Winning Space Race with Data Science

Mark Lester C. Real
01 October 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

On its website, Space X promotes Falcon 9 rocket launches for 62 million dollars; other suppliers charge upwards of 165 million dollars for each launch. A large portion of the savings is due to Space X's ability to reuse the first stage. So, if we can figure out whether the first stage will land, we can figure out how much a launch will cost. If another business wishes to submit a proposal for a rocket launch against space X, they can use this information. The project's objective is to build a pipeline for machine learning that can forecast if the initial stage will land successfully.

Problems you want to find answers

- What elements determine whether the rocket will successfully land?
- What are the way different factors combine to affect a landing's likelihood of success?
- What operational requirements must be met for a landing program to be successful?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was gathered through scraping Wikipedia's website and the SpaceX API.
  - Perform data wrangling
  - We used one-hot encoding for 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 was gathered by sending a get request to the SpaceX API.
  - The content of the response was then decrypted as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - The data was then cleansed, missing values were checked for, and filled in as appropriate.
  - Additionally, using BeautifulSoup, we scraped Wikipedia for information on Falcon 9 launch statistics.
  - The goal was to extract the launch records as an HTML table, parse the table, and then transform the table into a pandas dataframe for later analysis.

Data Collection – SpaceX API

- To gather data, sanitize the requested data, and do some simple data wrangling and formatting, we used the get request to the SpaceX API.
- The link to the notebook is https://github.com/engrmlr213/dscaps/blob/main/1_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

- With BeautifulSoup, we used web scraping to collect Falcon 9 launch data.
- The table was analyzed, then transformed into a pandas dataframe.
- The link to the notebook is:

