IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mohammed Jafri
June 24, 2024

Outline

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

Executive Summary

Summary of methodologies (In order)

- 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
- Predictive Analysis

Introduction

SpaceX creates, produces, and launches cutting-edge rockets and spaceships. The business was established in 2002 by CEO Elon Musk, with the goal of advancing space technology

Some of the problems included are:

1. Identifying all factors that influence landing outcome
2. Relationships between independent variables and how that affects the outcome
3. Best conditions to increase probability of successful landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The data for this projected was collected using SpaceX rest API along with webscrapping wikipedia
- Perform data wrangling
  - The 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 act of obtaining and examining precise information from a variety of sources in order to assess potential outcomes, trends, and probability, among other research concerns. The data for this project was collected using SpaceX REST API and web scrapping from wikipedia

For the REST API, we use a get request and decode the content as JSon and then turn it into a pandas dataframe using the function called json_normalize(). Following this, we clean the data and check for missing values and fill it with what we desire

For web scrapping, we use BeautifulSoup to extract data as an HTML table, parse the table and then convert it into a pandas dataframe

Data Collection – SpaceX API

Use get Request for data using API

Using Json_normalize() to convert result into dataframe

Clean data and fill missing values accordingly

https://github.com/mjaf04/Applied-Data-Science-Capstone/blob/main/Complete%20the%20Data%20Collection%20API%20Lab.ipynb

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
[77]
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
[78]
response = requests.get(spacex_url)
```

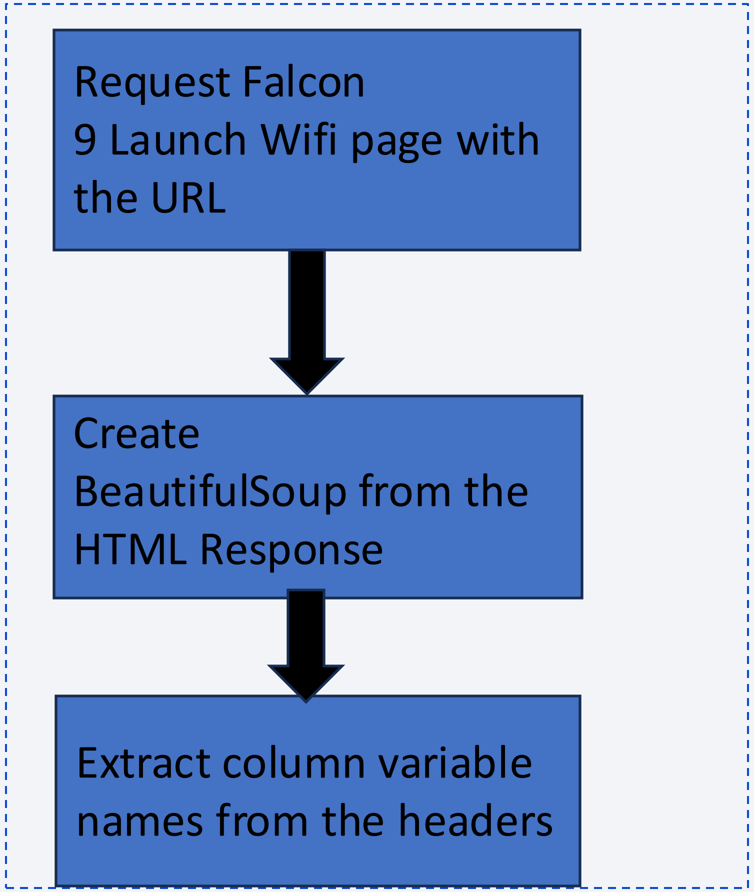
Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
[82]
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

```
# 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 payloads in a single roc
data = data[data['cores'].map(len)= Loading...
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)]
[84]
```

Data Collection - Scraping



First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url

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

Create a BeautifulSoup object from the HTML response

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

soup = BeautifulSoup(data)
```

Next, we just need to iterate through the <th> elements and apply the provided extract_column_from_header() to extract column name one by one

```
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

for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
```


https://github.com/mjaf04/Applied-Data-Science-Capstone/blob/main/Complete%20the%20Data%20Collection%20with%20WebScraping%20Lab.ipynb

Data Wrangling

1. We explored data to determine the label for training supervised models
2. Calculated number of launches (on each site), number and occurrence of each orbit, number and occurrence of mission outcome per orbit

