IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Suraj Kunthu
June 16, 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection with 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
  - EDA Results
  - Interactive Analytics screenshots
  - Machine Learning Results

Introduction

- Project background and context
  - In this capstone, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.
- Problems you want to find answers
  - Factors that determine successful rocket landing
  - What features help determine successful landing rate?
  - What are the ideal operational conditions for successful landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected with SpaceX API and BS4 to web scrape from Wikipedia
- Perform data wrangling
  - 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 dataset was collected by using the SpaceX REST API and web-scrapped from Wikipedia using the BeautifulSoup package.
- The SpaceX API data was then normalized to a dataframe
- Column and variable names were extracted from the Wikipedia Data to create a dataframe

Data Collection – SpaceX API

GET Request from SpaceX API

Normalize json response to a data frame

Clean and scrub data

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

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

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloa
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```


Github: https://github.com/surajkunthu/Data-Science-Projects/blob/main/IBM%20DS%20Professional%20Cert/Capstone/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

GET Request from Wikipedia

Use BeautifulSoup to create an object from HTML response

Extract column and variable names from HTML header

```
# use requests.get() method with the provided static_url
# assign the response to a object
falcon9WikiData = requests.get(static_url).text
```

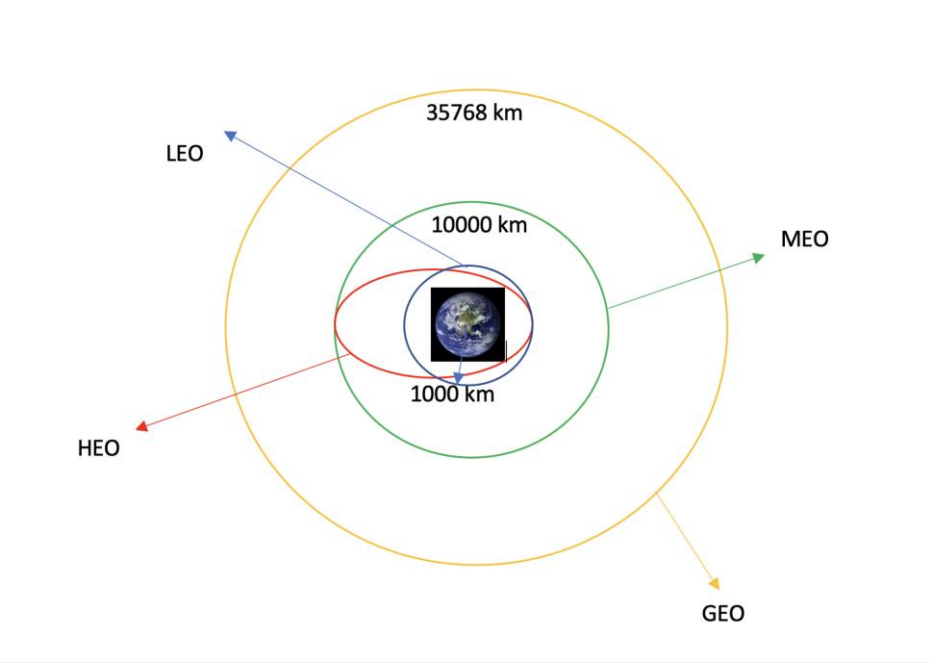
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(falcon9WikiData, 'html.parser')
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        print('-----------------------------------------------------')
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            print(flight_number)
            launch_dict['Flight No.'].append(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            print(date)
            launch_dict['Date'].append(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            print(time)
            launch_dict['Time'].append(time)
```

Github: https://github.com/surajkunthu/Data-Science-Projects/blob/main/IBM%20DS%20Professional%20Cert/Capstone/jupyter-labs-webscraping.ipynb

Data Wrangling

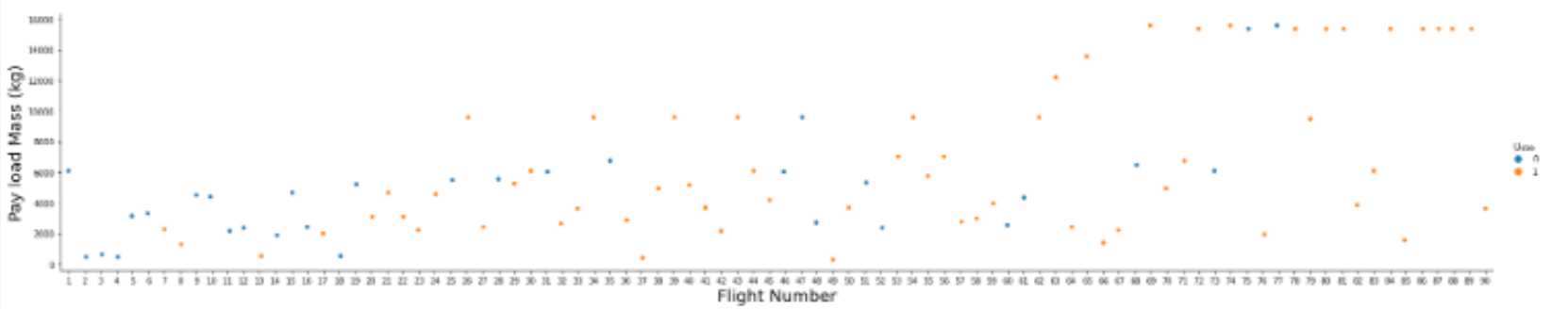
- Data was processed and cleaned for Exploratory Data Analysis (EDA).
  - First the number of launches per site was calculated.
  - Then the mission outcome per orbit type occurrence number was calculated.
  - Finally, we assigned a landing outcome label from the outcome column

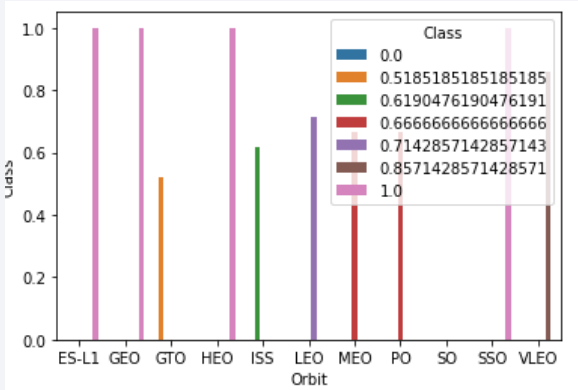


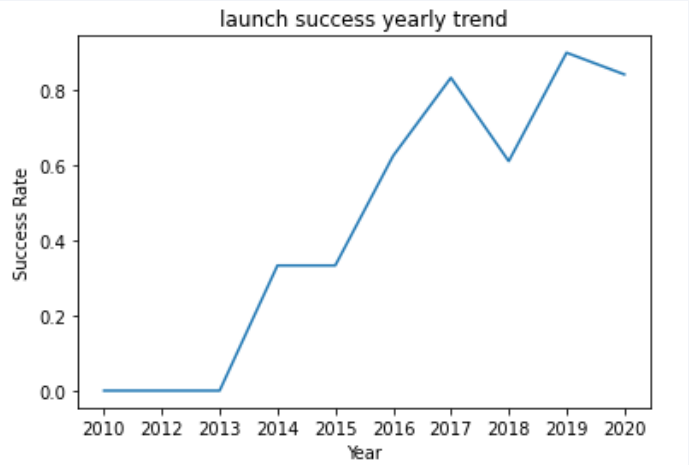
Github: https://github.com/surajkunthu/Data-Science-Projects/blob/main/IBM%20DS%20Professional%20Cert/Capstone/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- Used scatterplots to find the relationship between different attributes.
  - Payload vs Flight Number
  - Flight Number vs Launch Site
  - Payload vs Launch Site
  - Flight Number vs Orbit Site
  - Payload vs Orbit Type
- Then used a Bar graph to visual launch year trend
- Finally used Features Engineering to pair dummy variables against the categorical columns in a line plot.







Github: https://github.com/surajkunthu/Data-Science-Projects/blob/main/IBM%20DS%20Professional%20Cert/Capstone/jupyter-labs-eda-dataviz.ipynb