https://github.com/engrmlr213/dscaps/blob/main/2_Data%20Collection%20with%20Web%20Scraping.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

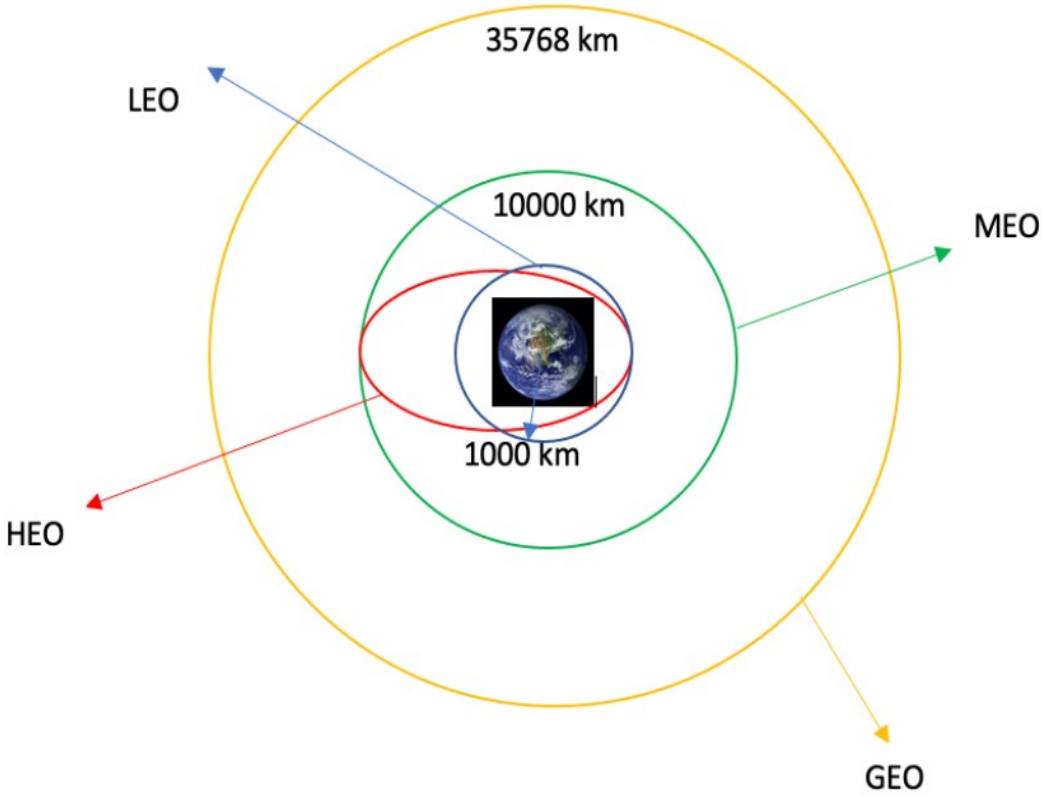
```
In [10]: column_names = []

         # 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

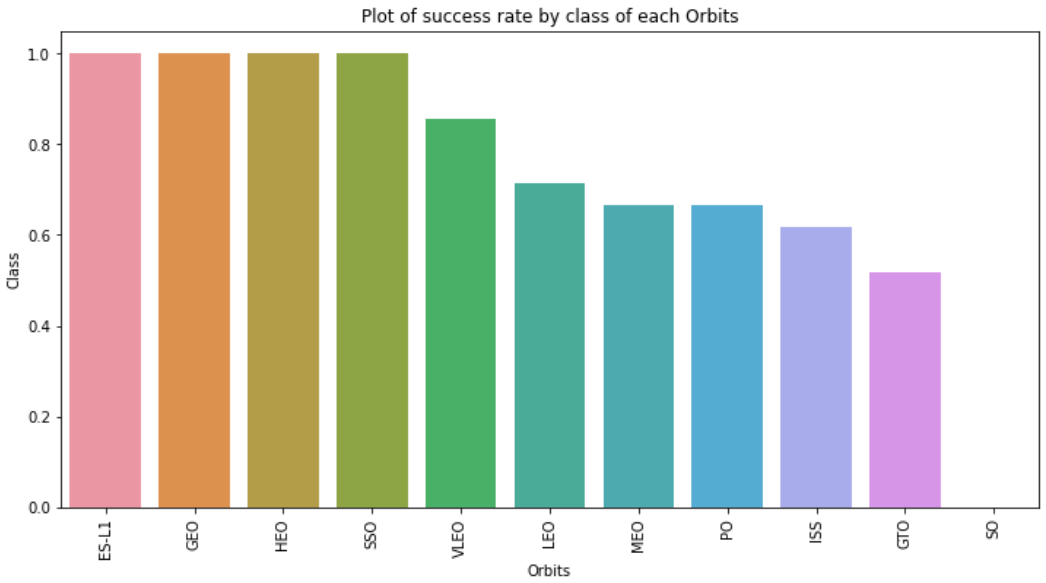
Data Wrangling

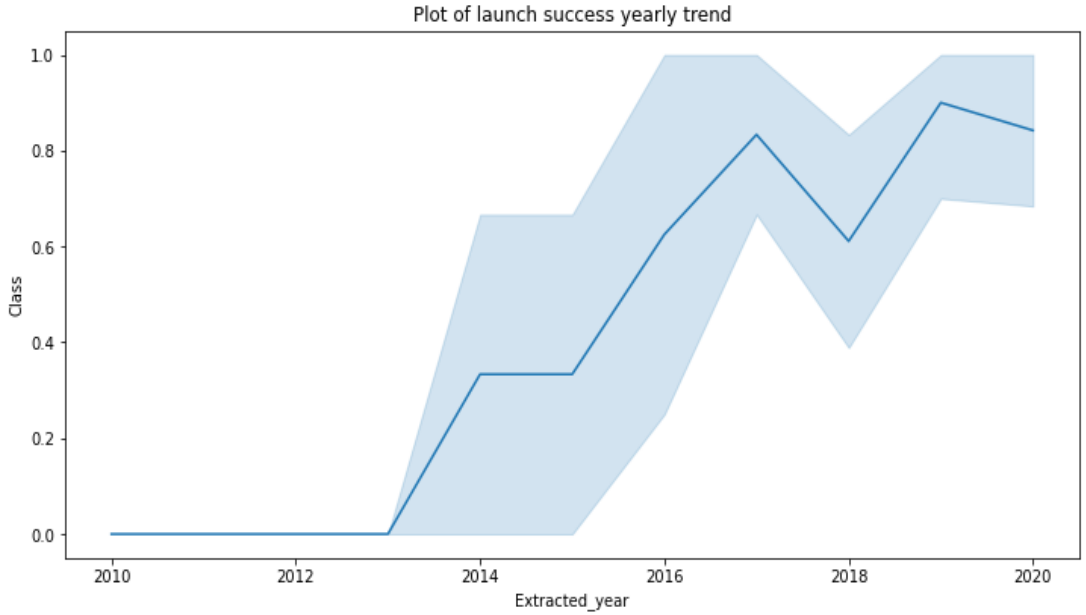


- Exploratory data analysis was done to establish the training labels.
- We determined the number of launches at each location as well as the frequency and number of orbits.
- We used the outcome column to build the landing outcome label and saved the data to CSV.
- The link to the notebook is https://github.com/engrmlr213/dscaps/blob/main/3_Data%20Wrangling.ipynb

EDA with Data Visualization

- By displaying the relationship between the flight number and the launch site, the payload and the launch site, the success rate of each orbit type, the flight number and the orbit type, and the yearly trend in launch success, we investigated the data.





- The link to the notebook is https://github.com/engrmlr213/dscaps/blob/main/4_EDA%20with%20Data%20Visualization.ipynb

EDA with SQL

- Without leaving the Jupyter notebook, the SpaceX dataset was loaded 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.
- The link to the notebook is https://github.com/engrmlr213/dscaps/blob/main/5_EDA%20with%20SQL_MLCR.ipynb

Build an Interactive Map with Folium

- On the folium map, we identified every launch point and added map elements like markers, circles, and lines to indicate whether a launch was successful or unsuccessful for each location.
- We classified the success or failure of the feature launch into classes 0 and 1, i.e., 0 for failure and 1 for success.
- The launch sites with a comparatively high success rate were determined using the color-labeled marker clusters.
- We measured the separations between a launch site and its environs. We responded to various queries, such as:

1) Are launch sites near railways, highways and coastlines.
2) Do launch sites keep certain distance away from cities.

Build a Dashboard with Plotly Dash

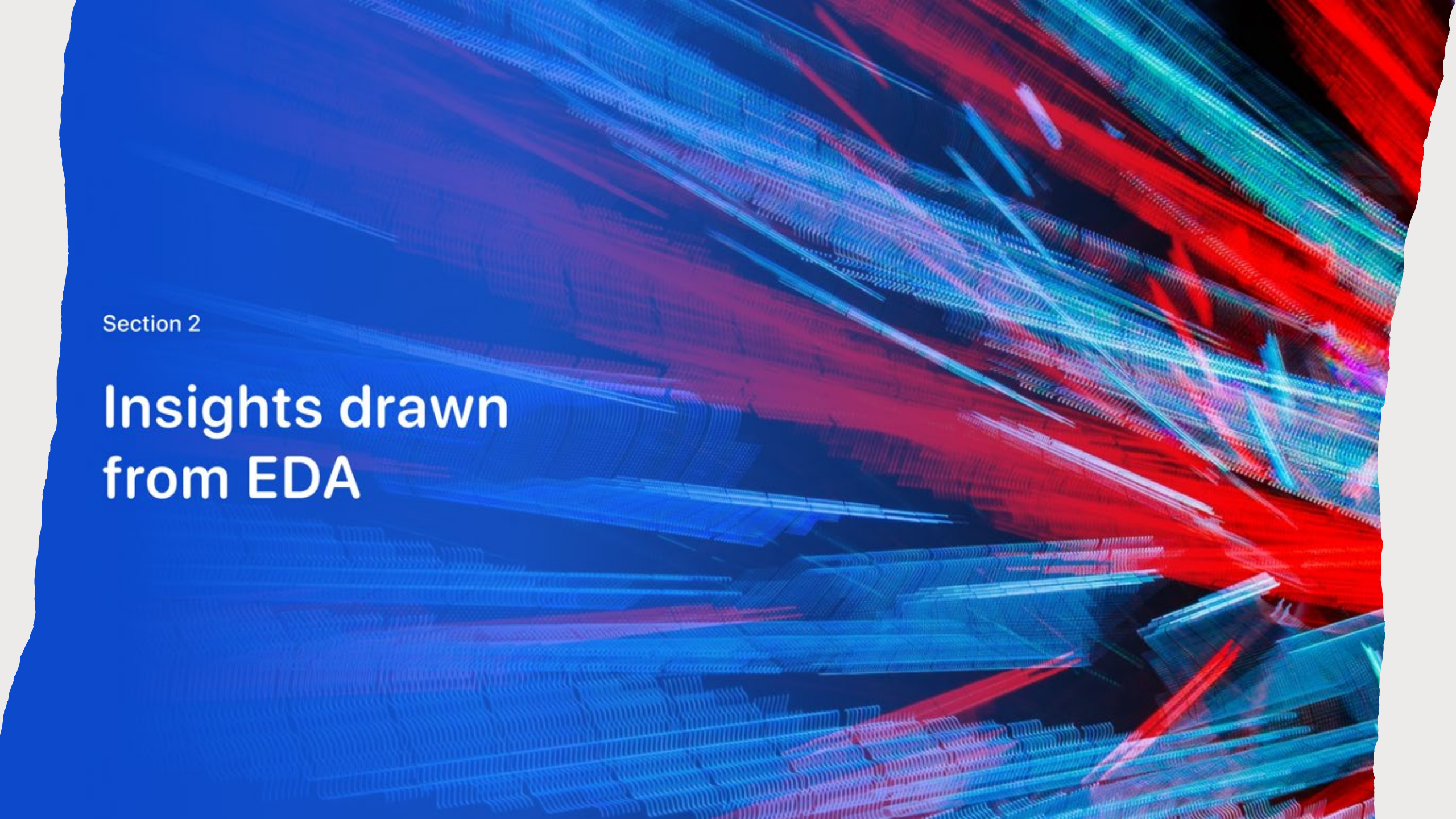
- Using Plotly dash, we created an interactive dashboard.