3. We created a landing outcome training label from 'Outcome' Column

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

```
# landing_outcomes = values on Outcome column

landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

`True Ocean` means the mission outcome was successfully landed to a specific region of the ocean while `False Ocean` means the mission outcome was unsuccessfully landed to a specific region of the ocean. `True RTLS` means the mission outcome was successfully landed to a ground pad `False RTLS` means the mission outcome was unsuccessfully landed to a ground pad. `True ASDS` means the mission outcome was successfully landed to a drone ship `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

https://github.com/mjaf04/Applied-Data-Science-Capstone/blob/main/Data%20Wrangling.ipynb

EDA with Data Visualization

1. Scatter Plots

- This was used to find relationships between several variables (eg: PayLoad and Launch Site). This plot shows dependancy of variables on each other. When you observe patterns, it is easy to tell which factors influence successful landing outcome

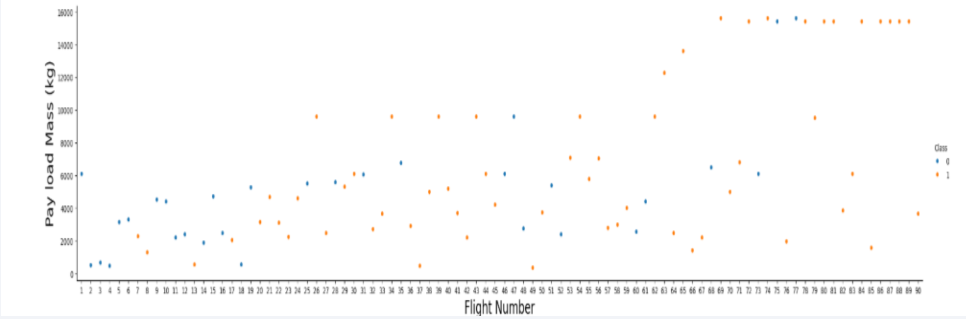
2. Bar graphs

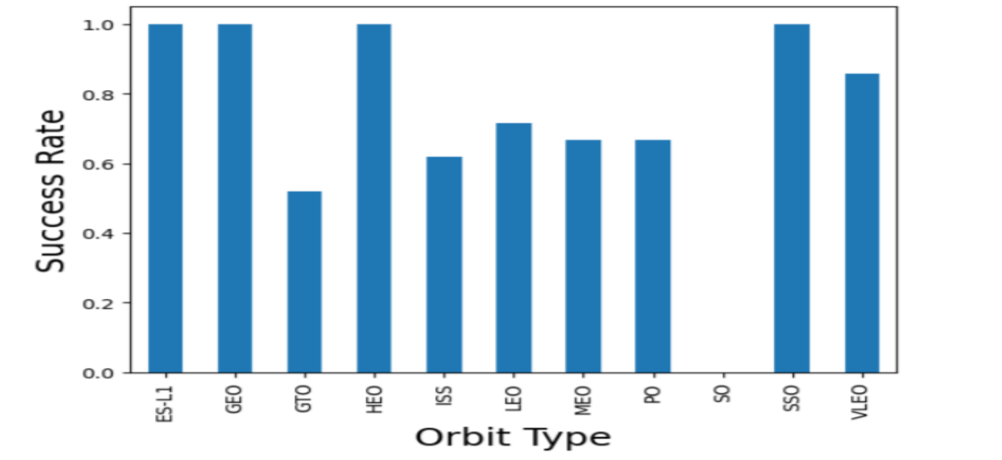
- Easiest way to interpret the relationship between variables. We used this graph to determine which orbits have highest success rate

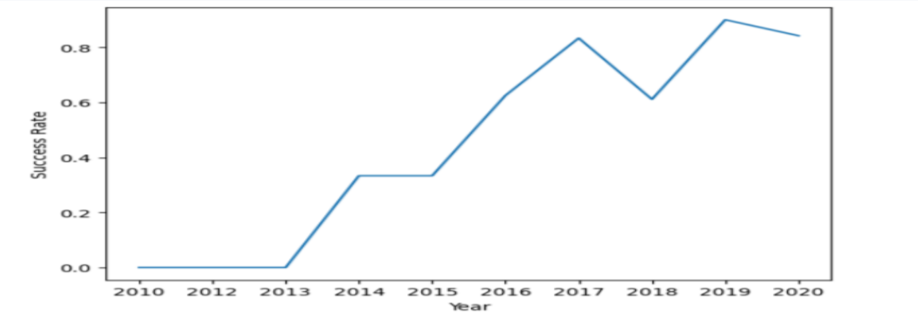
3. Line Chart

- To find trends and patterns of variables over time. We used this graph to visualize launch success yearly trend

https://github.com/mjaf04/Applied-Data-Science-Capstone/blob/main/EDA%20With%20Visualization%20Lab.ipynb







EDA with SQL

With SQL, we performed many queries to better understand the data: Some included (but not all)

1. Displaying 5 records where launch sites begin with the string 'CCA'.
2. Displaying the total payload mass carried by booster launched by NASA (CRS). - Displaying the average payload mass carried by booster version F9 v1.1.
3. Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000. -
4. Listing the total number of successful and failure mission outcomes.

https://github.com/mjaf04/Applied-Data-Science-Capstone/blob/main/Complete%20the%20EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

We placed a circle marker with the name of the launch location labeled around each launch site, using the latitude and longitude coordinates at each launch site.

Next, we used MarkerCluster() to assign Red and Green markers to classes 0 and 1 on the dataframe launch_outcomes(failure,success).

We next computed the launch sites' distances from various landmarks using Haversine's formula to determine the answers to the following questions:

- How near are the launch locations to roads, trains, and coastlines?
- How near are the launch locations to neighboring cities?

https://github.com/mjaf04/Applied-Data-Science-Capstone/blob/main/Launch%20Sites%20Locations%20Analysis%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

Using Plotly Dash, we created an interactive dashboard that lets the user alter the data as needed.

• We created pie charts that displayed each site's total launches. This chart is a great way of showing the differences between each site's launches and great for comparison

• Next, for each booster variant, we created a scatter graph that displayed the link between the outcome and payload mass (kg). Scatter plots are a great way to display what happens to one variable when another variable is changed

https://github.com/mijaf04/Applied-Data-Science-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

BUILDING THE MODEL

- Loaded the data set
- Transformed the data and then split it into test/training sets
- Chose a ML technique

EVALUATING THE MODEL

- Check accuracy of each model
- Plot the confusion matrices

IMPROVING THE MODEL

- Used featured engineering along with Algorithm tuning

DETERMINING MOST ACCURATE MODEL

