IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ruth Efrain
06.08.2024

Outline

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

Executive Summary

Summary of Techniques Applied

- Collection of data via API and web scraping
- Exploratory Data Analysis (EDA) with Visualization of Data
- EDA with SQL
- Interactive Map using Folium
- Dashboard utilizing Plotly Dash
- Predictive Analysis

A summary of all results

- Outcomes of exploratory data analysis (EDA)
- Dashboard with Interactive Maps
- Predictive Results

Introduction

- SpaceX, being an innovative company, has disrupted the space industry by providing rocket launches, specifically Falcon 9, at a cost as low as $62 million, in contrast to competitors who charge up to $165 million. The majority of these cost savings have been achieved through SpaceX's ingenious strategy of re-landing rockets following each launch and repurposing them for subsequent missions. By persisting in this behaviour, you have the potential to further reduce the price.

- As a data scientist for a startup aiming to rival SpaceX, my task is to create a machine learning pipeline that can forecast the likelihood of a successful first-stage landing. Based on the outcomes of this experiment, rivals can ascertain their initial offers in competition with SpaceX.

- The problems encompassed the identification of all variables that could impact the success of the landing, the examination of the relationship between each independent variable and the outcome, and the determination of the ideal conditions to maximise the probability of a successful landing.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Two data sources were used to create Space X dataframes
    - SpaceX API URL: https://api.spacexdata.com/v4/.
    - Web scraping (source: https://en.wikipedia.org/wiki/List_of_Falcon/_9/_and_Falcon_Heavy_launches)
- Perform data wrangling
  - After collecting and analysing data, a label was created to enrich the dataset.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - The data collected so far was standardised, divided into training and test datasets, and analysed using four different classification models. The accuracy of each model was evaluated using different combinations of variables.

Data Collection

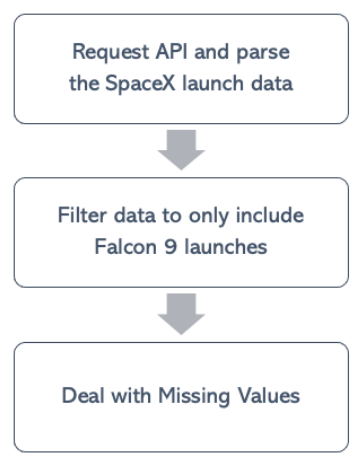
- Initially, the datasets were generated by utilising the SpaceX REST API, enabling us to engage with launch data provided by SpaceX.
- The API will provide us with data regarding launches, including details about the rocket type, payload, launch parameters, landing parameters, and the outcome of the landing.
- Our objective is to utilise this data to assess the probability of a successful landing for a SpaceX rocket. The user's text consists of a bullet point symbol.
- To utilise the SpaceX REST API, simply input the URL "https://api.spacexdata.com/v4" into the address bar of your web browser.
- Additionally, the utilisation of BeautifulSoup for webscraping Wikipedia is another technique employed to acquire Falcon 9 launch data.

Data Collection – SpaceX API

GITHUB

Add the GitHub URL of the completed SpaceX API calls notebook

https://github.com/cherubims/IBM-Capstone-SpaceX/blob/main/1.%20Data%20Collection%20via%20API/jupyter-labs-spacex-data-collection-api.ipynb




```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)

In [9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'

In [10]: response.status_code

Out[10]: 200

In [11]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())

In [16]: # Call getBoosterVersion
         getBoosterVersion(data)

In [18]: # Call getLaunchSite
         getLaunchSite(data)

In [19]: # Call getPayloadData
         getPayloadData(data)

In [20]: # Call getCoreData
         getCoreData(data)

[21]: launch_dict = {'FlightNumber': list(data['flight_number']),
       'Date': list(data['date']),
       'BoosterVersion':BoosterVersion,
       'PayloadMass':PayloadMass,
       'Orbit':Orbit,
       'LaunchSite':LaunchSite,
       'Outcome':Outcome,
       'Flights':Flights,
       'GridFins':GridFins,
       'Reused':Reused,
       'Legs':Legs,
       'LandingPad':LandingPad,
       'Block':Block,
       'ReusedCount':ReusedCount,
       'Serial':Serial,
       'Longitude': Longitude,
       'Latitude': Latitude}

In [24]: # Hint data['BoosterVersion']!='Falcon 1'
         data_falcon9=launch_data[launch_data["BoosterVersion"]!="Falcon 1"]
         data_falcon9

[25]: data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
      data_falcon9

In [27]: # Calculate the mean value of PayloadMass column
         mean_PayloadMass=data_falcon9["PayloadMass"].mean()

         # Replace the np.nan values with its mean value
         data_falcon9["PayloadMass"] = data_falcon9["PayloadMass"].replace(np.nan, mean_PayloadMass)

         data_falcon9.isnull().sum()

In [28]: data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

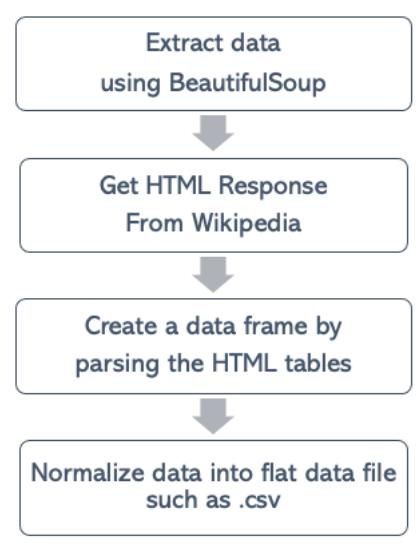
1. The following URL will send a request for rocket launch data to SpaceX's API.
2. In order to maintain consistency, we were used the following static response object to return JSON results.
3. The JSON data in the response object converted into a Pandas dataframe using .json_normalize().
4. To retrieve features from the dataframe, we employed custom functions.
5. The columns of a dataframe were constructed by assigning extracted lists to dictionaries.
6. With a filtering operation on the BoosterVersion column, we were able to extract only the launches involving a Falcon 9 and save them in a separate dataframe we called data_falcon9.
7. We reseted the FlightNumber column after removing some values.
8. We calculated the mean for the PayloadMass using the .mean(). Then, we used it in the .replace() function to replace np.nan values in the data.
9. Dataframe was saved as a comma-separated values (CSV) file.

Data Collection - Scraping

GITHUB

Add the GitHub URL of the completed web scraping notebook:

https://github.com/cherubims/IBM-Capstone-SpaceX/blob/main/2.%20Data%20Collection%20via%20Webscrape/jupyter-labs-webscraping.ipynb



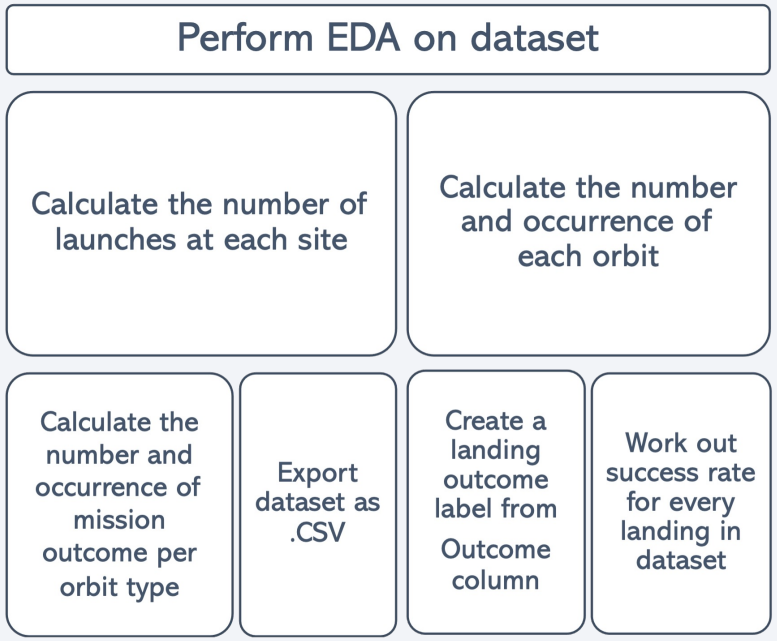

```
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        response=requests.get(static_url).text

In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup=BeautifulSoup(response)

In [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")

In [10]: column_names = []

         # Apply find_all() function with `th` element on first_launch_table
         # Iterate each th element and apply the provided extract_column_from_header() to get a column name
         # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

         element = first_launch_table.find_all('th')
         for row in range(len(element)):
             try:
                 name = extract_column_from_header(element[row])
                 if (name is not None and len(name) > 0):
                     column_names.append(name)
             except:
                 pass

In [12]: launch_dict= dict.fromkeys(column_names)

         # Remove an irrelvant column
         del launch_dict['Date and time ( )']

         # Let's initial the launch_dict with each value to be an empty list
         launch_dict['Flight No.'] = []
         launch_dict['Launch site'] = []
         launch_dict['Payload'] = []
         launch_dict['Payload mass'] = []
         launch_dict['Orbit'] = []
         launch_dict['Customer'] = []
         launch_dict['Launch outcome'] = []
         # Added some new columns
         launch_dict['Version Booster']=[]
         launch_dict['Booster landing']=[]
         launch_dict['Date']=[]
         launch_dict['Time']=[]

In [13]: 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 correspondlng 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:

In [14]: df=pd.DataFrame(launch_dict)