- We created pie graphs that display all of the launches made by particular sites.
- For each booster version, we created a scatter graph to highlight the relationship between the outcome and the payload mass (Kg).
- The link to the notebook is https://github.com/engrmlr213/dscaps/blob/main/app.py

Predictive Analysis (Classification)

- Using Numpy and Pandas, we loaded the data, transformed it, and divided it into training and testing sets.
- Using GridSearchCV, we constructed various machine learning models and tuned various hyperparameters.
- Our model was measured by accuracy, and it was enhanced through feature engineering and algorithm tuning.
- The most effective classification model was discovered.
- The link to the notebook is https://github.com/engrmlr213/dscaps/blob/main/8_Machine%20Learning%20Prediction.ipynb

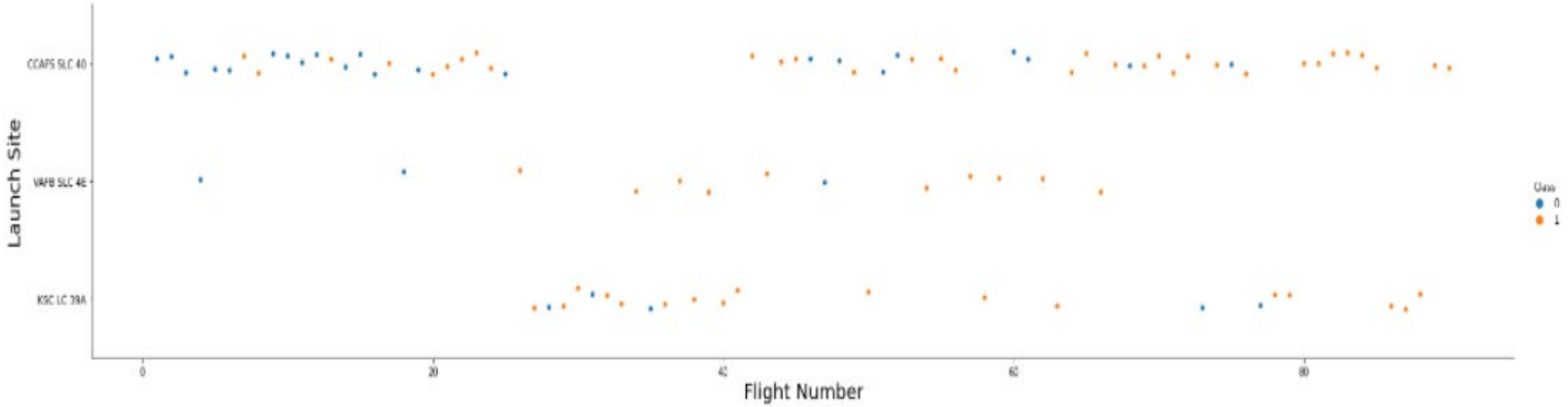
Results

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


Section 2
Insights drawn
from EDA

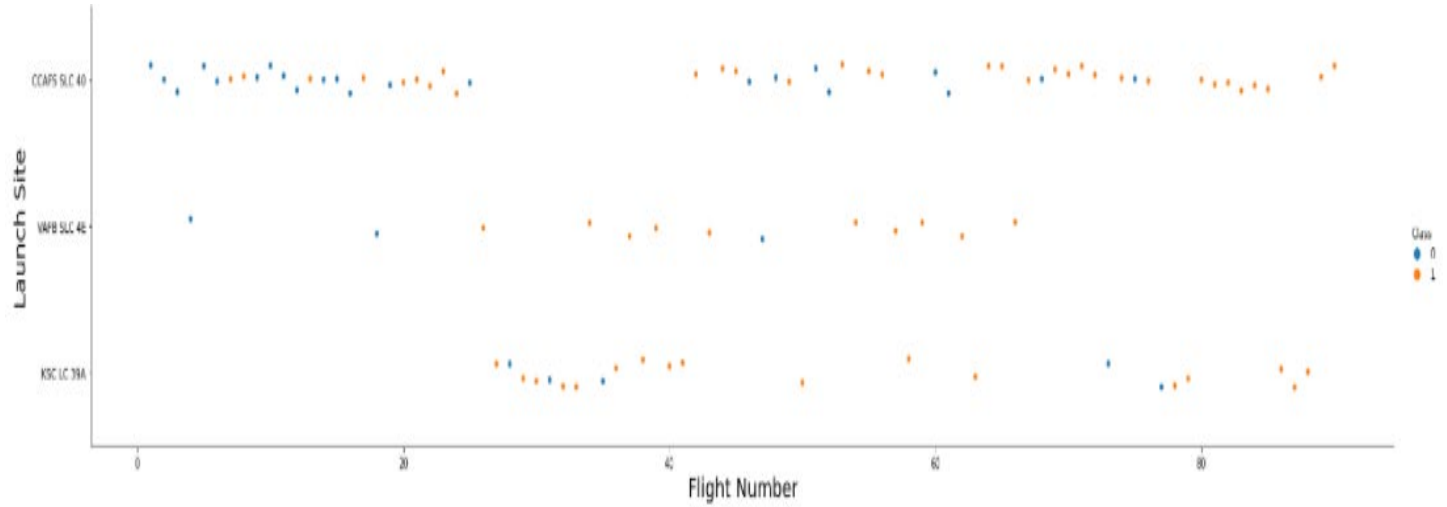
Flight Number vs. Launch Site

✓The plot led us to the conclusion that a launch site's success rate increases with the size of the flight quantity.