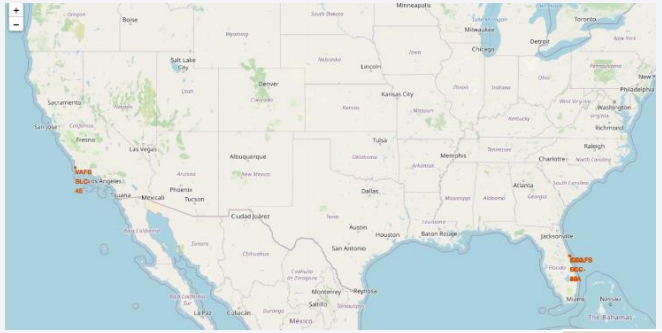
EDA with SQL

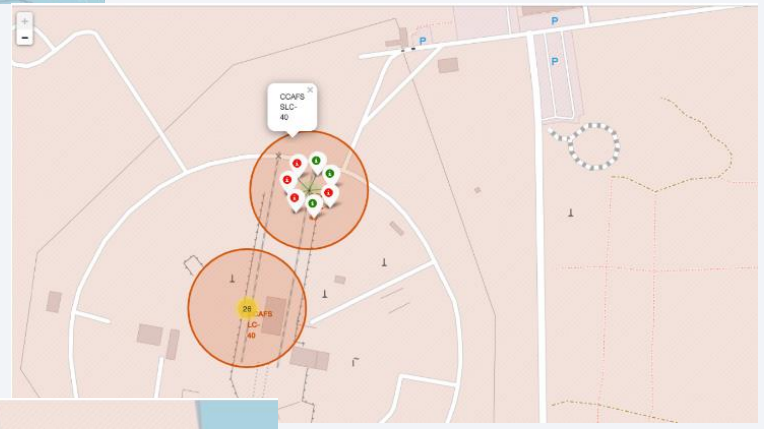
- Performed the following SQL queries (had to use PostgreSQL. IBM db2 did NOT work)
  - Display names of the unique launch sites in the space mission
  - Display 5 records where launch sites begin with the string 'CCA'
  - Display the total payload mass carried by boosters launched by NASA (CRS)
  - Display average payload mass carried by booster version F9 v1.1
  - List the date when the first successful landing outcome in ground pad was achieved
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - List the total number of successful and failure mission outcomes
  - List the names of the booster_versions which have carried the maximum payload mass with a subquery.
  - List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - Rank the count of landing outcomes (such as Failure (drone shop) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Github: https://github.com/surajkunthu/Data-Science-Projects/blob/main/IBM%20DS%20Professional%20Cert/Capstone/jupyter-labs-eda-sql.ipynb

Build an Interactive Map with Folium

- Utilized Folium in Python to plot SpaceX launch site data
- Assigned the geospatial data to a dataframe and added markers on the map to identify launch outcomes
- Then used Haversine's formula to calculate the distance of the launch sites to various landmarks.

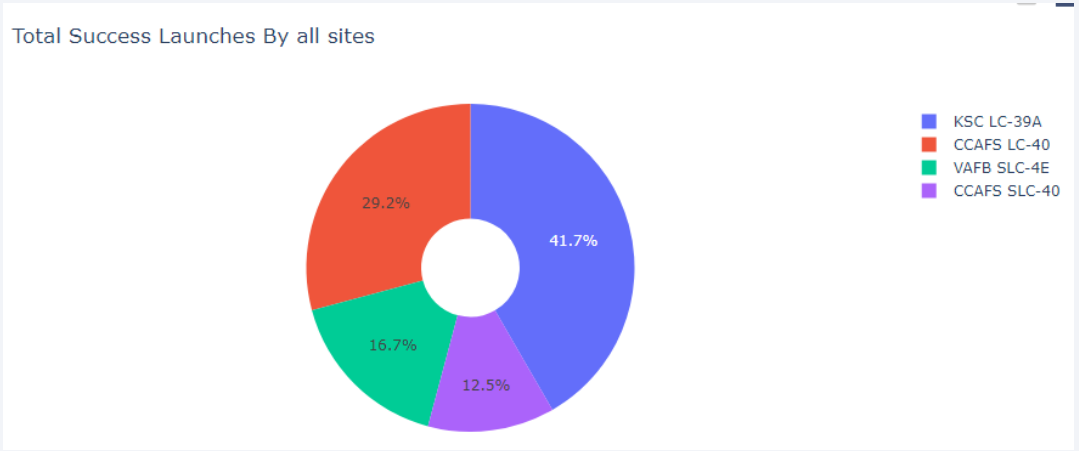


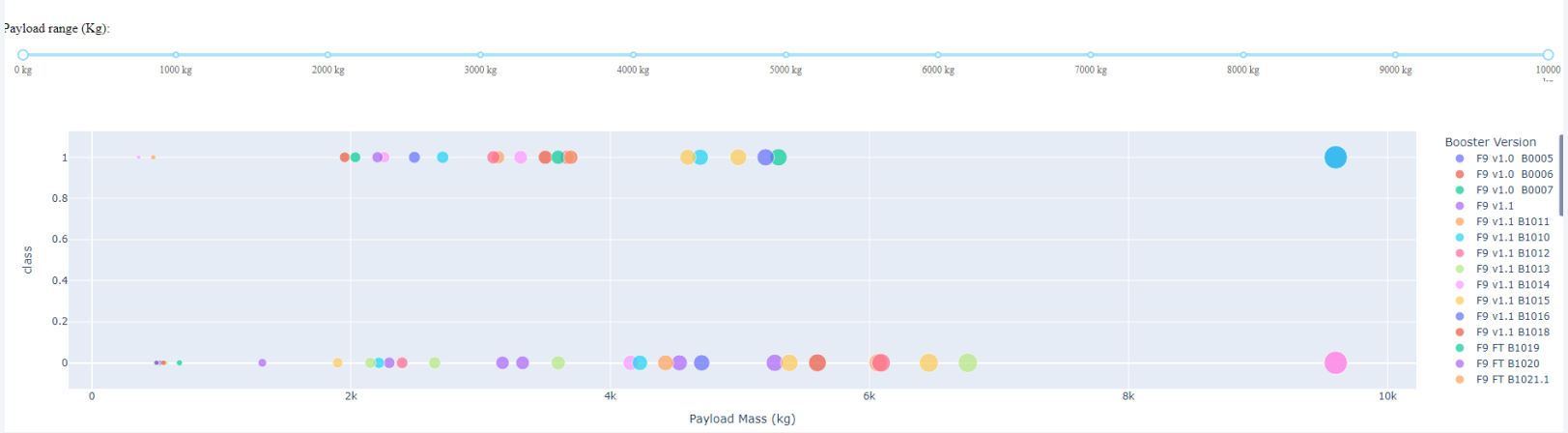


Github: https://github.com/surajkunthu/Data-Science-Projects/blob/main/IBM%20DS%20Professional%20Cert/Capstone/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Used Plotly to create an interactive dashboard in Python
- Plotted a pie chart to show Total success launches by sites
- Then used a scatter plot to show the relationship between outcome and payload mass for each booster version





