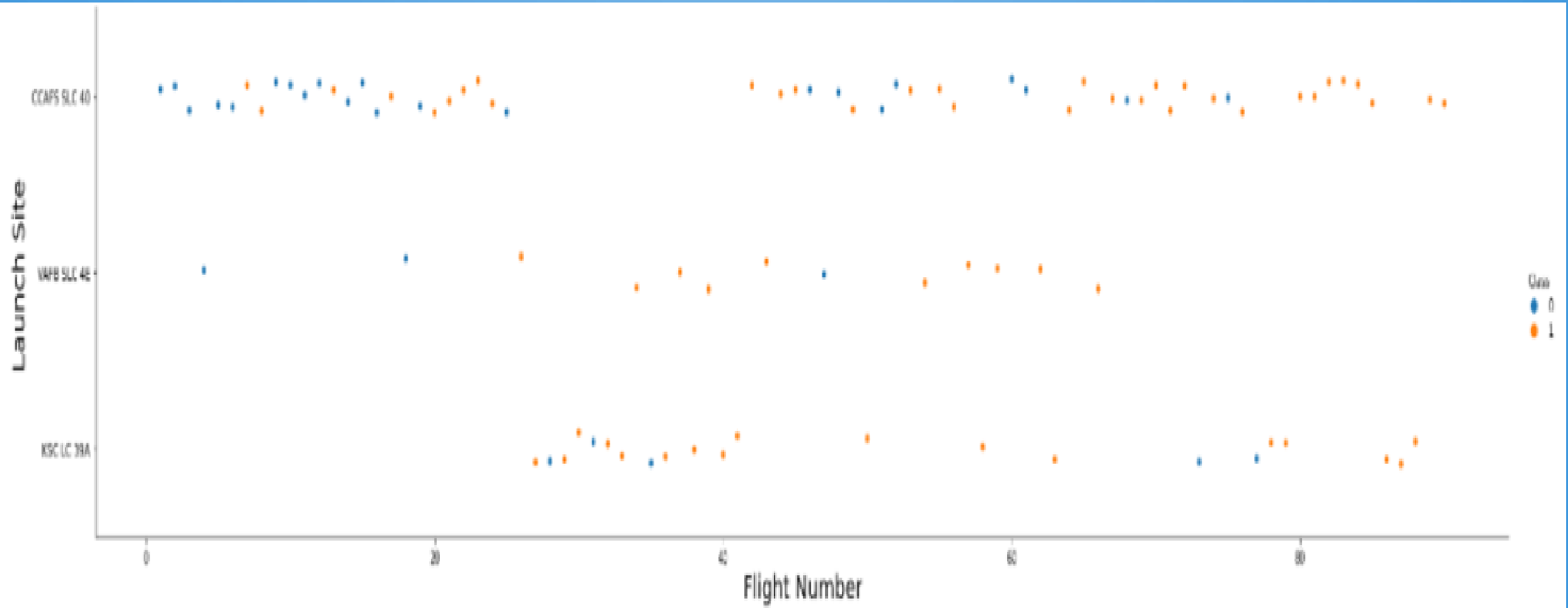
FLIGHT NUMBER VS. LAUNCH SITE

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



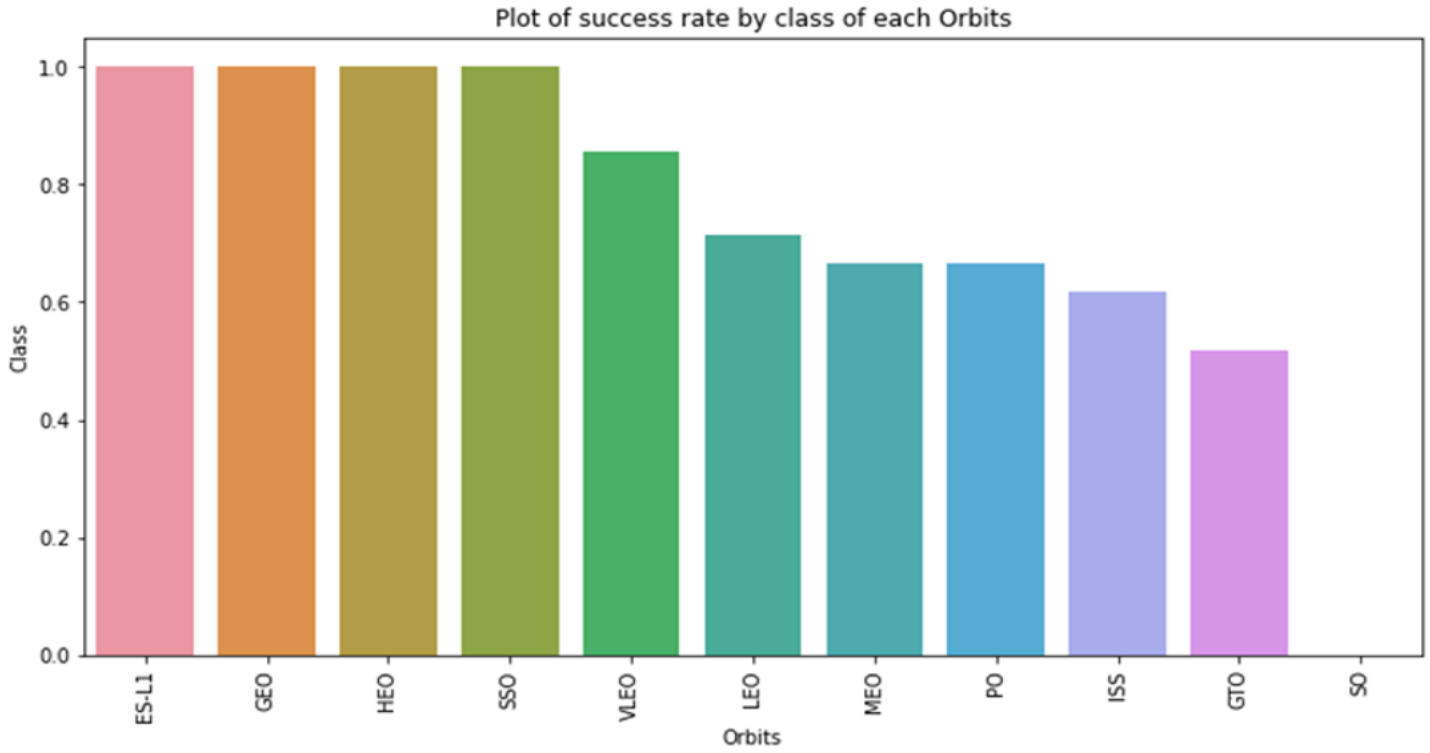
Payload vs. Launch Site

The greater the mass payload for launch site CCAFS SLC 40 the greater chance of success