In [15]: df.to_csv('spacex_web_scraped.csv', index=False)
```

1. An object is created with HTML page from the URL.
2. We used requests.get() method to have response from the HTML object.
3. We created a BeautifulSoup object from the HTML response.
4. We used the find_all function in the BeautifulSoup object, with element type `table` to assign the result to a list called `html_tables`.
5. We iterated through the <th> elements and applied the extract_column_from_header() to extract column name one by one.
6. We created an empty dictionary with keys from the extracted column names. Later, this dictionary was converted into a Pandas dataframe
7. We appended data to keys in the dictionary.
8. The dictionary was converted into a dataframe.
9. Dataframe saved as a CSV file.

Data Wrangling

GITHUB

- We conducted an Exploratory Data Analysis (EDA) to identify patterns in the data and establish the appropriate label for training supervised models.
- Within the dataset, multiple instances can be observed where the booster failed to achieve a successful landing. Occasionally, a landing was tried but was unsuccessful due to an accident. For instance, "True Ocean" indicates that the mission was successfully landed in a particular area of the ocean, while "False Ocean" indicates that the mission was unsuccessfully landed in a specific region of the ocean.
- True RTLS refers to the successful landing of a mission outcome on a ground pad.A False Return to Launch Site (RTLS) refers to a mission outcome where the landing on a ground pad was unsuccessful.
- True ASDS refers to the successful landing of a mission outcome on a drone ship.A "False ASDS" refers to a mission outcome where the landing on a drone ship was unsuccessful.
- In this laboratory experiment, our primary objective is to convert the resulting outcomes into Training Labels. Specifically, a value of '1' will indicate a successful landing of the booster, while a value of '0' will indicate an unsuccessful landing.



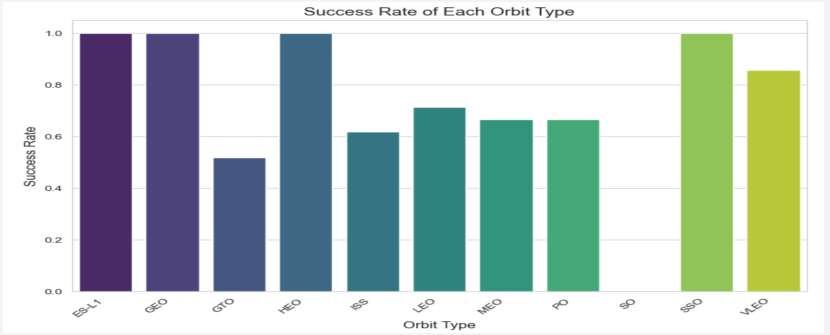
EDA with Data Visualization

GITHUB

Bar Graph being drawn:

- Success Rate VS. Orbit Type

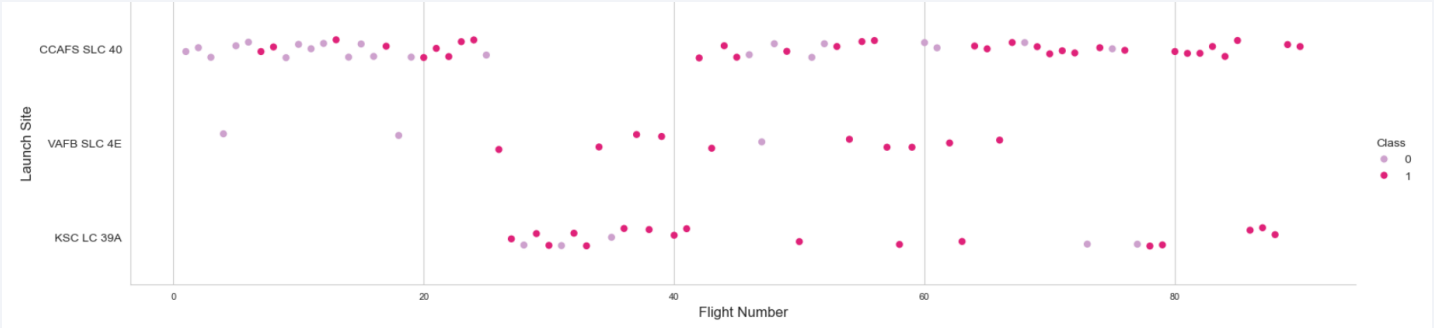
- A bar diagram makes it easy to compare data across groupings. On one axis, the graph shows categories and on the other, discrete values. Showing the two axes' relationship is the purpose. Bar charts can display large data changes over time.

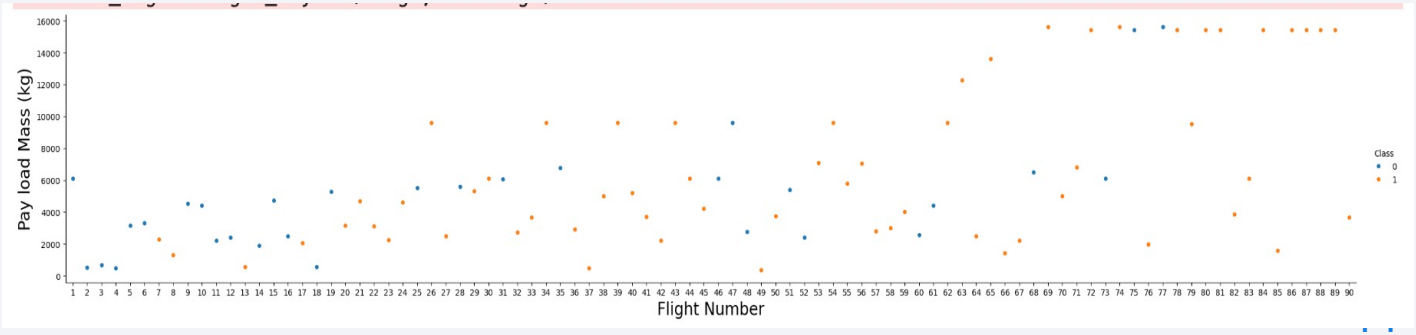


Some Scatter Graphs being drawn:

- Flight Number VS. Launch Site
- Flight Number VS. Payload Mass
- Orbit VS. Success Rate

Scatter plots reveal the extent to which one variable influences another. The connection between two metrics is known as a correlation. Data sets are typically quite large in scatter plots.





EDA with SQL

GITHUB

We executed SQL queries to extract details from the dataset. The following tasks are asked concerning the data to obtain the results from the dataset.

- Display the 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;
- Display average payload mass carried by booster version F9 v1.1;
- List the date when the first succesful landing outcome in ground pad was acheived;
- List the names of the boosters that have had success in drone ships and have a payload mass greater than 4000 but less than 6000;
- List the total number of successful and failed mission outcomes;
- List the names of the booster_versions which have carried the maximum payload mass;
- List the records that will display the month names, failure landing_outcomes in drone ships, ,booster versions, launch_site for the months in 2015;
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

Build an Interactive Map with Folium