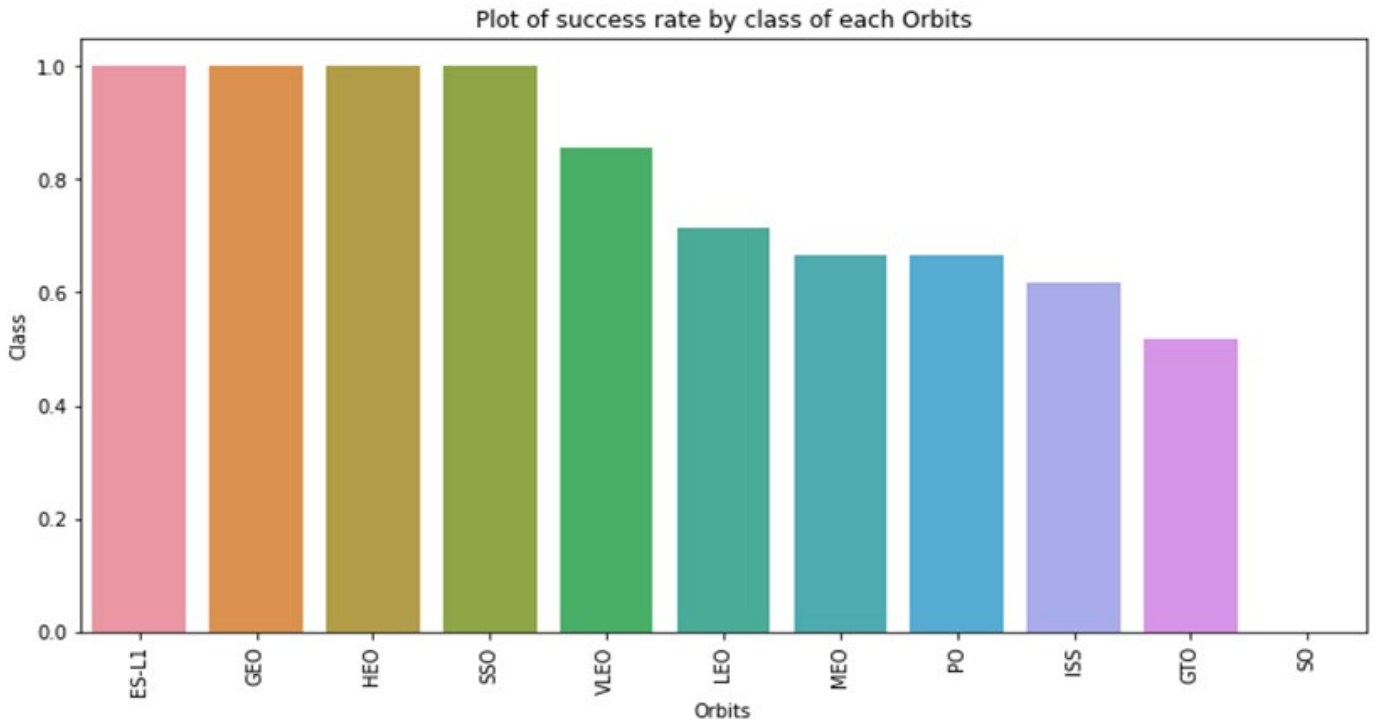
The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.

Payload vs. Launch Site



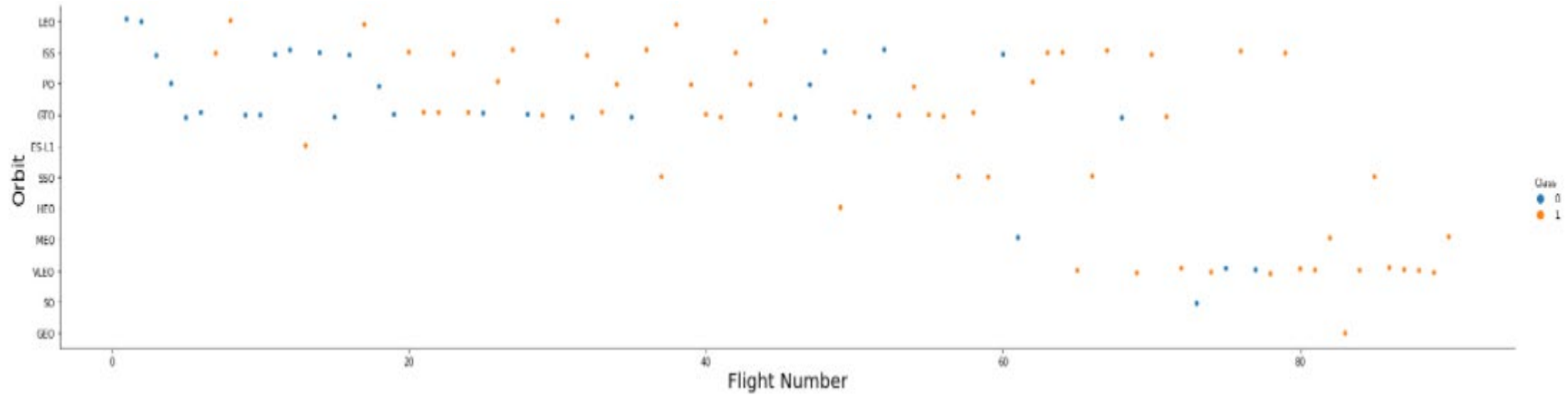
Success Rate vs. Orbit Type

- ✓ According to the figure, ES-L1, GEO, HEO, SSO, and VLEO had the highest success rates.



