Winning Space Race with Data Science

Alex J Torres

May 13th, 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API and Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA) with SQL and 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 while other providers have a cost upward of 165 million dollars each. Much of the savings is because Space X can reuse the first stage of launch. 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. The goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Questions we want to find answers for:

  - What factors determine if the rocket will land successfully?
  - What interaction, among various features, will determine the success rate of a landing?
  - What conditions needs to be in place to ensure 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 Data visualization and SQL

Perform interactive visual analytics using Folium and Plotly Dash

Perform predictive analysis using classification models

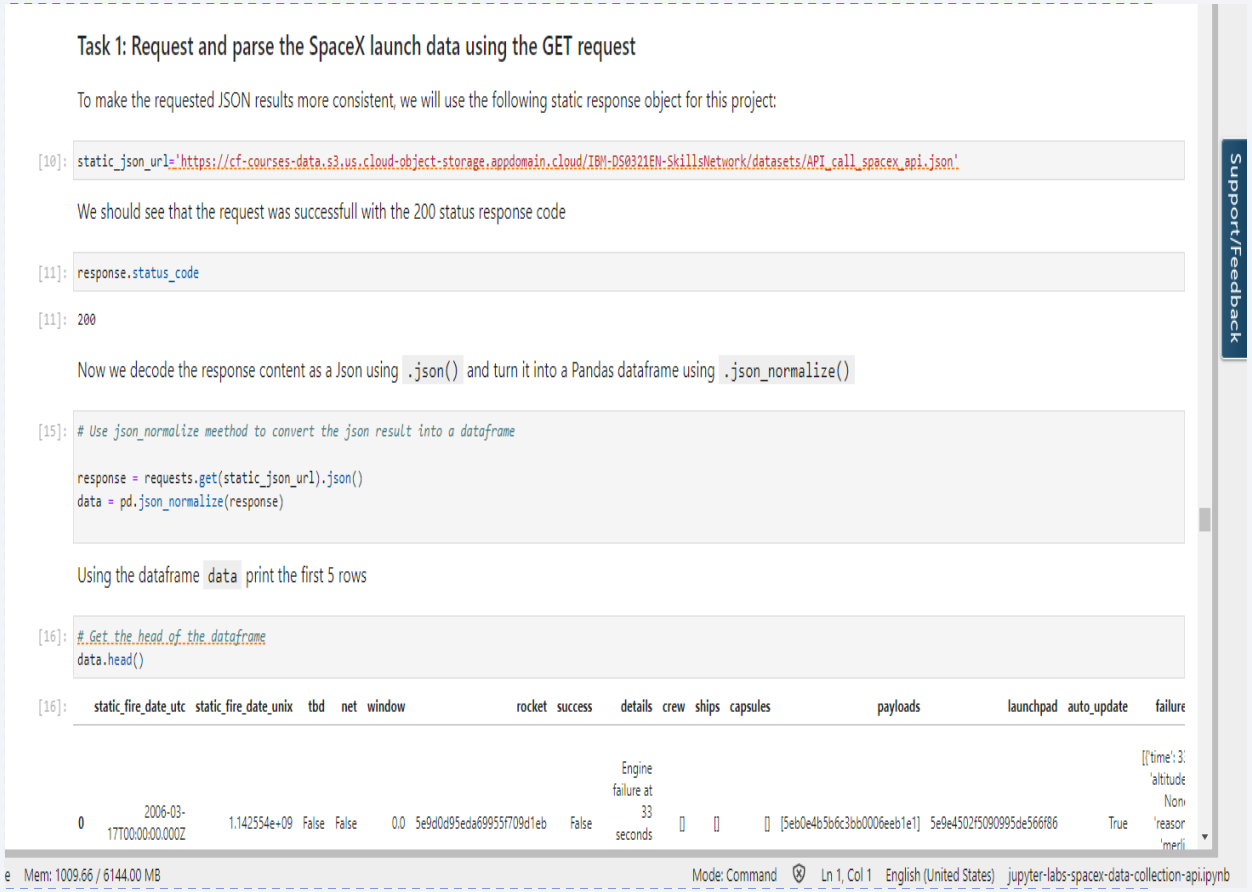
- How to build, evaluate, and determine the best classification models

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 turned it into a Pandas dataframe using .json_normalize().
  - Then, we cleaned the data, checked for missing values, and filled in missing values when 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 further analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data, and did some data wrangling and formatting.
- Link: https://github.com/AlexTorres1994/IBM-Course-10-SpaceX-Falcon-Capstone/blob/main/jupyter-labs-spacex-data-collection-api%20(1).ipynb



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:

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

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

```
[11]: response.status_code
[11]: 200
```

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

```
[15]: # Use json_normalize meethod to convert the json result into a dataframe

response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

Using the dataframe `data` print the first 5 rows

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

[16]:

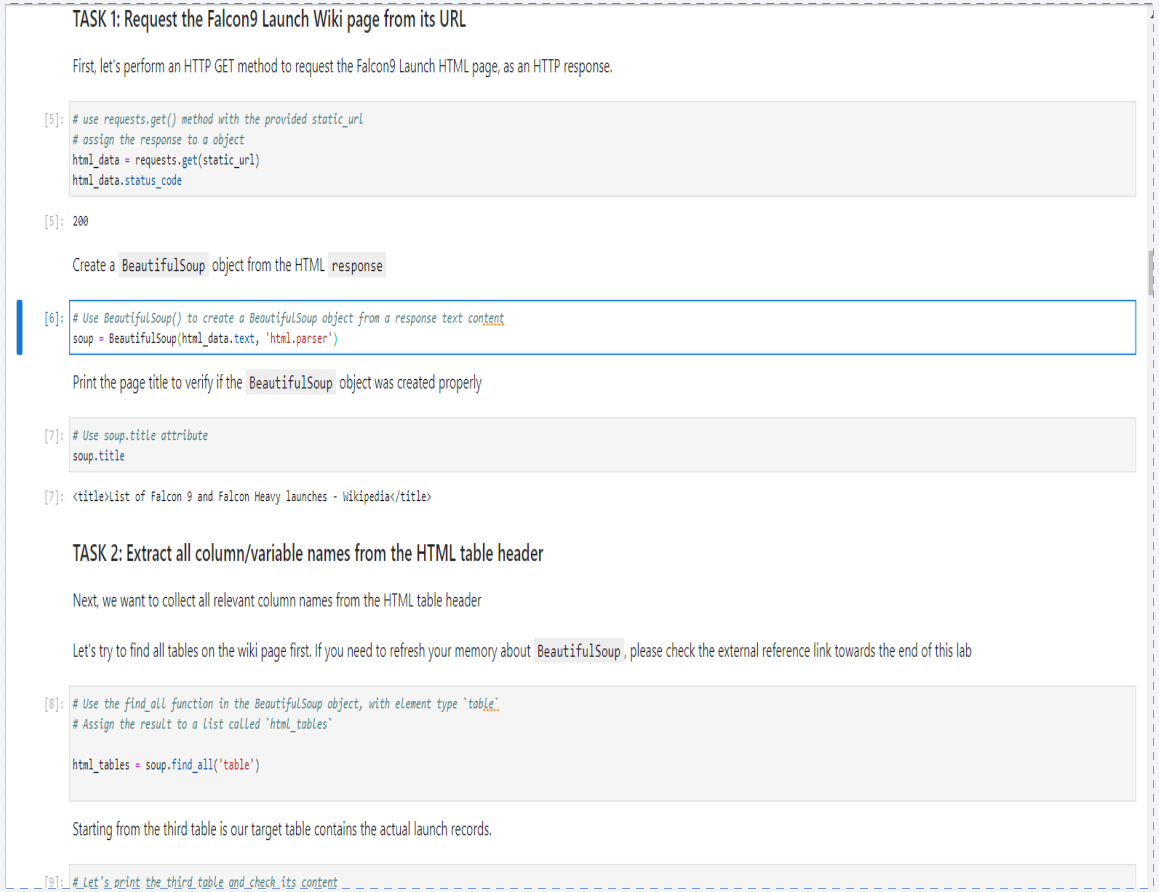
	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules	payloads	launchpad	auto_update	failure
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e4502f5090995de566f86	True	[{'time': 3: 'altitude Non 'reason 'merli

Support/Feedback

e Mem: 1009.66 / 6144.00 MB Mode: Command Ln 1, Col 1 English (United States) jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

- We applied web scraping to web scrap Falcon 9 launch records with BeautifulSoup.
- We analyzed the table and converted it into a pandas dataframe.
- Link: https://github.com/AlexTorres1994/IBM-Course-10-SpaceX-Falcon-Capstone/blob/main/jupyter-labs-webscraping.ipynb



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.