GITHUB

- The goal is to create an interactive map based on the launch data. We marked each launch site on the map by placing a marker in the shape of a circle at its latitude and longitude coordinates and labeling it with the location's name.

- The launch_sites(failure, success) dataframe was then classified as **class 0** and **class 1**, with red and green markers on the map, using **MarkerCluster().**

- Then, we utilized Haversine's formula to figure out how far away the launch sites were from a variety of landmarks, such as:
  - How close are the launch sites to railways, highways, and coastlines?
  - How close are the launch sites to nearby cities?

Build a Dashboard with Plotly Dash

GITHUB

- Using Plotly dash, we developed an interactive dashboard that allows the user to examine the data in any desired manner
- A pie chart is a visual depiction of data that illustrates the proportionate distribution of several categories. The pie chart in our dashboard depicts the aggregate number of launches from a certain site or all sites, and visually represents the proportional distribution of different categories of data.
- A scatter plot illustrates the relationship between two independent variables and is the most effective method for visualising a nonlinear pattern.Scatter graph illustrating the link between result and payload mass (in kilogrammes) for several versions of boosters

Predictive Analysis (Classification)

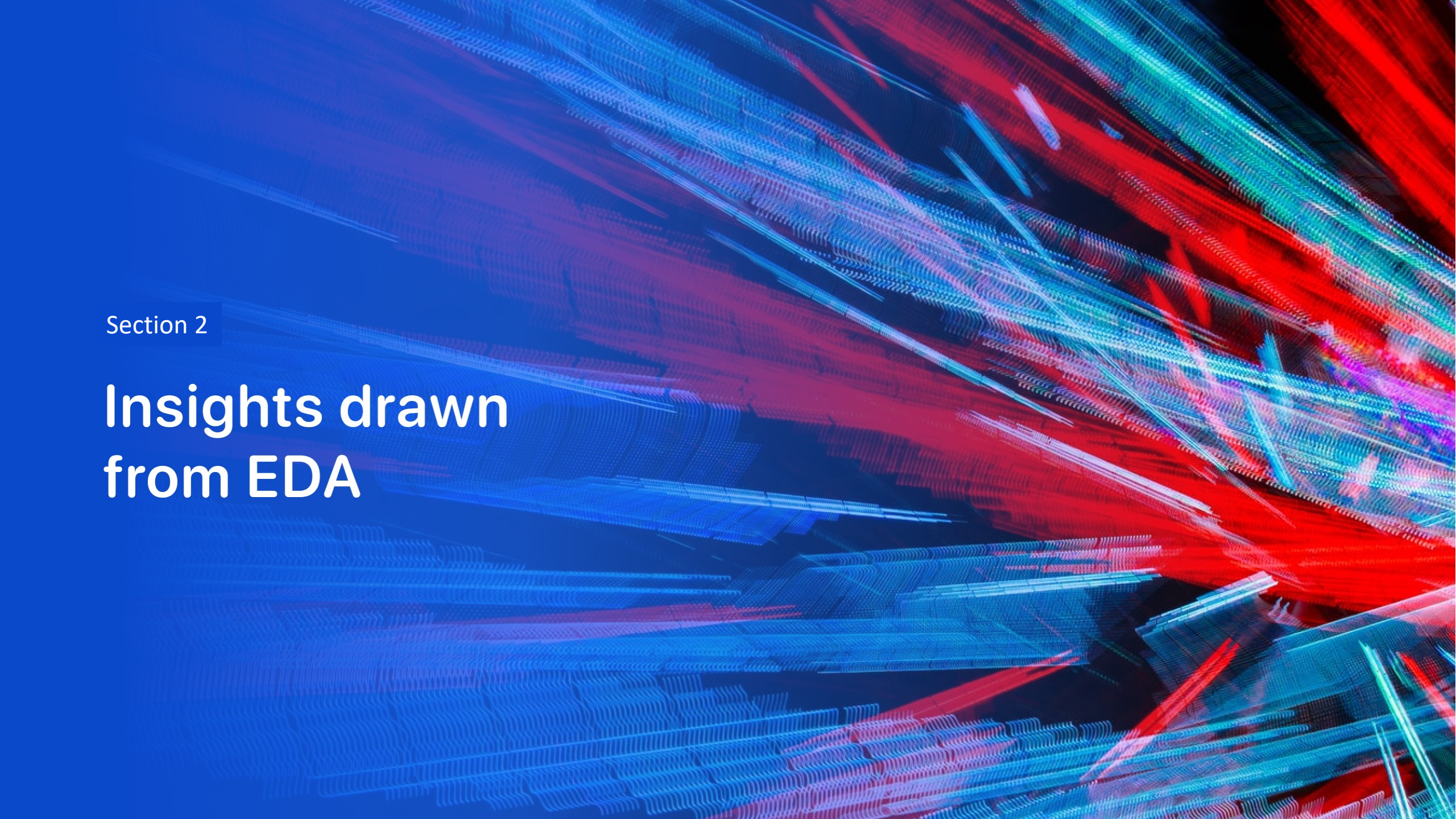
GITHUB

- Four classification methods — logistic regression, support vector machine, decision tree, and k nearest neighbors — were compared.

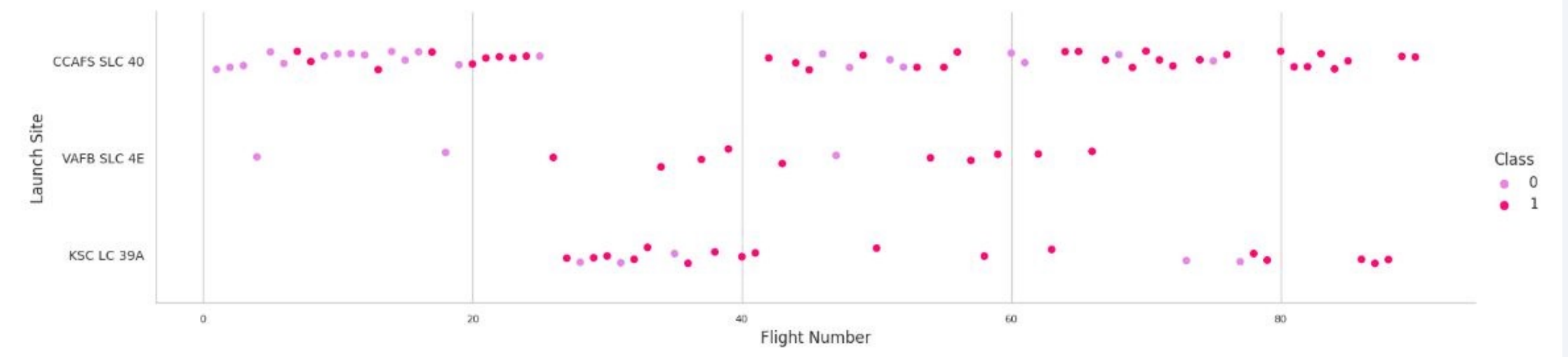
- BUILD THE MODEL -> EVALUATE THE MODEL -> IMPROVE THE MODEL -> FIND THE BEST MODEL.

Results

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

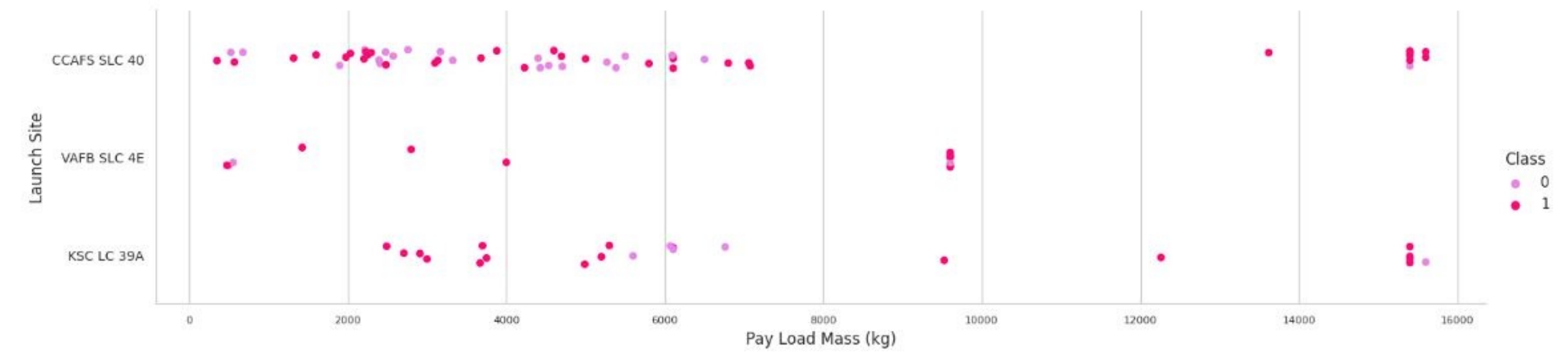

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



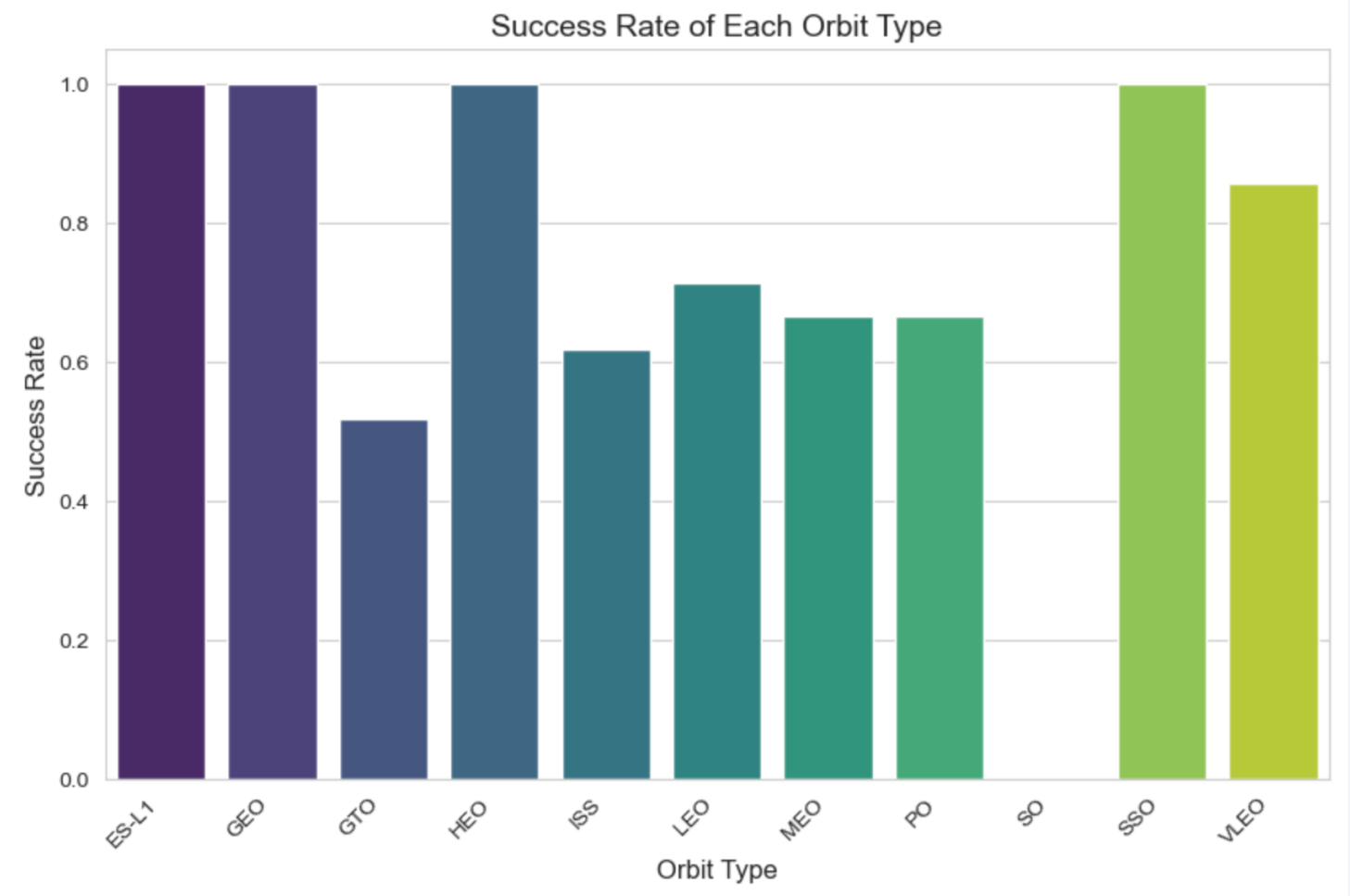
- CCAF5 SLC 40 has consistently been the most successful launch site recently, making it the best choice for future launches.
- The second launch site that has achieved success is Vandenberg Air Force Base Space Launch Complex 4E, which is then followed by Kennedy Space Centre Launch Complex 39A.
- Furthermore, it is possible to observe an increase in the rate of success as time progresses.

Payload vs. Launch Site



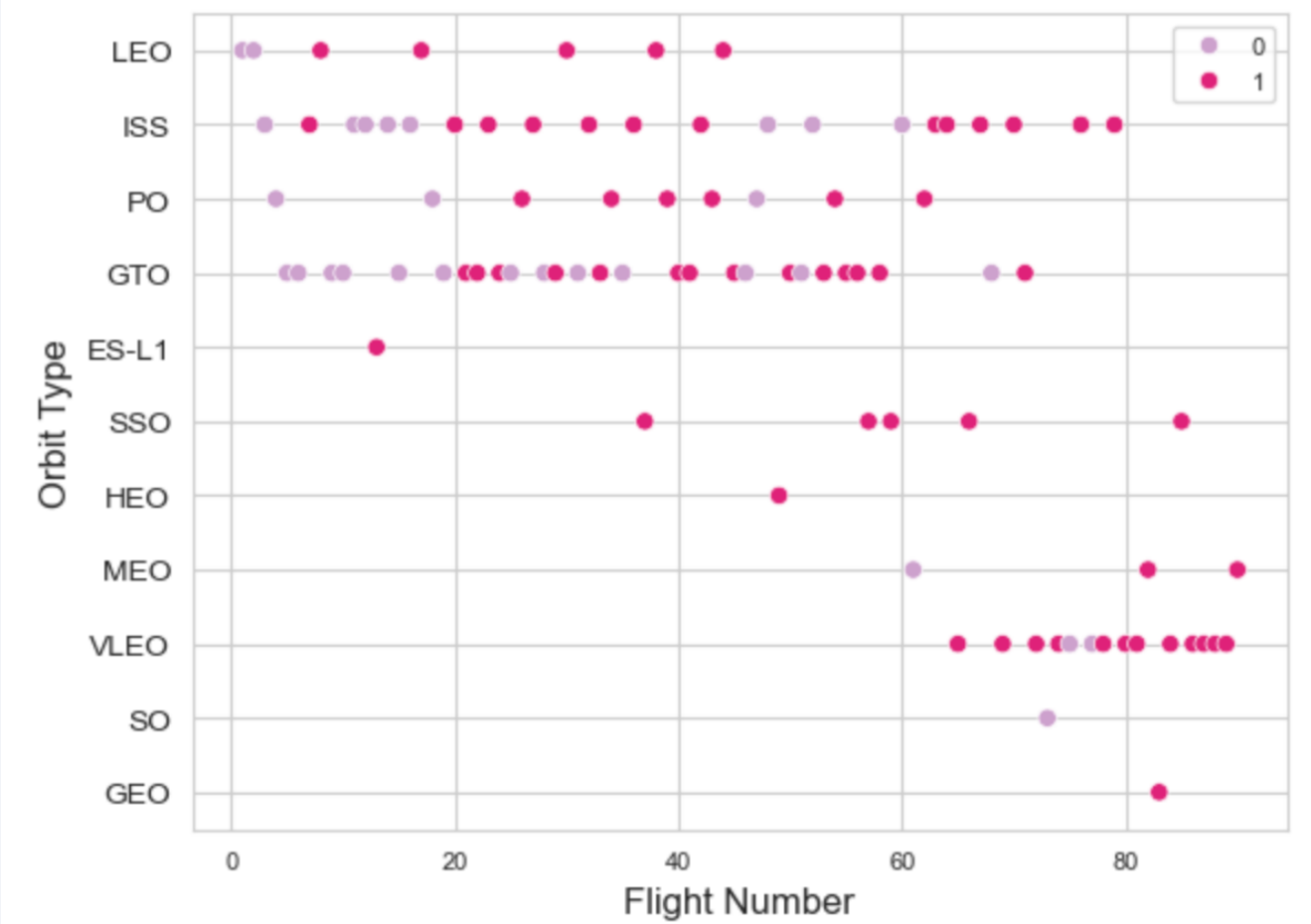
- The success rate for payloads weighing more than 9,000kg is exceptional. Payloads weighing over 12,000kg can only be launched from the CCAFS SLC 40 and KSC LC 39A launch facilities.
- This scatter figure illustrates that after the payload mass above 7,000 kg, the probability of success will increase significantly. However, there is no definitive proof that the success rate of the launch site is influenced by the mass of the payload.

Success Rate vs. Orbit Type



- SSO, HEO, GEO, and ES-L1 orbits all had 100% success rates, while SO orbits had 100% failure rates.
- The bar chart demonstrated how orbits could influence landing results.

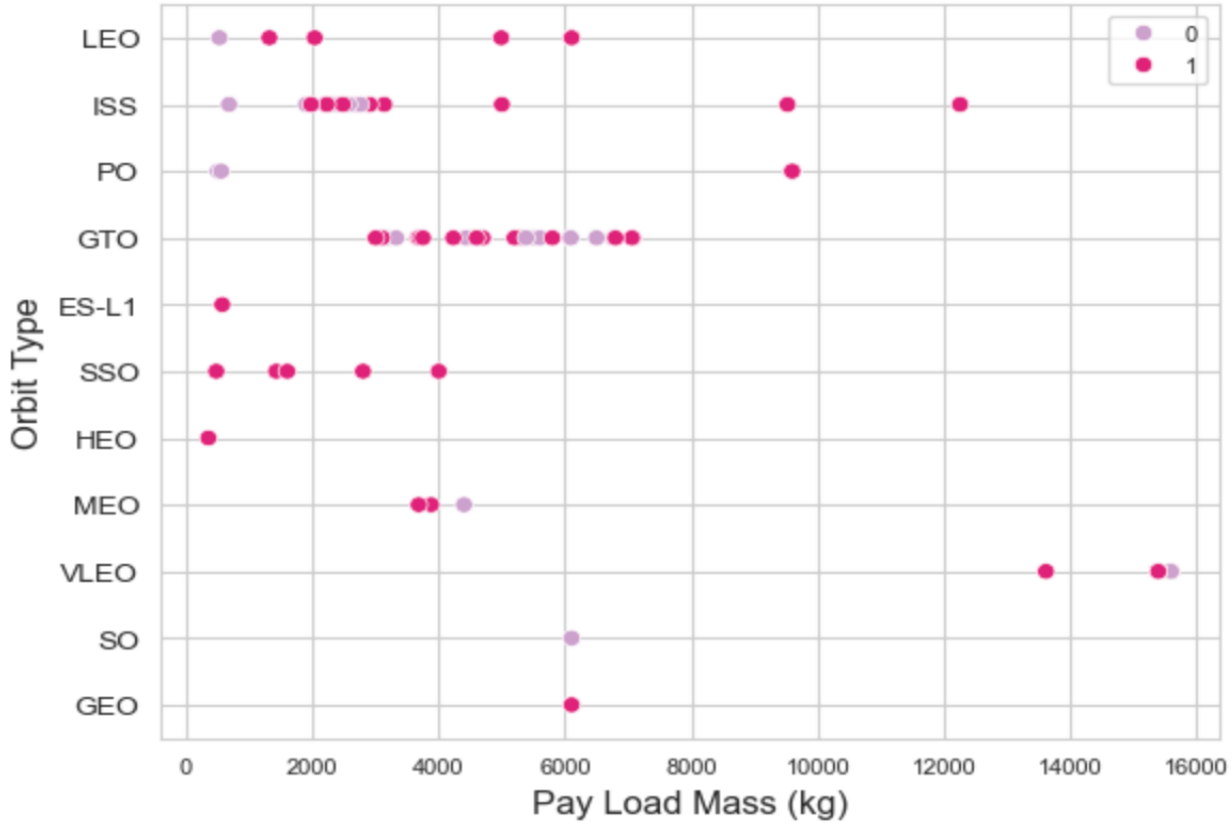
Flight Number vs. Orbit Type



- The scatter plot indicates that, apart from the GTO orbit, there is a positive correlation between the trip number and the success rate, particularly in the LEO orbit.
- Orbits that occur only once should also be removed from the previous statement, as it necessitates further information.
- Evidently, the success rates for all orbits have progressively improved over time.
- The VLEO orbit, in particular, now presents a novel economic opportunity because to its recent increase in frequency.

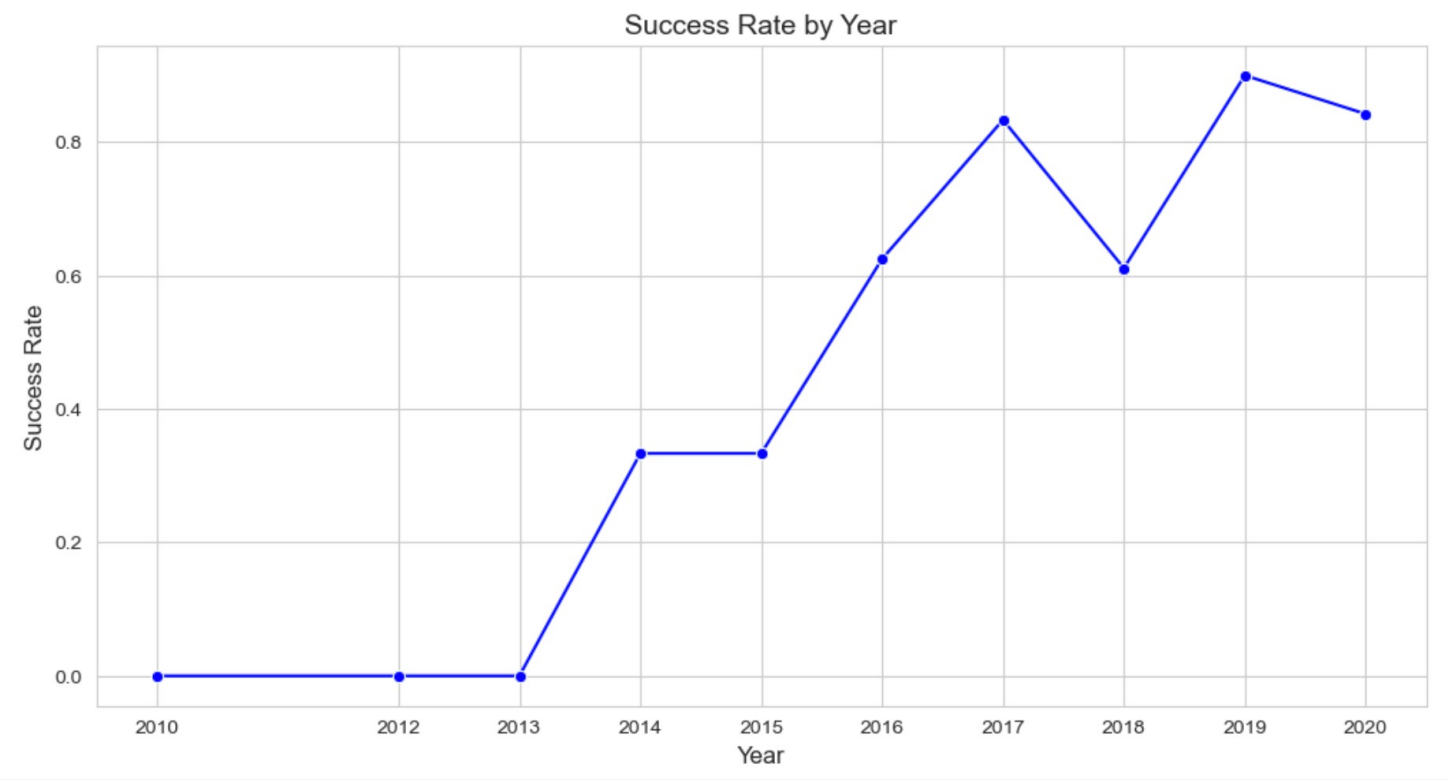
Payload vs. Orbit Type