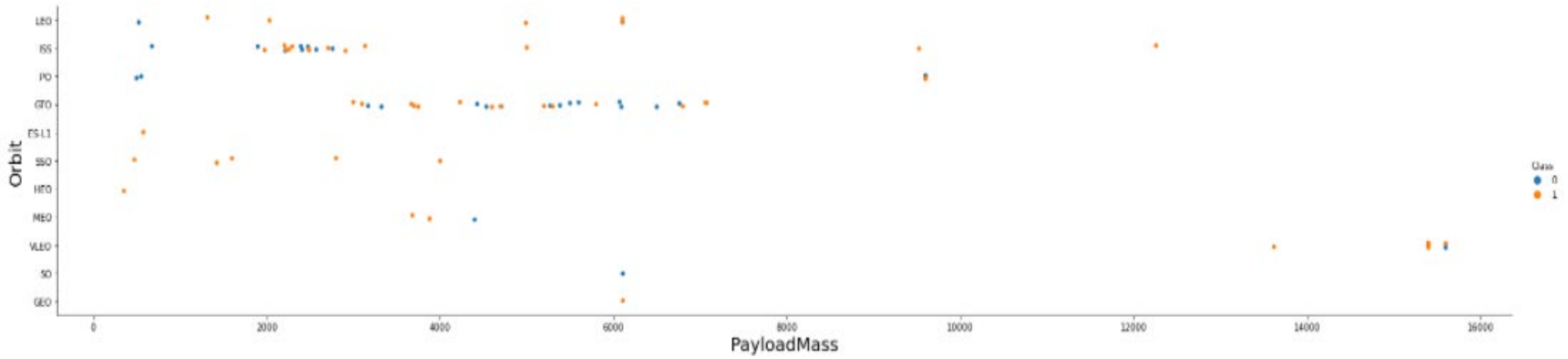
Flight Number vs. Orbit Type

✓The plot of the Flight Number versus Orbit type is shown below. We note that success in the LEO orbit is correlated with the number of flights, however there is no correlation between the number of flights and the GTO orbit.



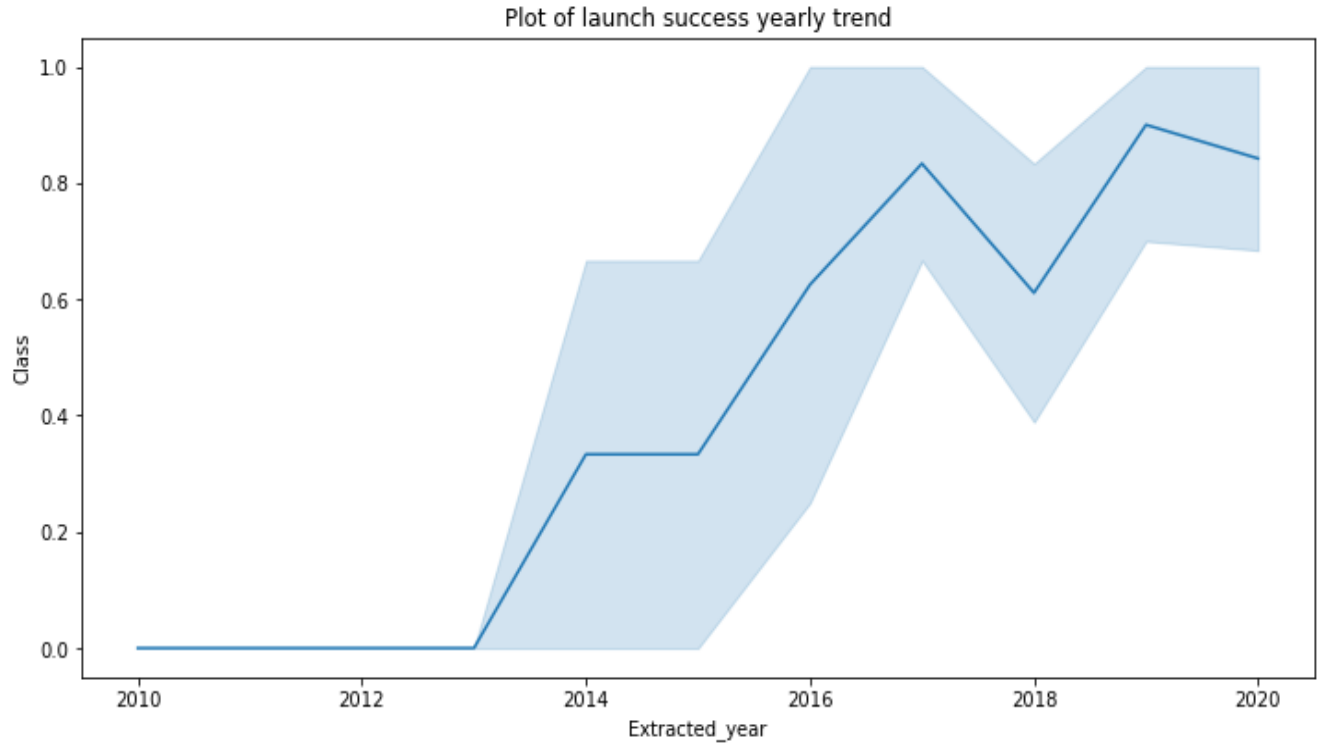
Payload vs. Orbit Type

✓We can see that successful landings with heavier payloads tend to occur more frequently in PO, LEO, and ISS orbits.



Launch Success Yearly Trend

- The plot reveals that the success rate has been rising since 2013 and will continue to do so until 2020.



All Launch Site Names

- To display just distinct launch sites from the SpaceX data, we utilized the keyword DISTINCT.

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'

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

- We performed the aforementioned query to show 5 records for launch sites that start with "CCA."

Total Payload Mass

- Using the following query, we arrived at the total payload carried by NASA boosters as being 45596:

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

- The average mass of the payload that booster version F9 v1.1 can carry was calculated to be 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
0	2928.4

First Successful Ground Landing Date

- We noted that the first successful landing result on the ground pad occurred on December 22, 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

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

- In order to find boosters that have successfully landed on drone ships, we employed the WHERE clause. We then used the AND condition to identify successful landings with payload masses larger than 4,000 but less than 6,000.

Total Number of Successful and Failure Mission Outcomes

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

- To filter for WHERE MissionOutcome was a success or failure, we used wildcards like "%."

Boosters Carried Maximum Payload

- Using a subquery in the WHERE clause and the MAX() method, we were able to identify the booster that had carried the most payload.

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

- For the year 2015, we filtered for failure landing outcomes in drone ship, their booster versions, and launch site names using combinations of the WHERE clause, LIKE, AND, and BETWEEN conditions.

