IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Giuseppe Gullì
20/01/2024

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

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 from Machine Learning Lab

Introduction

- Project background and context

  SpaceX is a company who has disrupt the space industry by offering a rocket launches specifically Falcon 9 as low as 62 million dollars; while other providers cost upward of 165 million dollar each. Most of this saving thanks to SpaceX astounding idea to reuse the first stage of the launch by re-land the rocket to be used on the next mission. Repeating this process will make the price down even further.

- Problems you want to find answers

  As a data scientist of a startup rivaling SpaceX, the goal of this project is to create the machine learning pipeline to predict the landing outcome of the first stage in the future.
  This project is crucial in identifying the right price to bid against SpaceX for a rocket launch.
  The problems include:
  • Identifying all factors that influence the landing outcome.
  • The relationship between each variables and how it is affecting the outcome.
  • The best condition needed to increase the probability of successful landing.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX REST API and Web Scraping from Wikipedia
- Perform data wrangling
  - Data was processed using 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

Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. As mentioned, the dataset was collected by REST API and Web Scraping from Wikipedia.

For REST API, its started by using the get request. Then, I decoded the response content as JSON and turn it into a pandas dataframe using json_normalize(). I then cleaned the data, checked for missing values and fill with whatever needed.

For Web Scraping, I used the BeautifulSoup 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

Get request for rocket launch data using API

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

response = requests.get(spacex_url)
```

Use json_normalize method to convert JSON result to dataframe

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

Performed data cleaning and filling the missing values

```
# 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
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)]
```

Link:

https://github.com/Gullo90/IBM_Data_Science_Capstone_Project/blob/main/01_jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

Request the Falcon9 Launch Wiki page from url

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

Create a BeautifulSoup from the HTML response

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

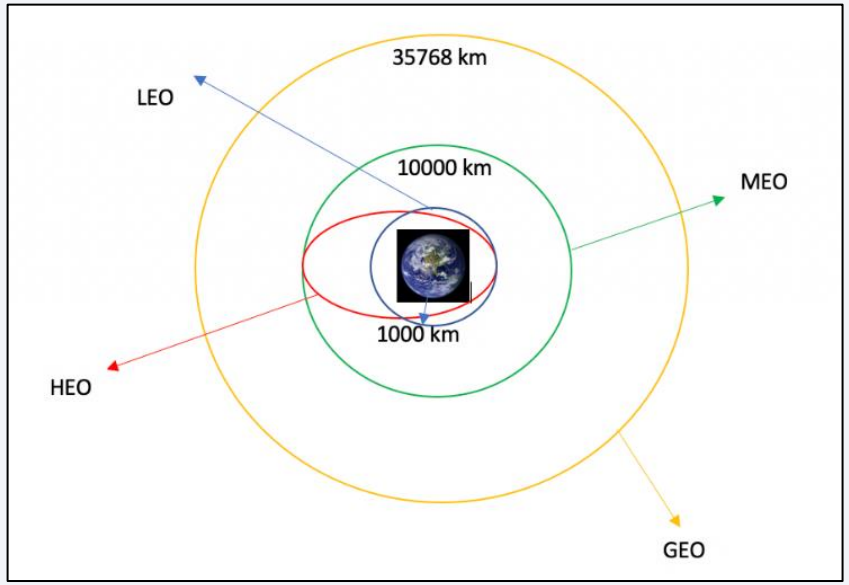
Extract all column/variable names from the HTML header

```
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"):
        #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.`
            launch_dict['Flight No.'].append(flight_number) #TODO-1
            #print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date) #TODO-2
            #print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time) #TODO-3
            #print(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict['Version Booster'].append(bv) #TODO-4
            if not(bv):
                bv=row[1].a.string
            print(bv)