```
plt.legend(loc="best", frameon=True, fontsize=8)
plt.show()
```



- Larger payloads benefit all orbits, including LEO, ISS, and P0. The impact is negative for MEO and VLEO orbits.
- There appears to be no relationship between the GTO orbit's depicted characteristics.
- Furthermore, extra data is required for SO, GEO, and HEO orbits to identify trends or patterns.
- It's important to note that bigger payloads may negatively effect GTO orbits but positively impact ISS and LEO orbits.

Launch Success Yearly Trend



- The line chart from 2013 to 2020 clearly depicts a consistent upward trend.
- If this pattern persists throughout the upcoming year and beyond, the rate of success will ultimately increase to 100%
- The success rate experienced an upward trend from 2013 to 2020.
- The initial three years seem to have been a period characterised by technological adaptation and progress.

All Launch Site Names

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

```
%%sql
SELECT DISTINCT Launch_Site
FROM SPACEXTBL;
```

[23] ✓ 0.0s

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

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

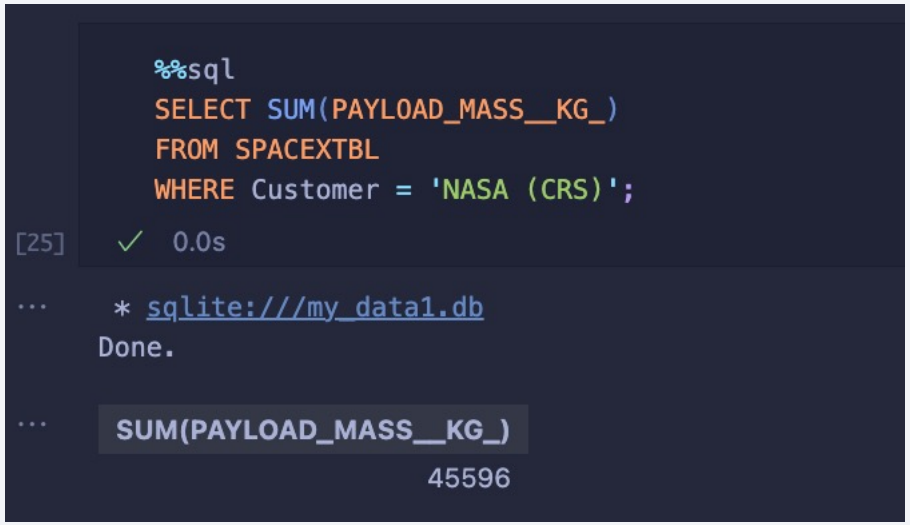
- To retrieve the unique names of the launch sites, I conducted a SQL query on the SPACEXTBL dataset using the keyword DISTINCT.
- According to the inquiry results, there are four launch places. They are obtained by filtering the dataset for distinct instances of "launch_ values."

Launch Site Names Begin with 'CCA'

Date	Time (UTC)	Booster_Version	Launch_Site
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40

- A query was executed to collect the results for the top 5 launch sites that begin with the characters 'CCA’.
- The "LIMIT 5" clause says that just the initial five records from SPACEXTBL will be retrieved.
- The "LIKE" phrase acts as a wildcard by matching any text, therefore the string "CCA%" indicates that the name of the "LAUNCH_SITE" must start with CCA.

Total Payload Mass



- Using the provided query, we successfully determined that NASA boosters have transported a cumulative payload of 45596 kilogrammes.
- The SUM function is utilised to consolidate the numbers within the column labelled PAYLOAD MASS KG.
- The WHERE clause allows us to narrow down our data collection and retrieve precise results for the Customer column that have a value of NASA (CRS).

Average Payload Mass by F9 v1.1

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Booster_Version = 'F9 v1.1';
```