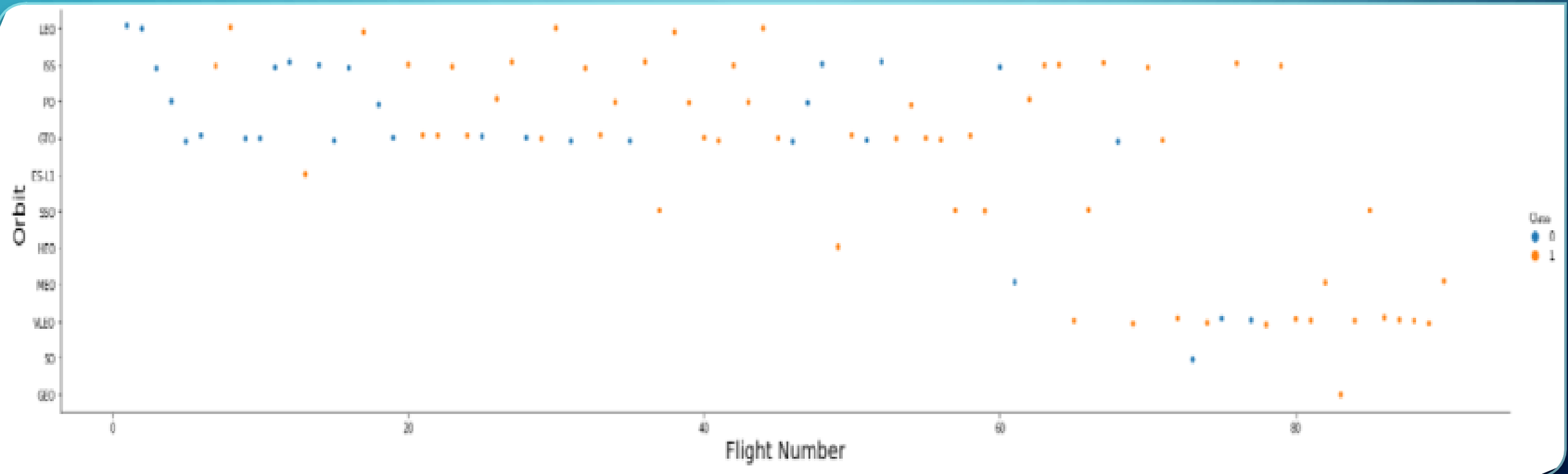
SUCCESS RATE VS. ORBIT TYPE

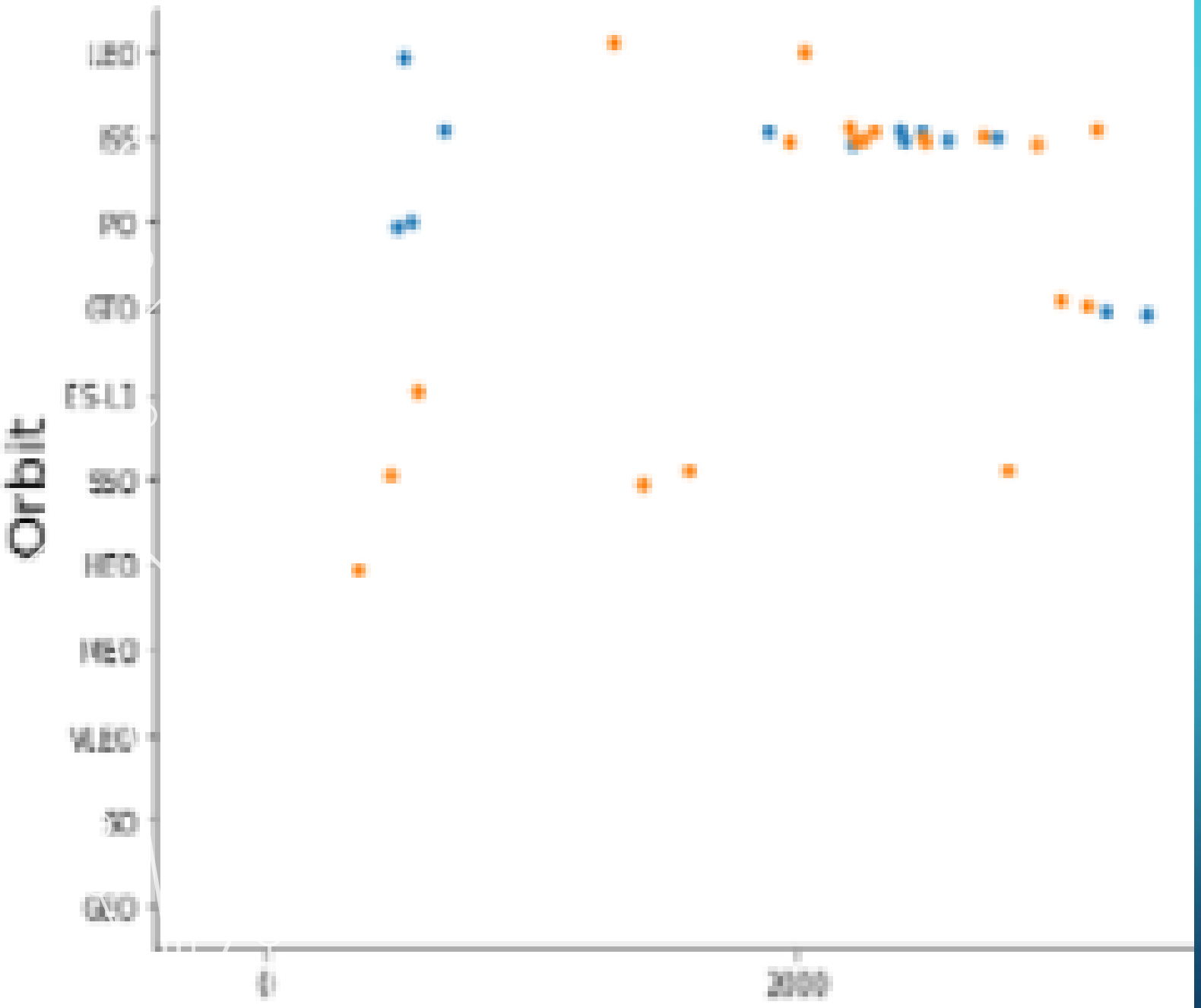
- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success orbit rates.