            # Launch Site
            # TODO: Append the bv into launch_dict with key `Launch Site`
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site) #TODO-5
```

Link:

https://github.com/Gullo90/IBM_Data_Science_Capstone_Project/blob/main/02_jupyter-labs-webscraping.ipynb

Data Wrangling



Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).

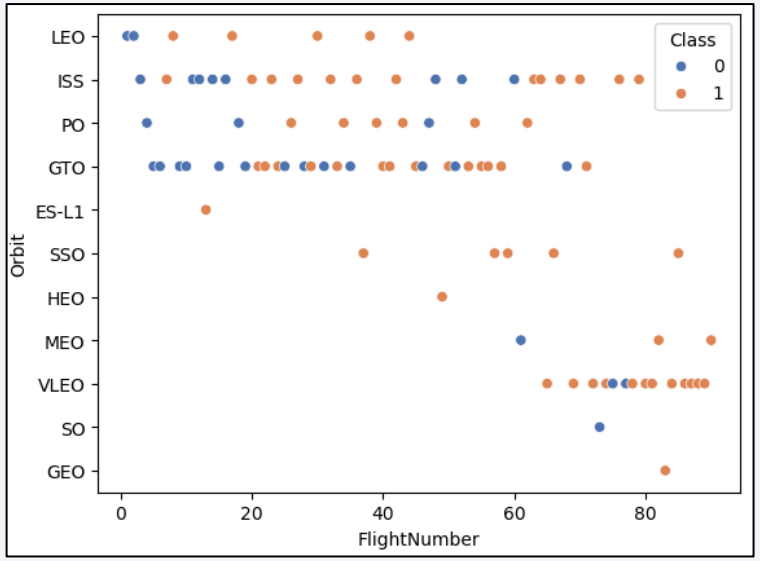
I calculated the number of launches on each site, then calculated the number and occurrence of mission outcome per orbit type.

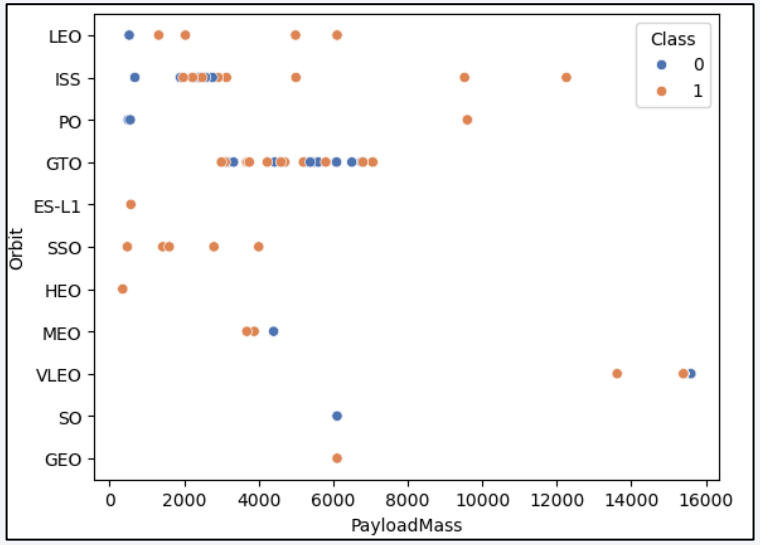
I then created a landing outcome label from the outcome column. This will make it easier for further analysis, visualization, and ML. Lastly, I exported the result to a CSV.

Link:

https://github.com/Gullo90/IBM_Data_Science_Capstone_Project/blob/main/03_labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization





I first started by using scatter graph to find the relationship between the attributes such as between:

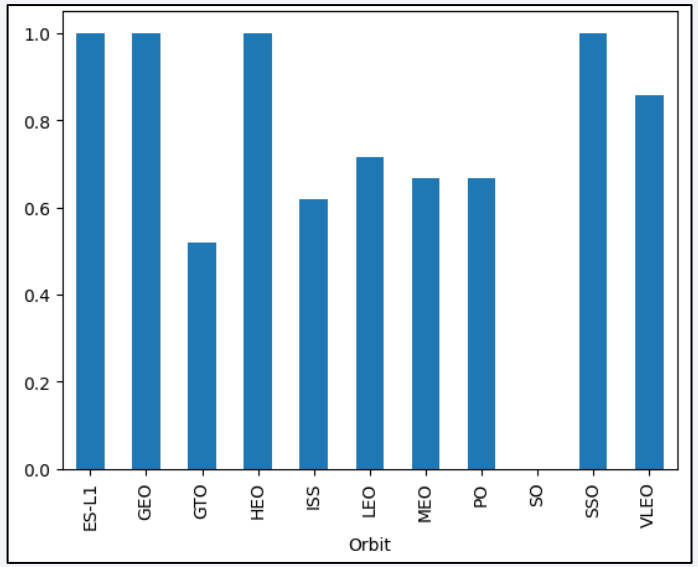
- Payload and Flight Number.
- Flight Number and Launch Site.
- Payload and Launch Site.
- Flight Number and Orbit Type.
- Payload and Orbit Type.

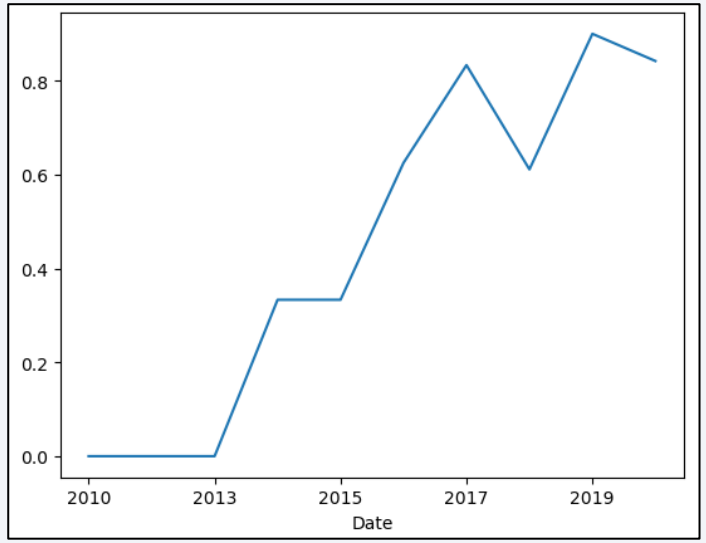
Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs, it's very easy to see which factors affecting the most to the success of the landing outcomes.

Link:

https://github.com/Gullo90/IBM_Data_Science_Capstone_Project/blob/main/05_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with Data Visualization





Once I got a hint of the relationships using scatter plot, I then used further visualization tools such as bar graph and line plots graph for further analysis.

Bar graphs is one of the easiest way to interpret the relationship between the attributes. In this case, I used the bar graph to determine which orbits have the highest probability of success.

I then used the line graph to show a trends or pattern of the attribute over time which in this case, is used for see the launch success yearly trend.

I then used Feature Engineering to be used in success prediction in the future module by created the dummy variables to categorical columns.

Link:

https://github.com/Gullo90/IBM_Data_Science_Capstone_Project/blob/main/05_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

Using SQL, I had performed many queries to get better understanding of the dataset:

- Displaying the names of the launch sites and displaying 5 records where launch sites begin with the string 'CCA'.