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

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

- In order to filter for landing outcomes BETWEEN 2010-06-04 and 2010-03-20, we choose Landing outcomes and the COUNT of landing outcomes from the data.
- The landing results were categorized using the GROUP BY clause, and they were then put in decreasing order using the ORDER BY clause.

Section 4

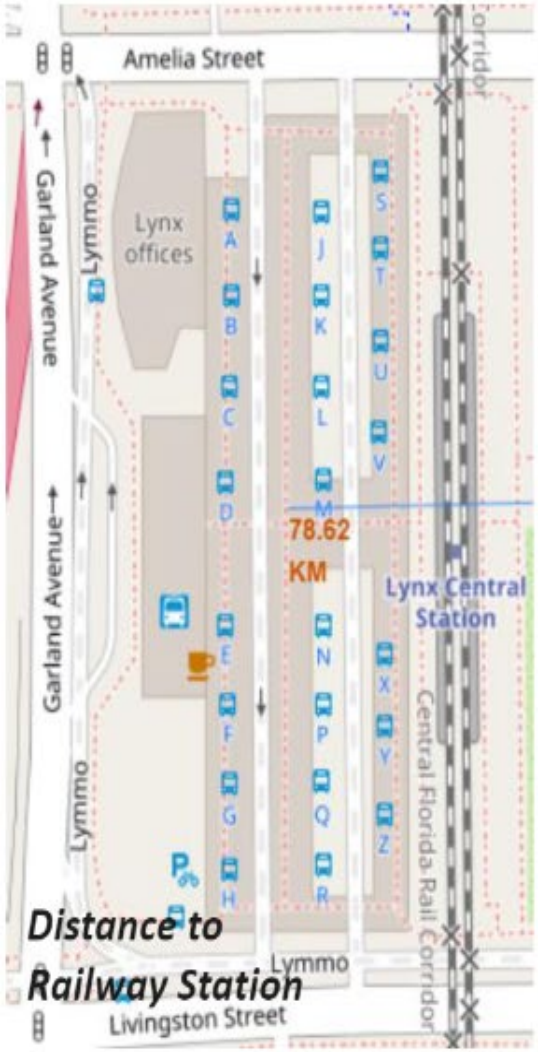
Launch Sites Proximities Analysis

All launch sites global map markers

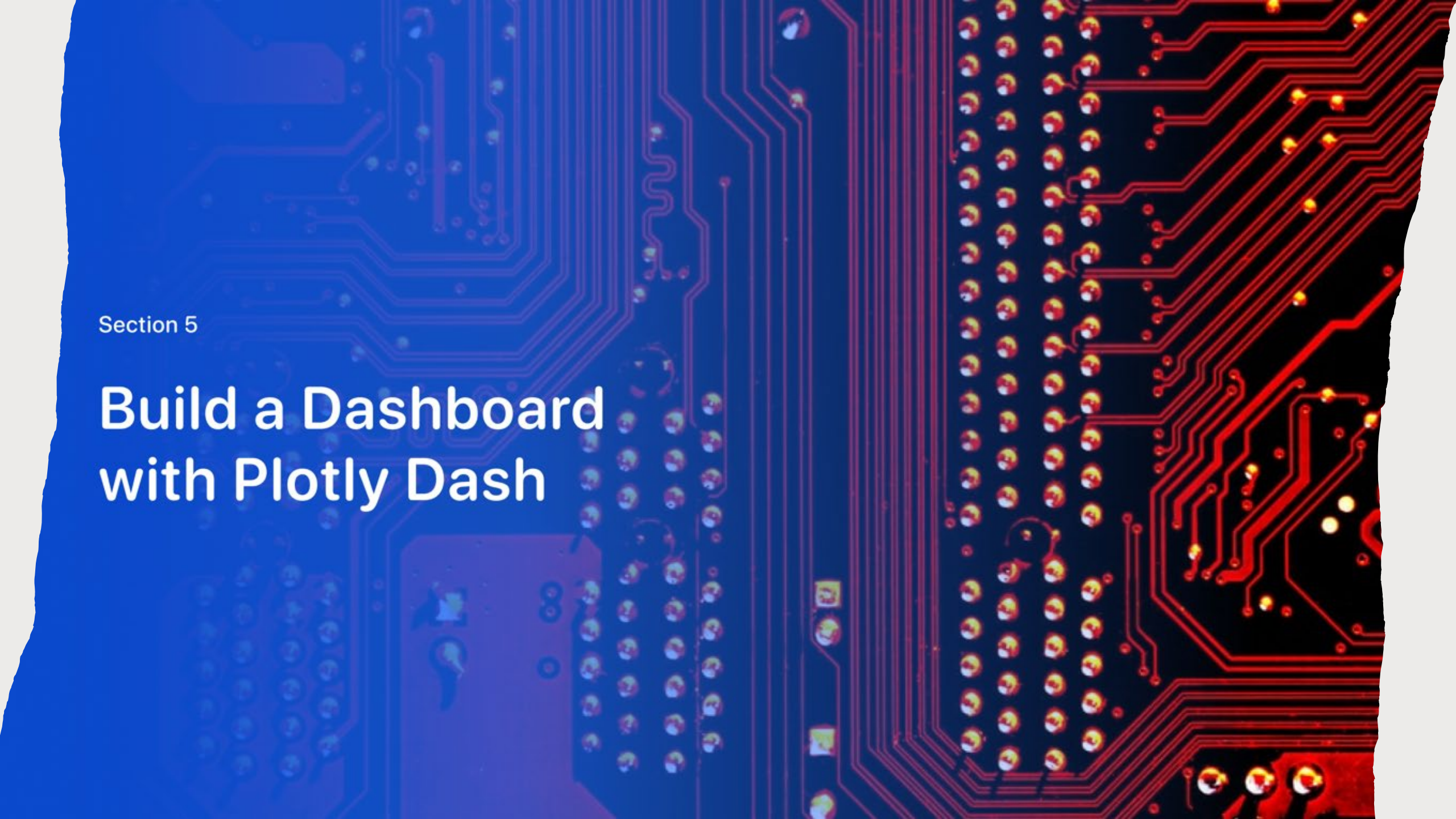
We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Markers showing launch sites with color labels

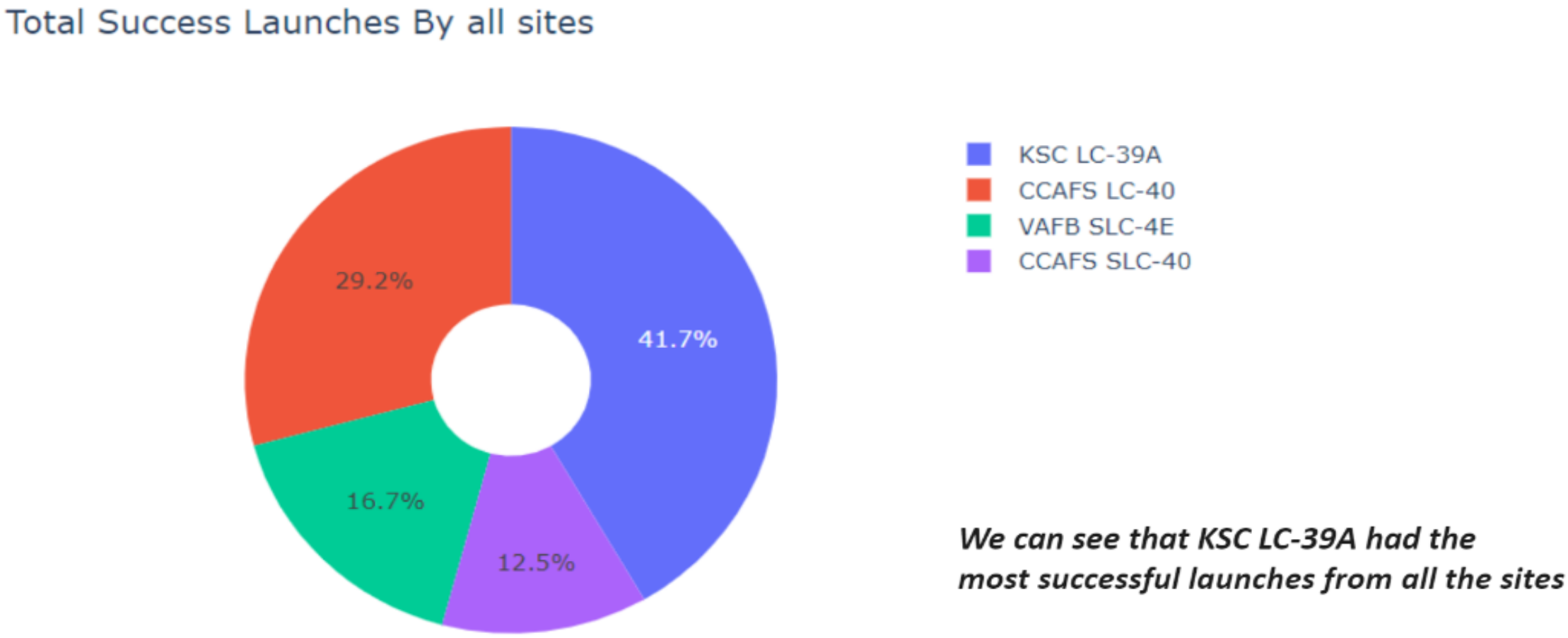
Launch Site distance to landmarks