```
[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
```

```
[5]: 200
```

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

```
[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

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

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

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

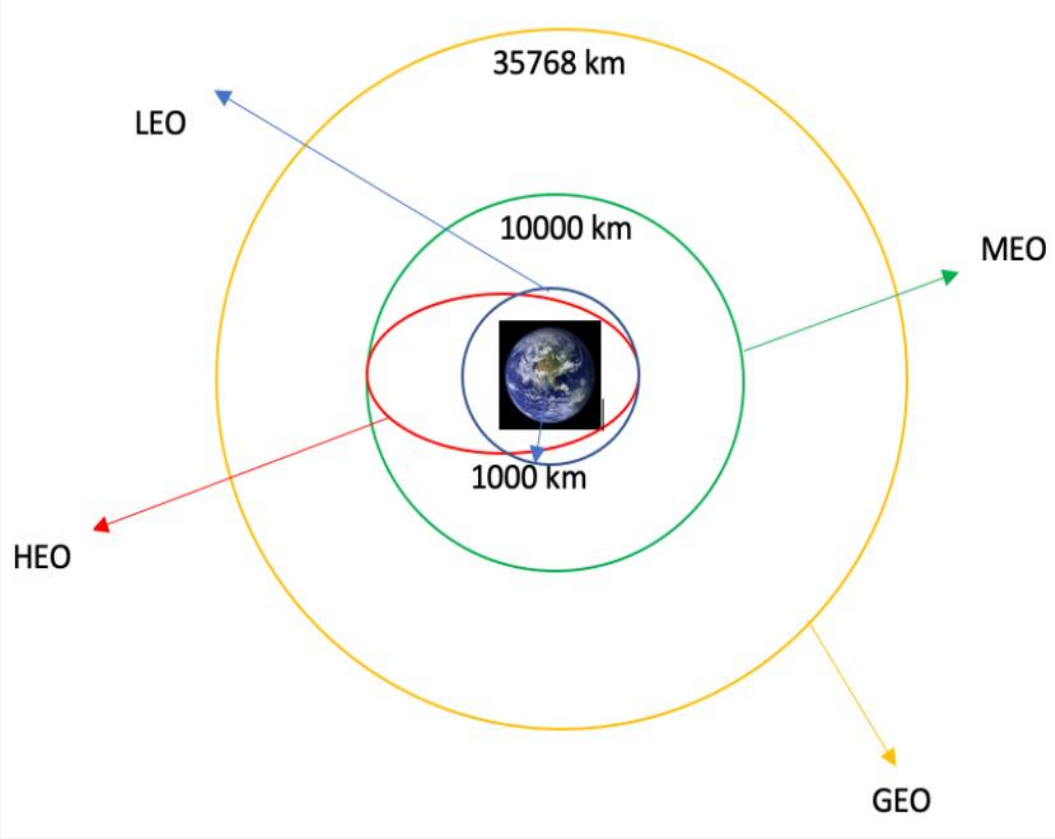
```
[8]: # Use the find_all function in the BeautifulSoup object, with element type `table`
     # Assign the result to a list called `html_tables`

     html_tables = soup.find_all('table')
```

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

```
[9]: # Let's print the third table and check its content
```

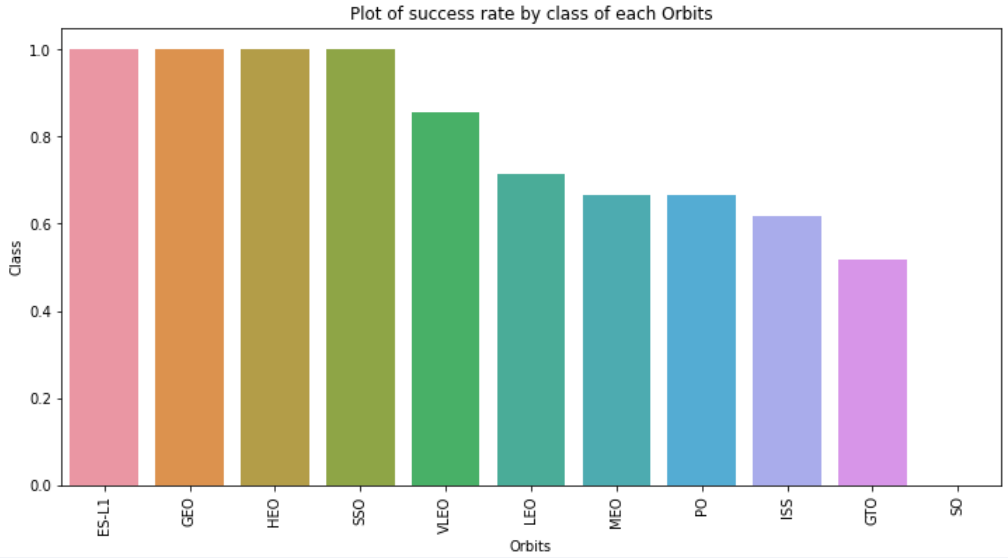
Data Wrangling

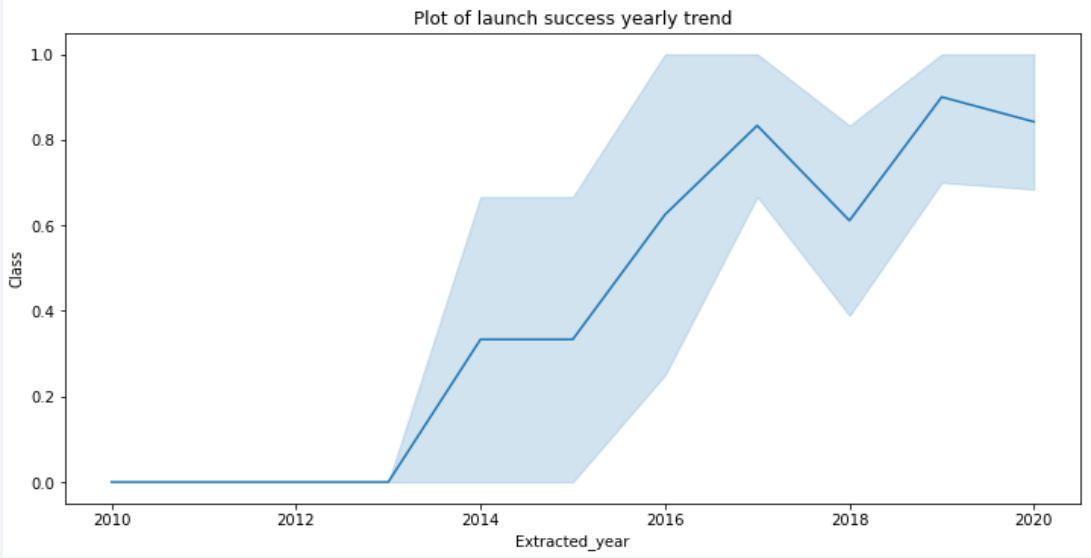
- We performed exploratory data analysis and determined the training labels.
- The number of launches at each site and the occurrence of each orbits were calculated.
- Landing outcome label from outcome column were created and the results exported to a csv.
- Link: https://github.com/AlexTorres1994/IBM-Course-10-SpaceX-Falcon-Capstone/blob/main/3.%20wrangling.ipynb



EDA with Data Visualization

- The data was explored by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number, orbit type, and the launch success yearly trend.
- Link: https://github.com/AlexTorres1994/IBM-Course-10-SpaceX-Falcon-Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb





EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the Jupyter notebook.
- EDA with SQL was applied to get insight from the data and wrote queries to find out:
  - 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 of the drone ship, their booster version, and launch site names.
- Link: https://github.com/AlexTorres1994/IBM-Course-10-SpaceX-Falcon-Capstone/blob/main/EDA_SQL.ipynb

Build an Interactive Map with Folium

- We marked all launch sites and added map objects such as markers, circles, and lines to mark the success or failure of launches for each site on the folium map.
- The launch outcomes (failure or success) were assigned to a 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 rates and calculated the distances between a launch site to it's proximities to various locations on a map.
- Link: https://github.com/AlexTorres1994/IBM-Course-10-SpaceX-Falcon-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