- Displaying the total payload mass carried by booster launched by NASA (CRS) and displaying the average payload mass carried by booster version F9 v1.1.

- Listing the date when the first successful landing outcome in ground pad was achieved.

- Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.

- Listing the total number of successful and failure mission outcomes.

- Listing the names of the booster_versions which have carried the maximum payload mass.

- Listing the failed landing_outcomes in drone ship, their booster versions, and launch sites names for in year 2015.

- Rank the count of landing outcomes or success between the date 2010-06-04 and 2017-03-20, in descending order.

Link:

https://github.com/Gullo90/IBM_Data_Science_Capstone_Project/blob/main/04_jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

To visualize the launch data into an interactive map. I took the latitude and longitude coordinates at each launch site and added a circle marker around each launch site with a label of the name of the launch site.

I then assigned the dataframe launch_outcomes (failure,success) to classes 0 and 1 with Red and Green markers on the map in MarkerCluster().

I then used the Haversine's formula to calculated the distance of the launch sites to various landmark to find answer to the questions of:

- How close the launch sites with railways, highways and coastlines?
- How close the launch sites with nearby cities?

Link:

https://github.com/Gullo90/IBM_Data_Science_Capstone_Project/blob/main/06_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- I built an interactive dashboard with Plotly Dash which allowing the user to play around with the data as they need.
- I plotted pie charts showing the total launches by a certain sites.
- I then plotted scatter graph showing the relationship with Outcome and Payload
- Mass (kg) for the different booster version.

Link:

https://github.com/Gullo90/IBM_Data_Science_Capstone_Project/blob/main/07_Build_Interactive_Dashboard_with_Ploty_Dash.py

Predictive Analysis (Classification)

Building the Model
- Load the dataset into NumPy and Pandas
- Transform the data and then split into training and test datasets
- Decide which type of ML to use
- Set the parameters and algorithms to GridSearchCV and fit it to dataset.

Evaluating the Model
- Check the accuracy for each model
- Get tuned hyperparameters for each type of algorithms.
- Plot the confusion matrix.

Improving the Model
- Use Feature Engineering and Algorithm Tuning

