IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Marycruz Vera Bedolla
7/16/2024

Outline

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

Executive Summary

- Summary of methodologies
  - Collecting data using API's & web scraping
  - Data wrangling
  - EDA using SQL
  - Data visualization
  - Interactive visual analytics using Folium
  - Machine Learning
- Summary of all results
  - EDA visualizations
  - Folium interactive map visualizations
  - Predictive analytics outcomes

Introduction

- Project background and context

  On its website, Space X promotes Falcon 9 rocket flights for a cost of 62 million dollars; in comparison, other suppliers charge upwards of 165 million dollars per launch; a large portion of the cost reduction can be attributed to Space X's ability to reuse the first stage. Thus, we can calculate the launch cost if we can ascertain if the first stage will land. If another business wishes to compete with Space X for a rocket launch, they can use this information. The project's objective is to develop a machine learning pipeline to determine whether the initial stage will land successfully.

- Problems you want to find answers
  - Factors Influencing Successful Landing
  - Feature Interaction
  - Optimal Operating Conditions

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Utilized SpaceX API and web scarping
- Perform data wrangling
  - Categorical features were encoded using a one-hot method.
- 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

- Describe how data sets were collected.
  - Utilizing a get call to the SpaceX API to gather data.
  - We used the.json() function call to decode the response content as JSON and the.json_normalize() method to convert it into a pandas dataframe.
  - We cleaned the data, looked for any missing or null values, and corrected them accordingly.
  - We used BeautifulSoup to perform web scraping to find launch records for Falcon 9.
  - The aim was to retrieve the launch records in the form of an HTML table, parse the information, and then transform it into a pandas dataframe for upcoming interpretation.

Data Collection – SpaceX API

- The SpaceX API's get request was utilized to gather data, clean the requested data, and do some simple data wrangling and formatting.
- SpaceX API calls notebook https://github.com/mvbedo/IBM-Space-Race-Case-Study/blob/2321b18b08f37e349cdf836edcb2c3f741cbae26/spacex-data-collection-api.ipynb

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

In [11]:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.c
```

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

In [12]:

```
response.status_code
```

Out[12]:

```
200
```

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

In [16]:

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

Using the dataframe `data` print the first 5 rows

In [17]:

```
# Get the head of the dataframe
data.head()
```

Data Collection - Scraping

- Using BeautifulSoup, we used web scraping to gather Falcon 9 launch records.
  The table was parsed, and a pandas dataframe was created.

- Completed web scraping notebook: https://github.com/mvbedo/IBM-Space-Race-Case-Study/blob/2321b18b08f37e349cdf836edcb2c3f741cbae26/webscraping.ipynb

TASK 1: Request the Falcon9 Launch Wiki page from its URL

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

In [20]:

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

# assign the response to a object
if response.status_code == 200:
    print("Request was successful.")
else:
    print("Request failed with status code:", response.status_code)
```

```
Request was successful.
```

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

In [21]:

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

Print the page title to verify if the `BeautifulSoup` object was created properly

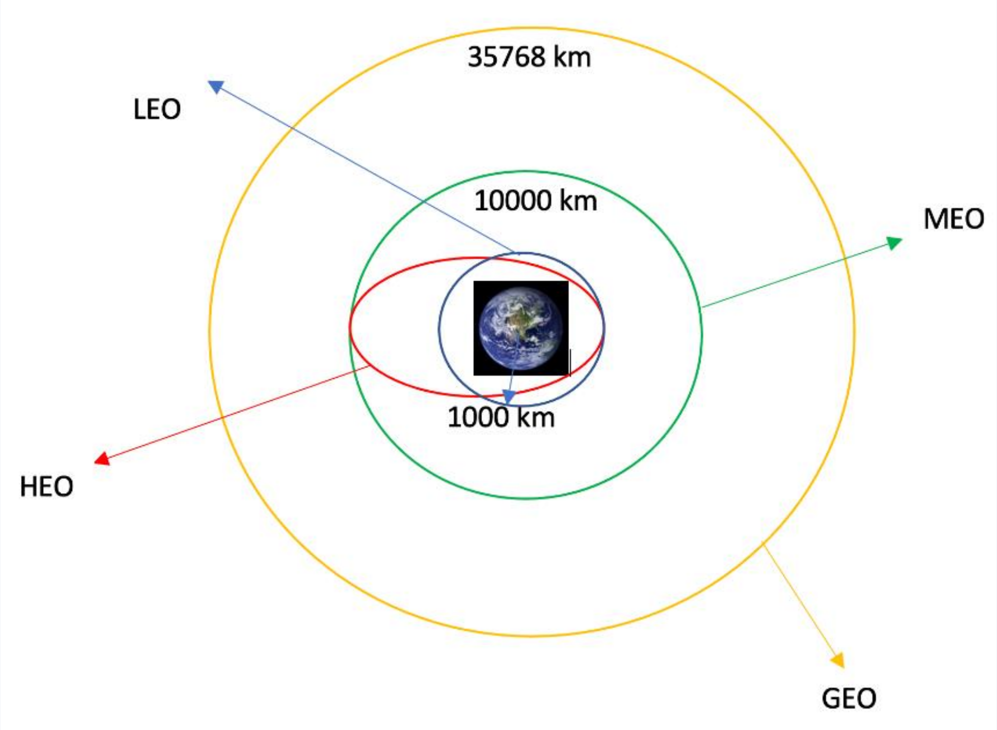
In [22]:

```
# Use soup.title attribute
print("Page title:", soup.title.string)
```