FLIGHT NUMBER VS. ORBIT TYPE

- The plot 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.



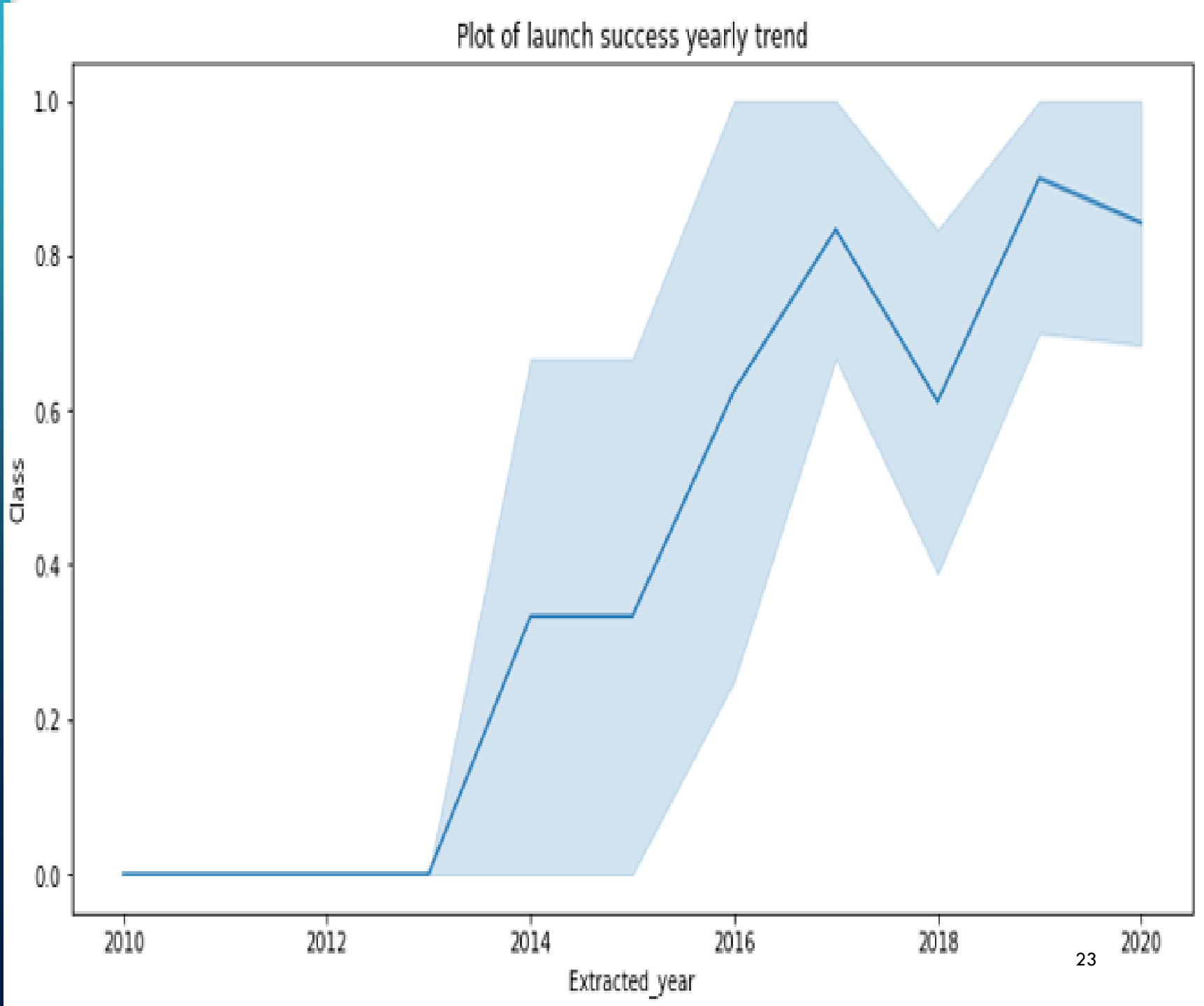


PAYLOAD VS. ORBIT TYPE

- 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.

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

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

LAUNCH SITE NAMES BEGIN WITH 'CCA'

I used the query above to display 5 records where launch sites begin with `CCA`.

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

In [11]:
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

TOTAL PAYLOAD MASS

I calculated the total payload carried by boosters from NASA resulting 45596 using the query.