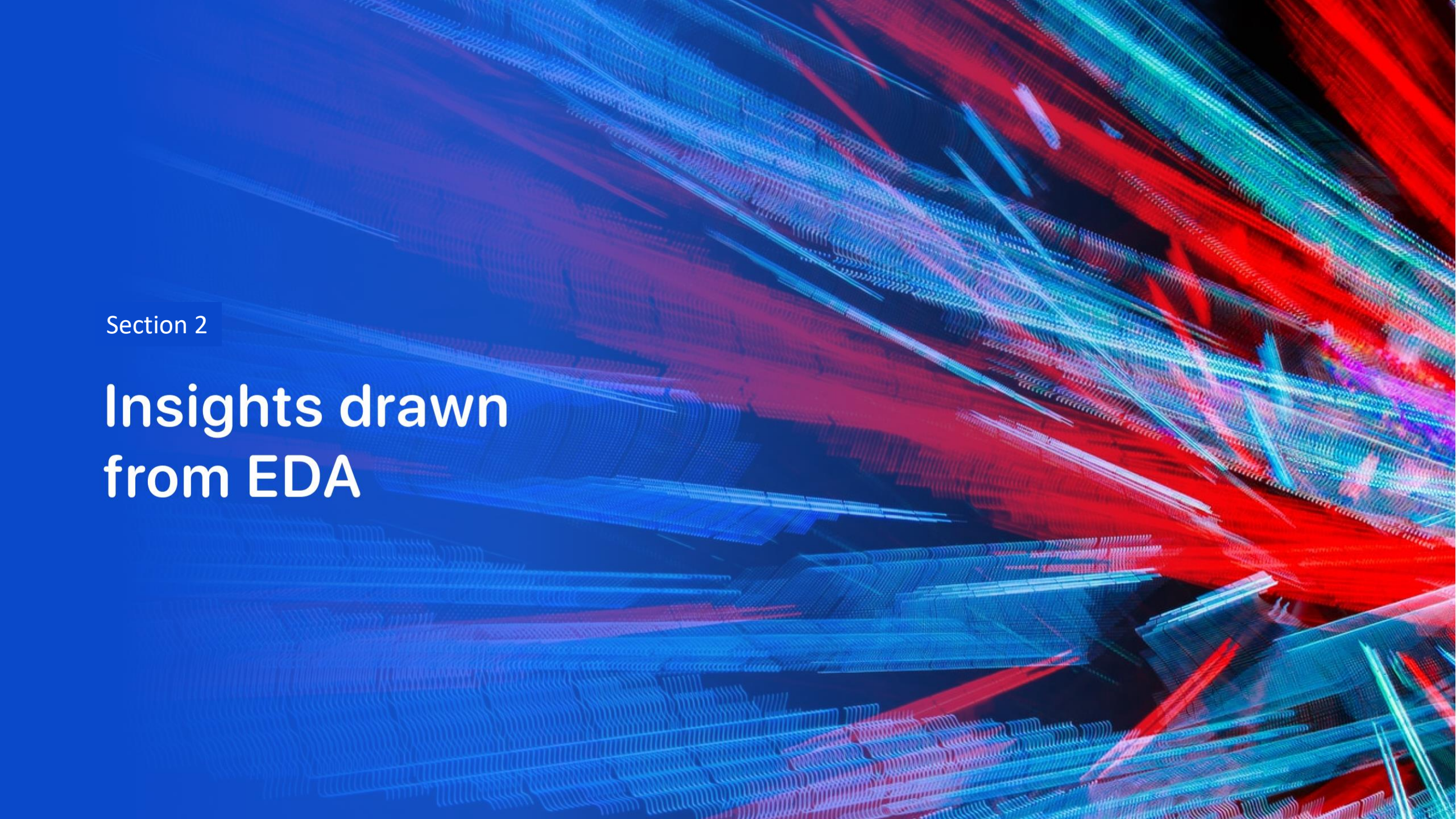
Find the Best Model
- The model with the best accuracy score will be the best performing model.

Link:

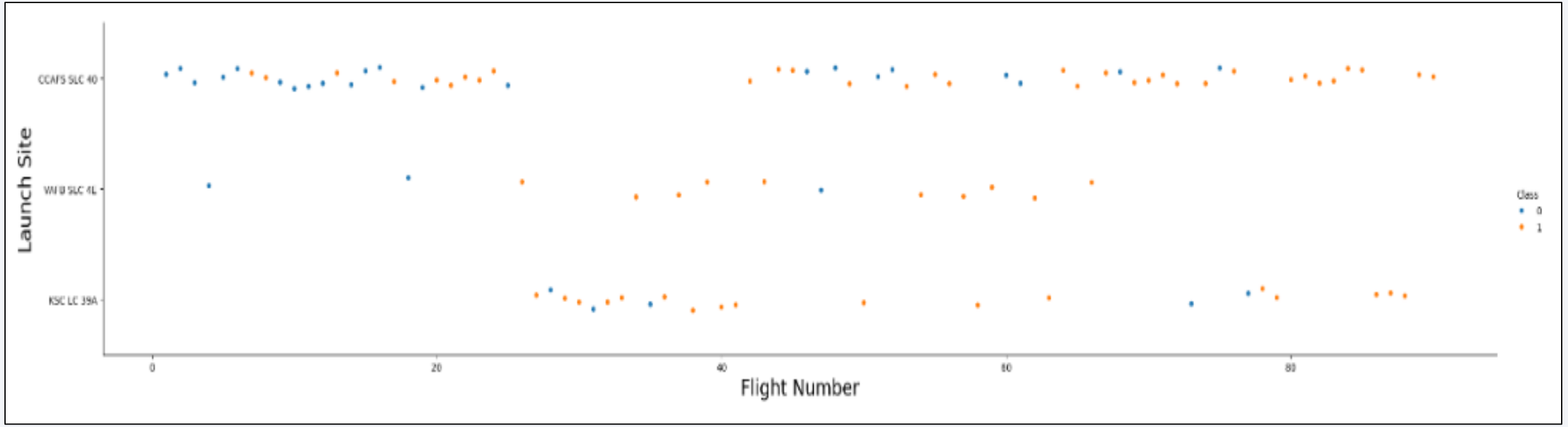
https://github.com/Gullo90/IBM_Data_Science_Capstone_Project/blob/main/08_SpaceX_Machine_Learning_Prediction.jupyterlite.ipynb

Results

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


Section 2
Insights drawn from EDA

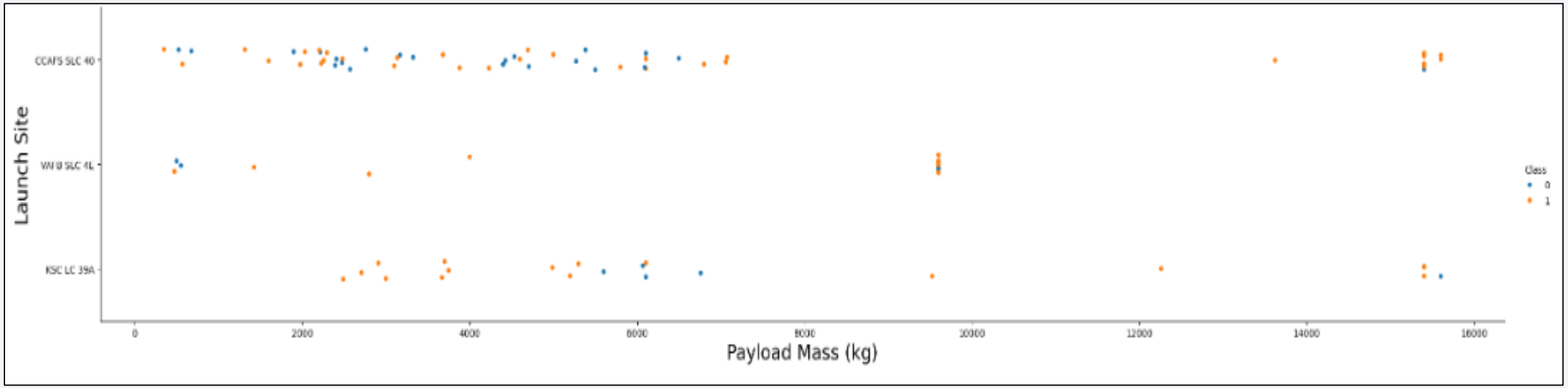
Flight Number vs. Launch Site



This scatter plot shows that the larger the flights amount of the launch site, the greater the success rate will be.

However, site CCAFS SLC40 shows the least pattern of this.

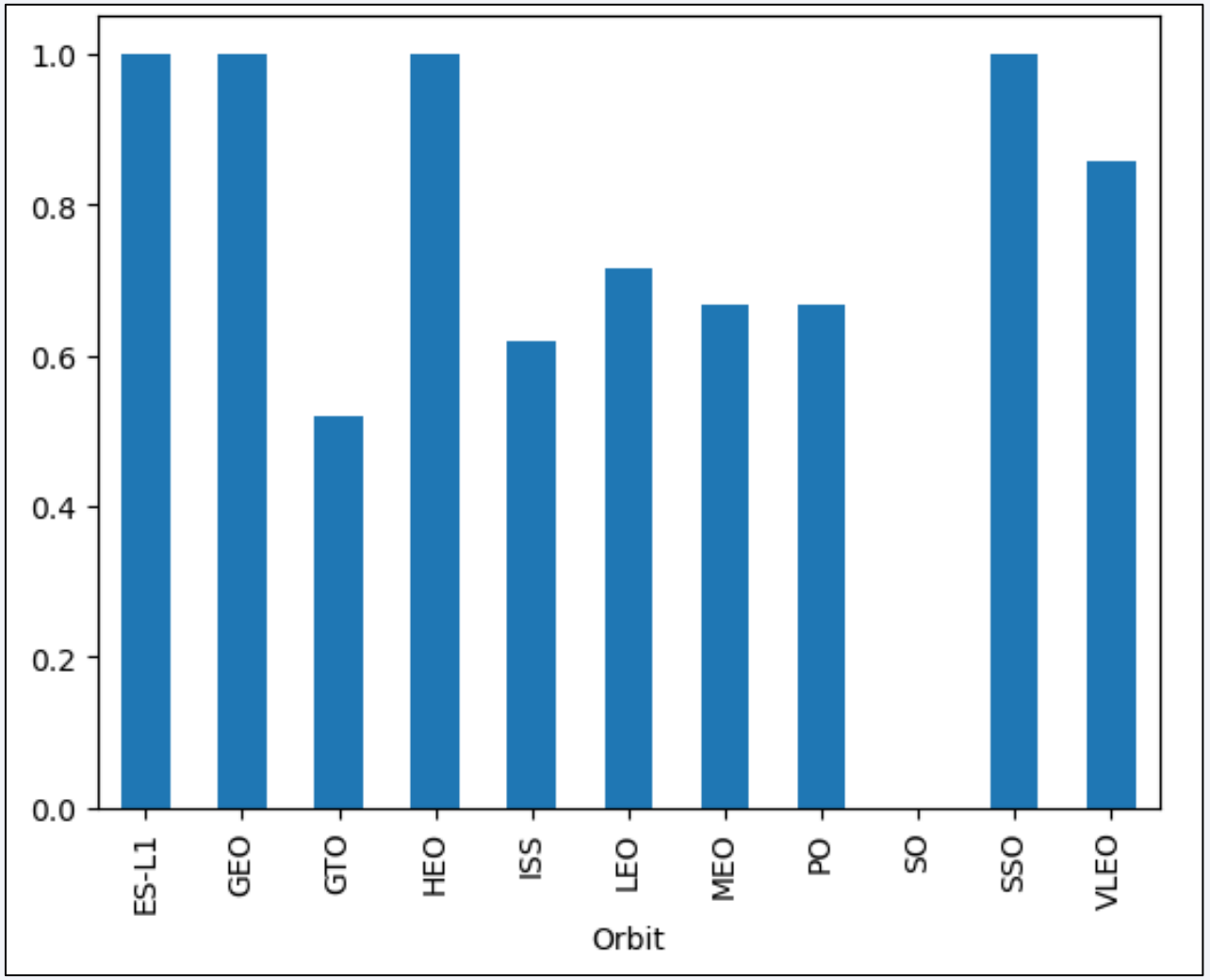
Payload vs. Launch Site



This scatter plot shows once the pay load mass is greater than 7000 kg, the probability of the success rate will be highly increased.

However, there is no clear pattern to say the launch site is dependent to the pay load mass for the success rate.

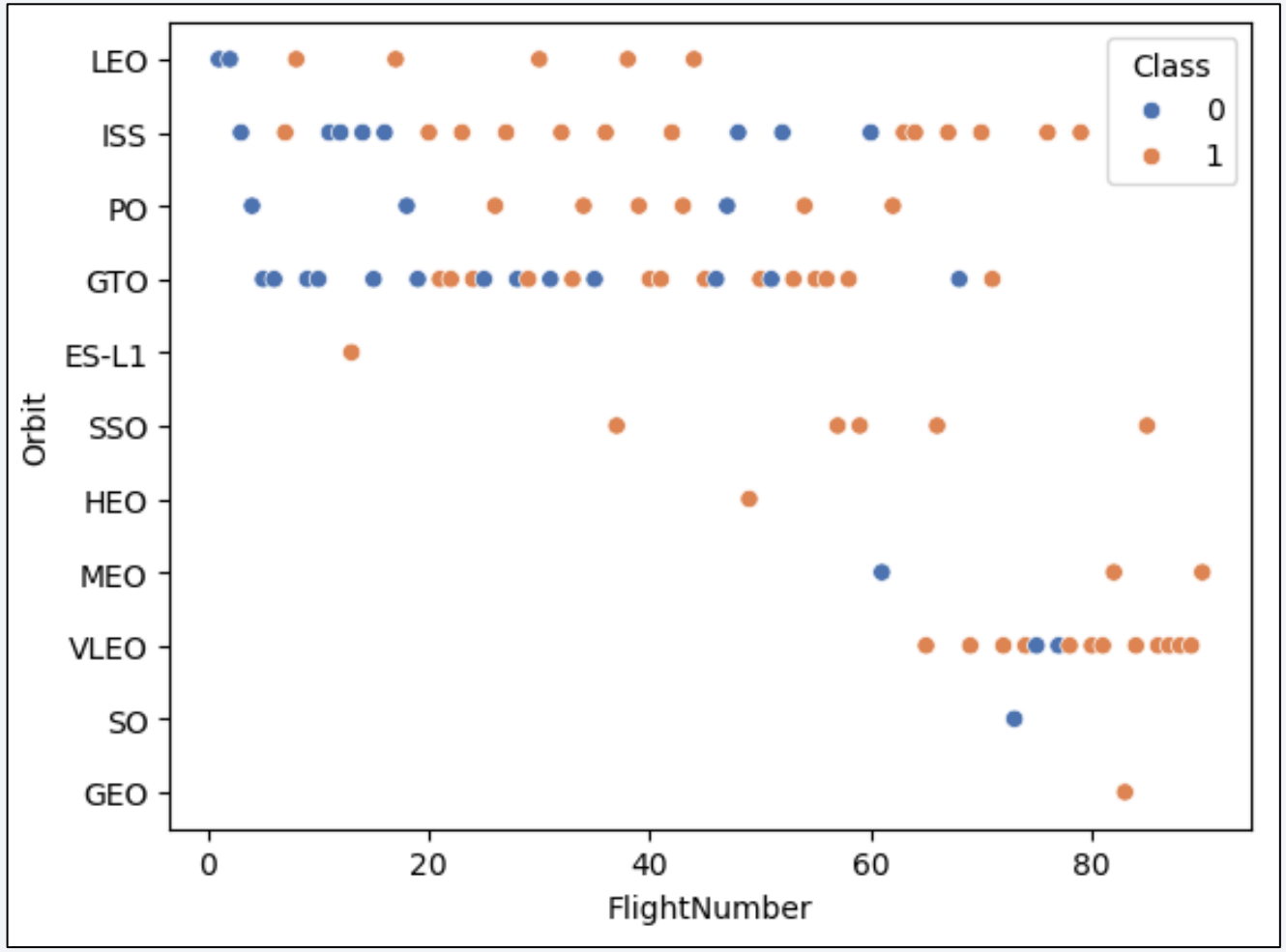
Success Rate vs. Orbit Type



This figure depicted the possibility of the orbits to influence the landing outcomes as some orbits has 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success.

However, deeper analysis show that some of this orbits has only 1 occurrence such as GEO, SO, HEO and ES-L1 which mean this data need more dataset to see pattern or trend before we draw any conclusion.

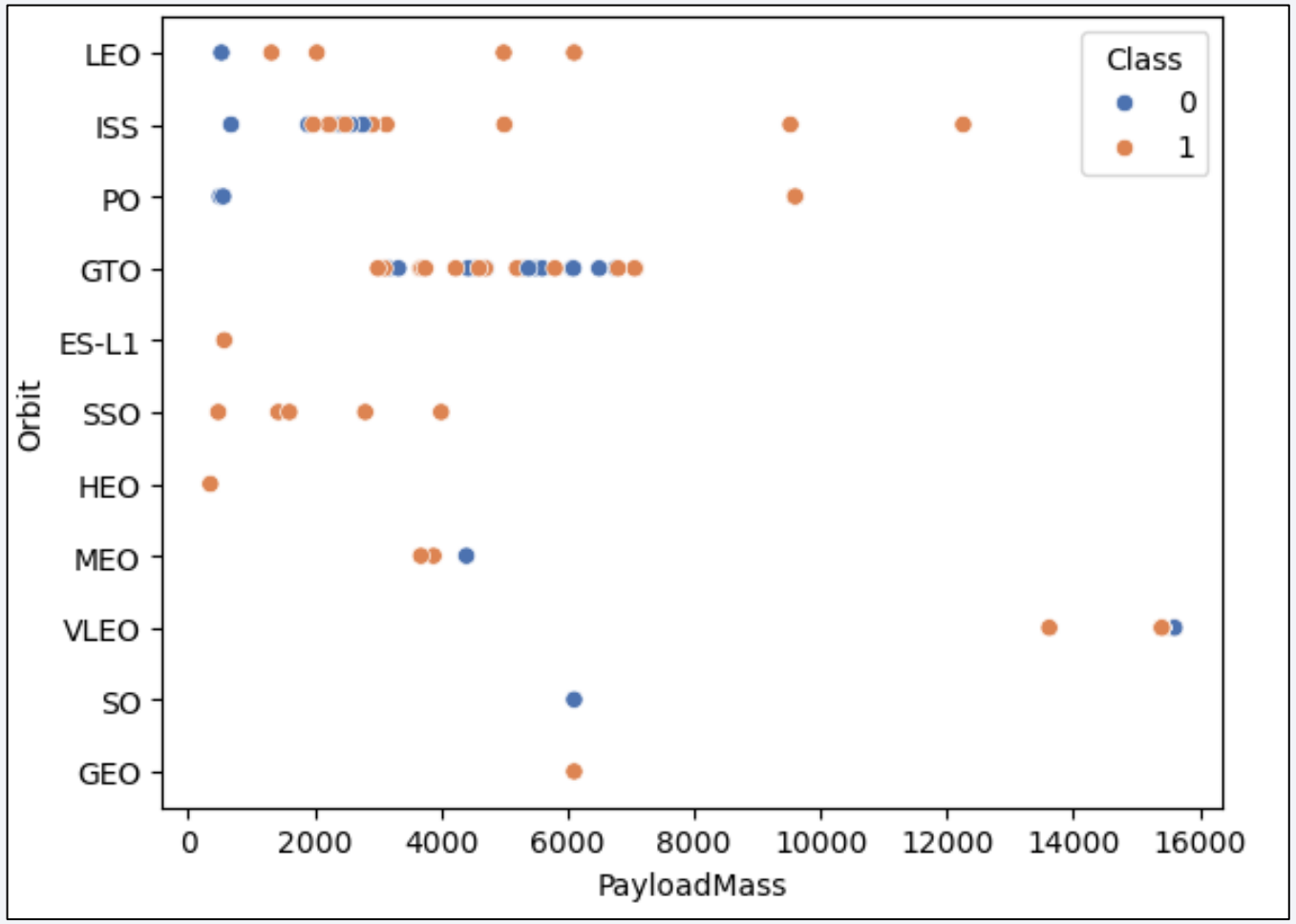
Flight Number vs. Orbit Type



This scatter plot shows that generally, the larger the flight number on each orbits, the greater the success rate (especially LEO orbit) except for GTO orbit which depicts no relationship between both attributes.

Orbit that only has 1 occurrence should also be excluded from above statement as it's needed more dataset.

Payload vs. Orbit Type



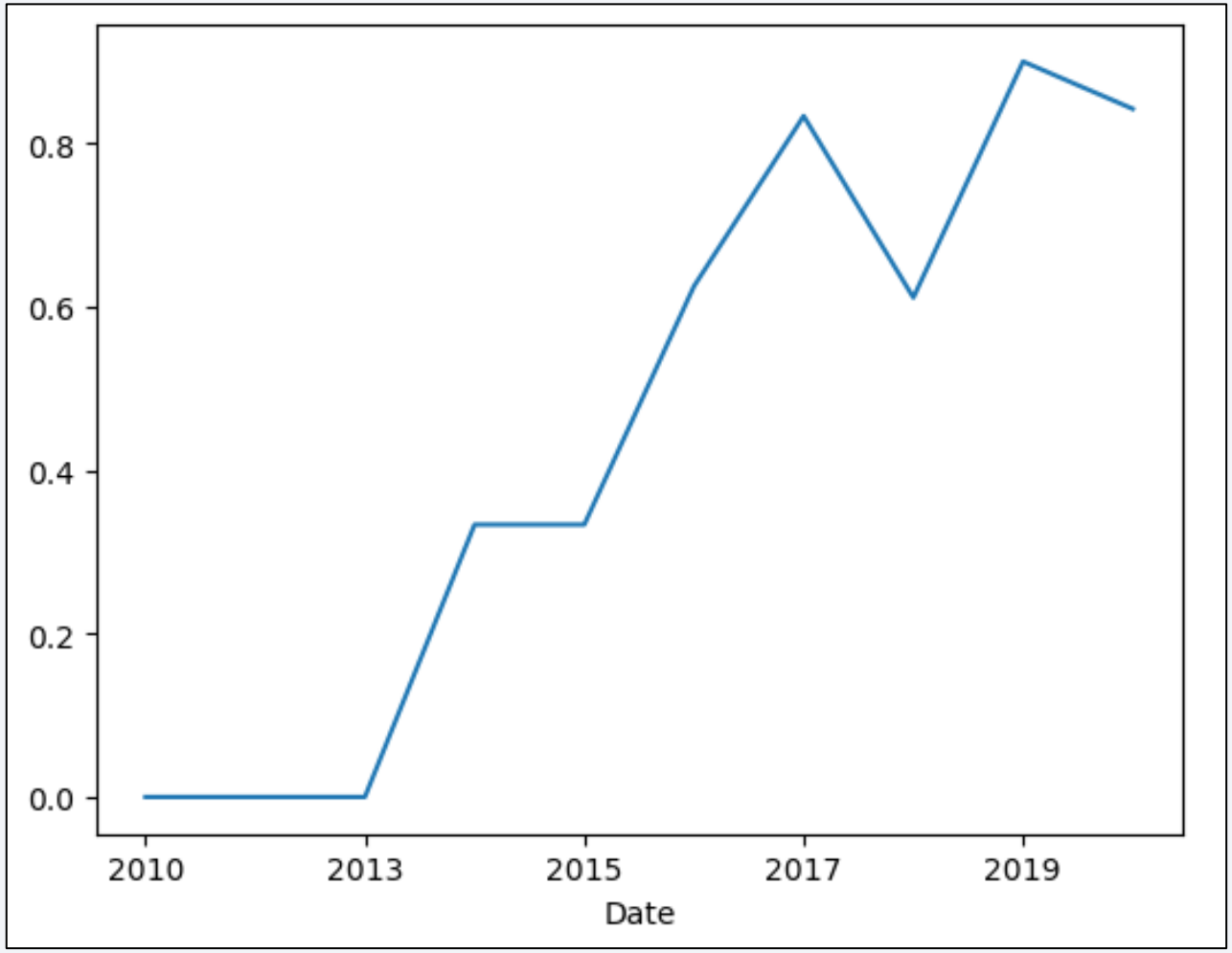
Heavier payload has positive impact on LEO, ISS and PO orbit.

However, it has negative impact on MEO and VLEO orbit.

GTO orbit seem to depict no relation between the attributes.

Meanwhile, again, SO, GEO and HEO orbit need more dataset to see any pattern or trend.

Launch Success Yearly Trend



This figures clearly depicted and increasing trend from the year 2013 until 2020.

If this trend continues for the next year onward, the success rate will steadily increase until reaching 1/100% success rate.

All Launch Site Names

I used the key word “DISTINCT” to show only unique launch sitesfrom the SpaceX data.