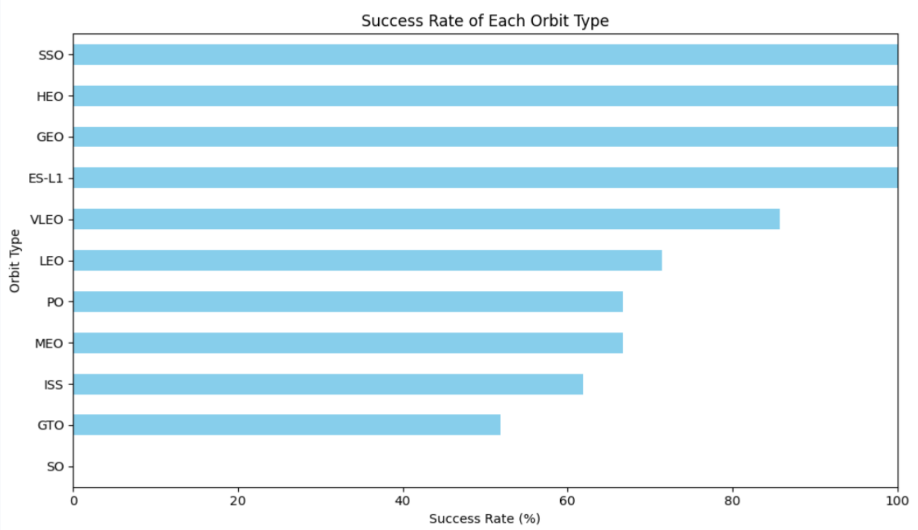
```
Page title: List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

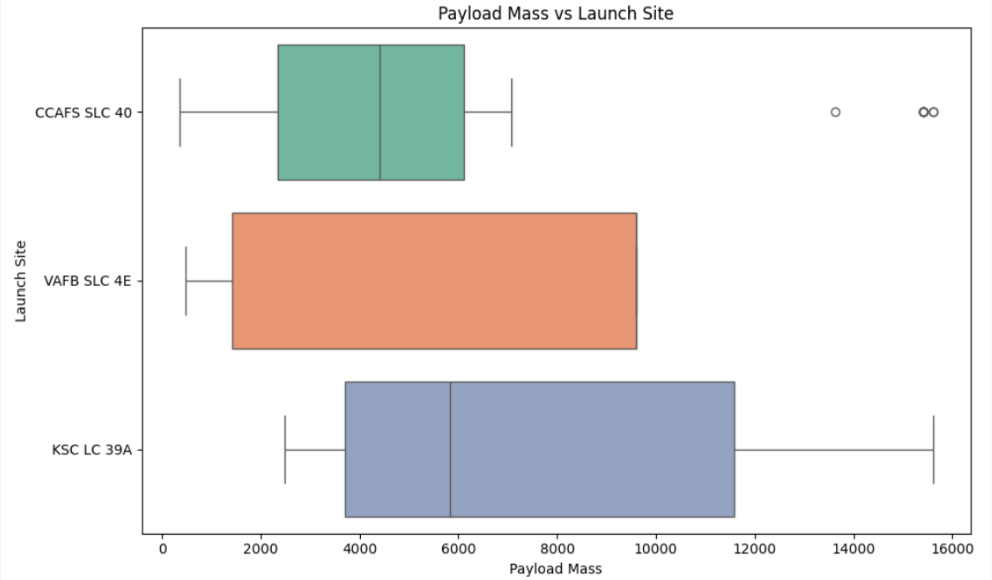
Data Wrangling

- We identified the training labels by doing an exploratory data analysis.
- We determined the quantity of launches at every location as well as the quantity and frequency of each orbit.
- From the outcome column, we generated the landing outcome label and exported the data to CSV.
- Completed data wrangling related notebooks: https://github.com/mvbedo/IBM-Space-Race-Case-Study/blob/2321b18b08f37e349cdf836edcb2c3f741cbae26/spacex-Data%20wrangling.ipynb



EDA with Data Visualization





- The relationship between the flight number and the launch site, the payload and the launch site, the success rate of each orbit type, the flight number and the orbit type, and the annual trend of launch success were all visualized as we investigated the data.
- Completed EDA with data visualization notebook: https://github.com/mvbedo/IBM-Space-Race-Case-Study/blob/2321b18b08f37e349cdf836edcb2c3f741cbae26/eda_data_viz.ipynb

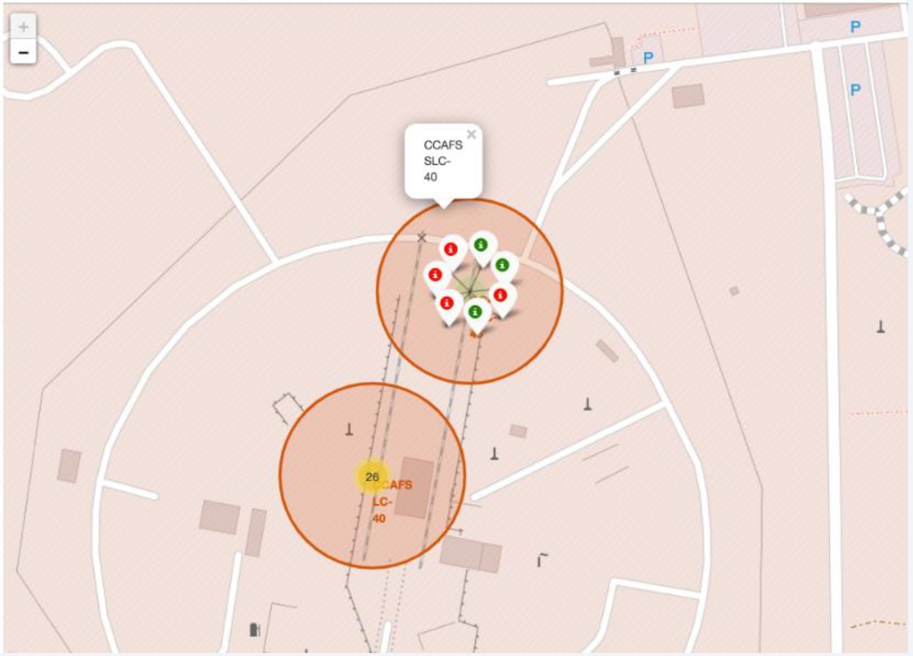
EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database.
- We used magic SQL to conduct EDA and query for:
  - The names of the space mission's distinctive launch sites.
  - The total mass of payload carried by NASA's CRS rockets
  - The F9 v1.1 booster's average payload mass
  - The total number of mission outcomes that were successful and unsuccessful
  - The drone ship's unsuccessful landing results, along with the names of the launch sites and booster versions.
- Completed EDA with SQL notebook: https://github.com/mvbedo/IBM-Space-Race-Case-Study/blob/2321b18b08f37e349cdf836edcb2c3f741cbae26/eda-sql-sqllite.ipynb

Landing_Outcome	Count_Landing_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