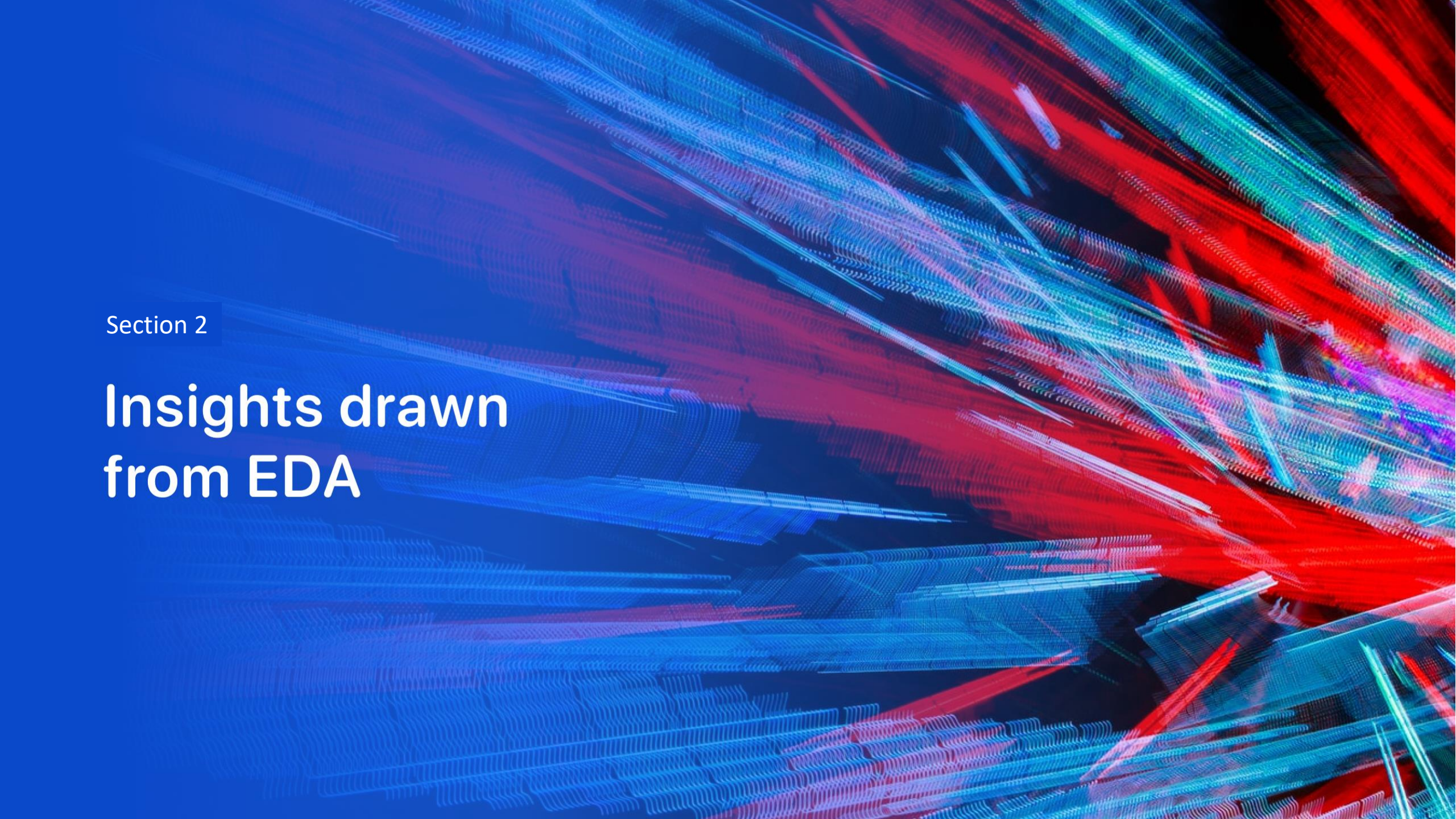
- We built an interactive dashboard with Plotly dash and created pie charts showing the total launches by certain sites.
- Afterwards, we plotted the findings in a scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster versions.
- Link: https://github.com/AlexTorres1994/IBM-Course-10-SpaceX-Falcon-Capstone/blob/main/app.py

Predictive Analysis (Classification)

- We loaded the data using Numpy and Pandas, transformed the data, and split our data into training andtesting.
- We built different machine learning models and tuned different hyperparameters using GridSearchCV.
- Accuracy was utilized as the metric for our model and improved the model using feature engineering and algorithm tuning.
- Finally, we found the best performing classification model for the project.
- Link: https://github.com/AlexTorres1994/IBM-Course-10-SpaceX-Falcon-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