```
%sql SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL
```

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

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

Launch Site Names Begin with 'CCA'

I used the query above to display 5 records where launch sites 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

I calculated the total payload carried by boosters from NASA as 99980 kg using the query below

```
%sql SELECT sum(PAYLOAD_MASS__KG_) FROM SPACEXTBL where Customer like 'NASA%'
```

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

sum(PAYLOAD_MASS__KG_)
99980

Average Payload Mass by F9 v1.1

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

```
%sql SELECT Avg(PAYLOAD_MASS__KG_) FROM SPACEXTBL where Booster_Version = 'F9 v1.1'
```

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

Avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

I used the min() function to find the result

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

```
%sql SELECT min(Date) FROM SPACEXTBL where "LANDING_OUTCOME" = 'Success (ground pad)'
```

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

min(Date)
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

```
%sql select Booster_Version from SPACEXTBL where "LANDING_OUTCOME" = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4
```

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

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

I used wildcard like '%' to filter for "COUNT" Mission_Outcome was a success or a failure.

```
%sql SELECT Mission_Outcome, count(*) FROM SPACEXTBL group by Mission_Outcome
```

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

Mission_Outcome	count(*)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

I determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.

```
%sql SELECT Booster_Version FROM SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

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

```
%sql SELECT substr(Date,6,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, [Landing_Outcome] FROM SPACEXTBL where [Landing_Ou
```

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

month	Date	Booster_Version	Launch_Site	Landing_Outcome
01	2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	2015-04-14	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.