Github: https://github.com/surajkunthu/Data-Science-Projects/blob/main/IBM%20DS%20Professional%20Cert/Capstone/spacex_dash_app.py

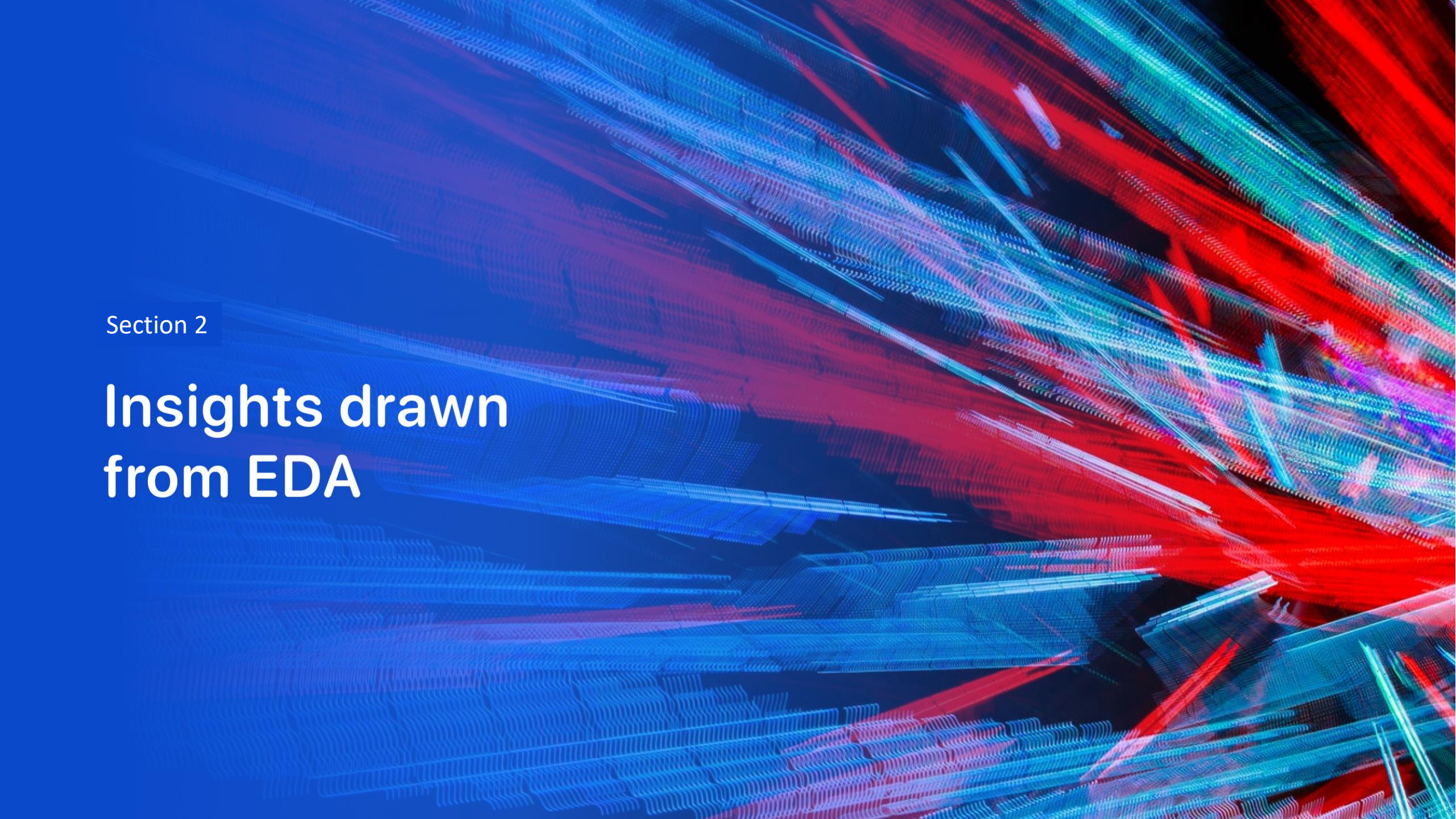
Predictive Analysis (Classification)

- Conducted predictive analysis by loading data with Pandas and NumPy, transformed data, and split into training and testing sets.
- Built machine learning models and tuned parameters with GridSearchCV
- Compared accuracies of different methods
  - Logistic Regression
  - Support Vector Machine (SVM)
  - Decision Tree
  - K-nearest neighbor

Github: https://github.com/surajkunthu/Data-Science-Projects/blob/main/IBM%20DS%20Professional%20Cert/Capstone/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

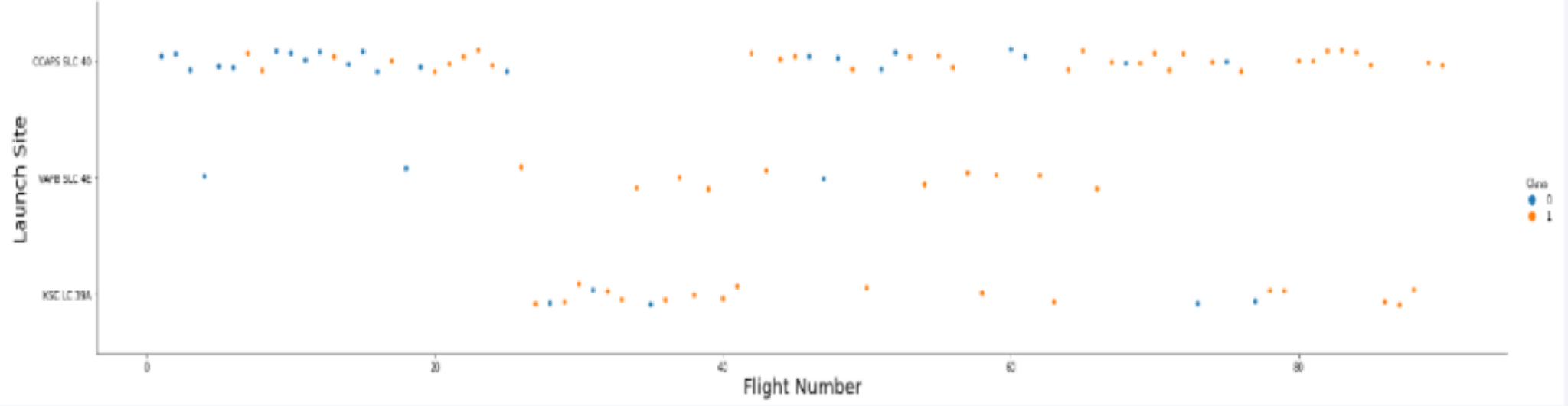
Results

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


Section 2
Insights drawn from EDA

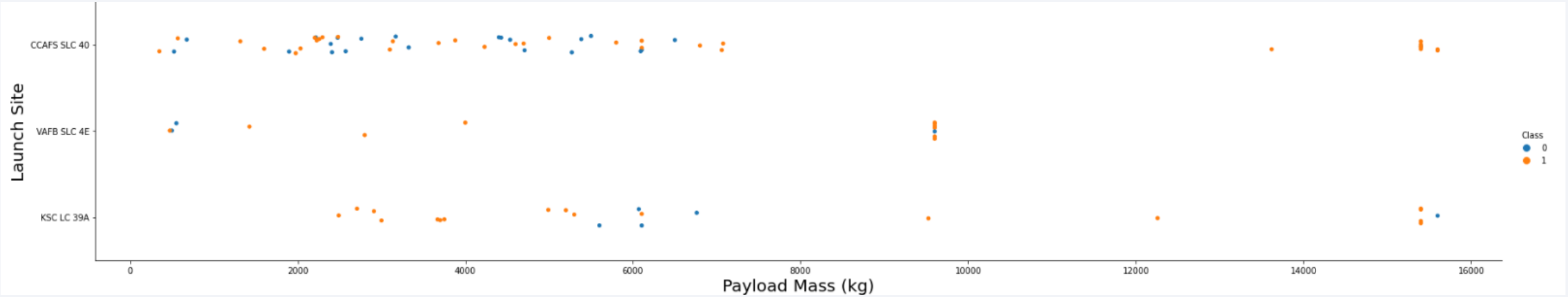
Flight Number vs. Launch Site

- From the scatterplot, we can conclude that the more flights there are at a launch site, the more likely the launch will be a success



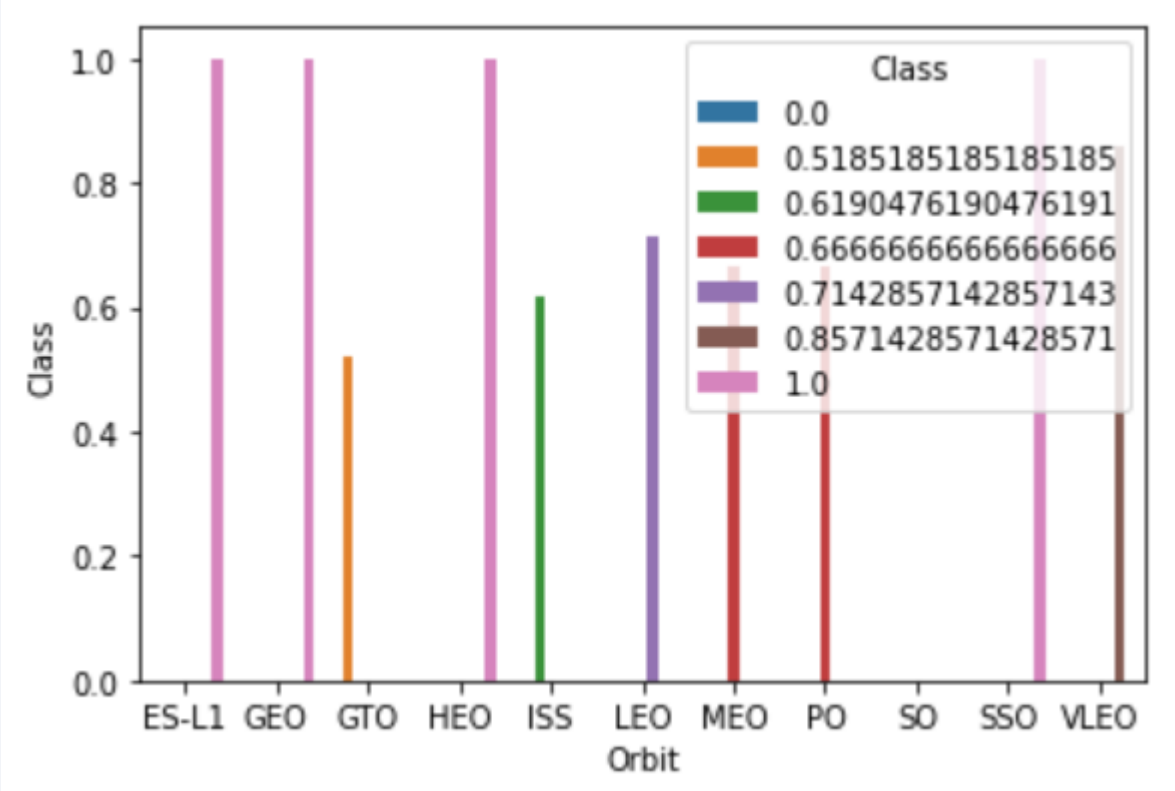
Payload vs. Launch Site

- From the scatterplot, we can conclude that as payload decreases the success rate for the rocket increases



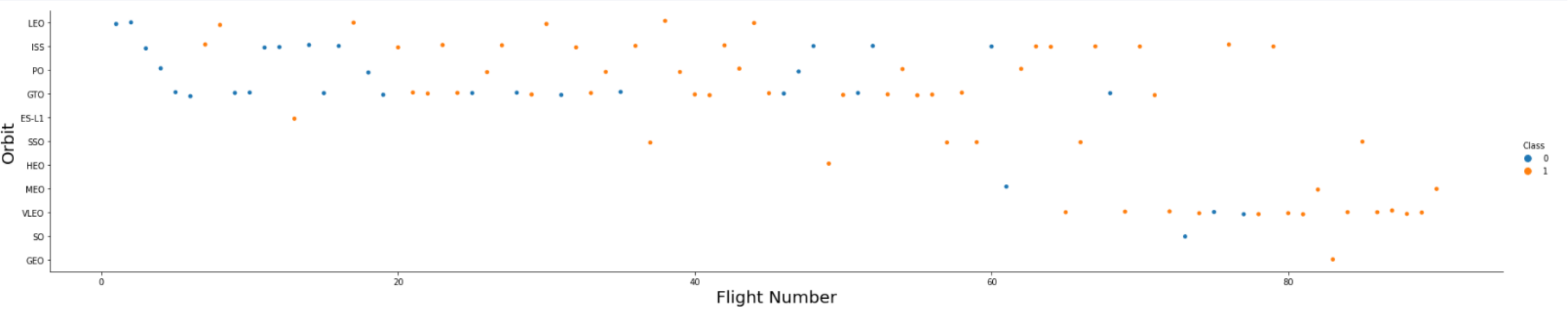
Success Rate vs. Orbit Type

- ES-L1, Geo, HEO, SSO, and VLEO had the most success rate



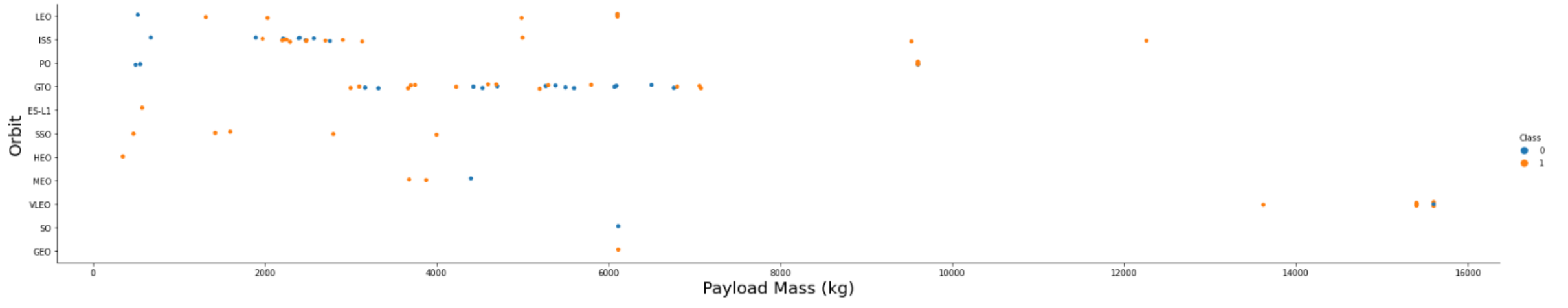
Flight Number vs. Orbit Type

- There appears to be no identifiable relationship between flight number and orbit type



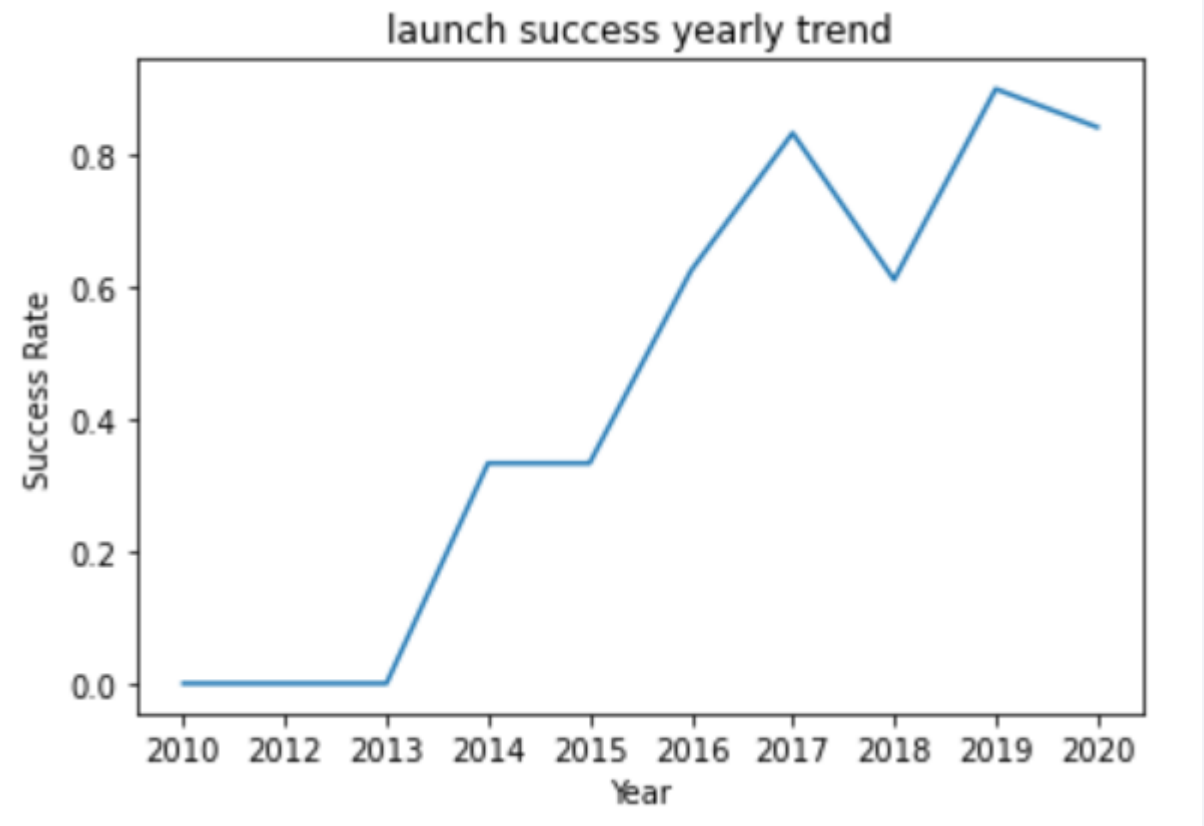
Payload vs. Orbit Type

- Heavier payloads seem to only benefit PO, LEO, ISS orbit landings



Launch Success Yearly Trend

- Success rate of launches improved from 2013 onwards.



All Launch Site Names

- Use DISTINCT to show unique launch sites