- The best model is the one with the highest accuracy

https://github.com/mjaf04/Applied-Data-Science-Capstone/blob/main/Complete%20the%20Machine%20Learning%20Prediction%20lab.ipynb

Results

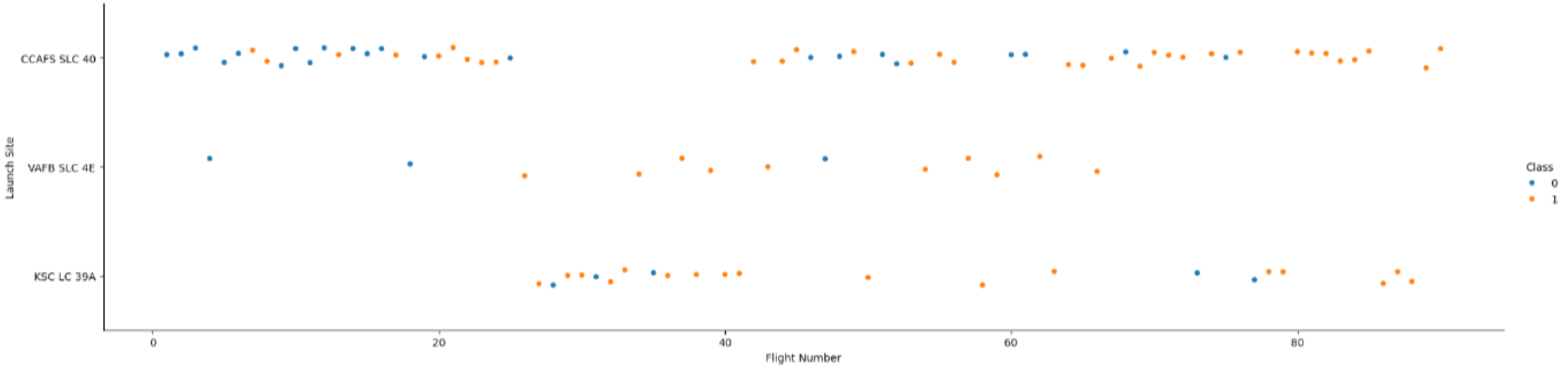
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

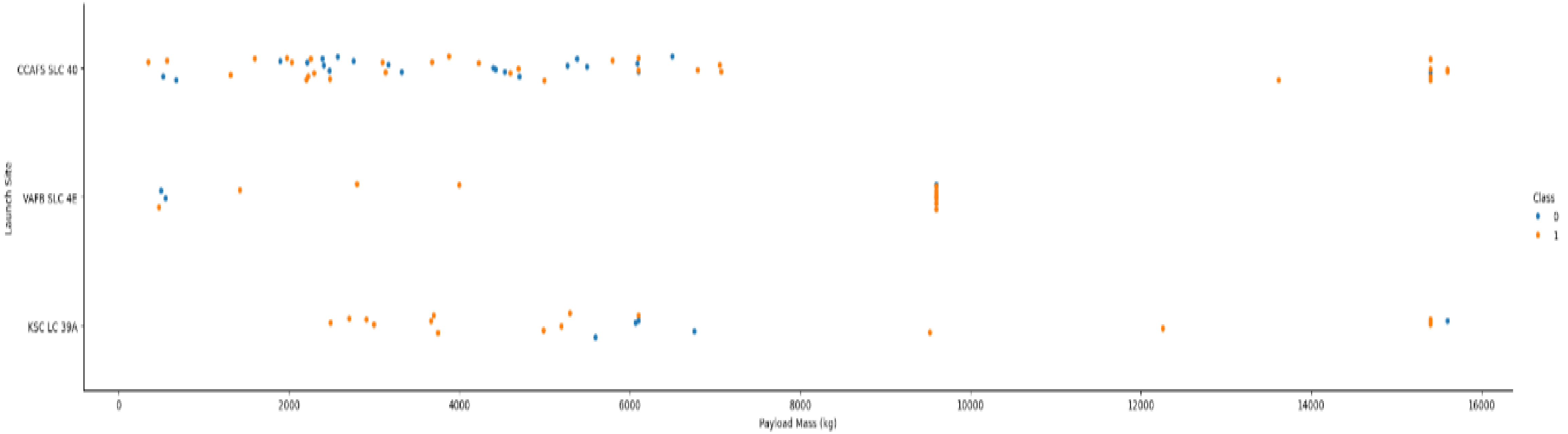
Flight Number vs. Launch Site

- This scatter plot demonstrates that the more flights there are, the
- of the launch site, the higher the likelihood of success.
- The location CCAFS SLC40, however, exhibits the least pattern of this.



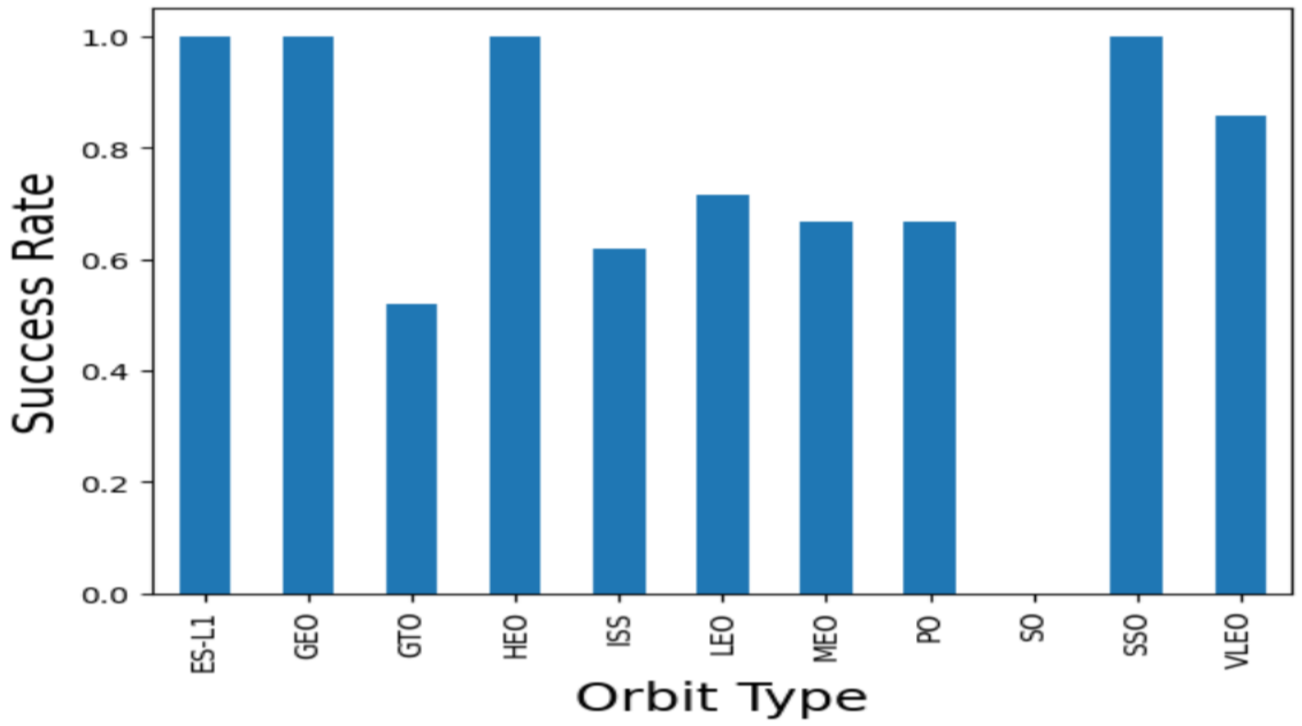
Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site
- Show the screenshot of the scatter plot with explanations



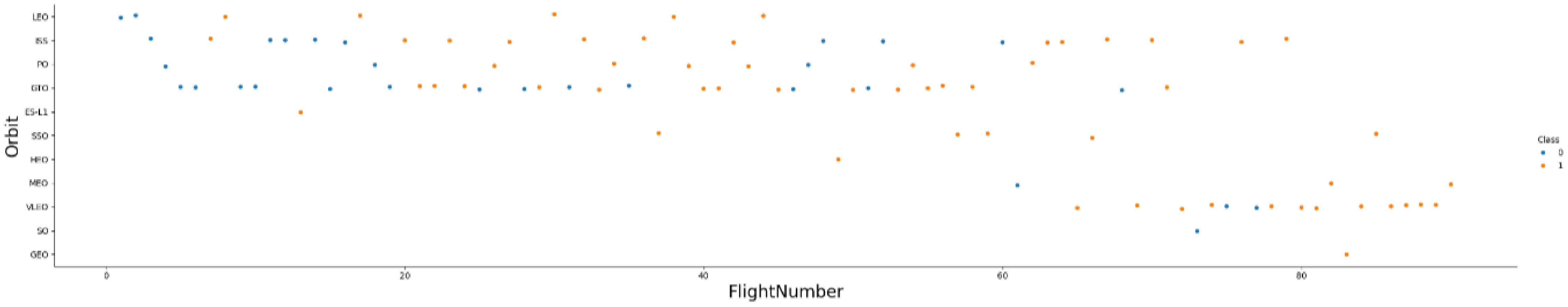
Success Rate vs. Orbit Type

- This chart illustrated how orbits could affect landing results. While SO orbit provided a 0% success rate, other orbits, like SSO, HEO, GEO, and ES-L1, had 100% success rates.



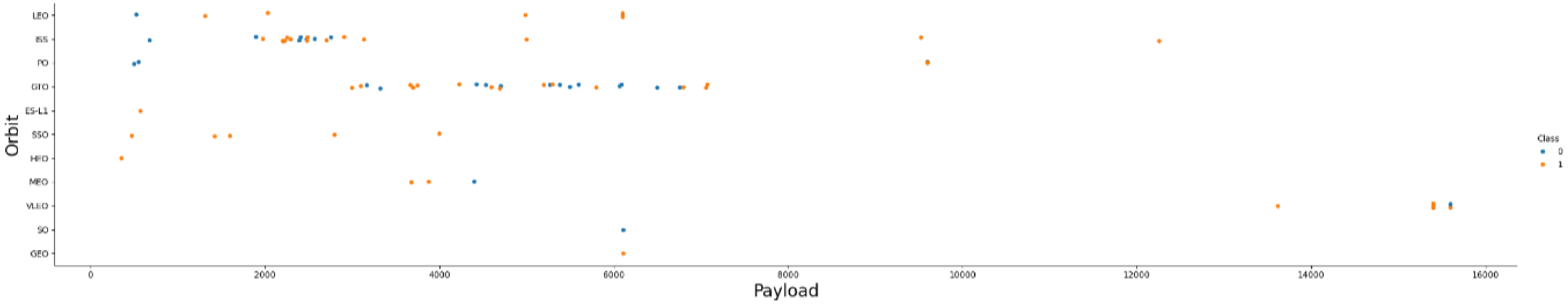
Flight Number vs. Orbit Type

- With the exception of the GTO orbit, which indicates no association between the two features, this scatter plot demonstrates that, generally speaking, the more the flight number on each orbit, the greater the success rate (particularly on the LEO orbit).



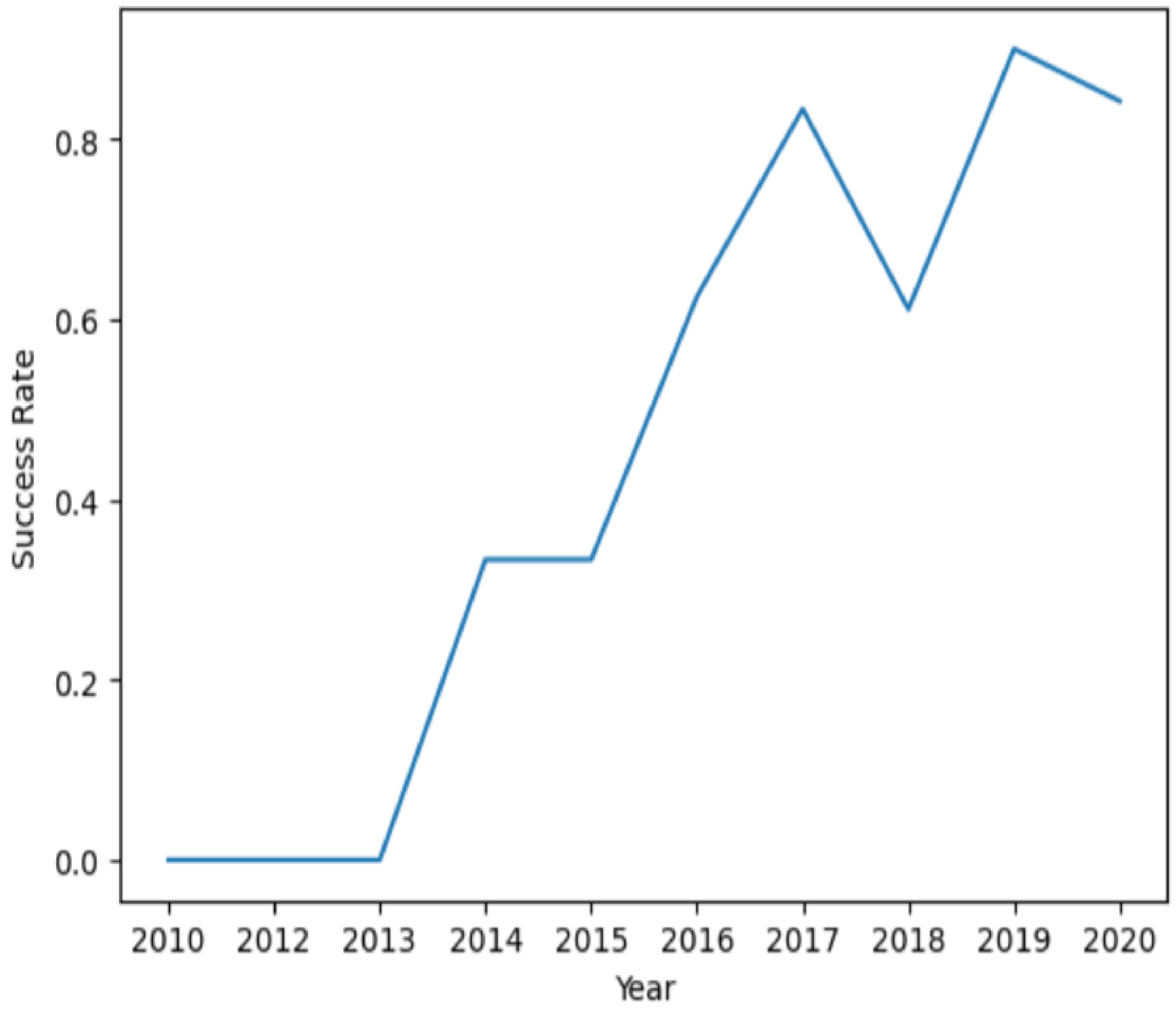
Payload vs. Orbit Type

- A heavier payload benefits P0 orbit, ISS, and LEO.
- On the other hand, it negatively affects the MEO and VLEO orbits.
- The GTO orbit appears to show no correlation between the characteristics.
- Furthermore, additional datasets are required for SO, GEO, and HEO orbit to detect any patterns or trends.



Launch Success Yearly Trend

- This graph clearly depicts an increasing trend from the year 2013 until 2020.



All Launch Site Names

To display only distinct launch sites from the SpaceX data, we employed the keyword DISTINCT.

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