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

```
%sql SELECT [Landing_Outcome], count(*) as count_outcomes FROM SPACEXTBL WHERE DATE between '2010-06-04' and '2017-03-20' g
```

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

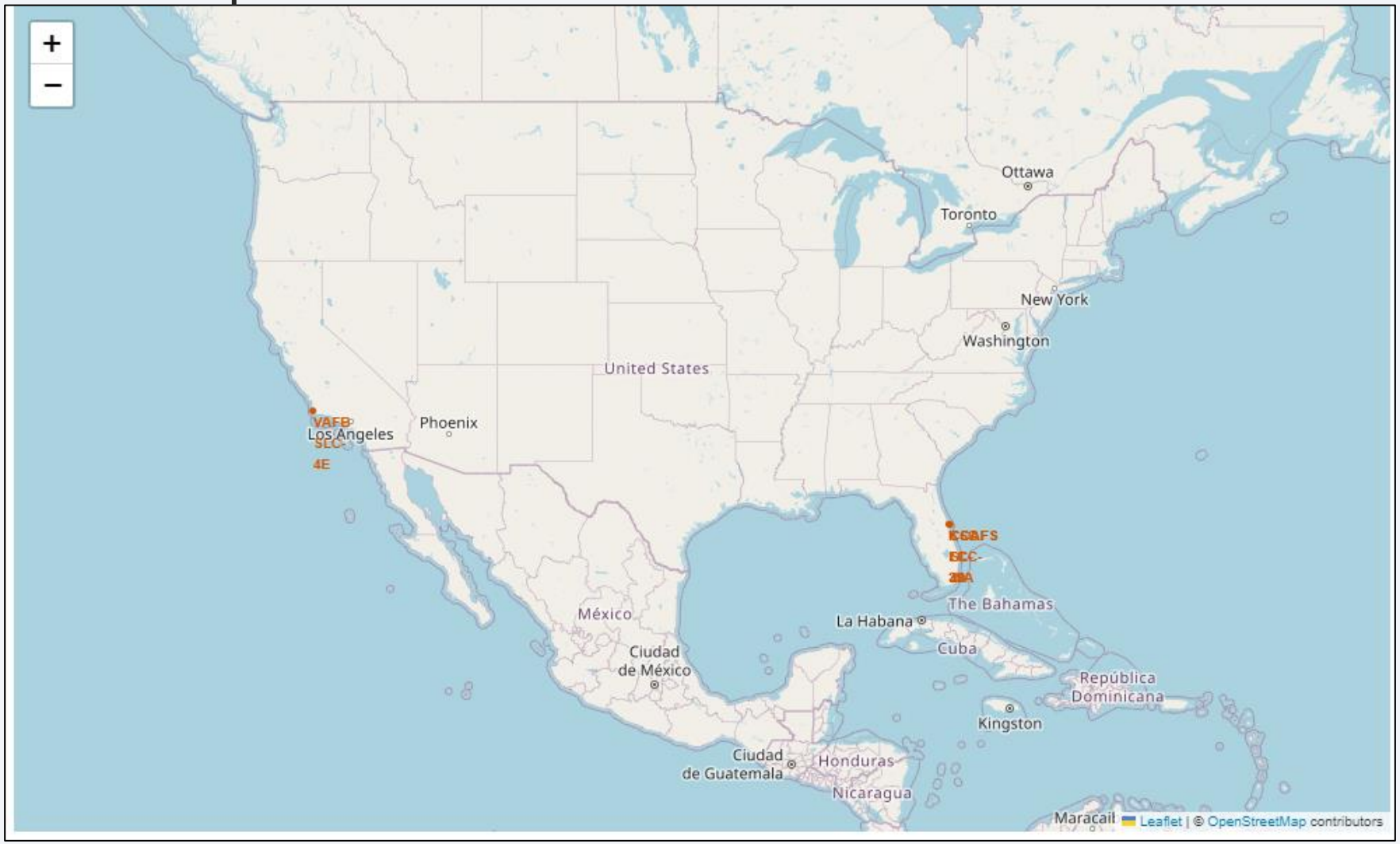
Landing_Outcome	count_outcomes
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

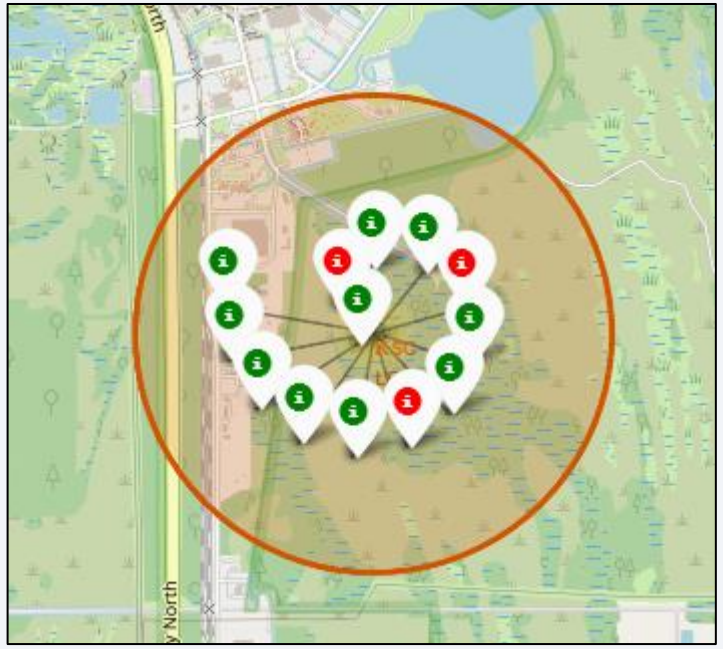
Launch Sites Proximities Analysis

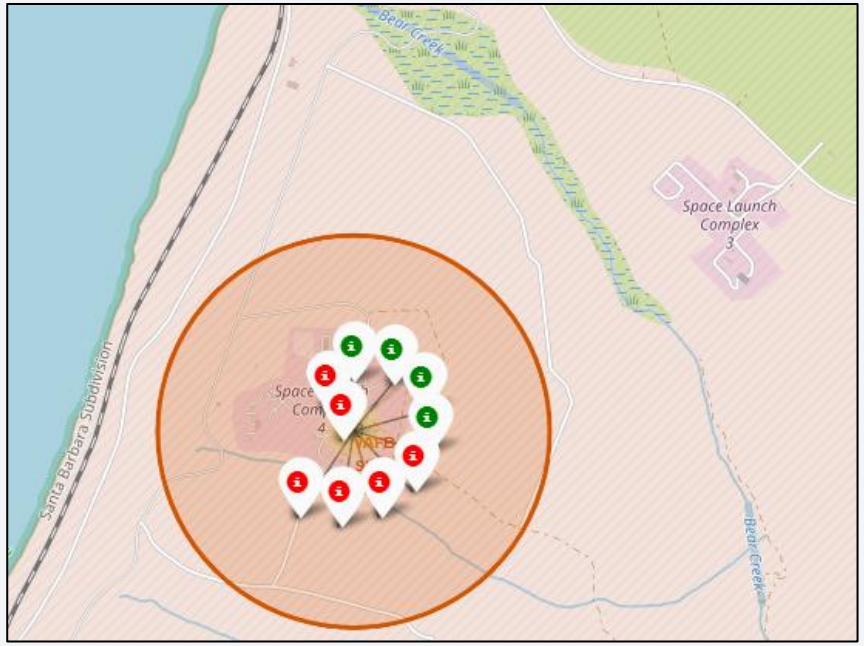
Location of all the Launch Sites

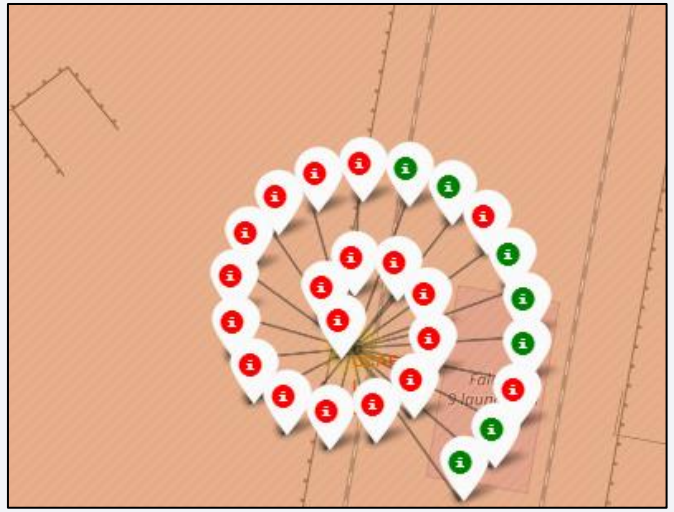
I saw that all the SpaceX launch sites are located inside the United States