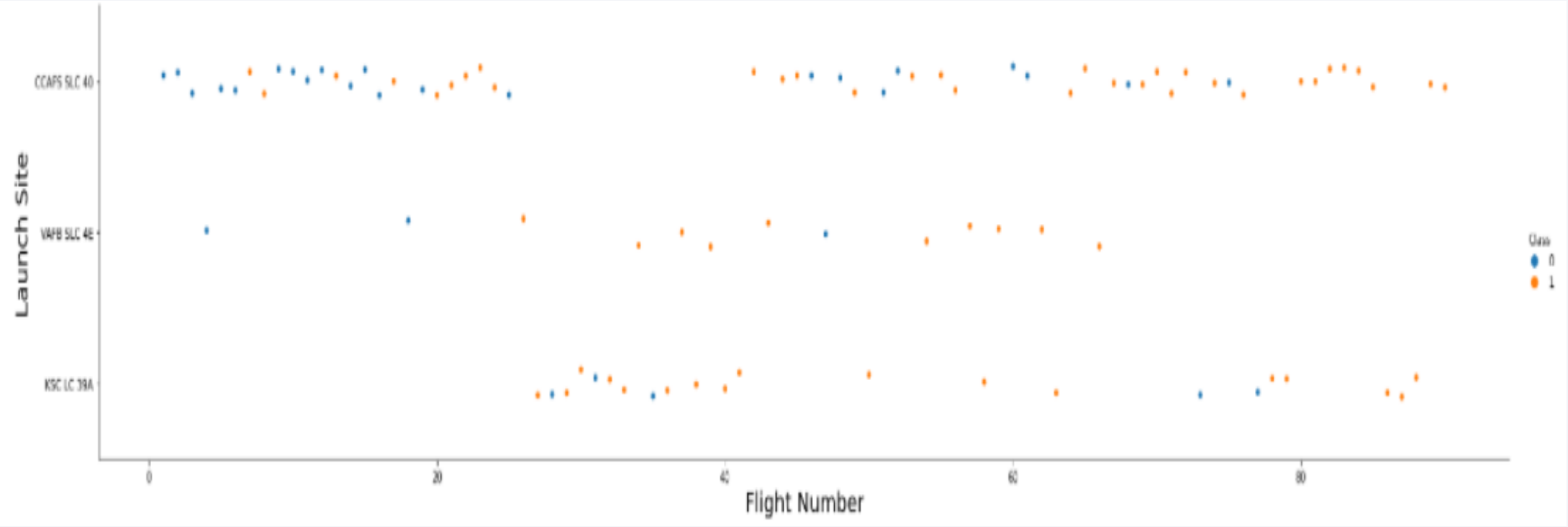
Results

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


Section 2
Insights drawn from EDA

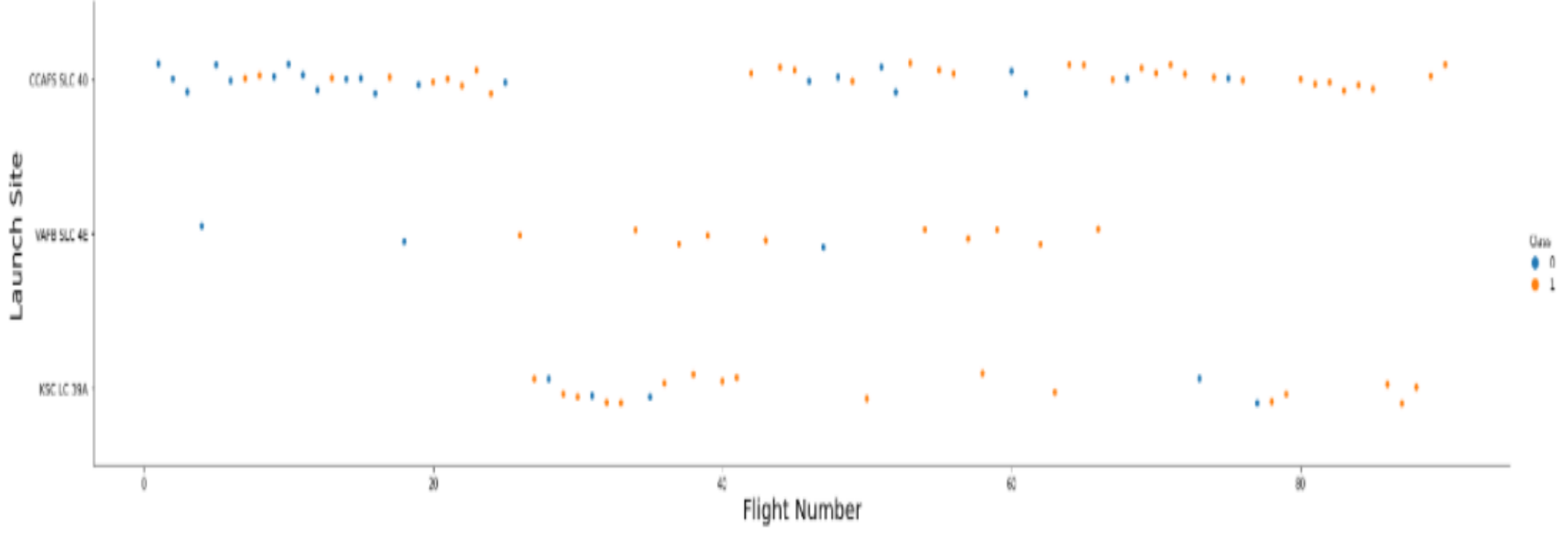
Flight Number vs. Launch Site

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



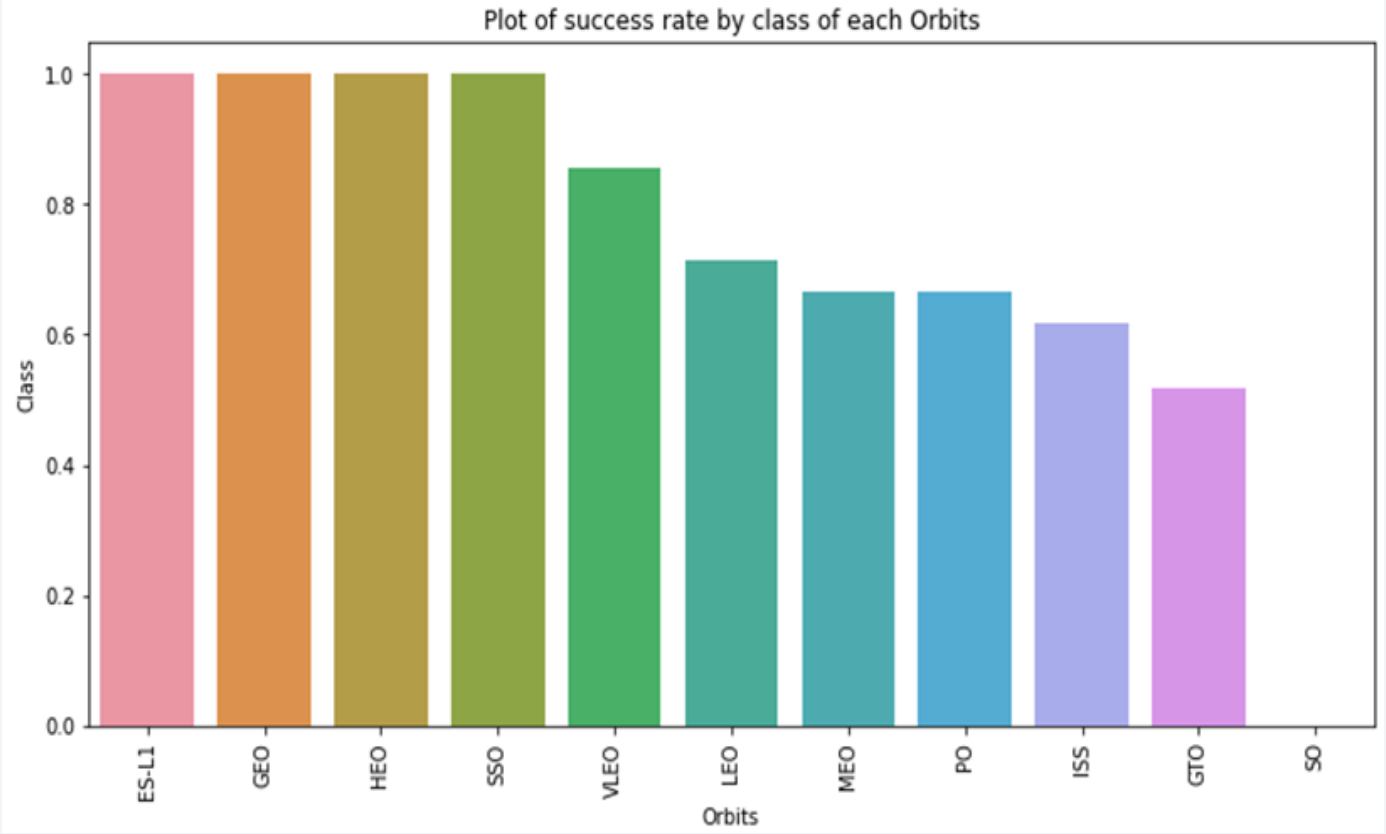
Payload vs. Launch Site

- From the plot below, we can determine that the greater the payload mass at the launch site, the higher the success rate for the rocket.



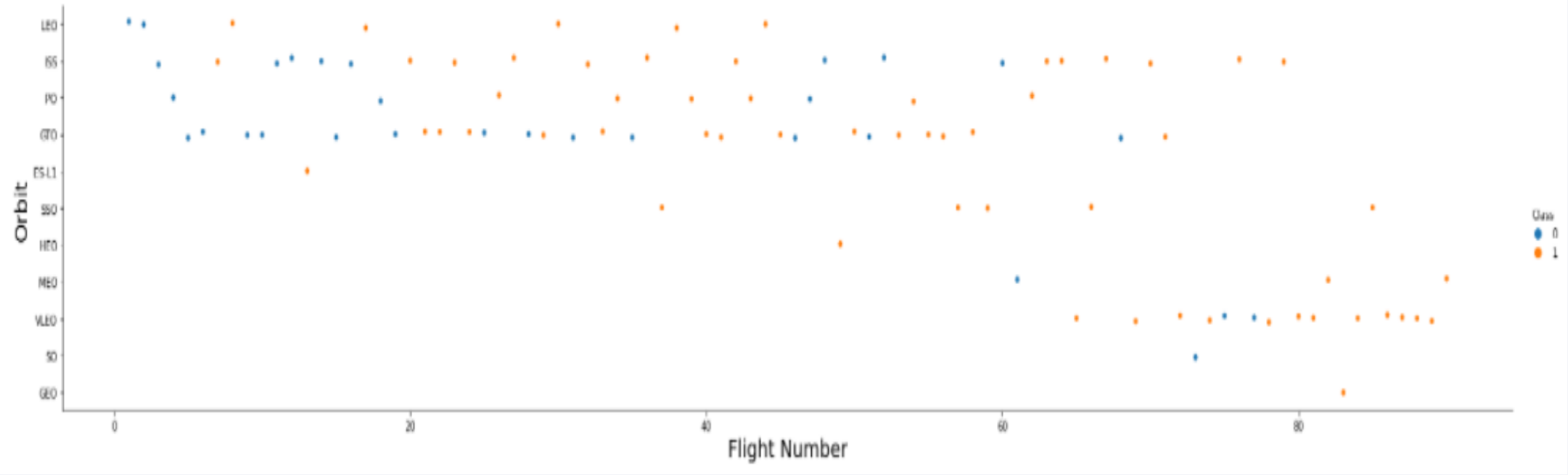
Success Rate vs. Orbit Type

- From the bar graph, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate of all the orbit types.