```
In [27]:
%sql SELECT "Landing_Outcome", COUNT(*) AS Count_Landing_Outcomes FROM SPACEXTB
```

Build an Interactive Map with Folium



- To indicate the success or failure of launches for each site on the folium map, we annotated all of the launch sites and added map elements like circles, lines, and markers.
- The feature launch outcomes—success or failure—were divided into classes 0 and 1.(0 = fail; 1 = success)
- The launch sites with a comparatively high success rate were determined by using the color-labeled marker clusters.
- We computed the separations between the proximities and launch sites. We provided answers to certain queries, such as:
  - Are launch locations close to roadways, trains, and coasts?
  - Are launch locations kept a specific distance from urban areas?
- Completed interactive map with Folium map: https://github.com/mvbedo/IBM-Space-Race-Case-Study/blob/2321b18b08f37e349cdf836edcb2c3f741cbae26/launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Using Plotly dash, we constructed an interactive dashboard.
- To visualize the total launches by a certain site, we created pie charts.
- We created a scatter plot that illustrated the association between the various booster versions' outcomes and payload masses (kg).
- Completed Plotly Dash lab: https://github.com/mvbedo/IBM-Space-Race-Case-Study/blob/2321b18b08f37e349cdf836edcb2c3f741cbae26/spacex_dash_app.py