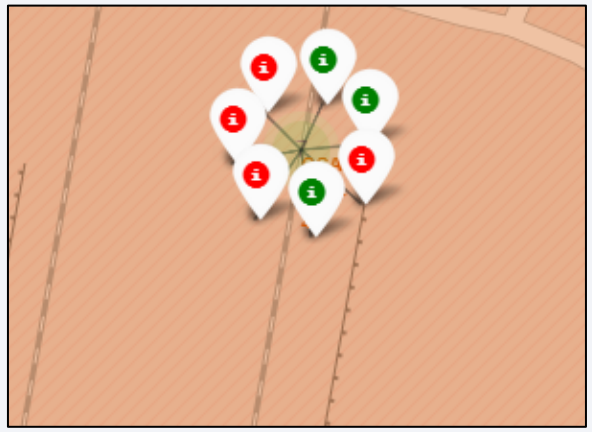


Markers showing launch sites with color labels





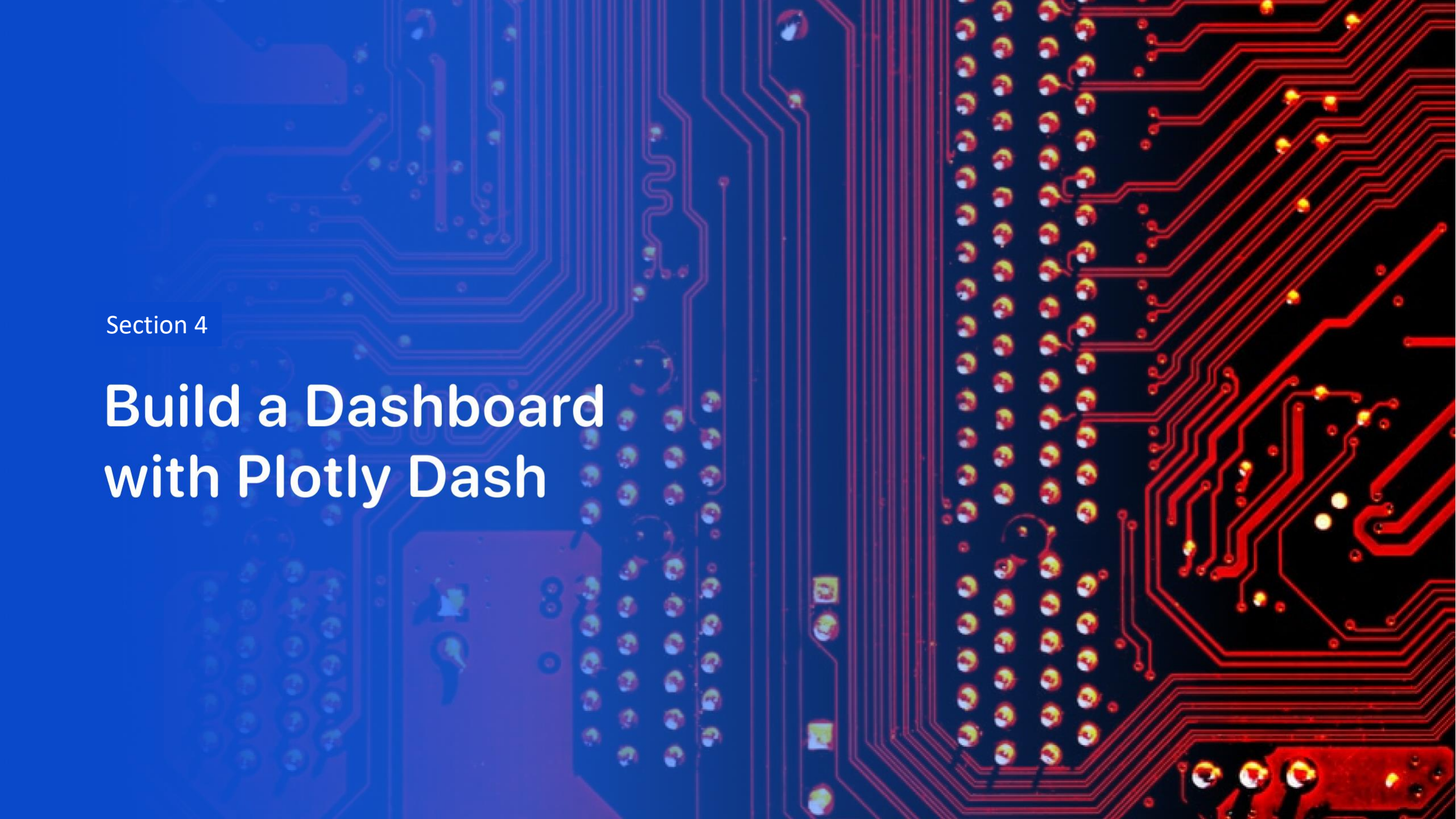




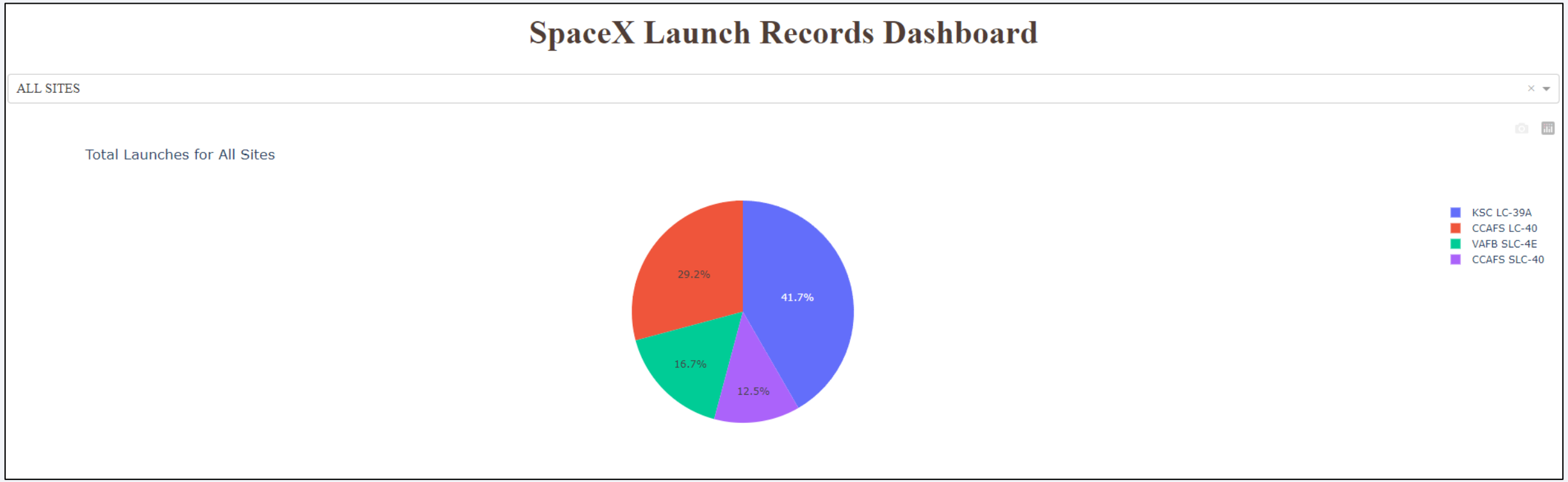
Green Markers show successful launches, instead Red Markers show failures