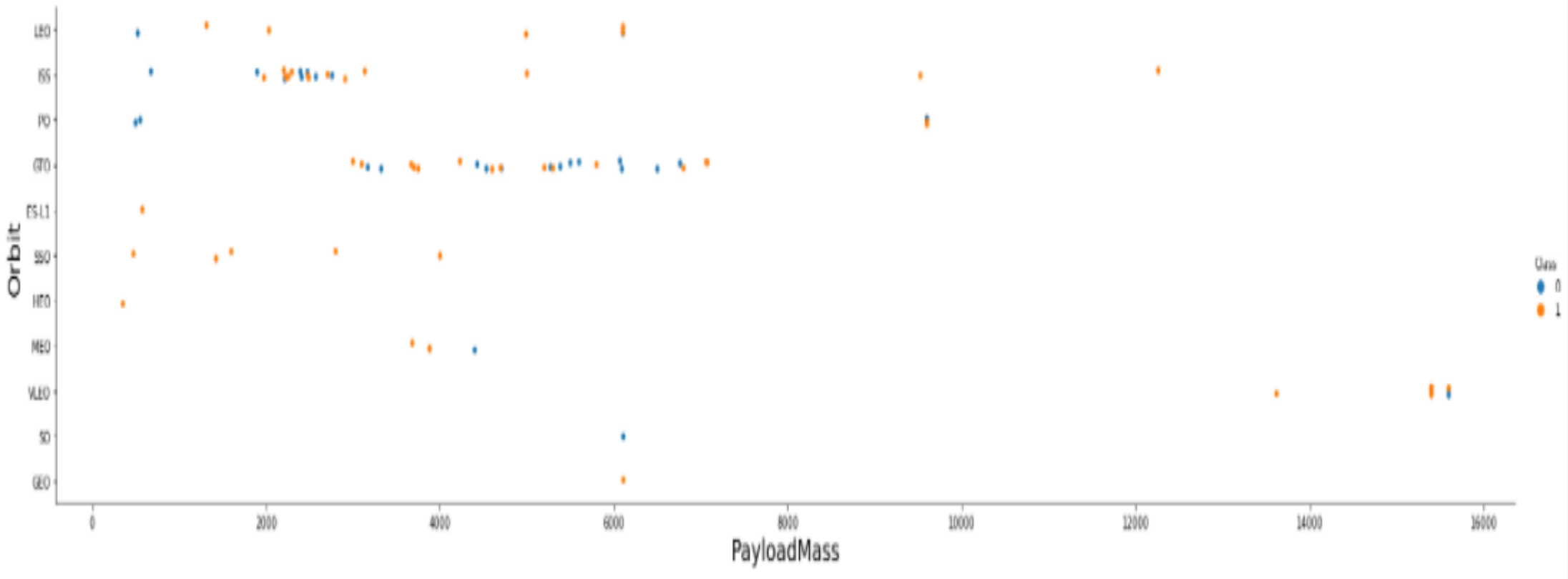
Flight Number vs. Orbit Type

- The plot below exhibits the Flight Number vs. Orbit type. We can observe that in the LEO orbit, success is related to the number of flights. However, in the GTO orbit, there is no relationship between flight number and the orbit.



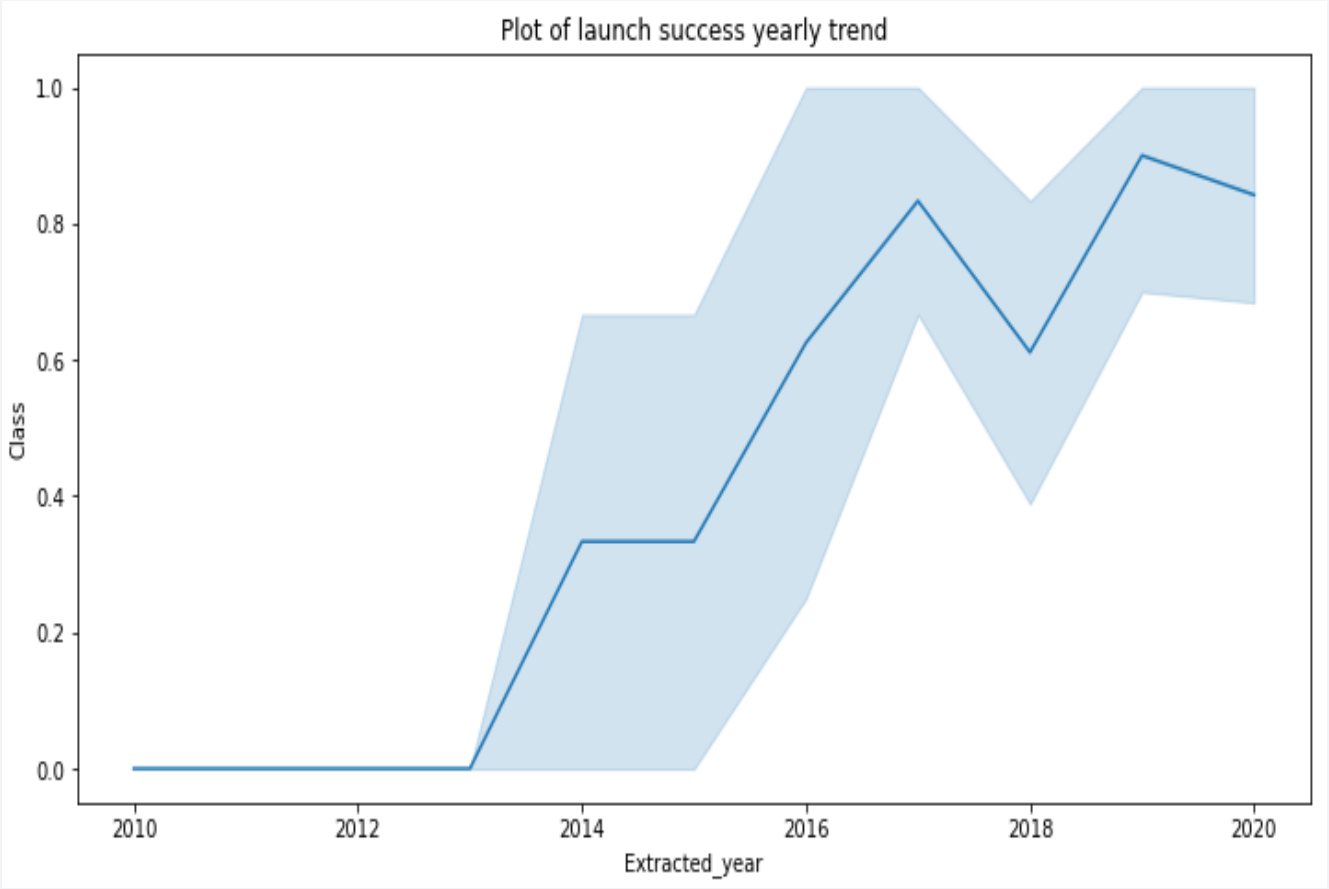
Payload vs. Orbit Type

- We can observe in the plot below that with heavy payloads, the success of landings are higher for PO, LEO, and ISS orbit types.



Launch Success Yearly Trend

- From the line graph, we can observe that the success rate of launch continuously increased starting in 2013, all the way until the end of 2020.



All Launch Site Names

- Utilizing the SELECT function with the key word DISTINCT allowed us to only show unique 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'

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

- In the above query, we utilized the SELECT function with WHERE and LIMIT clauses to display the first 5 records where launch sites begin with ‘CCA’

Total Payload Mass

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

- In the query above, utilizing the SELECT SUM function with the WHERE clause and LIKE condition, we can calculate the total payload carried by boosters from NASA as 45,596.

Average Payload Mass by F9 v1.1

- In this query, using the SELECT AVG function with the WHERE clause, we can calculate the average payload mass carried by booster version F9 v1.1 as 2,928.4

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

- In this query, using the SELECT MIN function, with the WHERE clause and LIKE condition, we determined that the date of the first successful landing on the ground pad was 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 this query, using the SELECT function, with WHERE clause, specifying Success for drone ship with AND condition to filter for specific boosters and payloads, we can determine the successful drone ship landings with a payload mass greater than 4,000 but less than 6,000.

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

- In this query, we used the SELECT COUNT function, with the WHERE clause and LIKE condition with the wildcard % to filter out the MissionOutcome that were successful or not.