```
%%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'

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

Total Payload Mass

Using the query shown, we calculated the total payload carried by boosters from NASA

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

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';
```

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

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

The average payload mass caried by booster version F9 v1.1 was 340.4 using this query

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

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.0%';
```

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

AVG(PAYLOAD_MASS__KG_)
340.4

First Successful Ground Landing Date

We used the min() function to find the result and the date was December 22, 2015

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
%%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

In order to identify successful landings with payload masses larger than 4000 but less than 6000, we employed the AND condition after using the WHERE clause to filter for boosters that have successfully landed on drone ships.

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

```
%%sql
SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' AND 4000 < PAYLOAD_MASS__KG_ < 6000;
```

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

Booster_Version
F9 FT B1021.1
F9 FT B1022
F9 FT B1023.1
F9 FT B1026
F9 FT B1029.1
F9 FT B1021.2
F9 FT B1029.2
F9 FT B1036.1
F9 FT B1038.1
F9 B4 B1041.1
F9 FT B1031.2
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

Total Number of Successful and Failure Mission Outcomes

To filter for WHERE Mission Outcome was successful or unsuccessful, we utilized wildcards like %.

List the total number of successful and failure mission outcomes

```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

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

Boosters Carried Maximum Payload

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

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

```
%%sql
SELECT DISTINCT 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

We used LIKE, BETWEEN, AND conditions so we can filter for failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%%sql
SELECT LANDING_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE Landing_Outcome = 'Failure (drone ship)' AND subs
```

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

Landing_Outcome	Booster_Version	Launch_Site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

Using the WHERE clause, we filtered for landing outcomes BETWEEN 2010-06-04 and 2010-03-20. We first chose the landing outcomes and the COUNT of landing outcomes from the data. The landing outcomes were sorted using the GROUP BY clause, and the grouped landing outcomes were arranged in descending order using the ORDER BY clause.