```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

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

Launch Site Names Begin with 'CCA'

- Queried the data to show launch sites that only begin with ‘CCA’

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

	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

- Calculated the total payload mass with the following query

```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

- Calculated the average payload mass by the F9 v1.1 with the following query

```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

- Found the first successful landing date with the following query

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Listed the names of boosters which have successfully landed on drone ship with WHERE and had payload mass greater than 4000 but less than 6000 with the operators > and <

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

	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

- Filtered with WHERE and % to retrieve if Mission Outcome was a Success or Failure

```
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%'
        '''
```

	successoutcome
0	100

	failureoutcome
0	1

Boosters Carried Maximum Payload

- Used WHERE and MAX to find the maximum carried payload by booster version

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

	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

- Used WHERE, LIKE, AND, and BETWEEN to filter for 2015 failed outcome launch records

```
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)
```

	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

- Used COUNT, WHERE, and BETWEEN to filter the data
- Then GROUP BY and ORDER BY to organize the resulting data

```
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)
```

	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

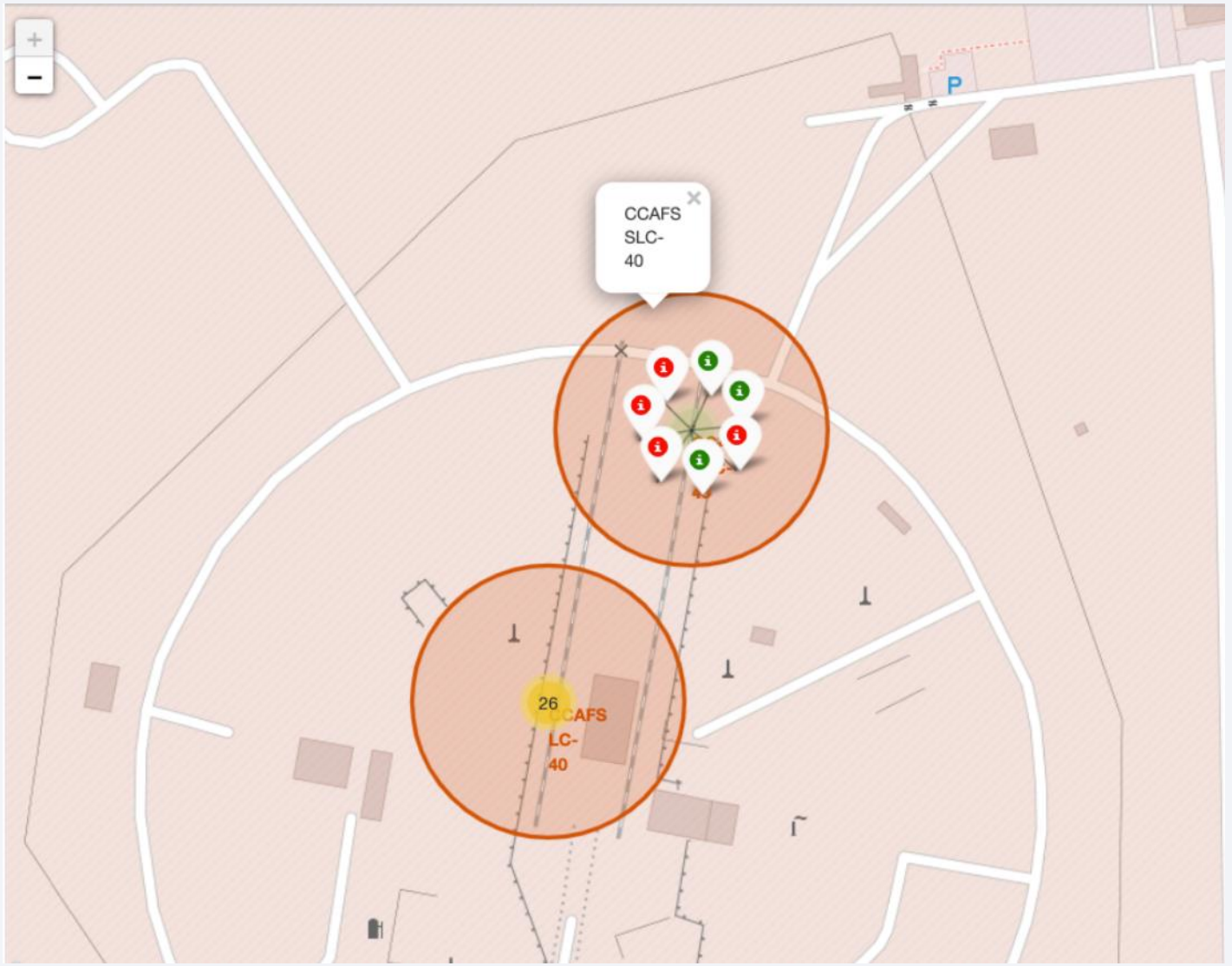
Launch Sites Proximities Analysis

SpaceX Launch Sites in the United States

- Launch sites in California and Florida

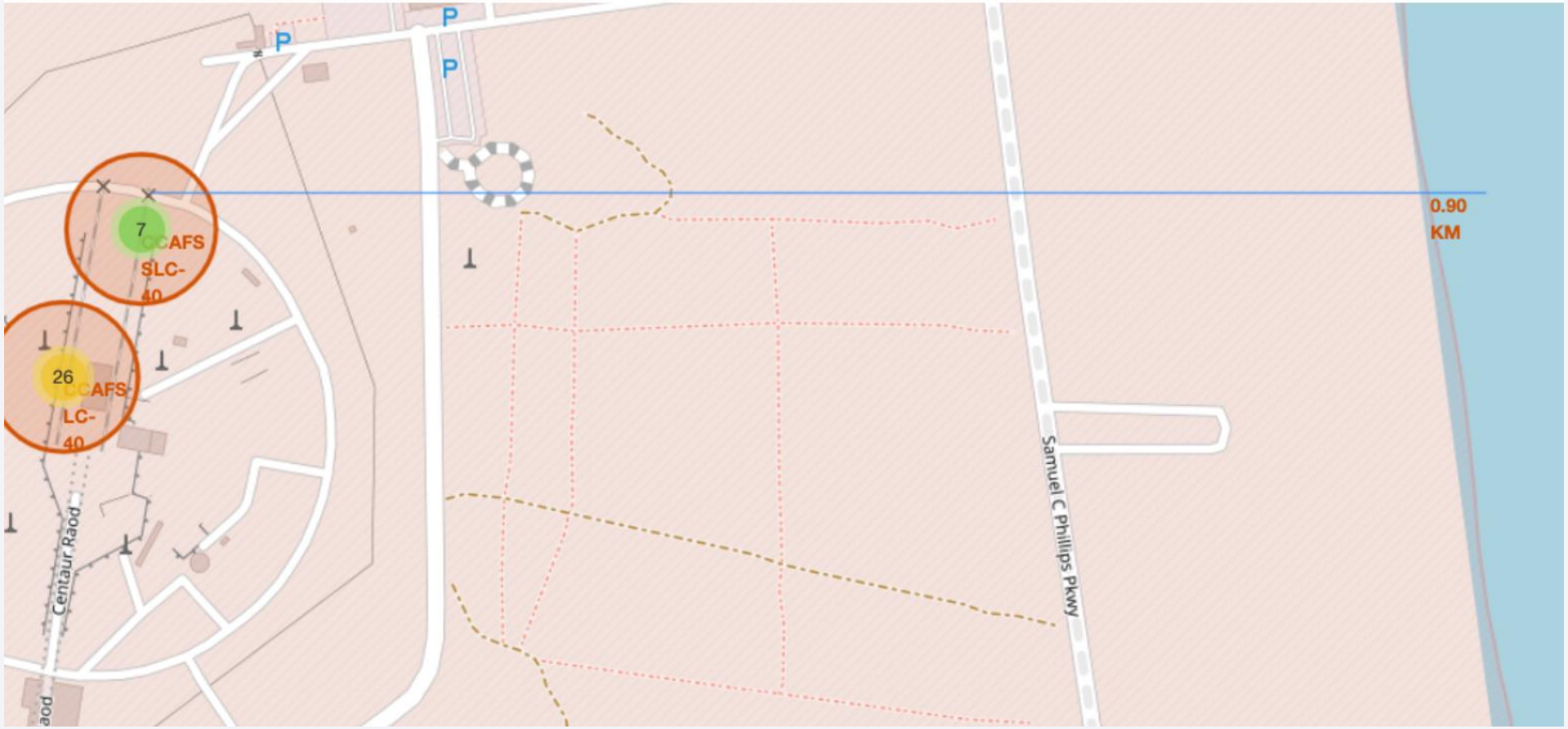
Folium Map with markers on launch sites

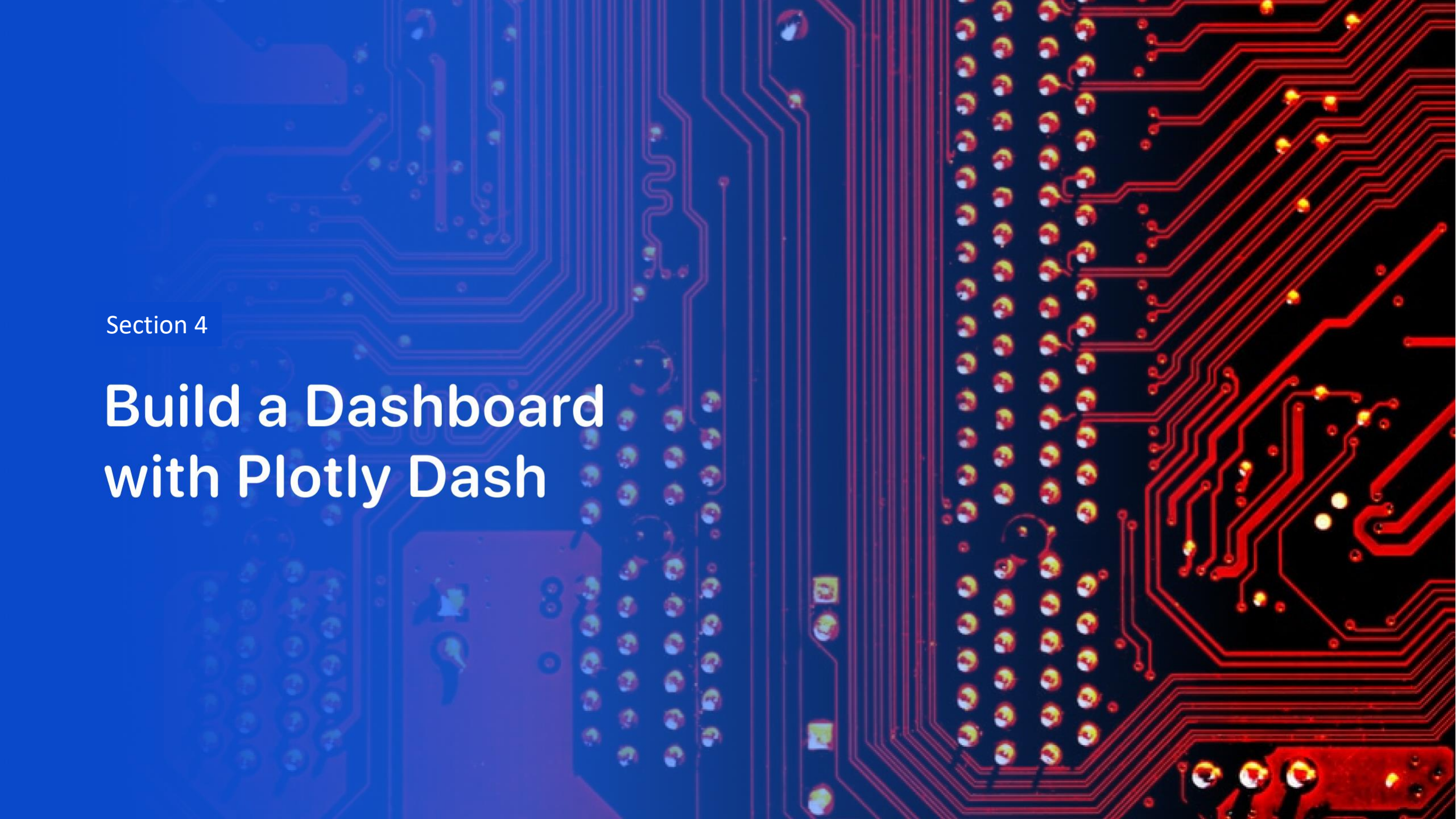
- Green – success
- Red - Failure



Folium Map – Launch site distance

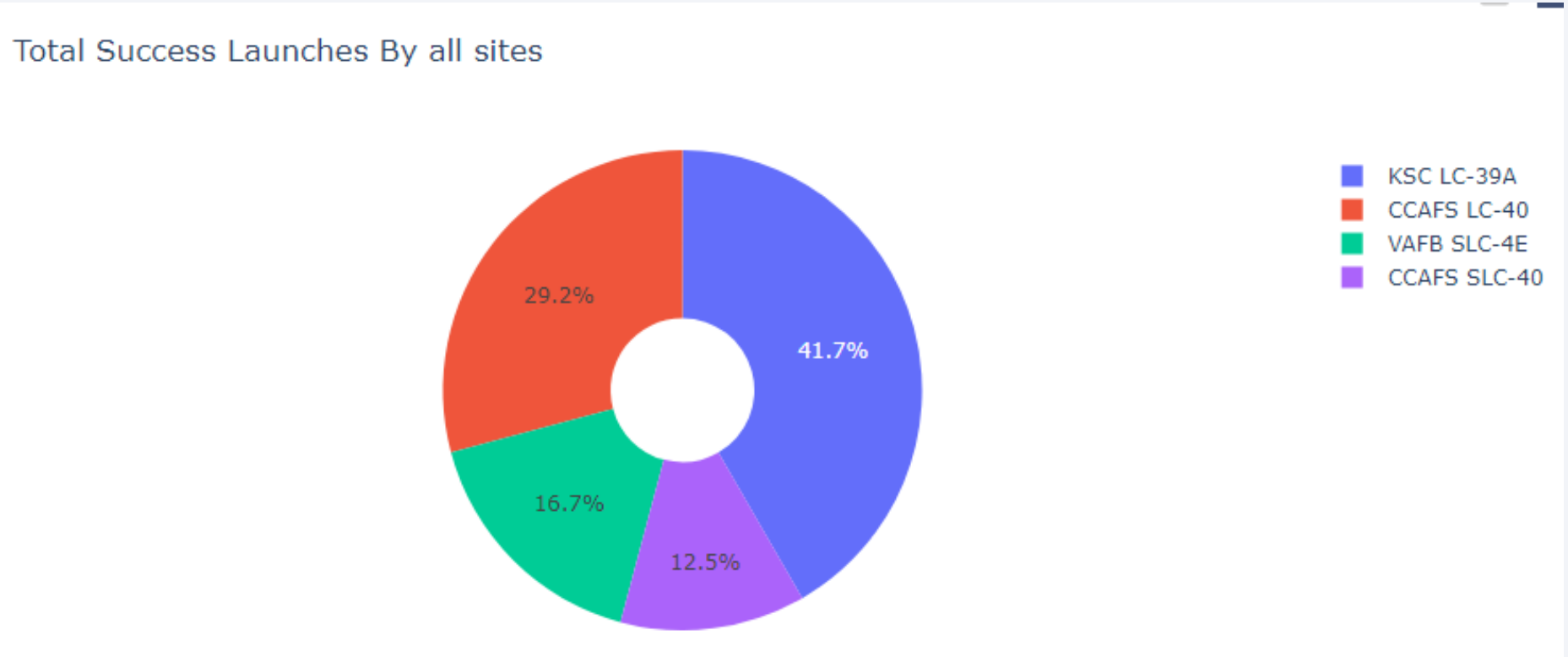
- Blue line shows launch site distance to coastline