```
# Run the app
if __name__ == '__main__':
    app.run_server(debug=True, port=8051)
```

Predictive Analysis (Classification)

- Using pandas and numpy, we loaded the data, processed it, and divided it into training and testing sets.
- Using GridSearchCV, we constructed several machine learning models and adjusted various hyperparameters.
- Our model's accuracy served as its metric, and it was enhanced through feature engineering and algorithm tuning.
- The top-performing categorization model has been identified.
- Completed predictive analysis lab: https://github.com/mvbedo/IBM-Space-Race-Case-Study/blob/2321b18b08f37e349cdf836edcb2c3f741cbae26/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

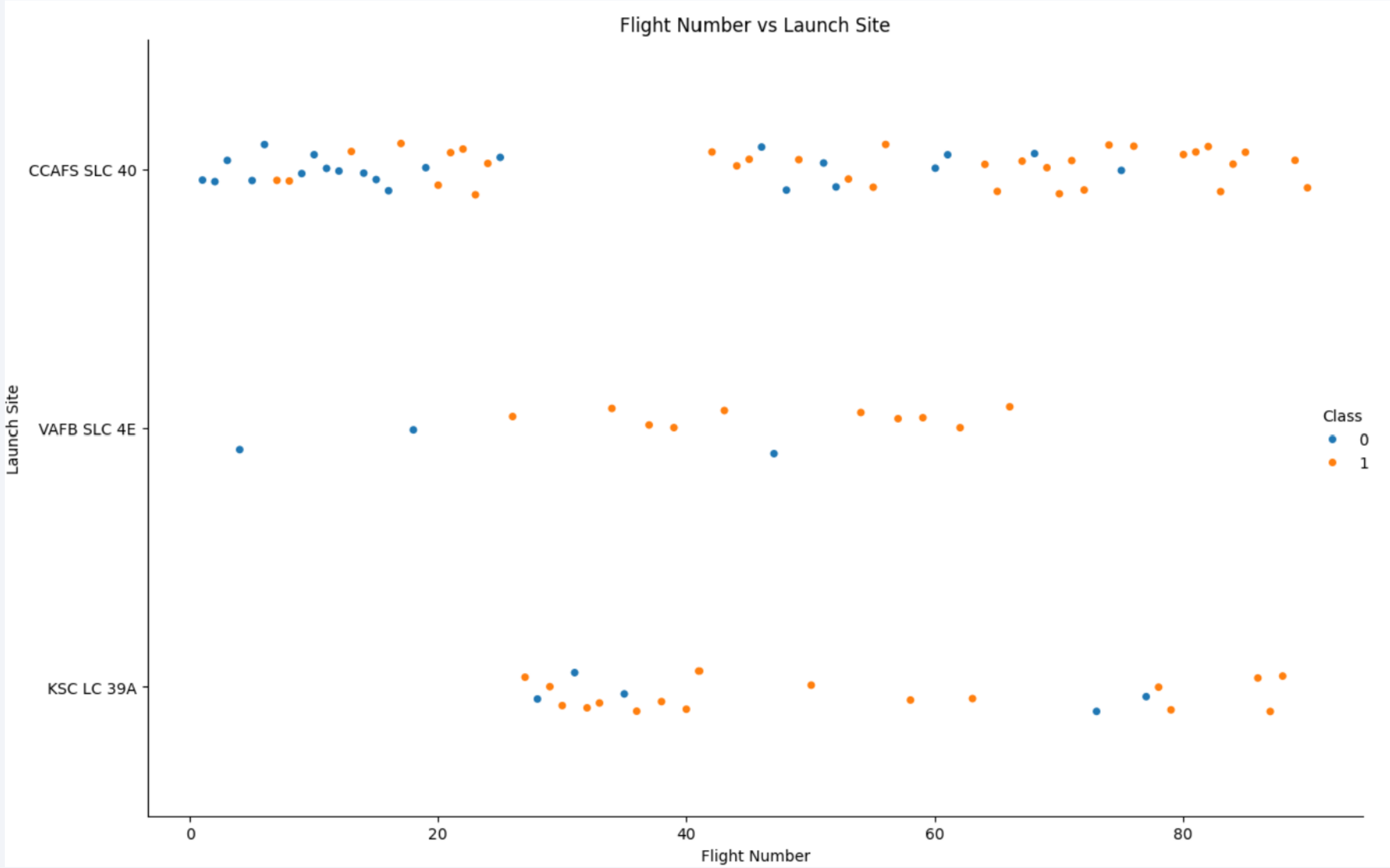