```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

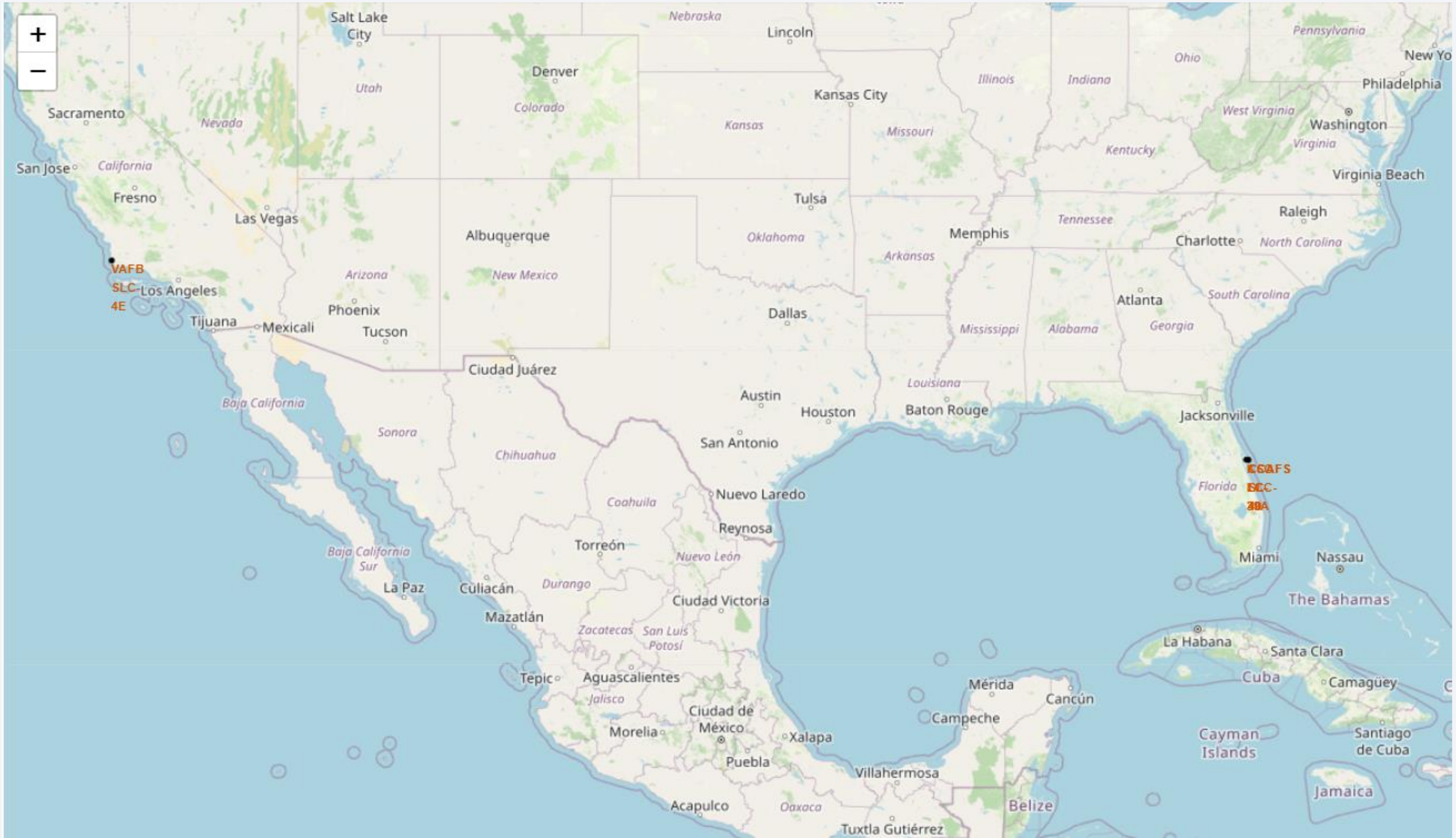
Landing Outcome	Total Count
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