Section 4
Build a Dashboard with Plotly Dash

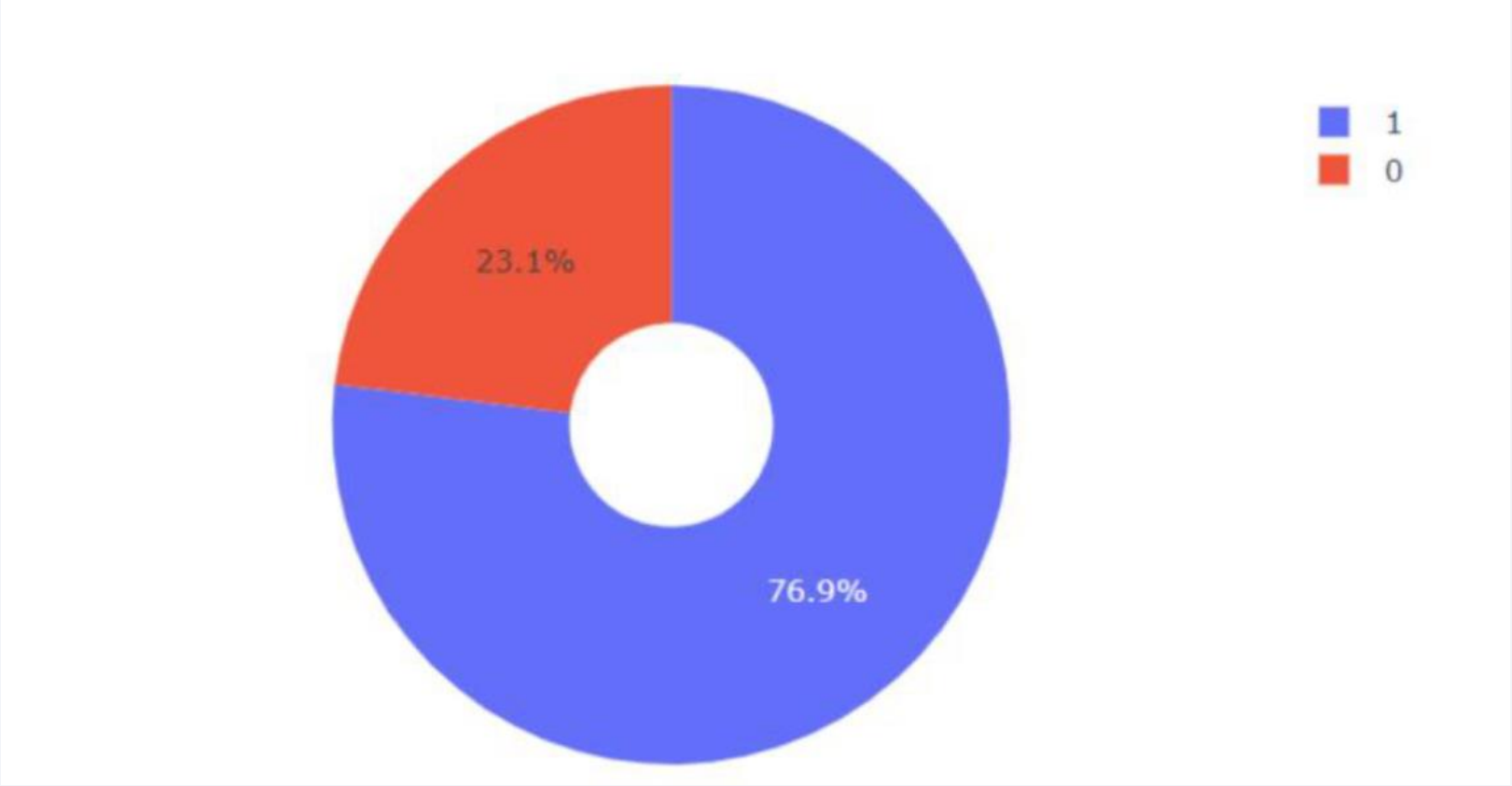
Pie Chart of Launch site success

- KSC LC-39A has had the most success



Pie Chart of highest Launch site success ratio

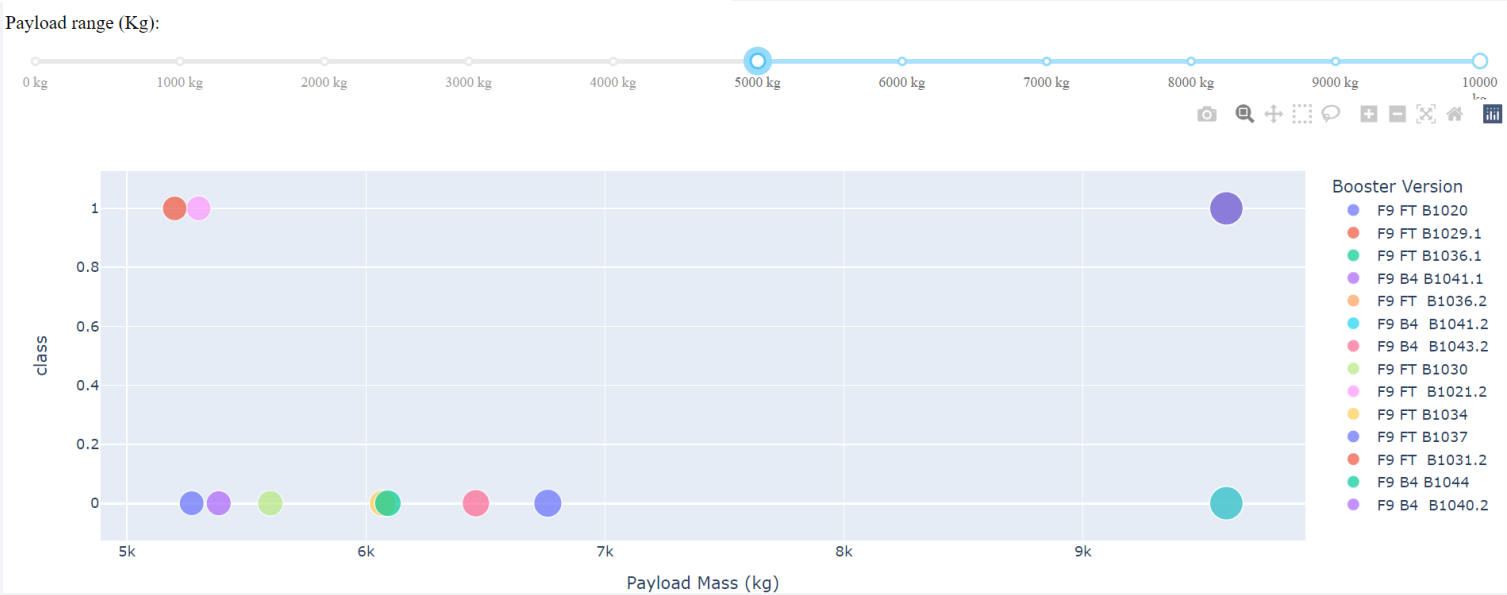
- KSC LC-39A has a 76.9% success rate and a 23.1% failure rate



Payload vs Launch Outcome Scatter Plot

- 0 kg – 5000 kg
- Success rate seems to increases as payload approaches to 5000 kg, then falls off as it approaches 10000 kg

- 5000 kg – 10000 kg



Section 5

Predictive Analysis (Classification)

Classification Accuracy

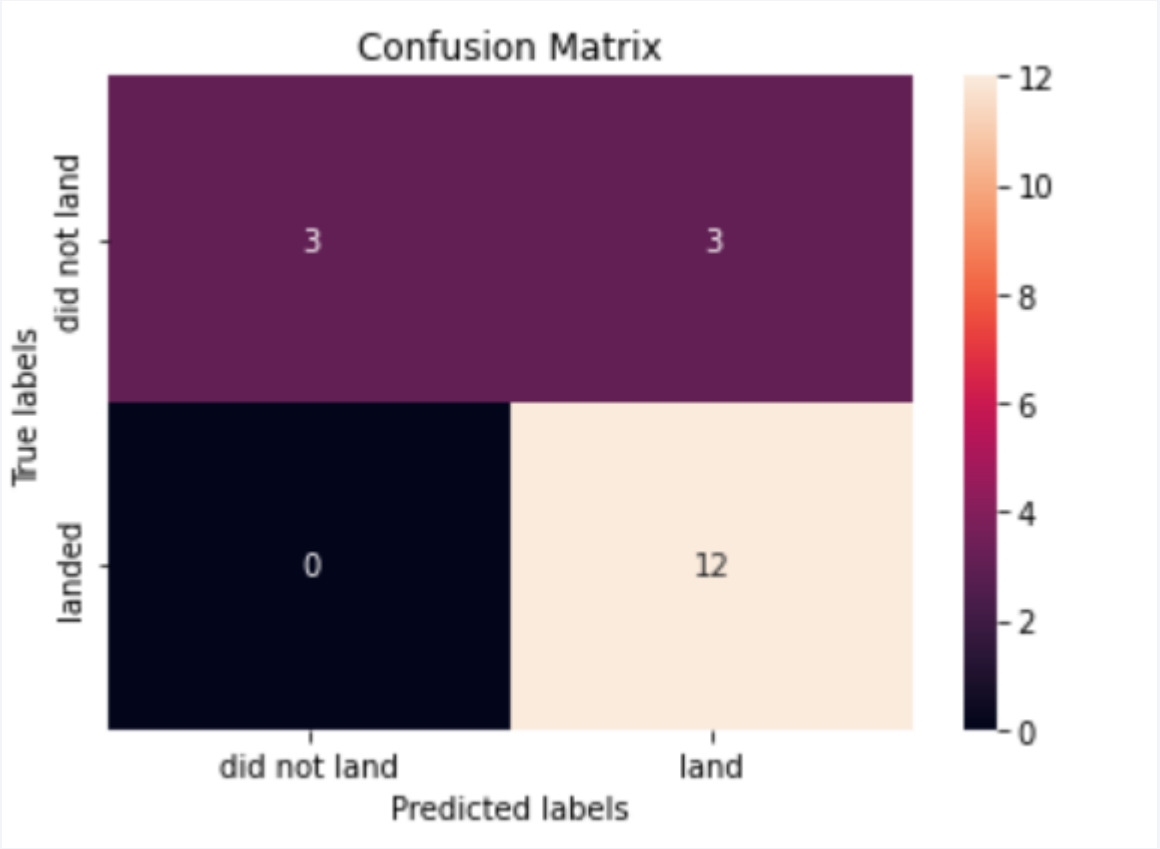
- The decision tree method proves to have the highest classification accuracy

```
print('Logistics Regression Method Accuracy:', logreg_cv.score(X_test, Y_test))
print( 'Support Vector Machine Method Accuracy:', svm_cv.score(X_test, Y_test))
print('Decision tree Method Accuracy:', tree_cv.score(X_test, Y_test))
print('K-nearest neighbors Method Accuracy:', knn_cv.score(X_test, Y_test))
```

```
Logistics Regression Method Accuracy:  0.8464285714285713
Support Vector Machine Method Accuracy: 0.8333333333333334
Decision tree Method Accuracy:  0.875
K-nearest neighbors Method Accuracy: 0.8482142857142858
```

Confusion Matrix

- The classifier is able to decipher between different classes. However, false positives can be a problem.



Conclusions

- The more launches conducted at a launch site indicate a greater success rate
- ES-L1, GEO, HEO, SSO, and VLEO have had the most success for orbits
- Launch success rates steadily increased from 2013-2020
- KSC LC-39A has had the most success at launches compared to any other site
- The decision tree classification method is the ideal machine learning algorithm to use since it has the highest accuracy of 87.5%

Appendix

- Github: https://github.com/surajkunthu/Data-Science-Projects/tree/main/IBM%20DS%20Professional%20Cert/Capstone

Thank you!