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

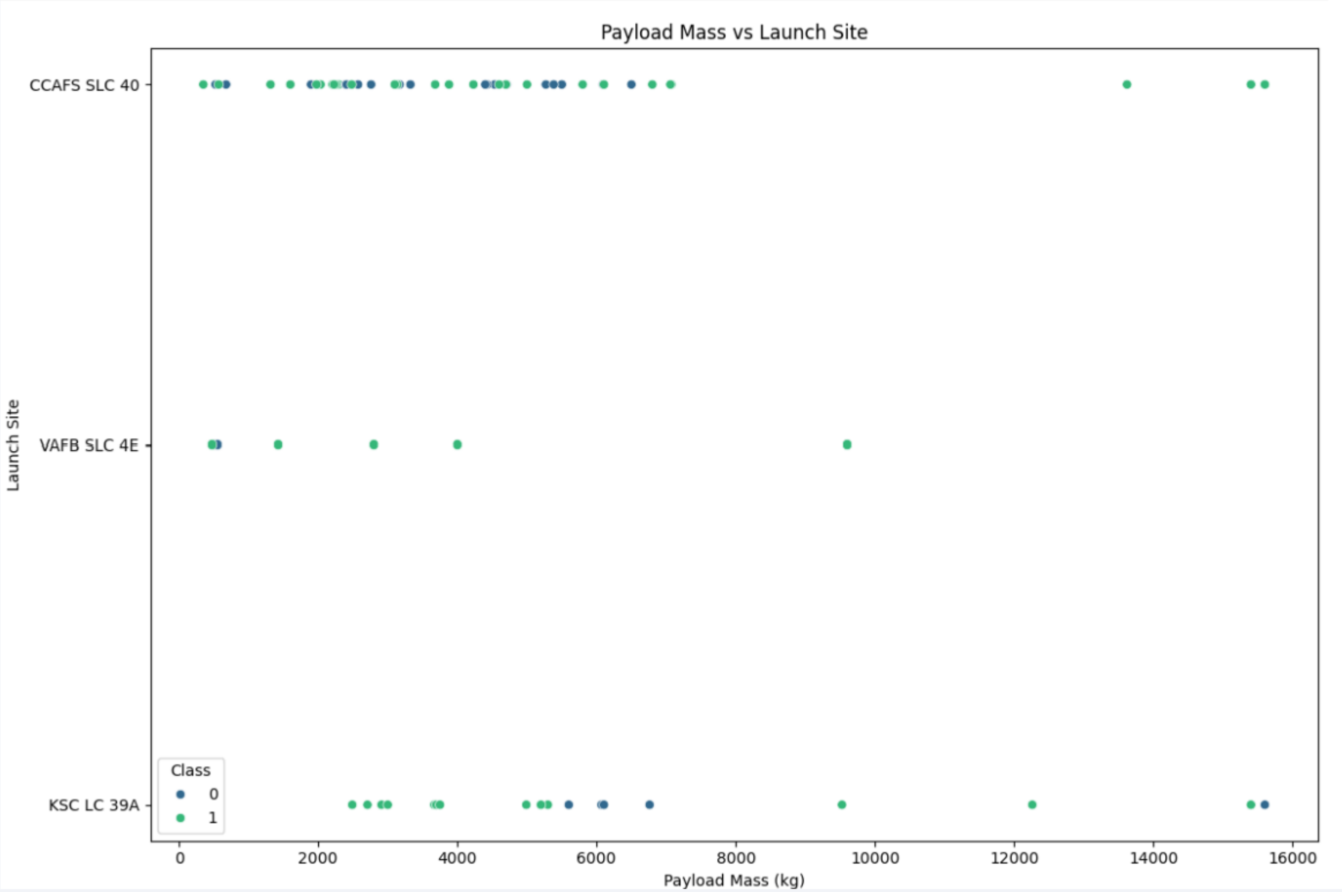
Flight Number vs. Launch Site

- We deduced from the plot that a launch site's success rate increases with the number of flights conducted there.



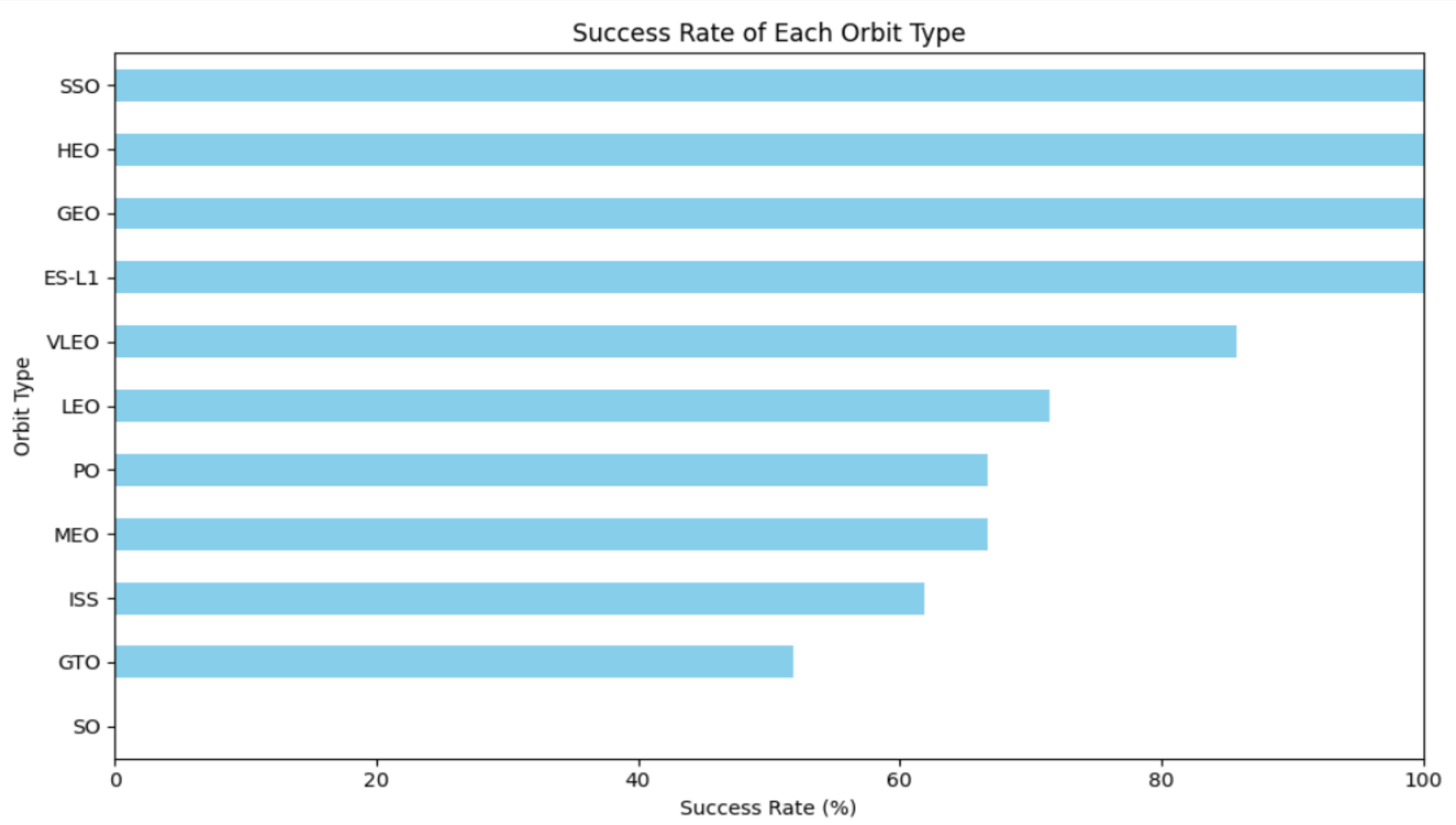
Payload vs. Launch Site

- Overall, the figure indicates that other criteria other than payload mass are probably more important in determining a launch's success, even though it does show some variation in launch success rates and payload masses across different sites.



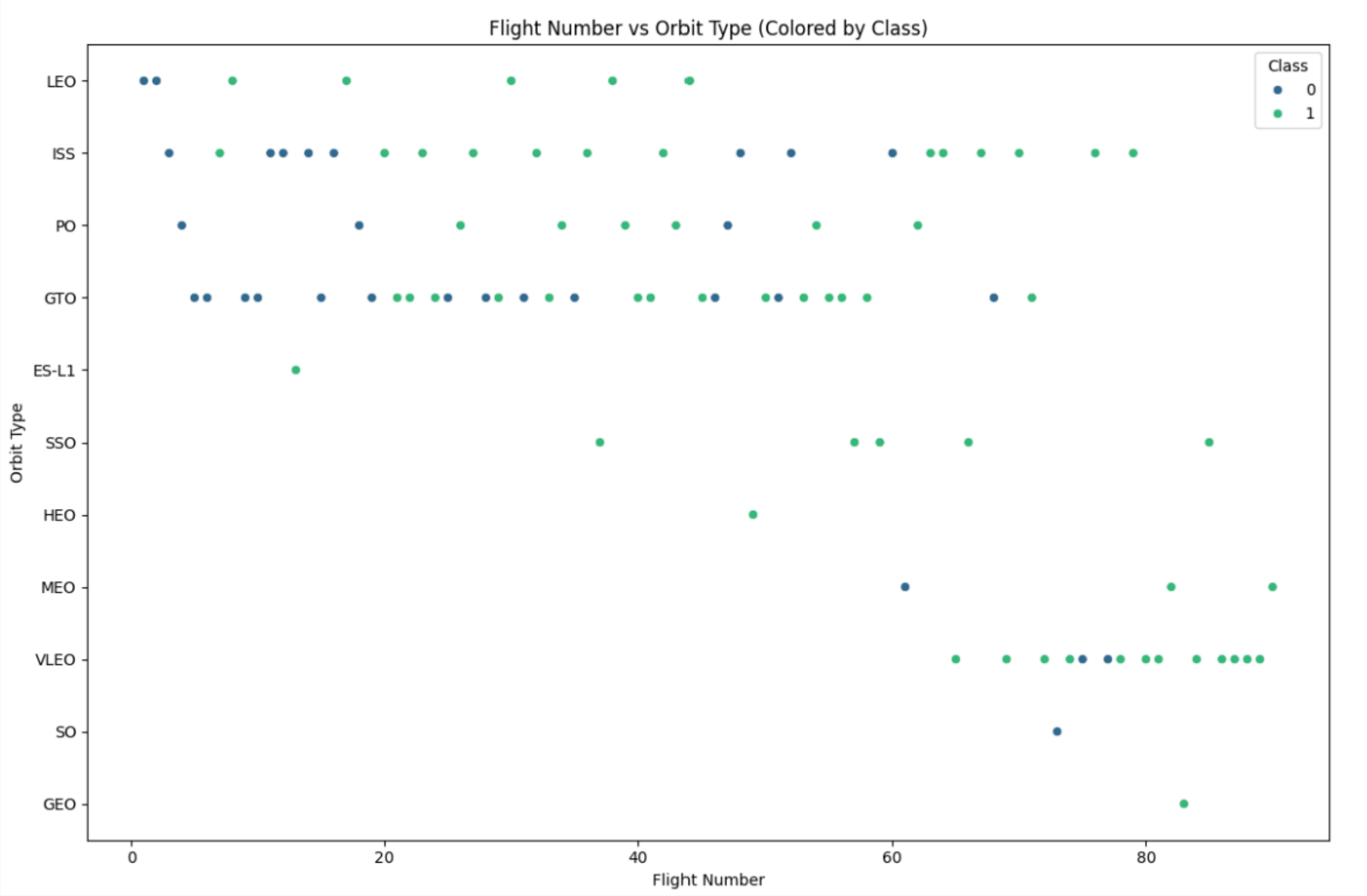
Success Rate vs. Orbit Type

- The plot indicates that the highest success rates were attained by ES-L1, GEO, HEO, SSO, and VLEO.