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)
```

Out[16]:

The total number of successful mission outcome is:

	successoutcome
0	100

The total number of failed mission outcome is:

	failureoutcome
0	1

Boosters Carried Maximum Payload

- In this query, we used the SELECT function, with the WHERE clause using the subquery MAX() function to determine the booster that carried the maximum 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

- In this query, using a combination of the SELECT function, WHERE clause, and the LIKE, AND, and BETWEEN conditions, we were able to filter for only the failed landing outcomes in the drone ships, their booster version, and their launch site in the 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

- In this query, using the SELECT and COUNT function, with the WHERE DATE BETWEEN clause, we can filter the landing outcomes between June 6, 2010 and March 20, 2017.
- Using the GROUP BY and ORDER BY clauses, we can group and order the landings in descending order of landing outcomes.

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

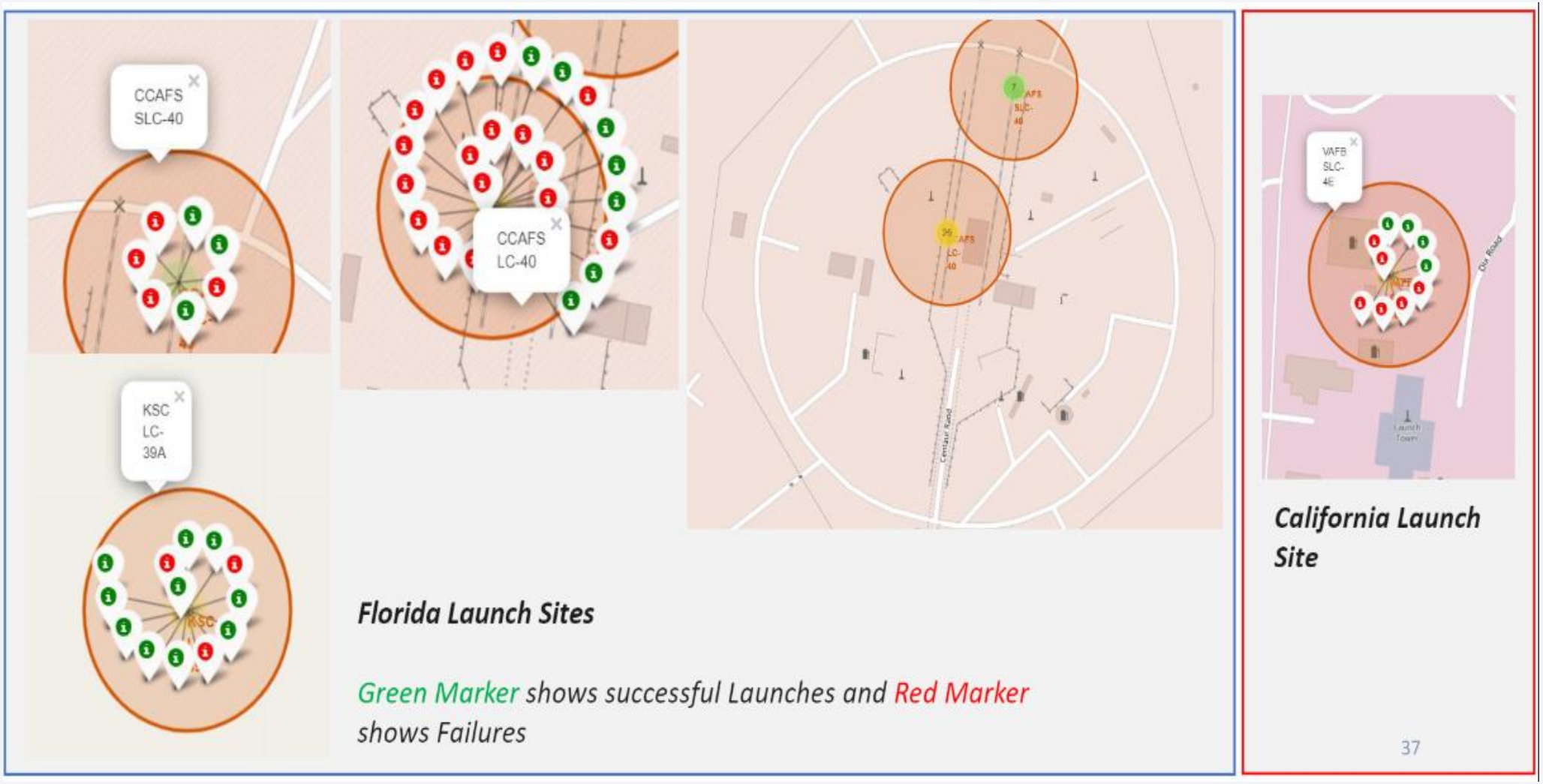
Section 3

Launch Sites Proximities Analysis

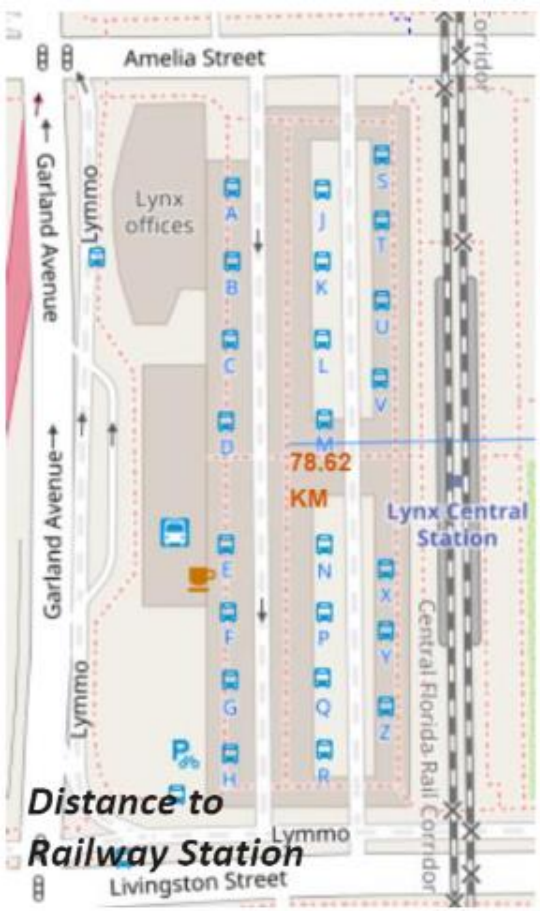
All launch sites global map markers

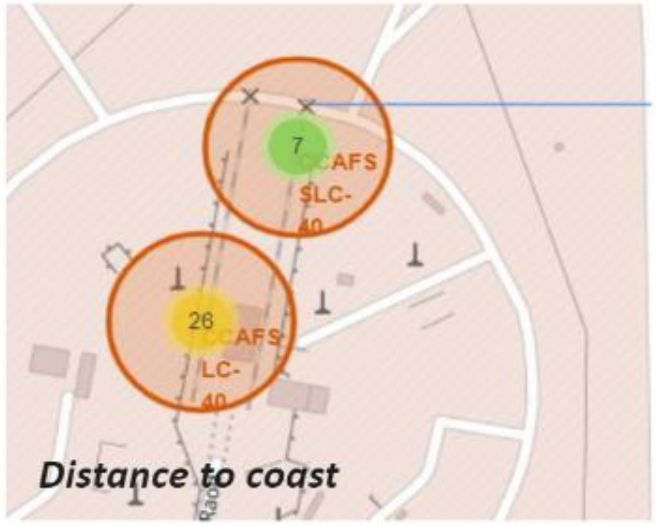
- In the photo above, we can see the SpaceX launch sites are in both coasts of the United States of America, being situated in Florida and California.

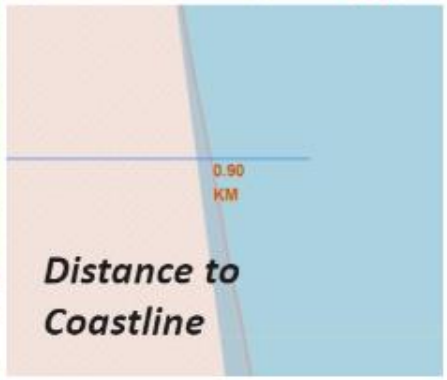
Markers showing launch sites with color labels



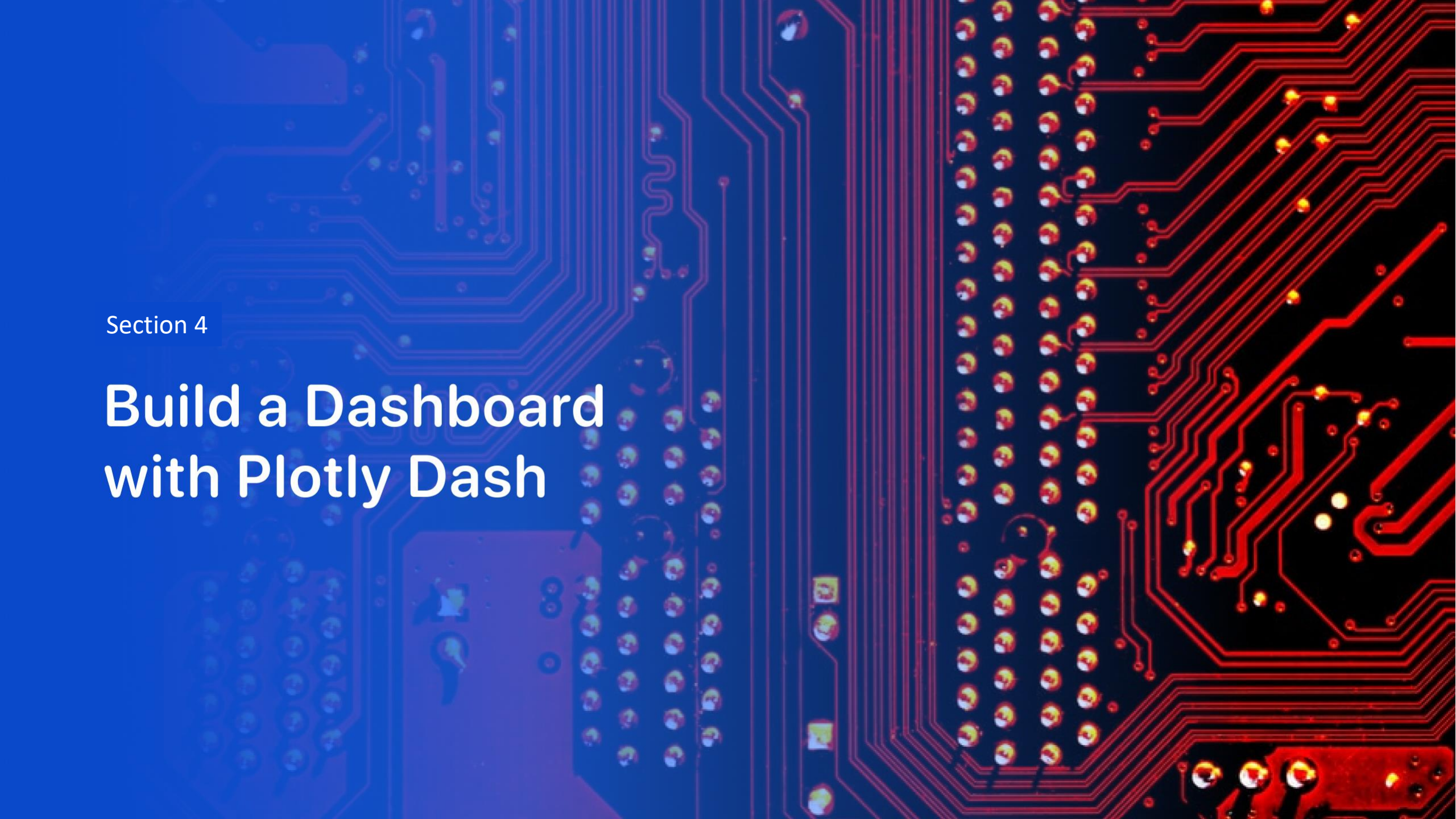
Launch Site distance to landmarks



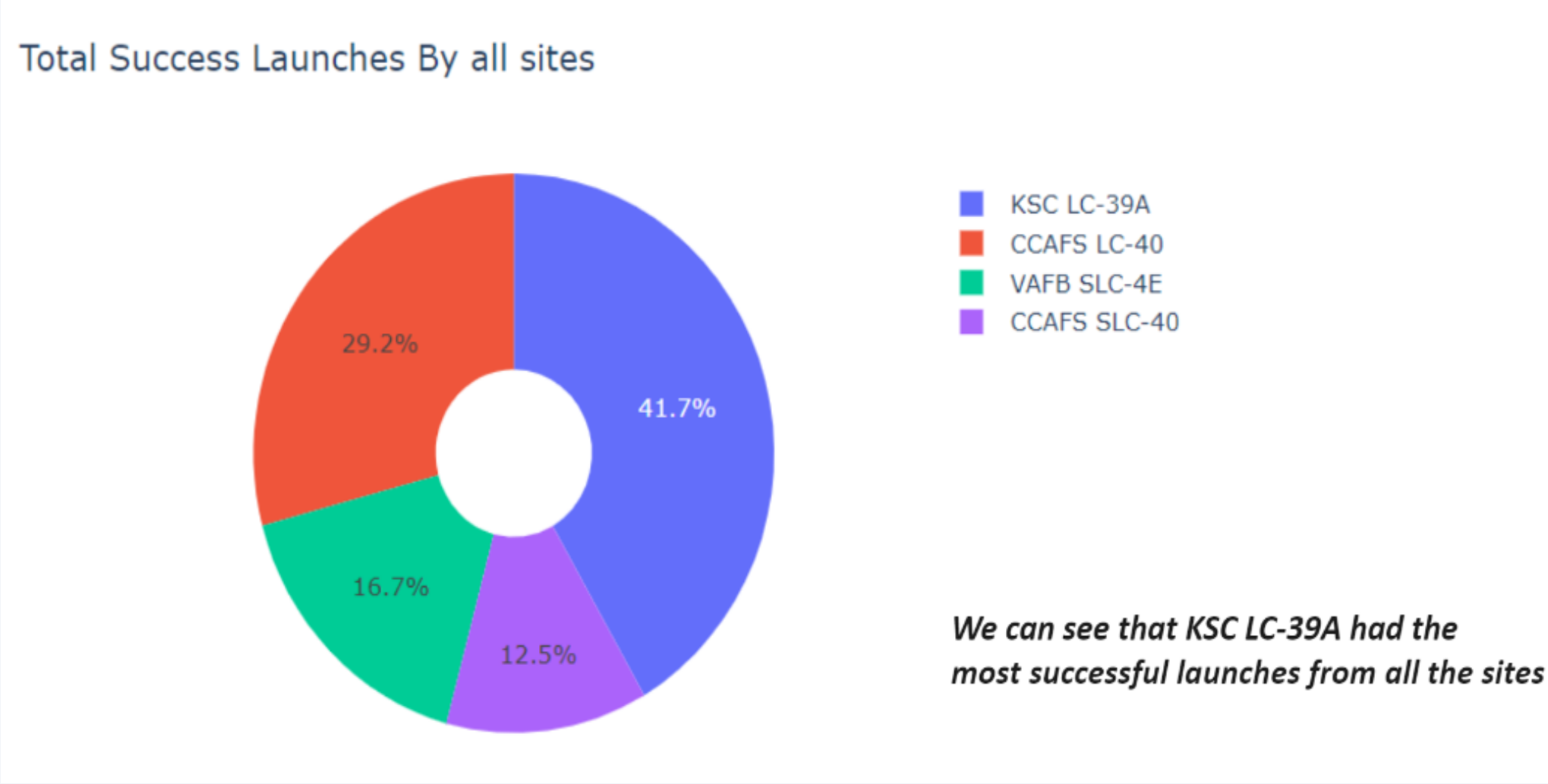




- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

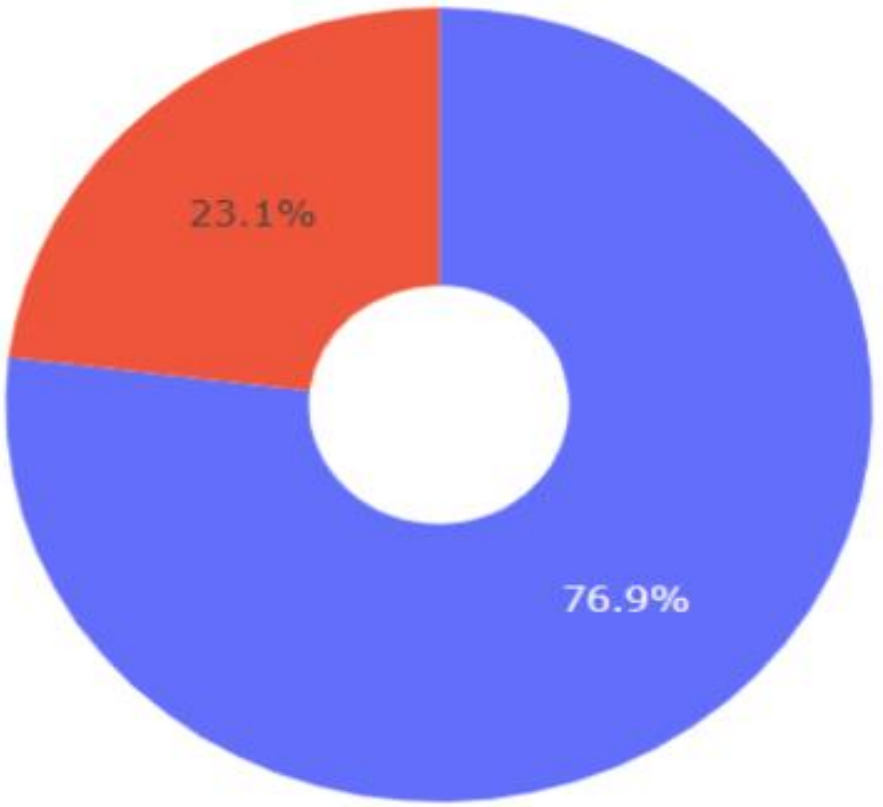

Section 4
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



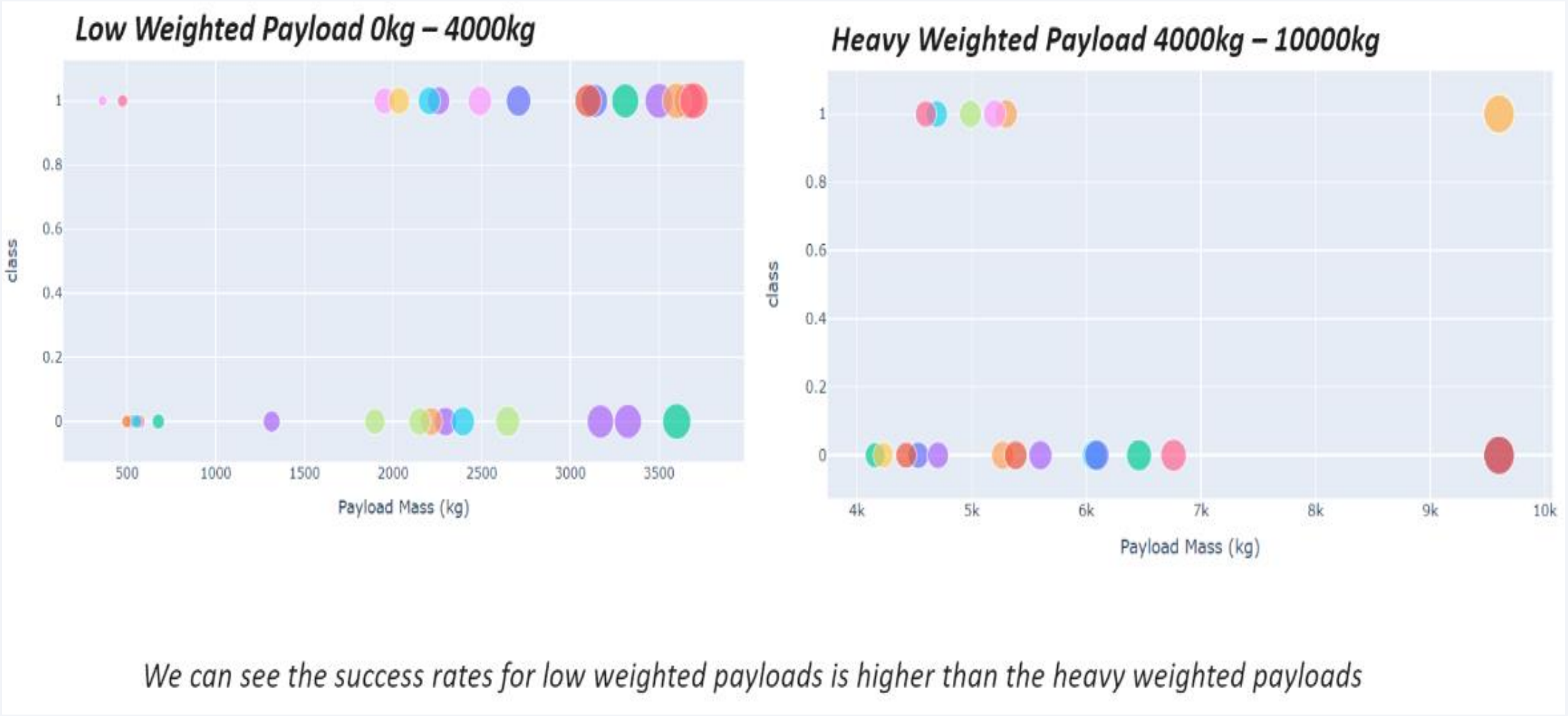
We can see that KSC LC-39A had the most successful launches from all the sites

Pie chart showing the KSC LC-39A Launch site which had the highest launch success ratio



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, with different payload selected in the range slider



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