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

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''

create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

AVERAGE PAYLOAD MASS BY F9 V1.1

I calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

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

In [13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''

create_pandas_df(task_4, database=conn)
```

Out[13]:

avg_payloadmass
2928.4

FIRST SUCCESSFUL GROUND LANDING DATE

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

In [14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''

create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

SUCCESSFUL DRONE SHIP LANDING WITH PAYLOAD BETWEEN 4000 AND 6000

I 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

In [15]:

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

TOTAL NUMBER OF SUCCESSFUL AND FAILURE MISSION OUTCOMES

- I used wildcard like '%' to filter for **WHERE** MissionOutcome was a success or a failure.

List the total number of successful and failure mission outcomes

In [16]:
```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

```
The total number of failed mission outcome is:
```

Out[16]:

	failureoutcome
0	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.

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

In [17]:
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 LAUNCH RECORDS

- I 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

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

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''

create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

RANK LANDING OUTCOMES BETWEEN 2010-06-04 AND 2017-03-20

I 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.

I then applied 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 landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:

```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

Out[19]:

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

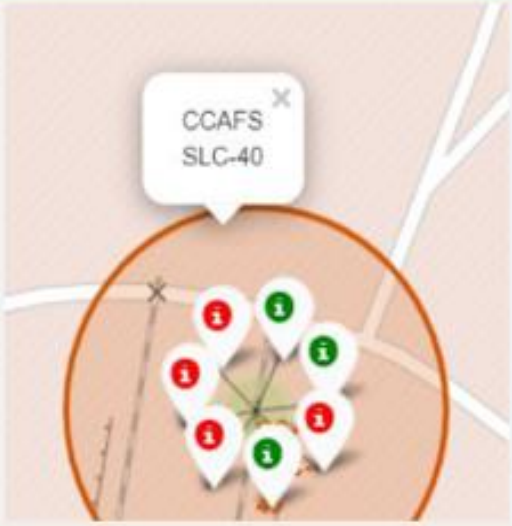
Section 3

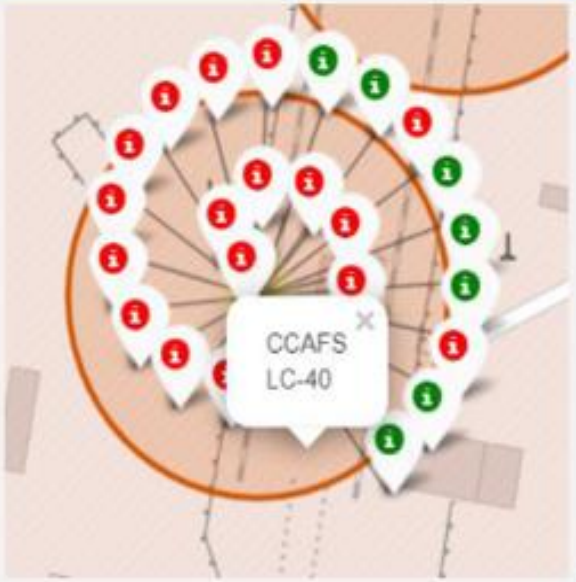
Launch Sites Proximities Analysis

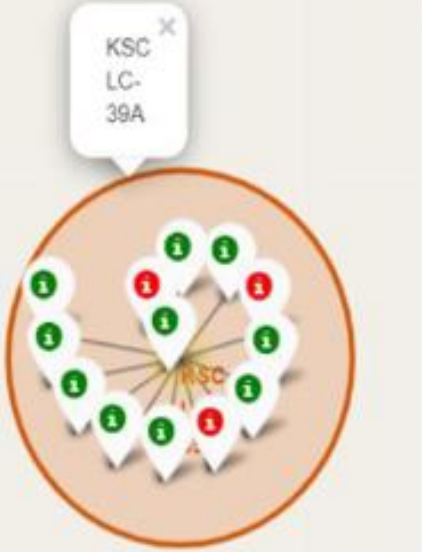