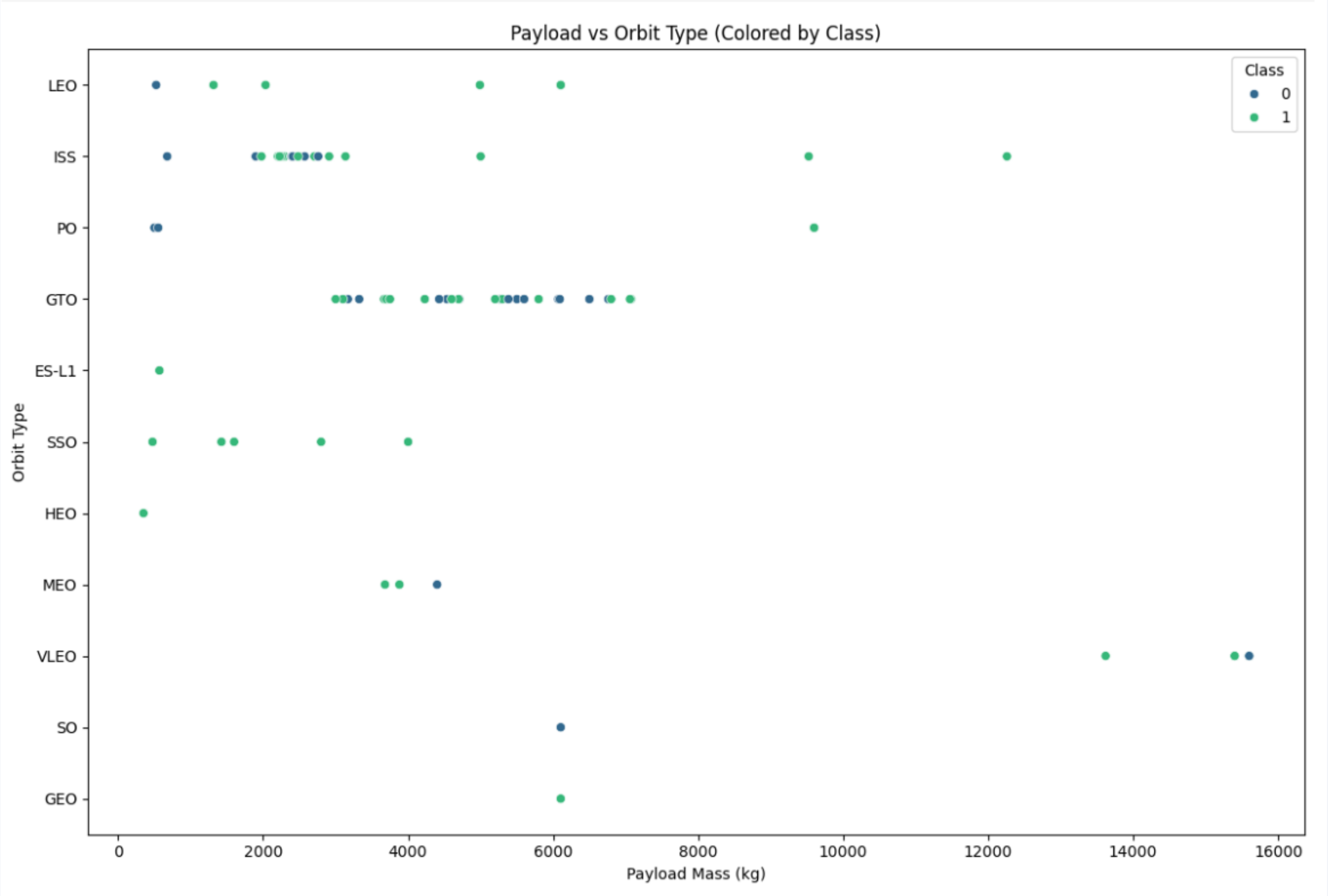
Flight Number vs. Orbit Type

- As you can see, there seems to be no correlation between flight number and success in GTO orbit, while in LEO orbit, the number of flights appears to be related to success.



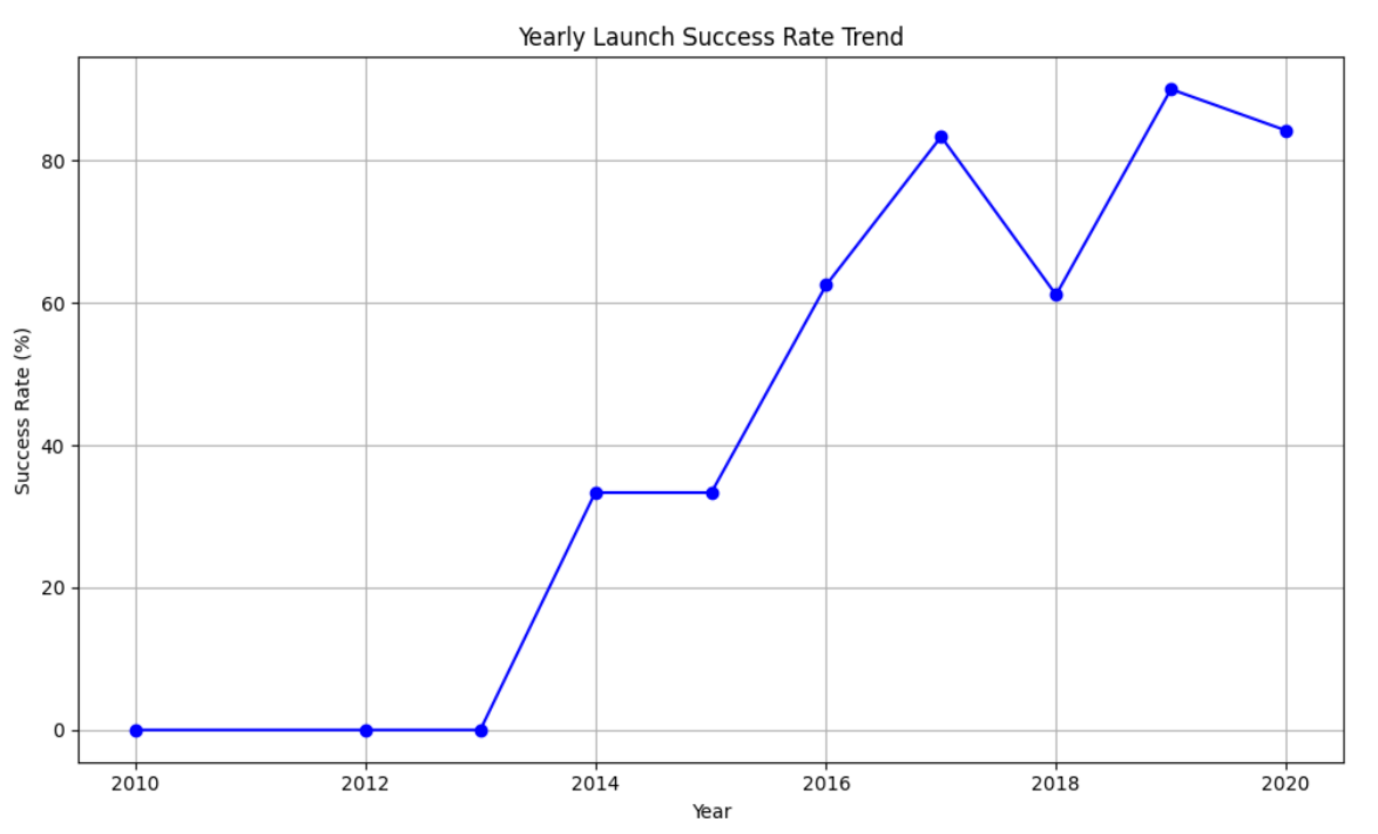
Payload vs. Orbit Type

- Polar, LEO, and ISS have higher success rates—or positive landing rates—when carrying big payloads.
- Unfortunately, for GTO, it is difficult to discern between the two as there is a positive landing rate and a negative landing (a mission that was failed).nations



Launch Success Yearly Trend

- As you can see, from 2013 till 2020, the success rate increased.



All Launch Site Names

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