Section 5

Predictive Analysis (Classification)

Classification Accuracy

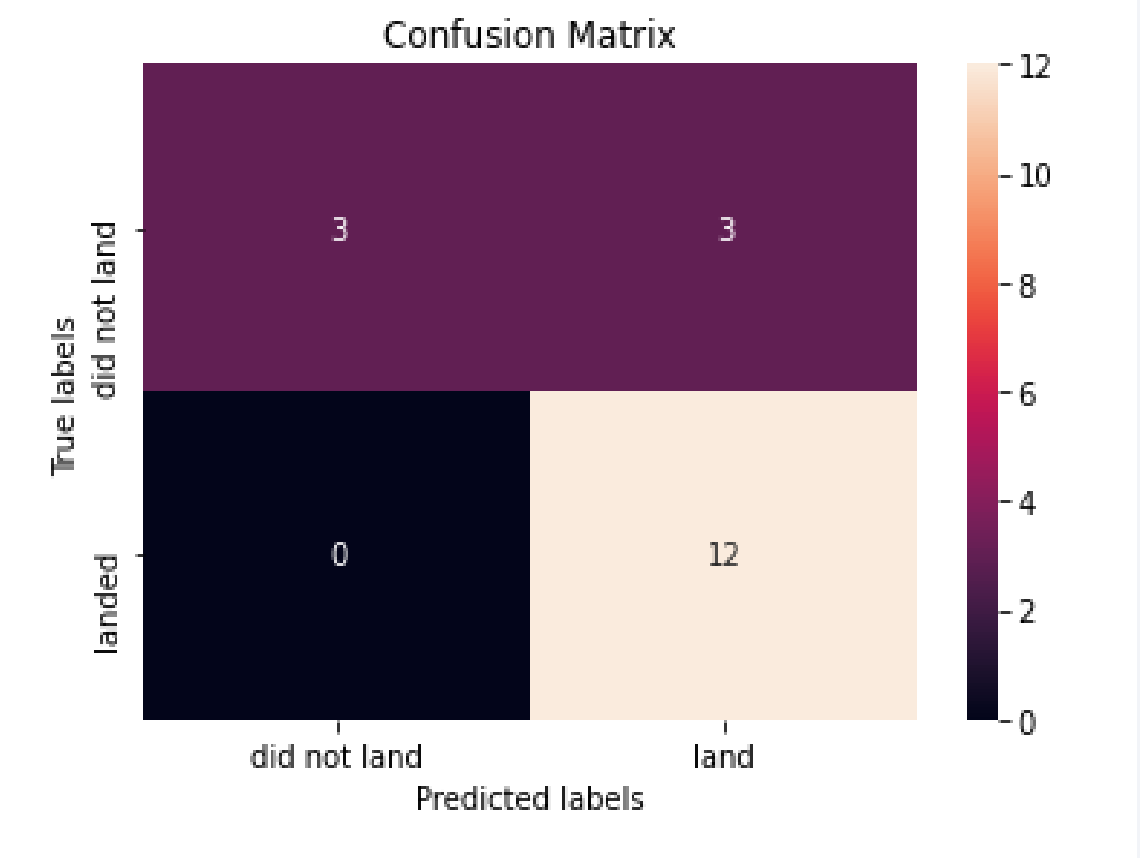
- The decision tree classifier is the classifying model with the highest accuracy in classification compared to all the other models.

```
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 this classifier can help distinguish between different classes.
- However, one major problem with this classifier was the false positives (unsuccessful landings) that were marked and included in the Decision Tree classifier.



Conclusions

After all the Data collecting, webscraping, wrangling, EDA SQL and visualization, we can conclude that:

- The larger the flight amount at a launch site, the greater the success rate was at the launch site.
- Launch success rate started to increase in 2013 until 2020, which was the end of the graph.
- Orbits ES-L1, GEO, HEO, SSO, and VLEO had the most success rates compared to all the other orbits.
- KSC LC-39A had the most successful launches of all of the sites.
- The Decision tree classifier is the best machine learning algorithm for the project.

Thank you!