SpaceX launch site Markers

We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

SpaceX launch flights labels





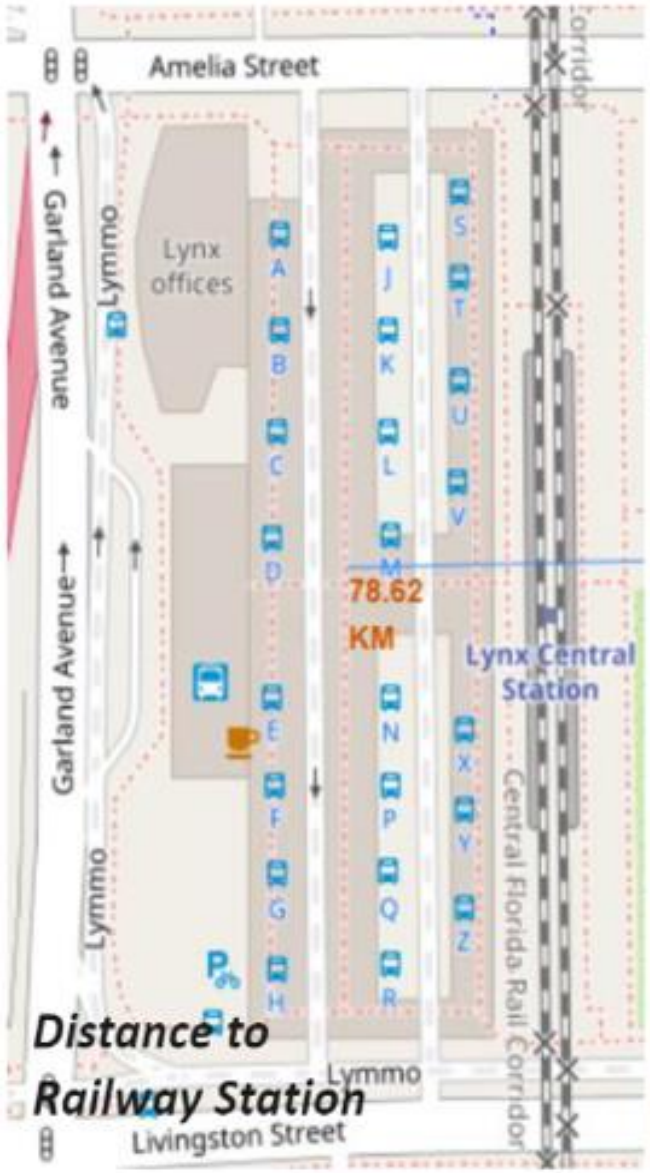


Florida Launch Sites

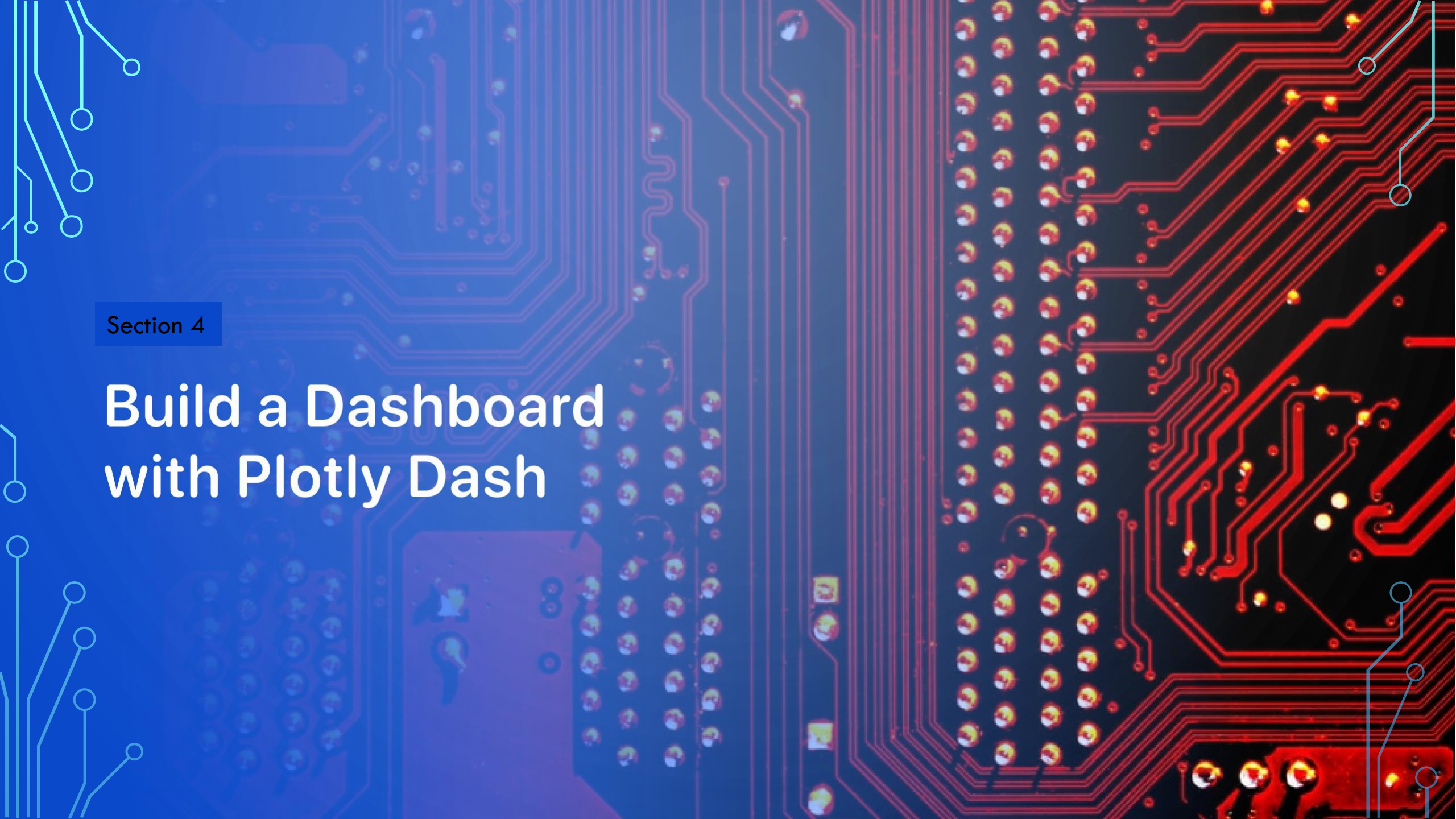
Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

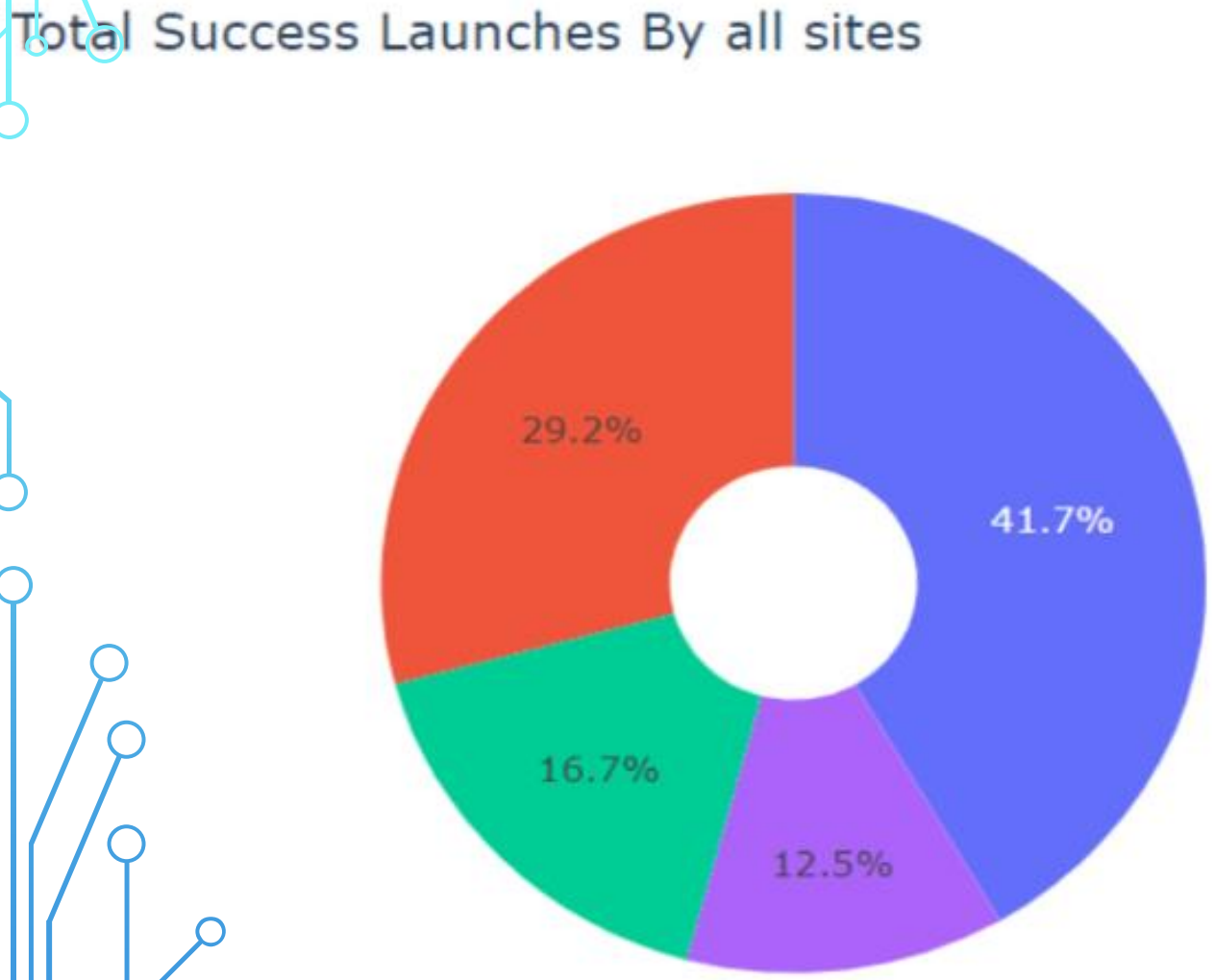
Launch sites distance from markers



- 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

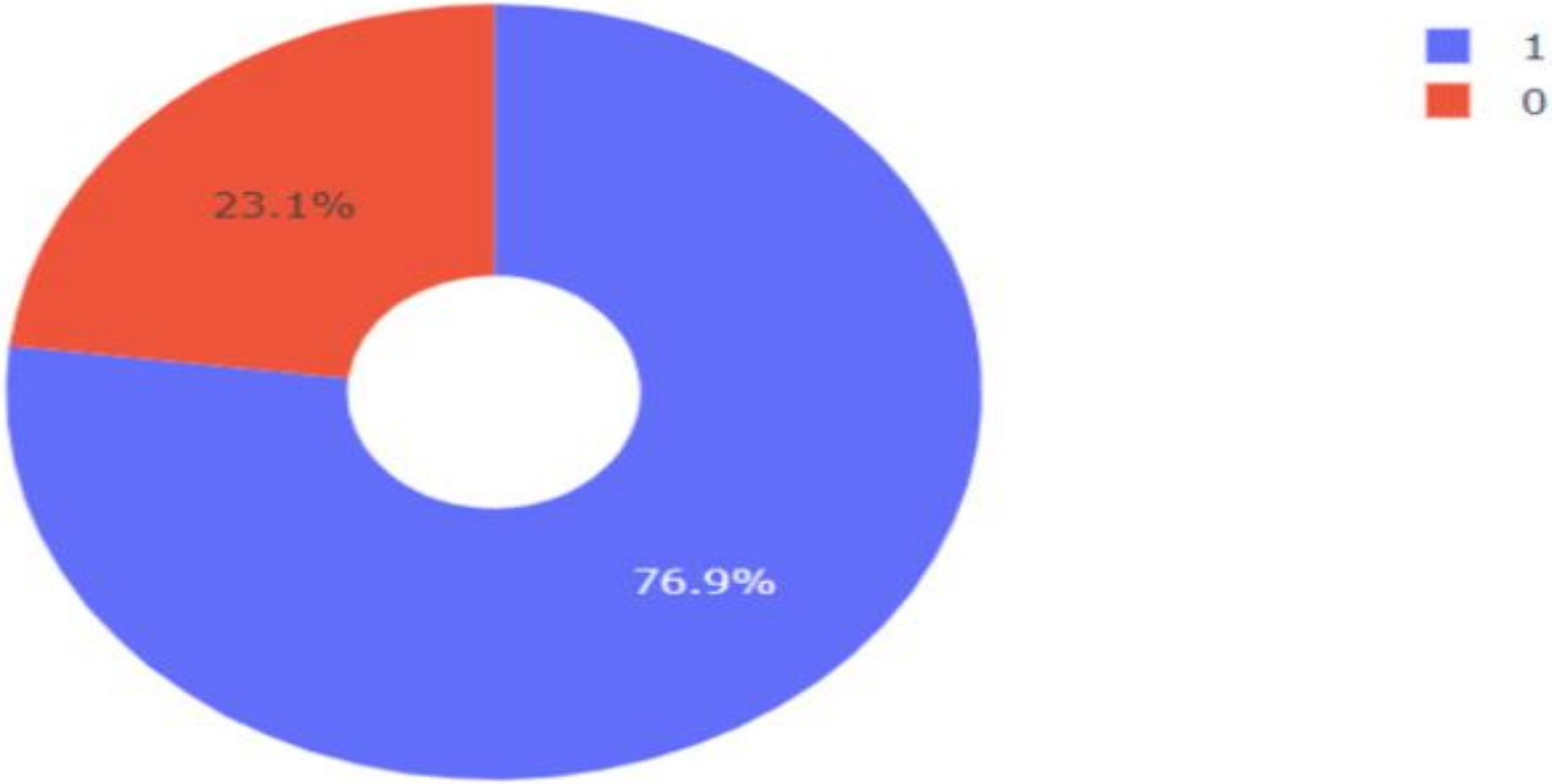

Section 4
Build a Dashboard with Plotly Dash

SUCCESS PERCENTAGE OF EACH LAUNCH SITE



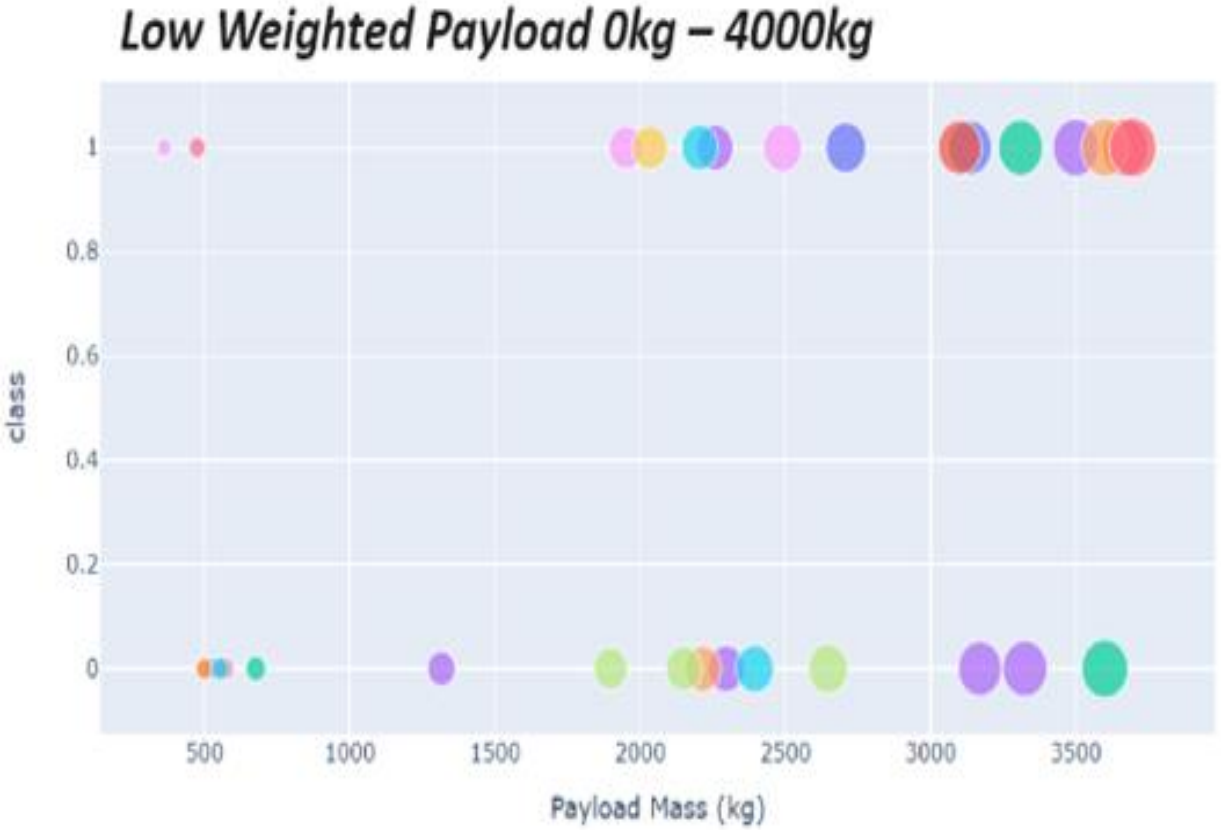
We can see that KSC LC-39A had the most successful launches from all the sites

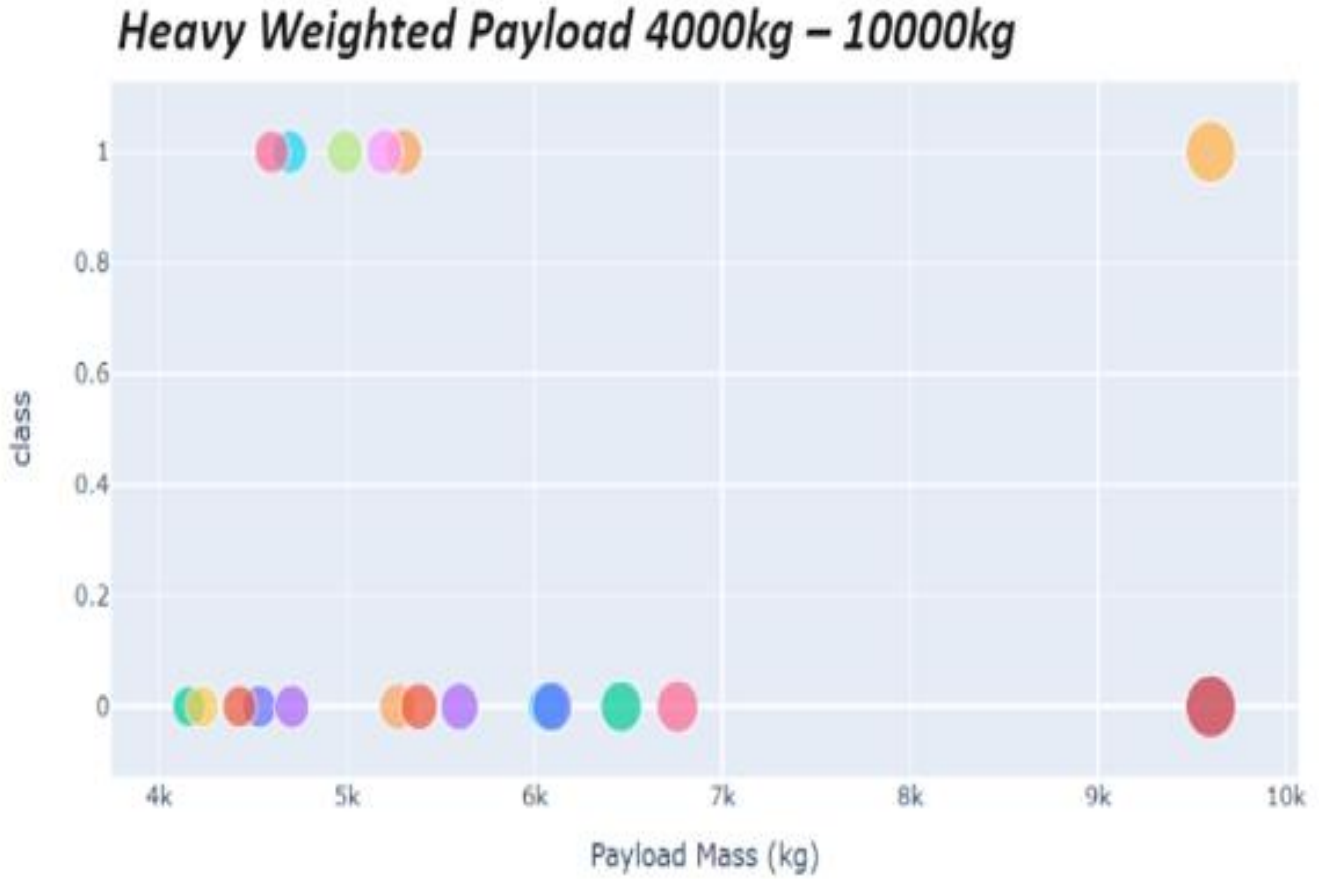
KSC LC-39A success rate