```
In [11]:
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTBL

 * sqlite:///my_data1.db
Done.
Out[11]:
```

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

Launch Site Names Begin with 'CCA'

- The aforementioned query was utilized to show five records whose launch sites start with 'CCA'.

In [12]:
```
%sql SELECT * FROM SPACEXTBL WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5
```

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

Out[12]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landin
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	

Total Payload Mass

- Using the following query, we were able to determine that 45596 was the entire payload carried by NASA's boosters.

```
In [13]:  %sql SELECT SUM("PAYLOAD_MASS__KG_") AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL WHERE "Customer" = 'NASA (CRS)'

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

Out[13]:

TOTAL_PAYLOAD_MASS
45596

Average Payload Mass by F9 v1.1

- We determined that the booster version F9 v1.1's average payload mass was 2928.4.

```
In [14]: %sql SELECT AVG("PAYLOAD_MASS__KG_") AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL WHERE "Booster_Version" = 'F9 v1.1'

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

Out[14]:

AVERAGE_PAYLOAD_MASS
2928.4

First Successful Ground Landing Date

- We noted that December 22, 2015, was the day of the first successful landing on a ground pad.

```
In [15]: %sql SELECT MIN("Date") AS FIRST_SUCCESSFUL_LANDING_DATE FROM SPACEXTBL WHERE "Landing_Outcome" = 'Success (ground pad)

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

Out[15]:

FIRST_SUCCESSFUL_LANDING_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.

```
In [16]: FROM SPACEXTBL WHERE "Landing_Outcome" = 'Success (drone ship)' AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000

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

Out[16]:

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

Total Number of Successful and Failure Mission Outcomes

- Calculate To filter for WHERE MissionOutcome was successful or unsuccessful, we utilized wildcards like %.

```
In [17]: %sql SELECT "Mission_Outcome", COUNT(*) AS TOTAL_COUNT FROM SPACEXTBL GROUP BY "Mission_Outcome"

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

Out[17]:

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

Boosters Carried Maximum Payload

- Using a subquery in the WHERE clause and the MAX() method, we were able to identify the booster that had transported the maximum payload.

```
In [19]: SELECT DISTINCT "Booster_Version" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL)

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

Out[19]:

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

- In order to filter for failure landing outcomes in drone ship, their booster versions, and launch location names for the year 2015, we combined the WHERE clause, LIKE, AND, and BETWEEN conditions.

In [26]:

```
sql_query = """
SELECT
    CASE
        WHEN substr(Date, 6, 2) = '01' THEN 'January'
        WHEN substr(Date, 6, 2) = '02' THEN 'February'
        WHEN substr(Date, 6, 2) = '03' THEN 'March'
        WHEN substr(Date, 6, 2) = '04' THEN 'April'
        WHEN substr(Date, 6, 2) = '05' THEN 'May'
        WHEN substr(Date, 6, 2) = '06' THEN 'June'
        WHEN substr(Date, 6, 2) = '07' THEN 'July'
        WHEN substr(Date, 6, 2) = '08' THEN 'August'
        WHEN substr(Date, 6, 2) = '09' THEN 'September'
        WHEN substr(Date, 6, 2) = '10' THEN 'October'
        WHEN substr(Date, 6, 2) = '11' THEN 'November'
        WHEN substr(Date, 6, 2) = '12' THEN 'December'
    END AS Month_Name,
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTBL
WHERE substr(Date, 0, 5) = '2015'
    AND "Landing_Outcome" LIKE 'Failure%'
    AND "Landing_Outcome" LIKE '%drone ship%'
"""