- 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

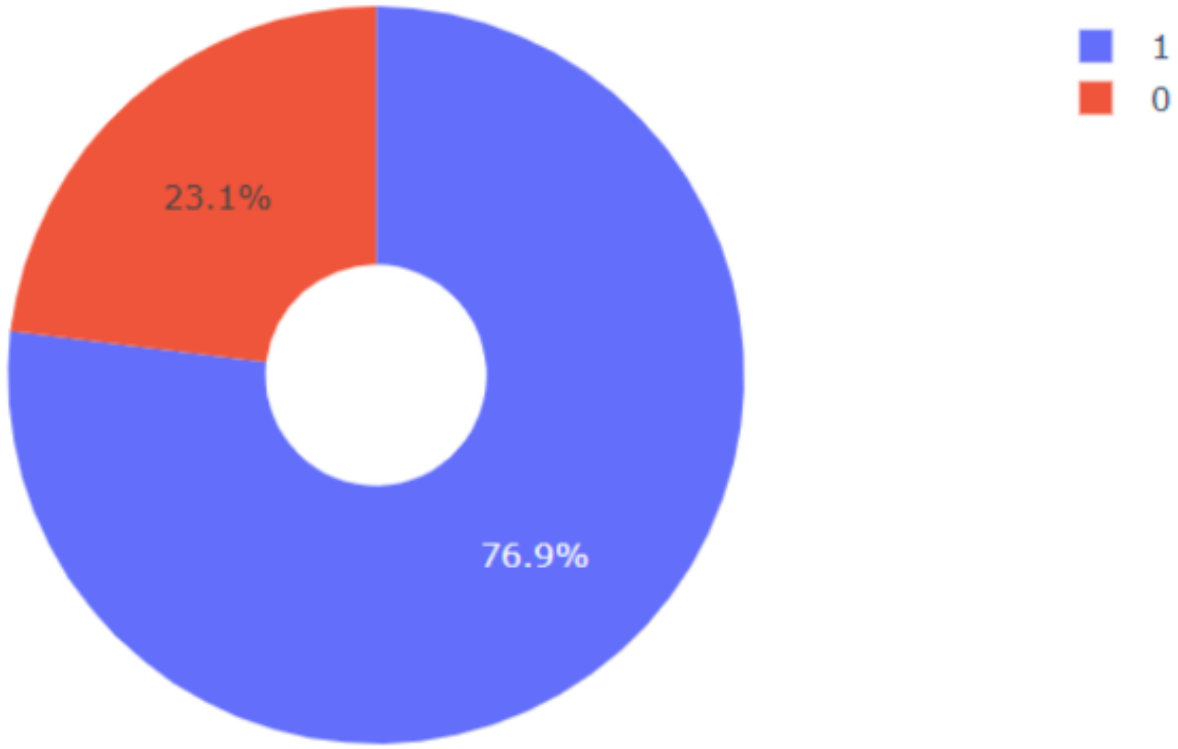

Section 5
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



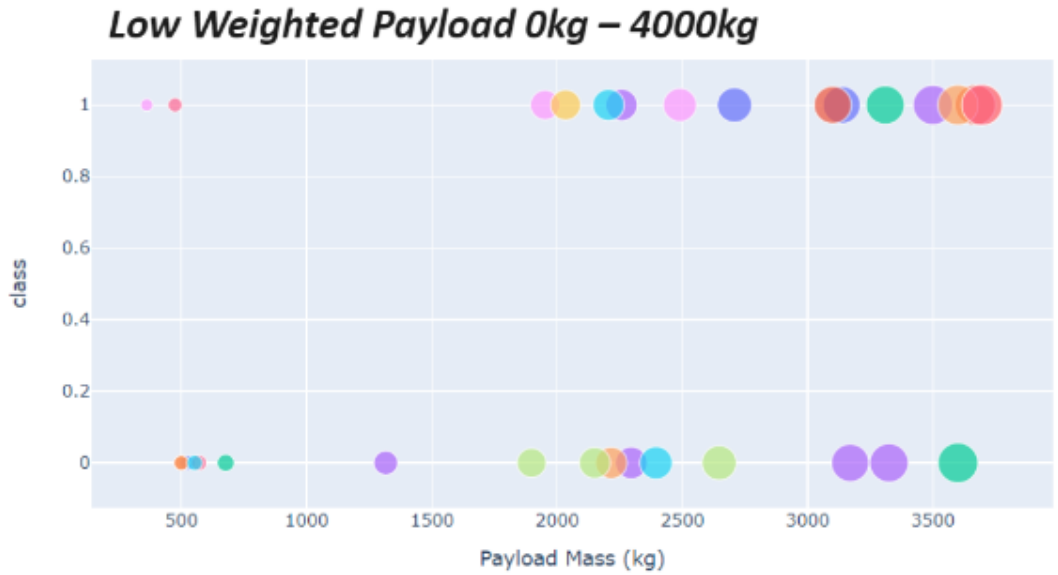
We can see that KSC LC-39A had the most successful launches from all the sites

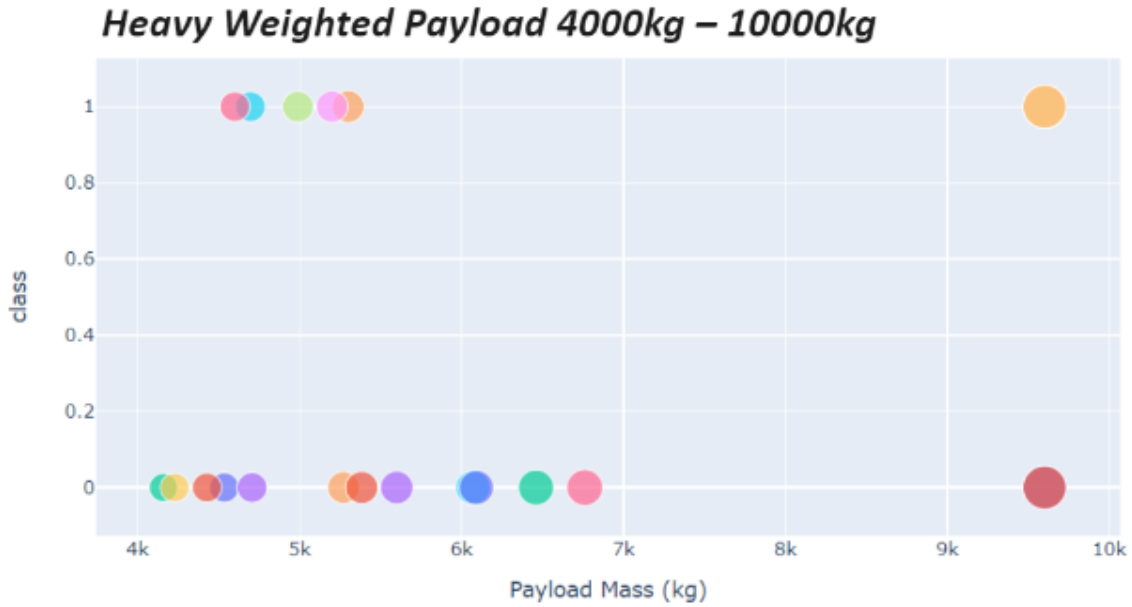
Pie chart showing the Launch site with the highest launch success ratio



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

Payload vs. Launch Outcome scatter plot for all sites, with various payloads selected using the range slider





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

Section 6

Predictive Analysis (Classification)

Classification Accuracy

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