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,





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

Section 5

Predictive Analysis (Classification)

CLASSIFICATIO N ACCURACY

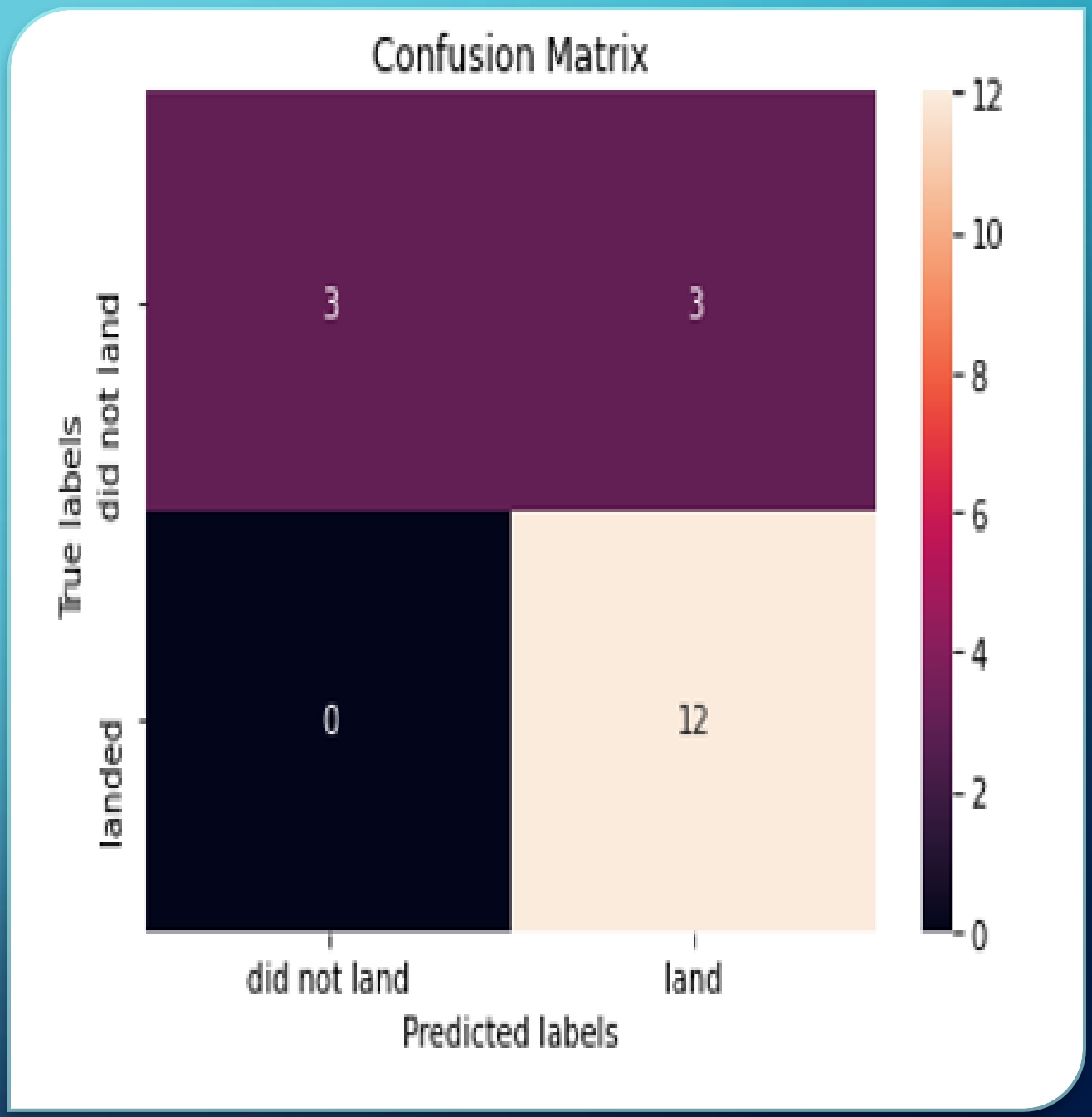
The decision tree classifier is the model with the highest classification accuracy.

```
models = {'KNeighbors':knn_cv.best_score_,
          'DecisionTree':tree_cv.best_score_,
          'LogisticRegression':logreg_cv.best_score_,
          'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

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 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 Decision tree classifier is the best machine learning algorithm for this task.

Thank you!