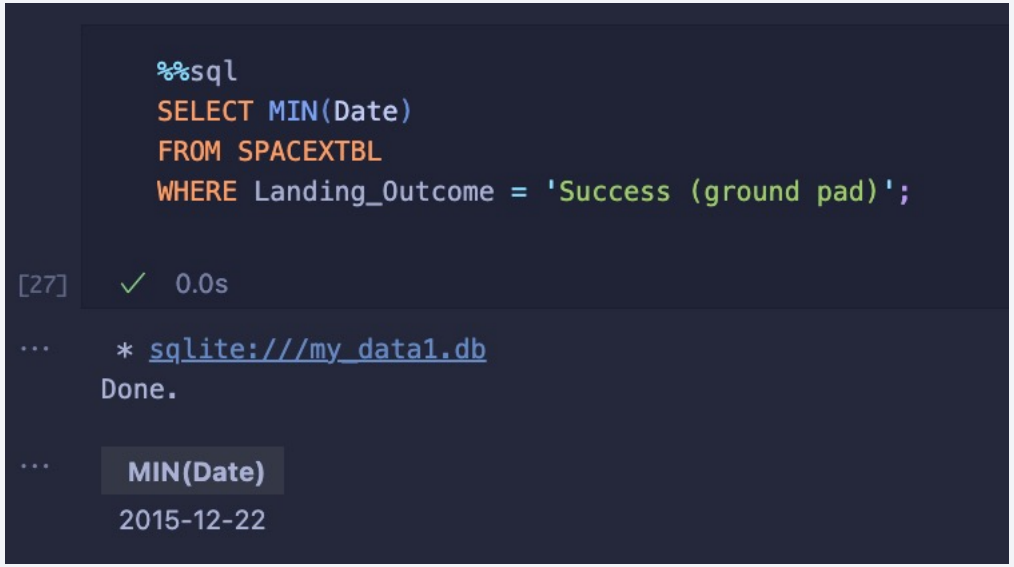
[26] ✓ 0.0s

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

AVG(PAYLOAD_MASS__KG_)
2928.4

- By utilising this query, we have effectively ascertained that NASA launchers have transported a total payload of 45596 kilogrammes.
- The SUM function is used to aggregate the numbers in the column labelled PAYLOAD MASS KG.
- The WHERE clause enables us to refine our data collection and return accurate results for the Customer column that specifically have a value of NASA (CRS).

First Successful Ground Landing Date



- The successful outcome of the ground pad landing was determined by performing the following query to retrieve the date. In order to confirm this, we utilised the MIN() function.
- On December 22, 2015, we witnessed the inaugural successful touchdown on a terrestrial platform. The minimal date in the Date column was computed using the MIN() function.
- In order to refine the data set, we utilise the WHERE clause to search for the specific value "Success (ground pad)" within the Landing_Outcome_Success field.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql
SELECT DISTINCT Booster_Version
FROM SPACEXTBL
WHERE Landing_Outcome = 'Success (drone ship)'
  AND PAYLOAD_MASS__KG_ > 4000
  AND PAYLOAD_MASS__KG_ < 6000;
```

[28] ✓ 0.0s

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

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

- By employing the following query, we extracted Boosters that accomplished a successful landing on a drone ship and possessed a payload mass above 4000 but falling short of 6000.
- We chose the column labelled BOOSTER_VERSION from the table. In order to refine the data gathering, we employ the WHERE clause to explicitly state that the Landing_Outcome must be "Success" (drone ship).
- Additional filtering conditions are specified in the AND clause. Both the Payload MASS KG 4000 and 6000 criteria are mandatory.

Total Number of Successful and Failure Mission Outcomes

```
%%sql
SELECT Mission_Outcome, COUNT(*)
FROM SPACEXTBL
GROUP BY Mission_Outcome;
```

[29] ✓ 0.0s

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

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

- I used the WHERE clause in nested queries to narrow down the data, the LIKE '%' statement to filter the "Success" and "Failure" statements in the MISSION OUTCOME column, then subqueries to aggregate the results.
- The query was used to get the summary below, which involved counting records for each subquery.

Boosters Carried Maximum Payload

```
%%sql
SELECT Booster_Version
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL
);
```

[30] ✓ 0.0s

```
 * 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

- Using a WHERE clause and the MAX() method in a subquery, we were able to determine which booster had the largest mass.
- Before extracting the maximum payloads from the table with the WHERE clause, we used the DISTINCT clause in the main query to retrieve only the unique booster versions from the BOOSTER VERSION column.