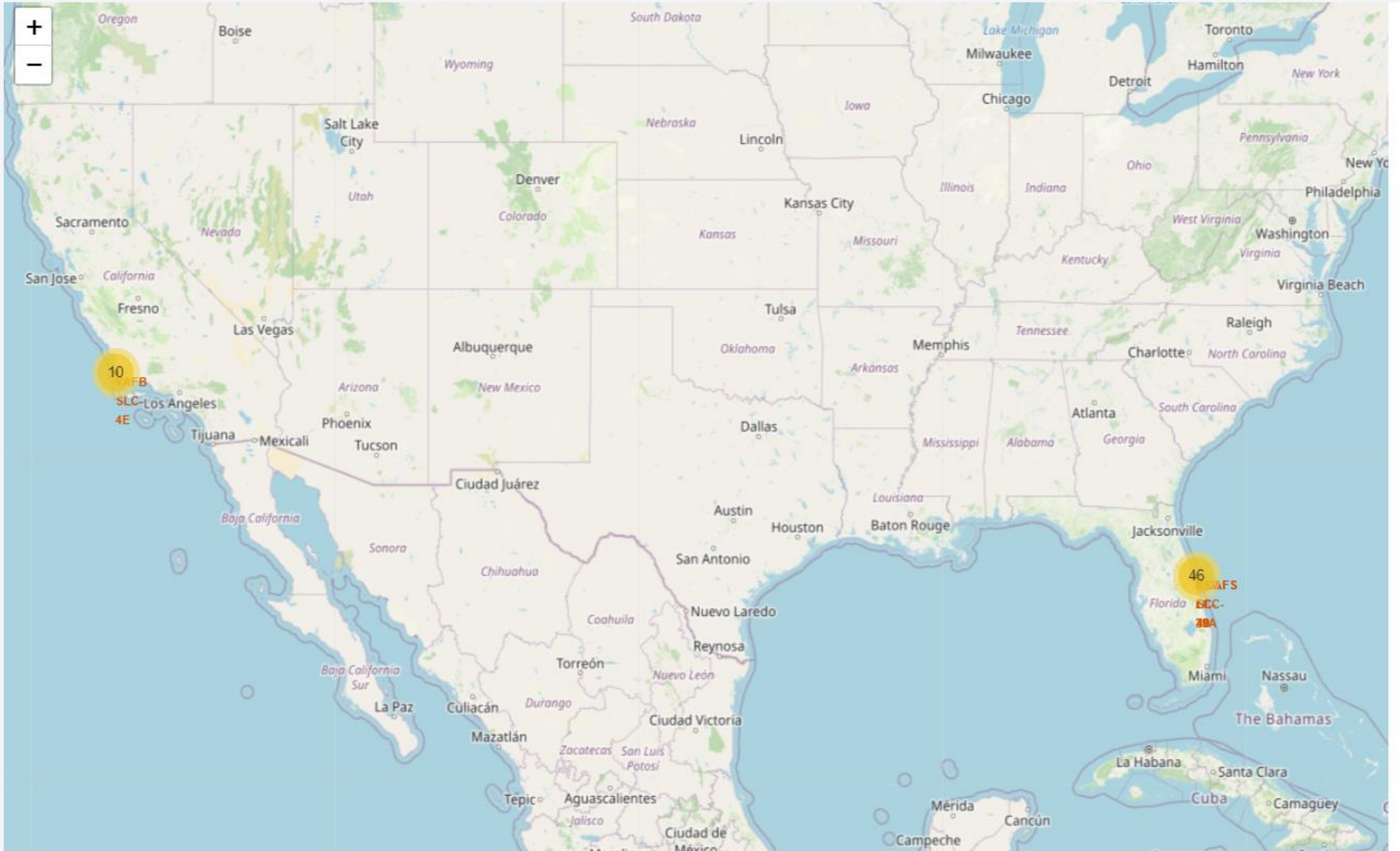
<Folium Map Screenshot 1>

All SpaceX launch sites are located within the United States of America

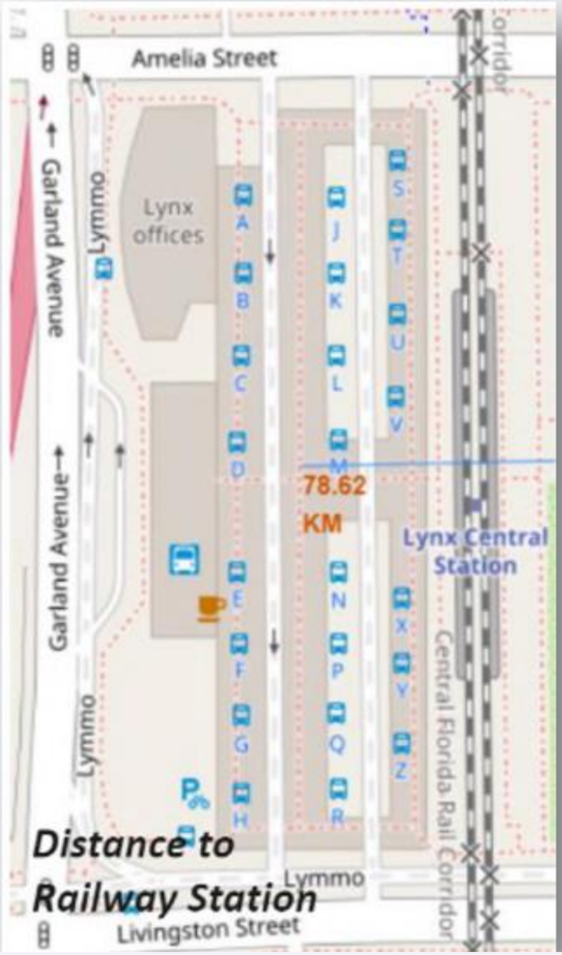


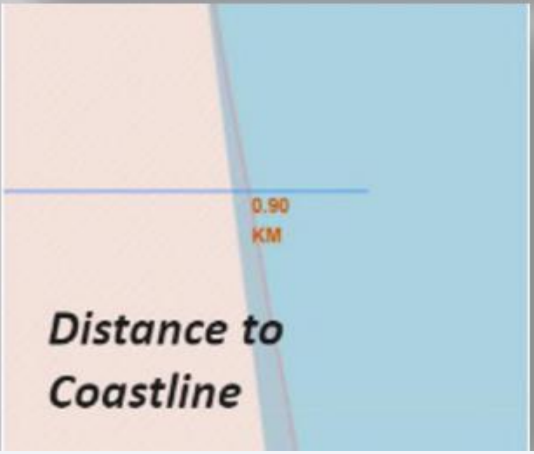
<Folium Map Screenshot 2>

Number of launches at each launch site location



<Folium Map Screenshot 3>





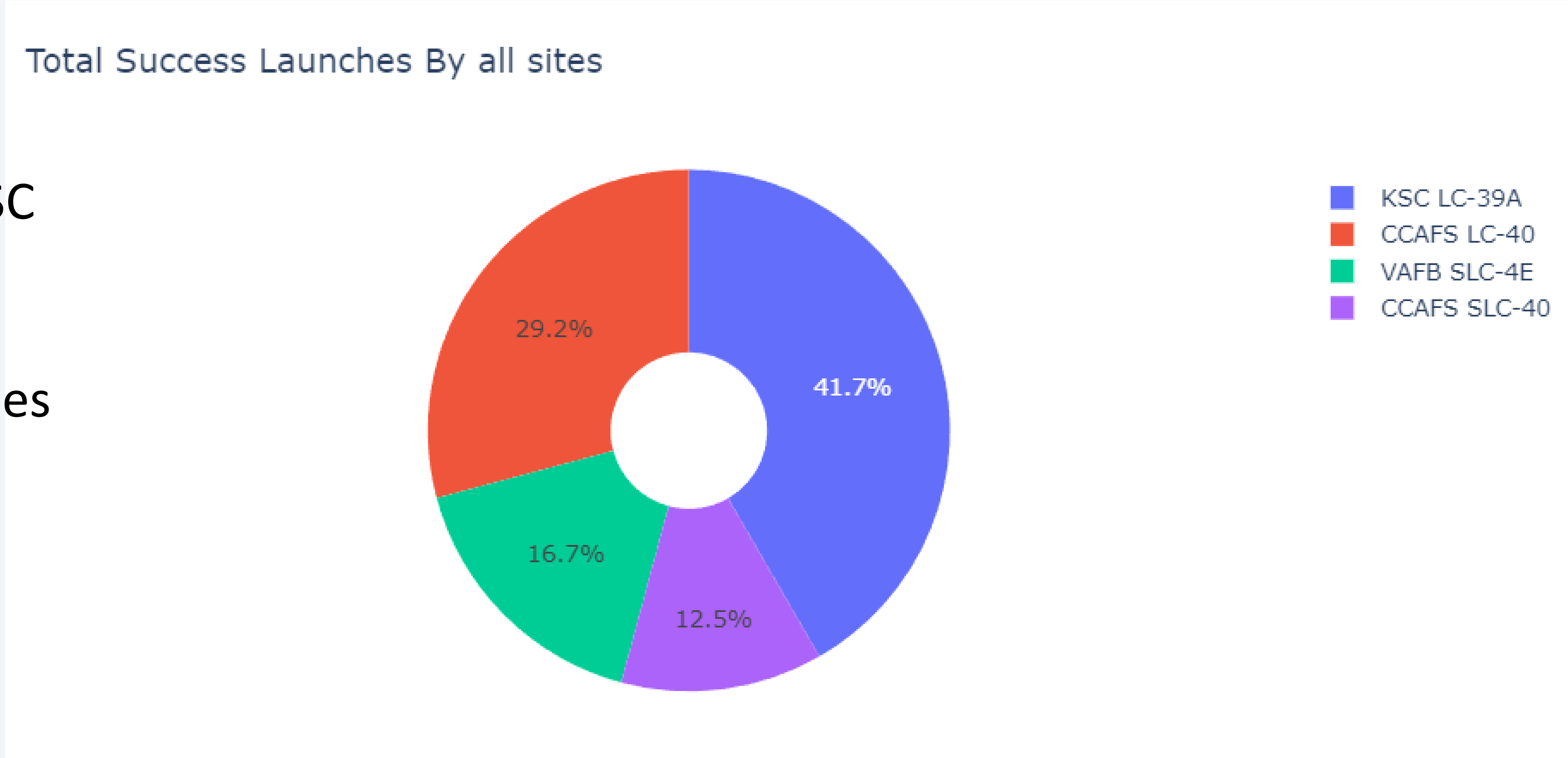
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