Launch Sites 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

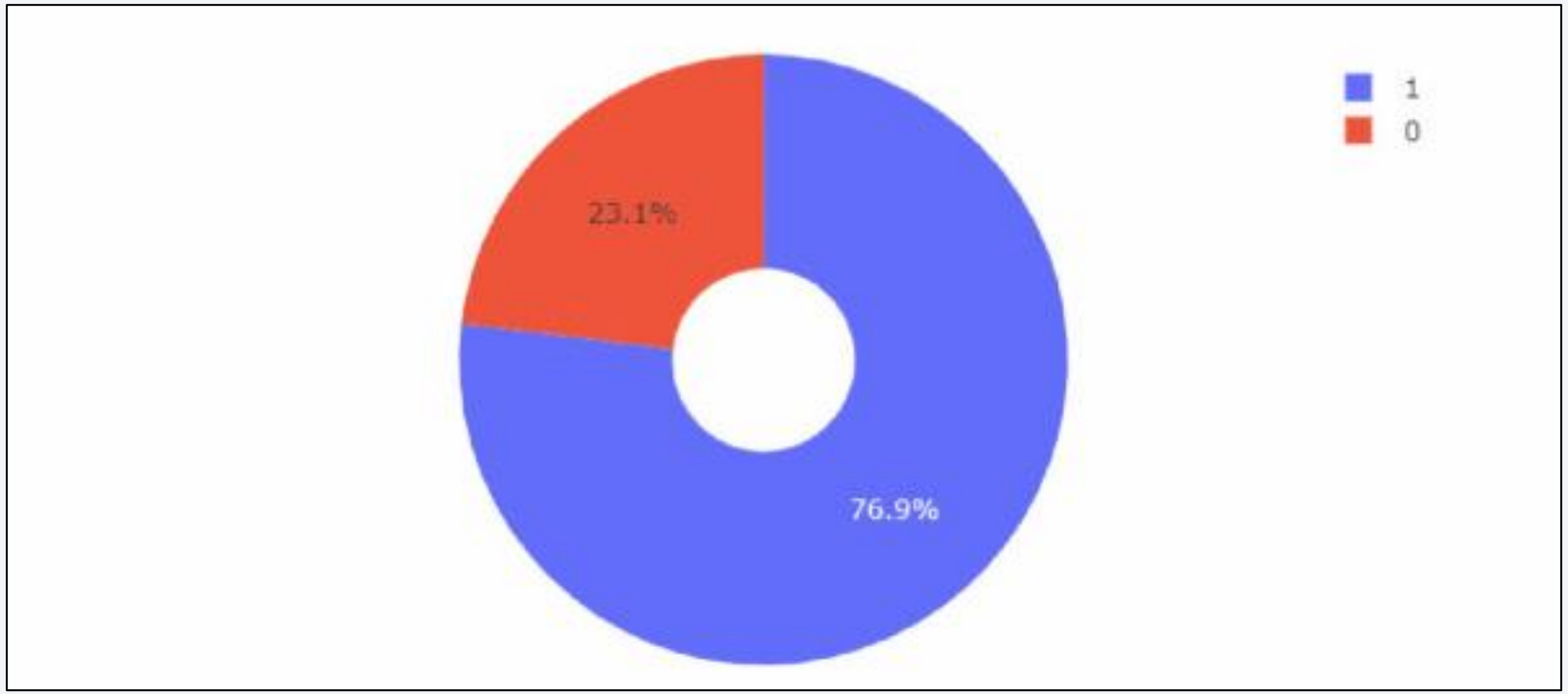

Section 4
Build a Dashboard with Plotly Dash

The success percentage by each sites



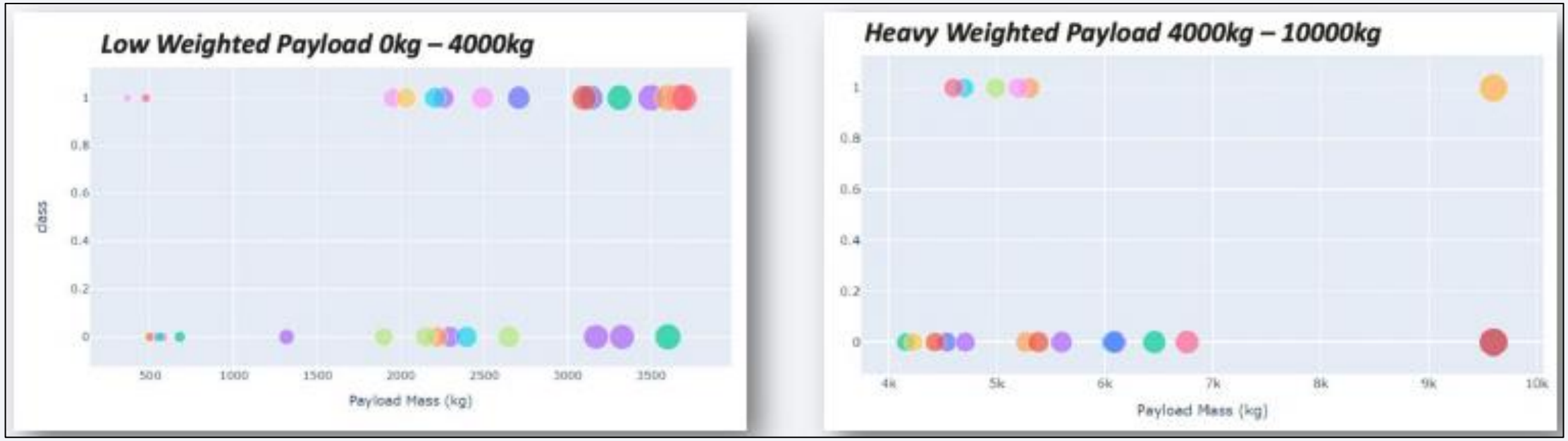
I could see that KSC LC-39A had the most successful launches from all the sites.

The highest launch-success ratio: KSC LC-39A



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

Payload vs Launch Outcome Scatter Plot



I could see that all the success rate for low weighted payload is higher than heavy weighted payload

Section 5

Predictive Analysis (Classification)

Classification Accuracy

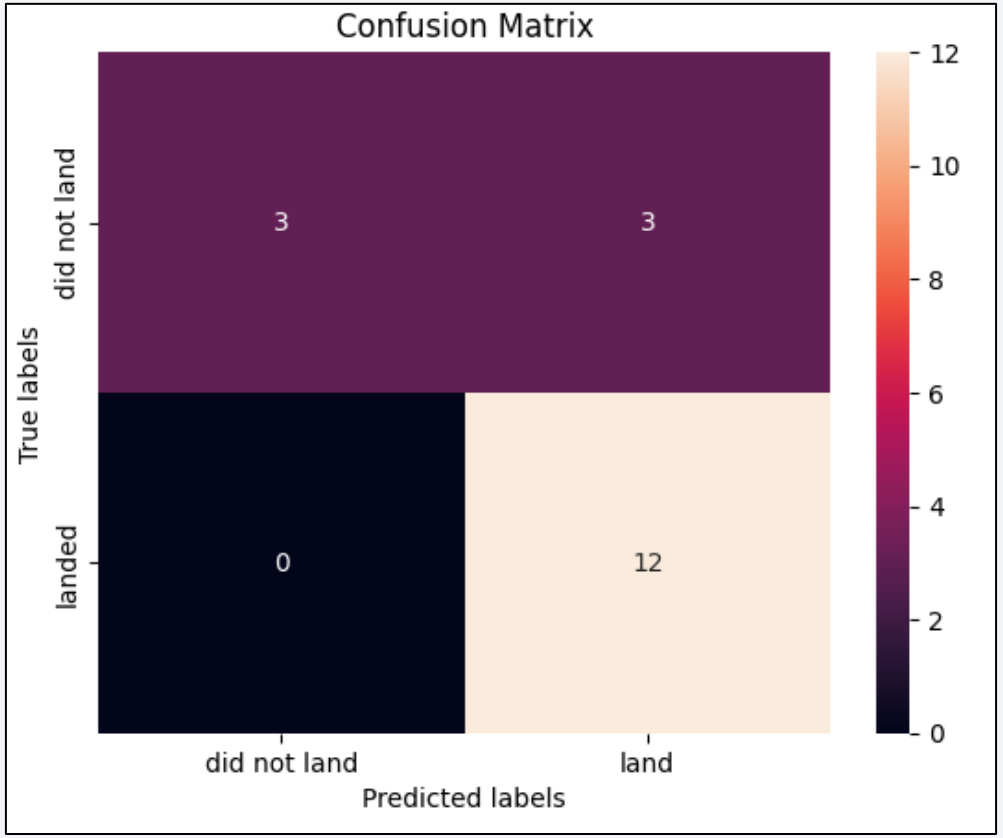
As we can see, by using the code as below: we could identify that the best algorithm to be the Tree Algorithm which have the highest classification accuracy.

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.9017857142857142
Best Params is : {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_sampl
es_split': 10, '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

I can conclude that:

- The Tree Classifier Algorithm is the best Machine Learning approach for this dataset.
- The low weighted payloads (which define as 4000kg and below) performed better than the heavy weighted payloads.
- Starting from the year 2013, the success rate for SpaceX launches is increased, directly proportional time in years to 2020, which it will eventually perfect the launches in the future.
- KSC LC-39A have the most successful launches of any sites; 76.9%
- SSO orbit have the most success rate; 100% and more than 1 occurrence.

Thank you!