2015 Launch Records

```
%%sql
SELECT
    CASE SUBSTR(Date, 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
    END AS Month,
    Landing_Outcome,
    Booster_Version,
    Launch_Site
FROM SPACEXTBL
WHERE Landing_Outcome = 'Failure (drone ship)'
  AND SUBSTR(Date, 1, 4) = '2015';
```

[31] ✓ 0.0s

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

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

- To extract only months, we picked the BOOSTER_VERSION and LAUNCH_SITE columns and filtered the DATE column with substr(Date, 4, 2).
- Then, we used the WHERE clause to extract "Failure (drone ship)" values from the LANDING OUTCOME column, and the AND statement in the WHERE clause to select the year 2015 using substr(Date,7,4)='2015'.

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

```
%%sql
SELECT Landing_Outcome, COUNT(*) AS Outcome_Count
FROM SPACEXTBL
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY Outcome_Count DESC;
```

[32] ✓ 0.0s

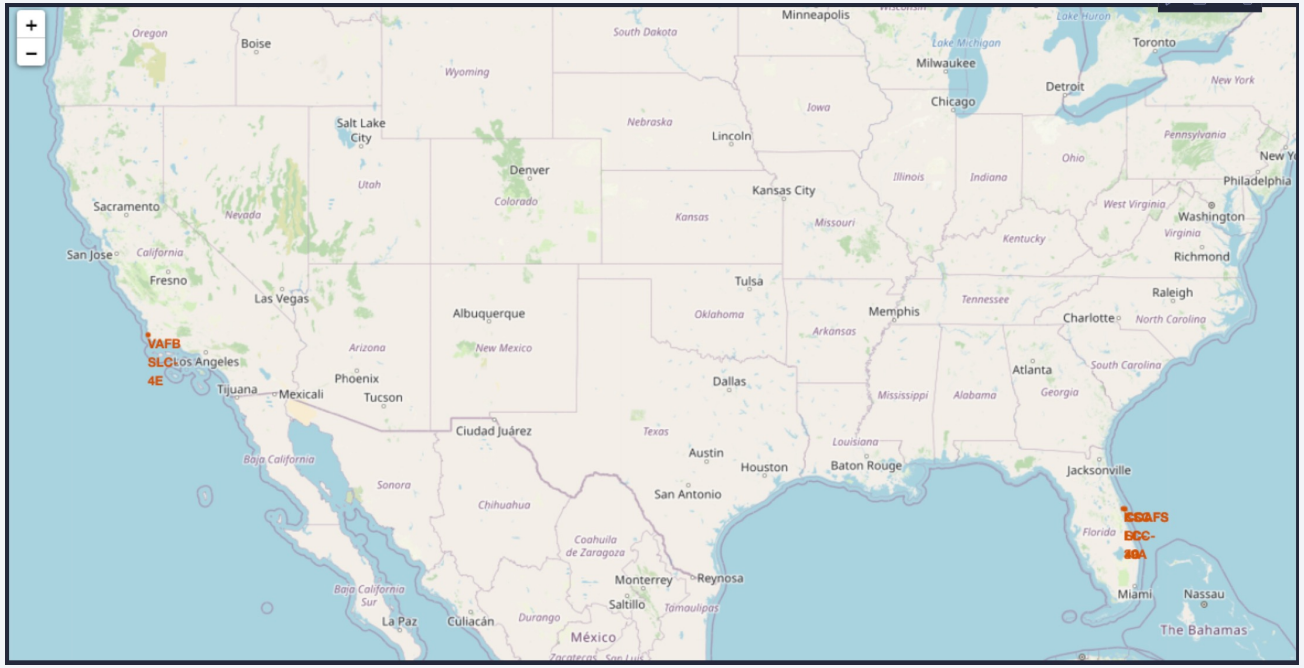
```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	Outcome_Count
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

- We extracted Landing_Outcomes and the COUNT of landing outcomes from the data, then used the WHERE clause to filter for landing outcomes between 2010-06-04 and 2010-03-20.
- We used the GROUP BY clause to group the landing outcomes and the ORDER BY clause to sort the grouped landing outcomes in descending order.

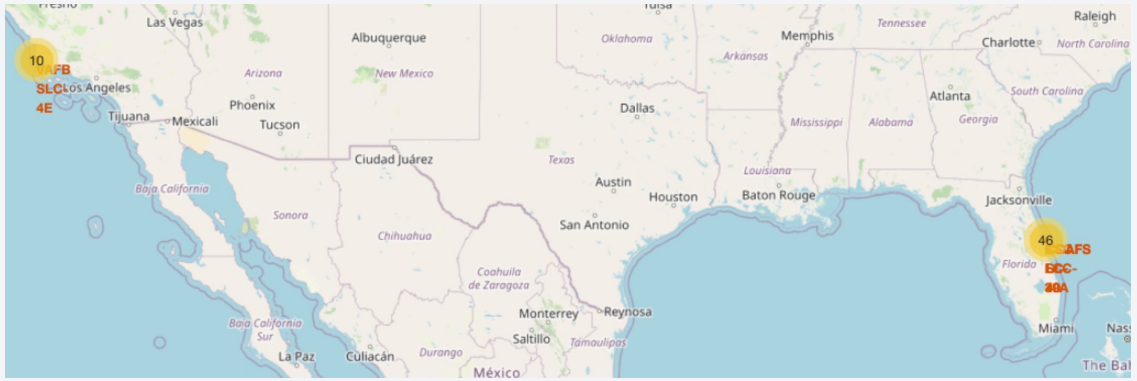

Section 3
Launch Sites
Proximities Analysis

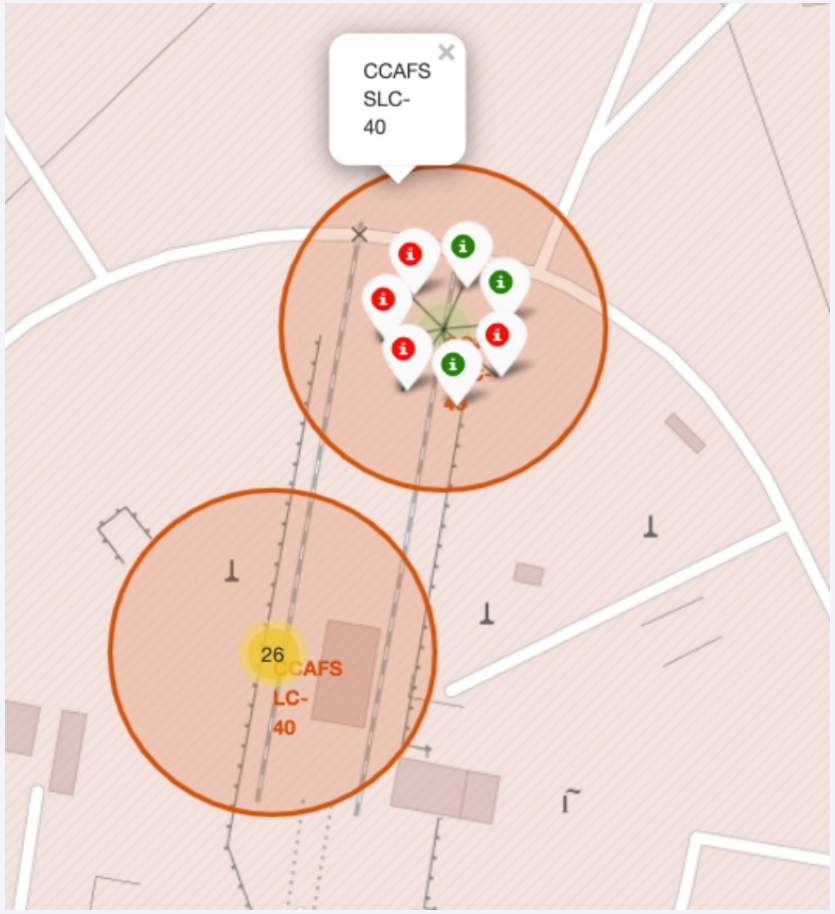
Marked Launched Sites



- As can be seen on the map, each and every one of SpaceX's launch operations are situated wholly within the borders of the United States of America.

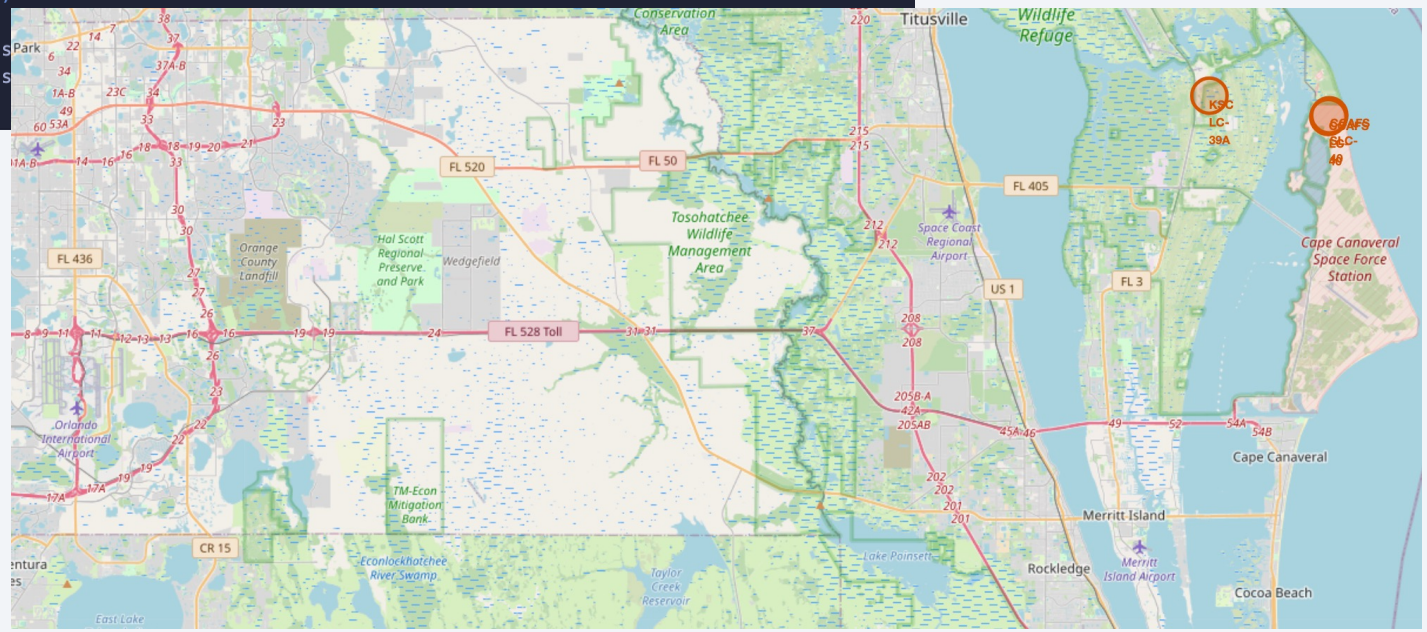
Colour-labeled launch outcomes



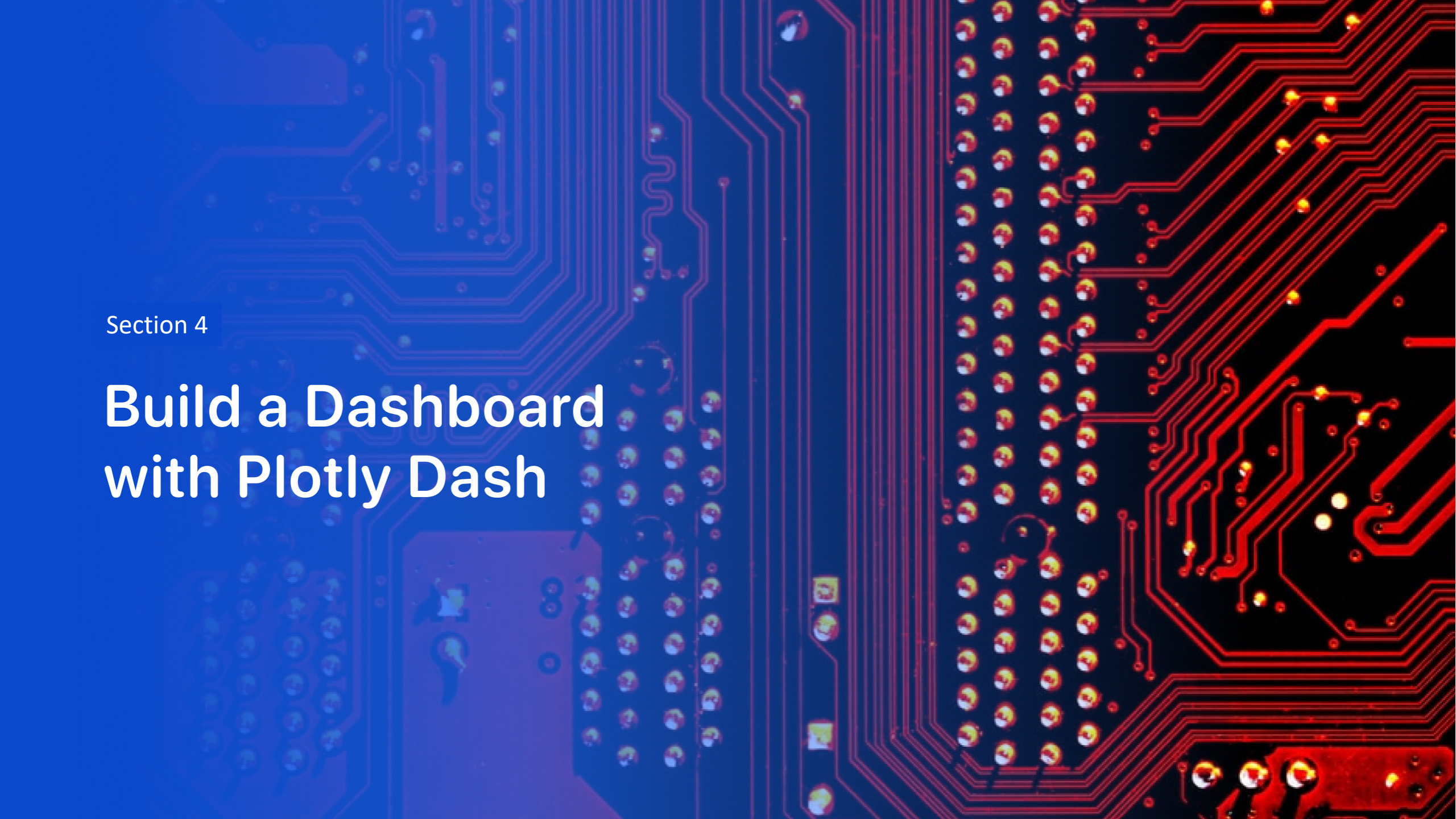


<Folium Map Screenshot 3>

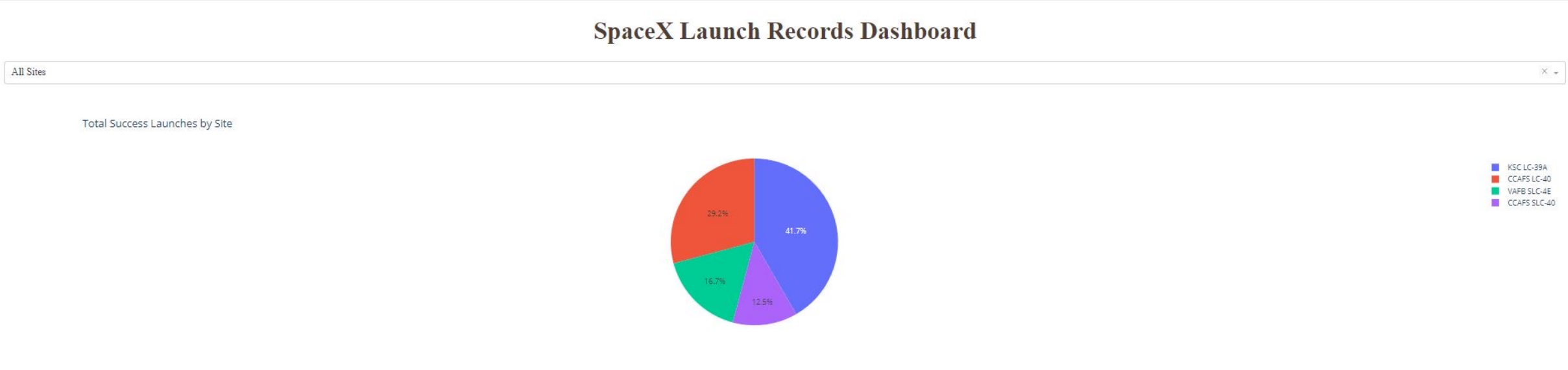