Build a Dashboard with Plotly Dash

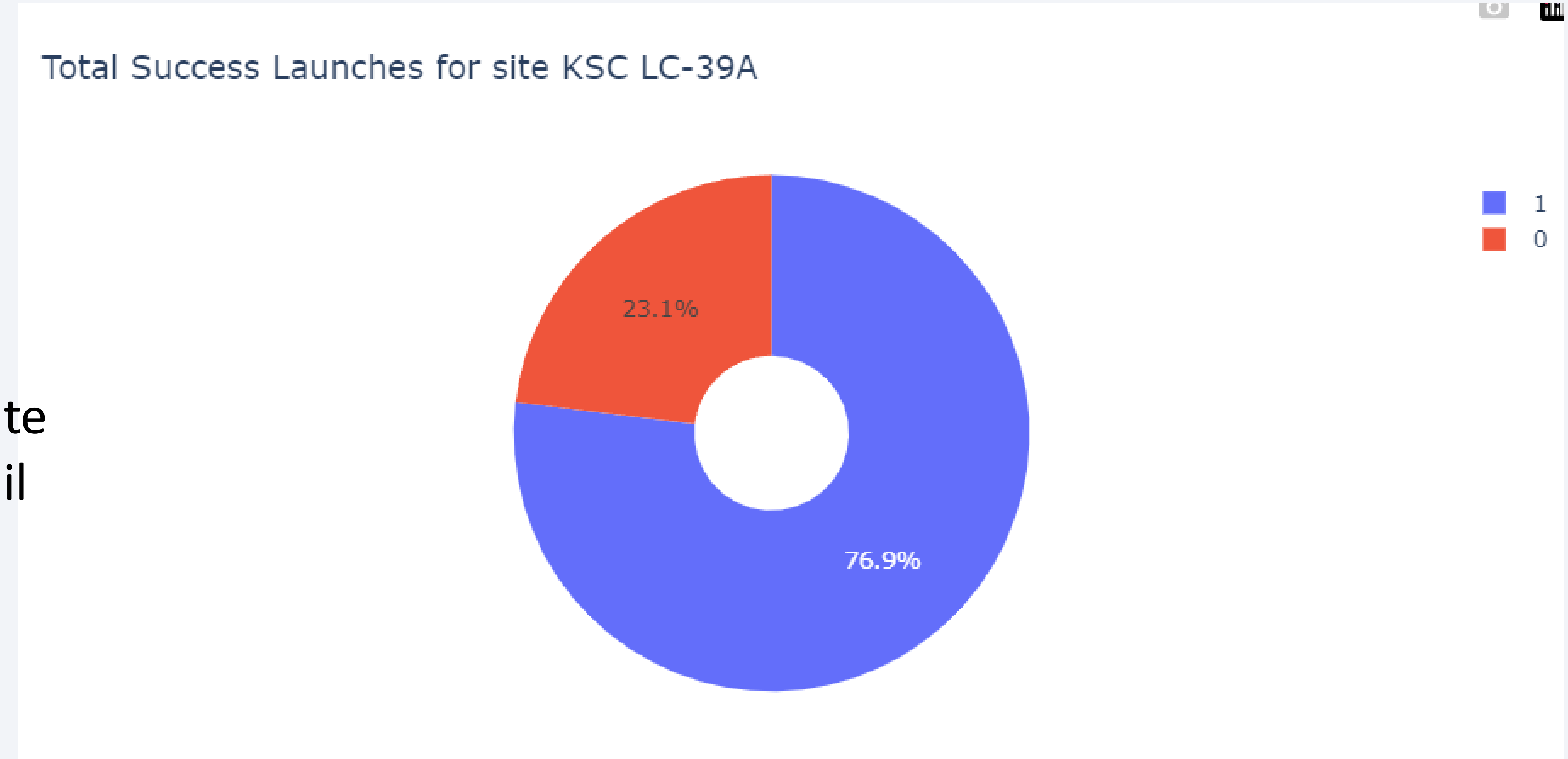
<Dashboard Screenshot 1>

As we can see, KSC LC-39A had the most successful launches from all sites

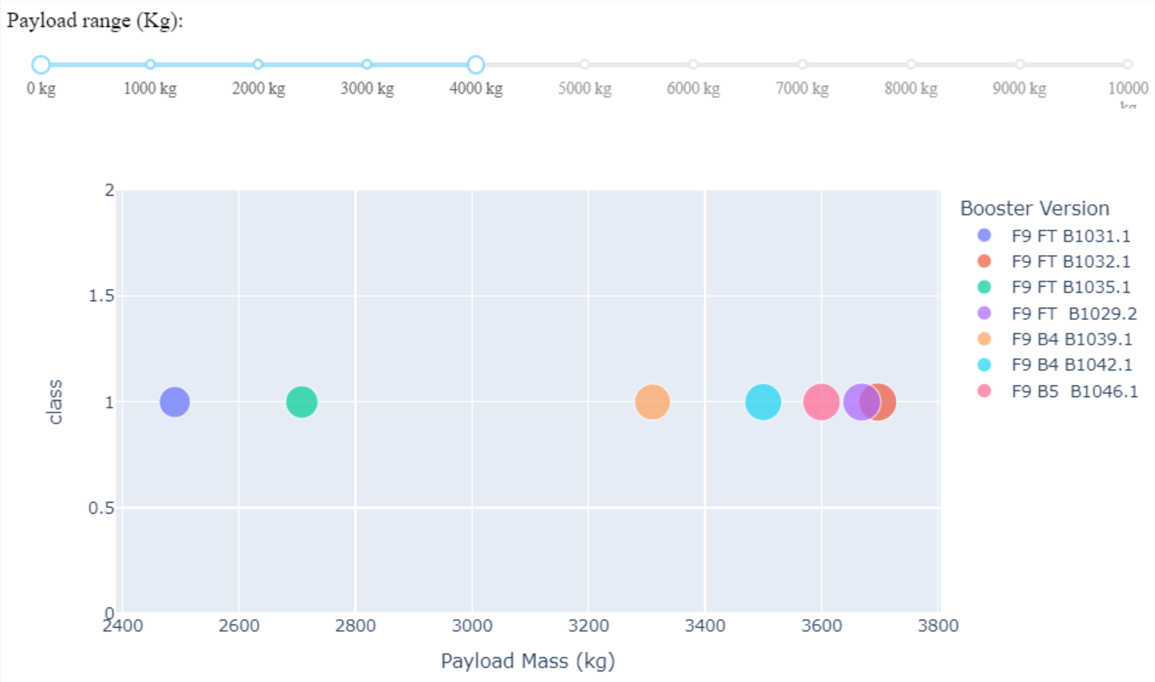


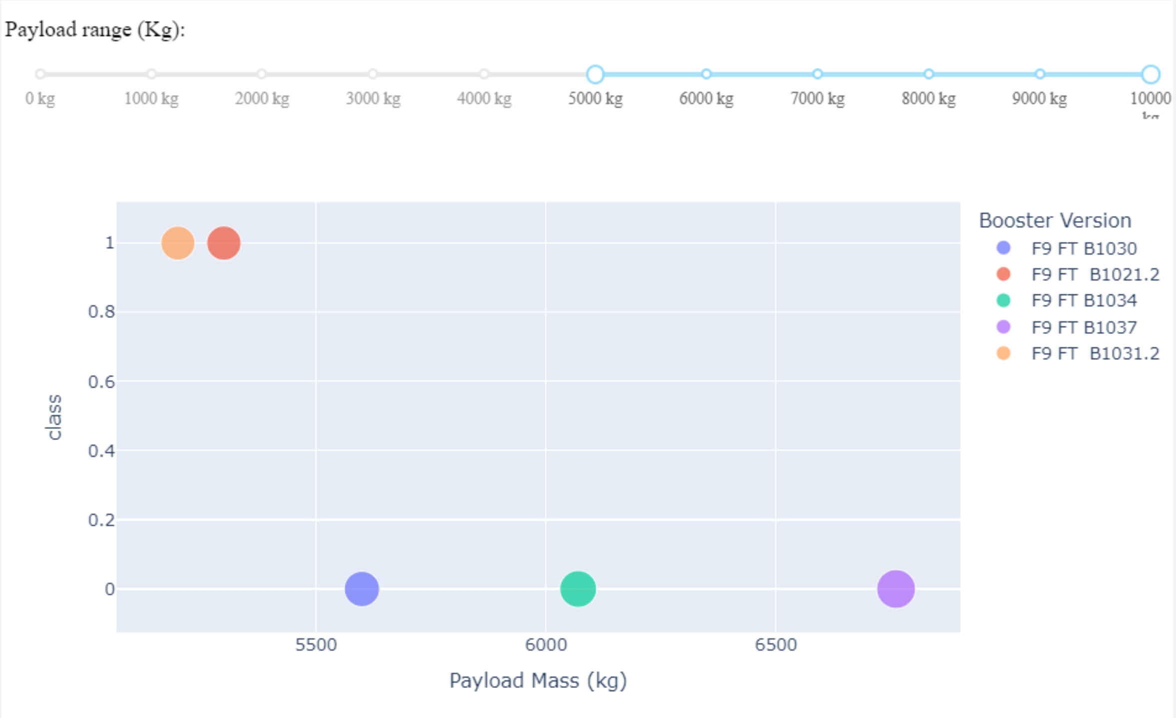
<Dashboard Screenshot 2>

KSC LC-39A had a 76.9% success rate with only 23.1% fail rate



<Dashboard Screenshot 3>





We can see that all the success rate for low weighted payload is higher than heavy weighted payload

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Find the method performs best:

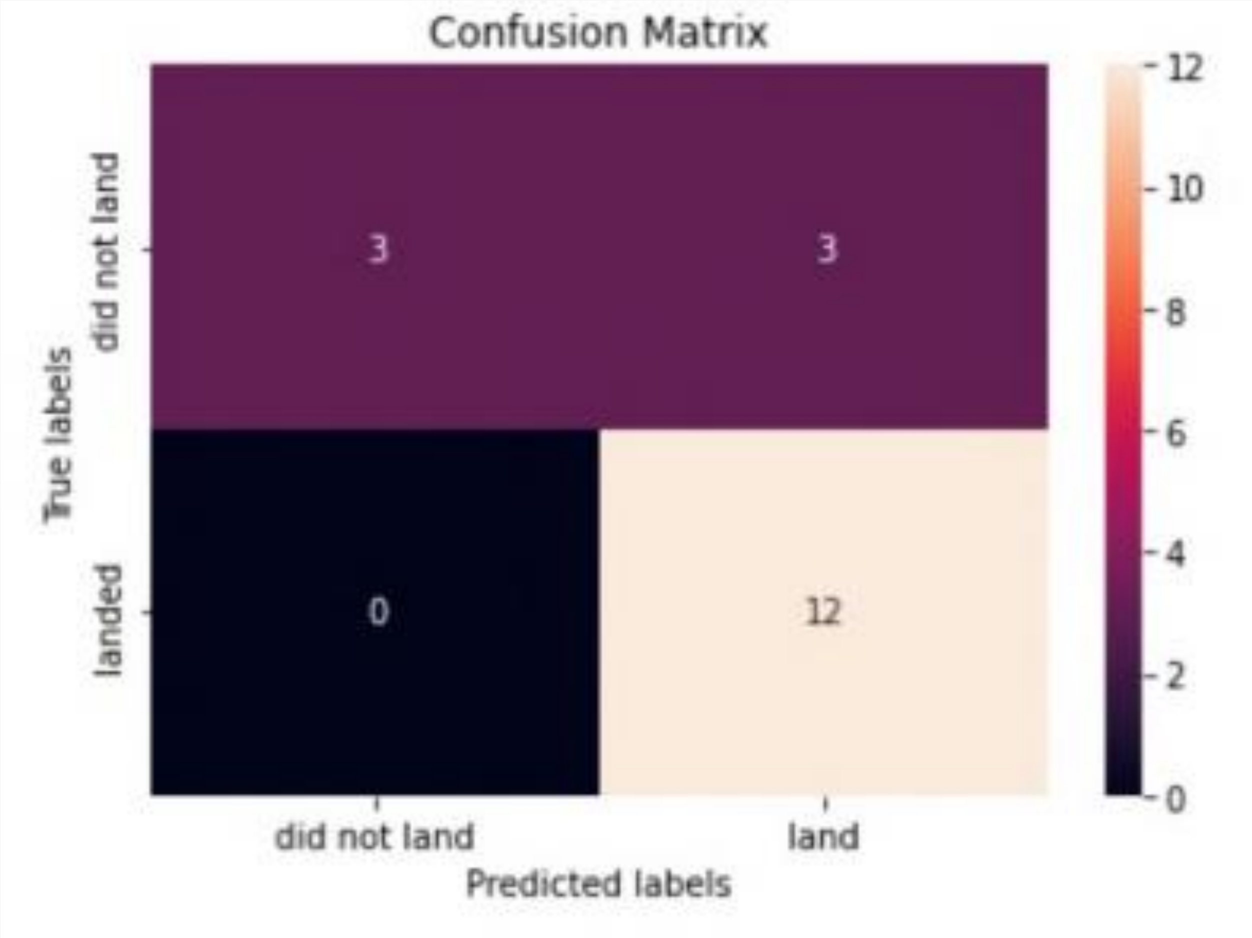
```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params 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 0.8910714285714285
Best Params is : {'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
```

The Tree algorithm has the highest classification accuracy

Confusion Matrix

The decision tree classifier's confusion matrix demonstrates the classifier's ability to discriminate between the various classes. False positives are the main issue.i.e., the classifier interprets an unsuccessful landing as a successful landing.



Conclusions

We can draw the following conclusions:

• For this dataset, the Tree Classifier Algorithm is the best machine learning technique.

• Since 2013, SpaceX launches have had a higher success rate, which increases in direct proportion to the number of years until 2020, when it will finally perfect future launches.

• Compared to big weighted payloads, light weighted payloads—defined as those weighing less than 4,000 kg—performed better.

• Of all the locations, KSC LC-39A had the most successful launches (76.9%).

• The SSO orbit has the highest success rate, occurring more than once and at 100%.

Thank you!