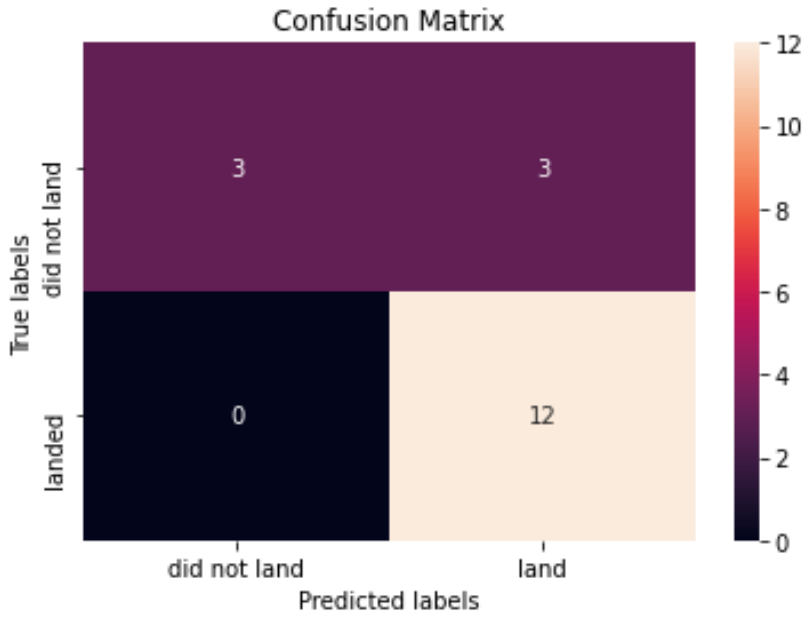
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 decision tree classifier's confusion matrix demonstrates that it is capable of differentiating between the various classes. False positives, or failure landings reported as successful landings by the classifier, are the main issue.



Conclusions

What we may infer is that:

1. The success rate at a launch site increases with the size of the flight quantity.
2. The success rate of launches increased from 2013 to 2020.
3. The highest success rate was in the ES-L1, GEO, HEO, SSO, and VLEO orbits.
4. Of all the sites, KSC LC-39A had the most successful launches.
5. The best machine learning algorithm for this task is the decision tree classifier.

Thank you!