```
# Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on the map
formatter = "function(num) {return L.Util.formatNum(num, 5);};"
mouse_position = MousePosition(
    position='topright',
    separator=' Long: ',
    empty_string='NaN',
    lng_first=False,
    num_digits=20,
    prefix='Lat:',
    lat_formatter=formatter,
    lng_formatter=formatter,
)
```



- To discover trends in the locational properties of launch sites and landmarks, I used the Haversine formula to determine the distance between them. The picture depicts the distance between launch points CCAFS LC-40 and the city centre, as well as KSC LC-39A and the coast.


Section 4
Build a Dashboard with Plotly Dash

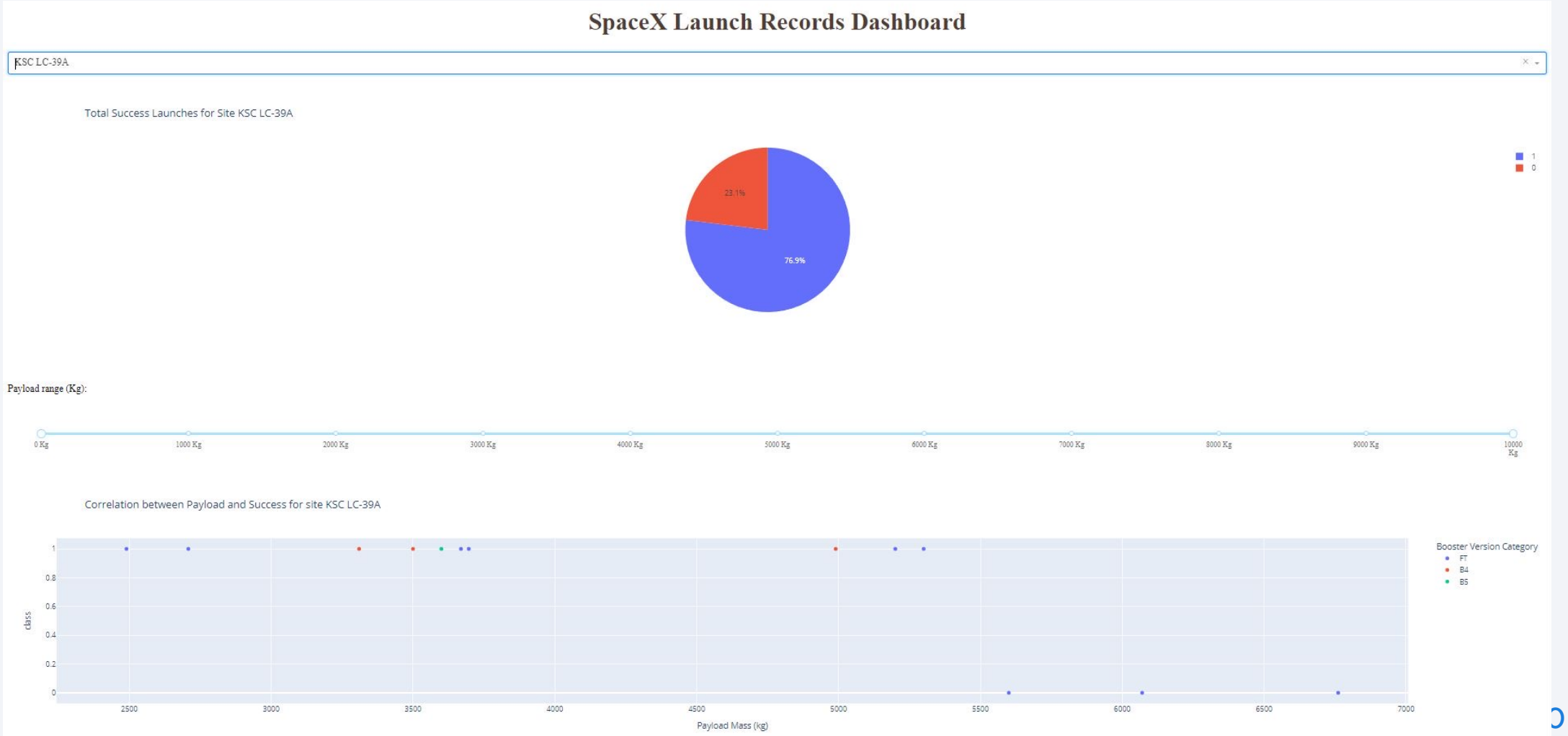
Success count for all sites



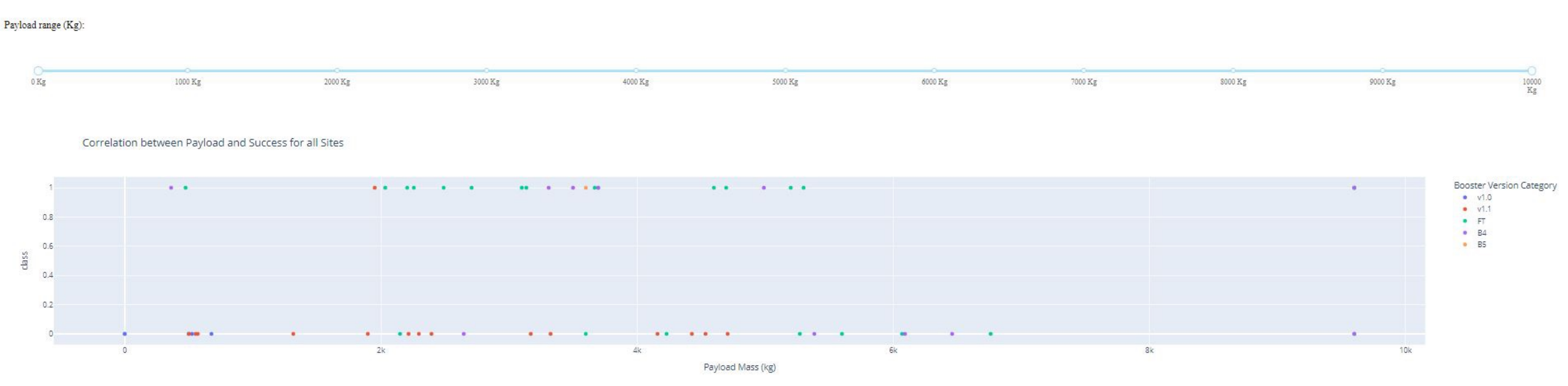
- KSC LC-39A had the greatest successful launch rate of any facility, at 41.7%. The second launch site, CCAFS LC-40, has a success percentage of 29.2%.

- With a success rate of 16.7%, VAFB SLC-4E looks to be the third-highest rate location. CCAFS SLC-40 is the fourth launch location, with a success rate of 12.5%.

Highest Launch Success Ratio



Scatter Plot:



- Most launches with payloads under 2,000 kg were unsuccessful.
- The majority of launches weighing 2,000-4,000 kg were successful.
- The most successful launches weighed between 2,000 and 4,000kg while the ones with payloads weighing 4,000-6,000 kg had a higher failure rate than successful ones.
- Except for one, all launches weighing more than 6,000 kg were unsuccessful.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
# Since their accuracies are all the same, we pick based on their best scores
algo_score = {'Logistic regresssion': [logreg_cv.best_score_], 'SVM': [svm_cv.best_score_], 'Decision tree': [tree_cv.best_score_], 'KNN': [knn_cv.best_score_]}
df = pd.DataFrame.from_dict(algo_score, orient='index', columns=['Best scores'])
df
```

[99] ✓ 0.0s

	Best scores
Logistic regresssion	0.846429
SVM	0.848214
Decision tree	0.889286
KNN	0.848214

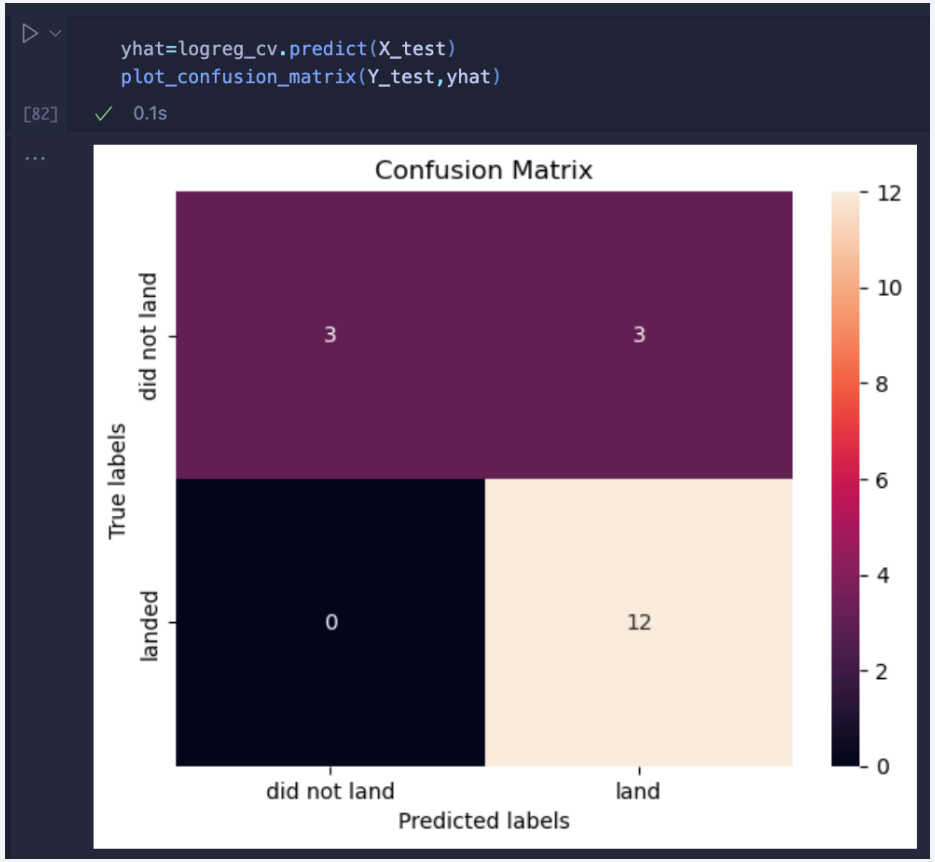
```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
best_algorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',best_algorithm,'with a score of',algorithms[best_algorithm])
if best_algorithm == 'Tree':
    print('Best Params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

[101] ✓ 0.0s

```
Best Algorithm is Tree with a score of 0.8892857142857142
Best Params is : {'criterion': 'gini', 'max_depth': 18, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter': 'best'}
```

- The tabular output demonstrates that the accuracy of all classification algorithms is equal for the test dataset.
- However, the classification tree approach exhibits the highest accuracy for the training dataset.
- In order to verify this result, the code showcases that the classification tree approach is the most precise model.

Confusion Matrix



- As can be seen from the tabular output, the accuracy of all classification algorithms for the test dataset is same. On the other hand, the classification tree approach has the highest accuracy for the training dataset.

- The code illustrates that the classification tree approach is the most accurate model, which is tested to determine whether or not this outcome is true.

Conclusions

- Based on our analysis, we have reached the following conclusion on this dataset:
    - Why Lighter payloads (4,000 kilogrammes or fewer) performed far better than heavier payloads.
    - If you launch more than 6,000 kilogrammes, there is a good chance that you will fail.
    - The success rate of SpaceX launches has been steadily growing since 2013, and this trend is directly related to the number of years until 2020, when SpaceX expects to have perfected its launches.
    - The LC-39A launch site at KSC has the best success rate (76.9%) of any launch location.
    - SSO orbits have the highest success rate, either 100% or many occurrences. The most common orbital routes are GEO, HEO, SSO, and ES L1.
    - The Tree Classifier Algorithm has been determined to be the most efficient approach of machine learning.

Thank you!