#excute query
%sql $sql_query
```

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

Out[26]:

Month_Name	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

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

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

In [27]:
```
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY "Landing_Outcome" ORDER BY Count_Landing_Outcomes DESC
```

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

Out[27]:

Landing_Outcome	Count_Landing_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

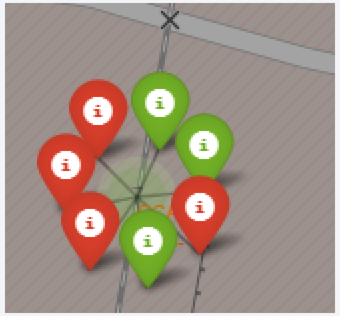
Launch Sites Proximities Analysis

Global Map Markers for All Launch Sites

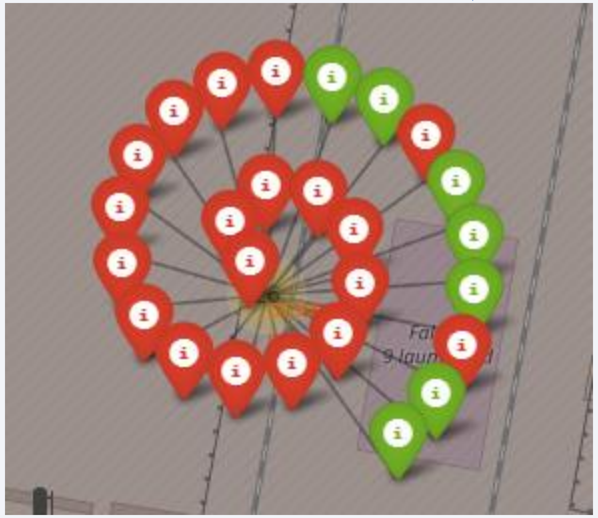
All launch sites are located in the Western hemisphere and in the United States, particularly on the east and western coasts.

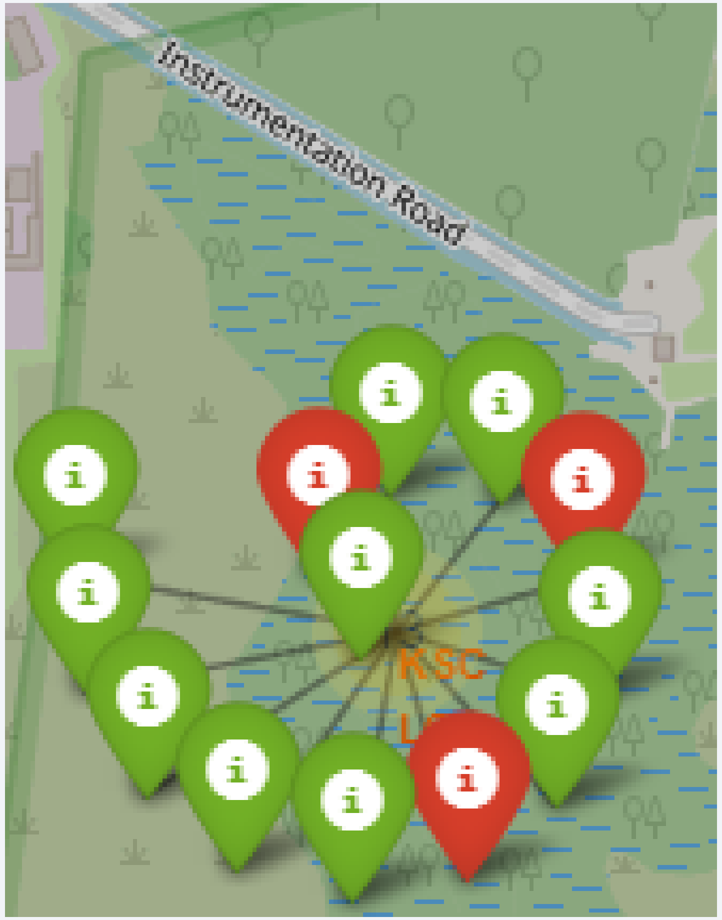
<Folium Map Screenshot 2>

Green represents successful launches while red represents failures.



Florida Launch Sites

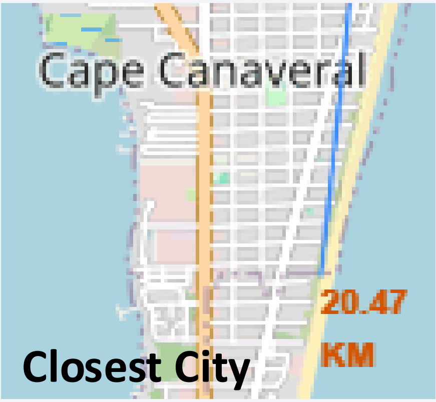


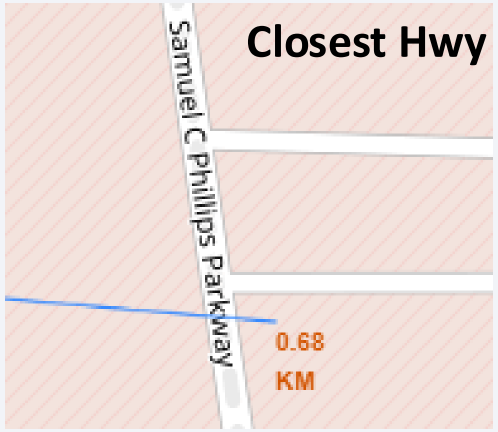


California Launch Site

Distances from Launch Site to Landmarks

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



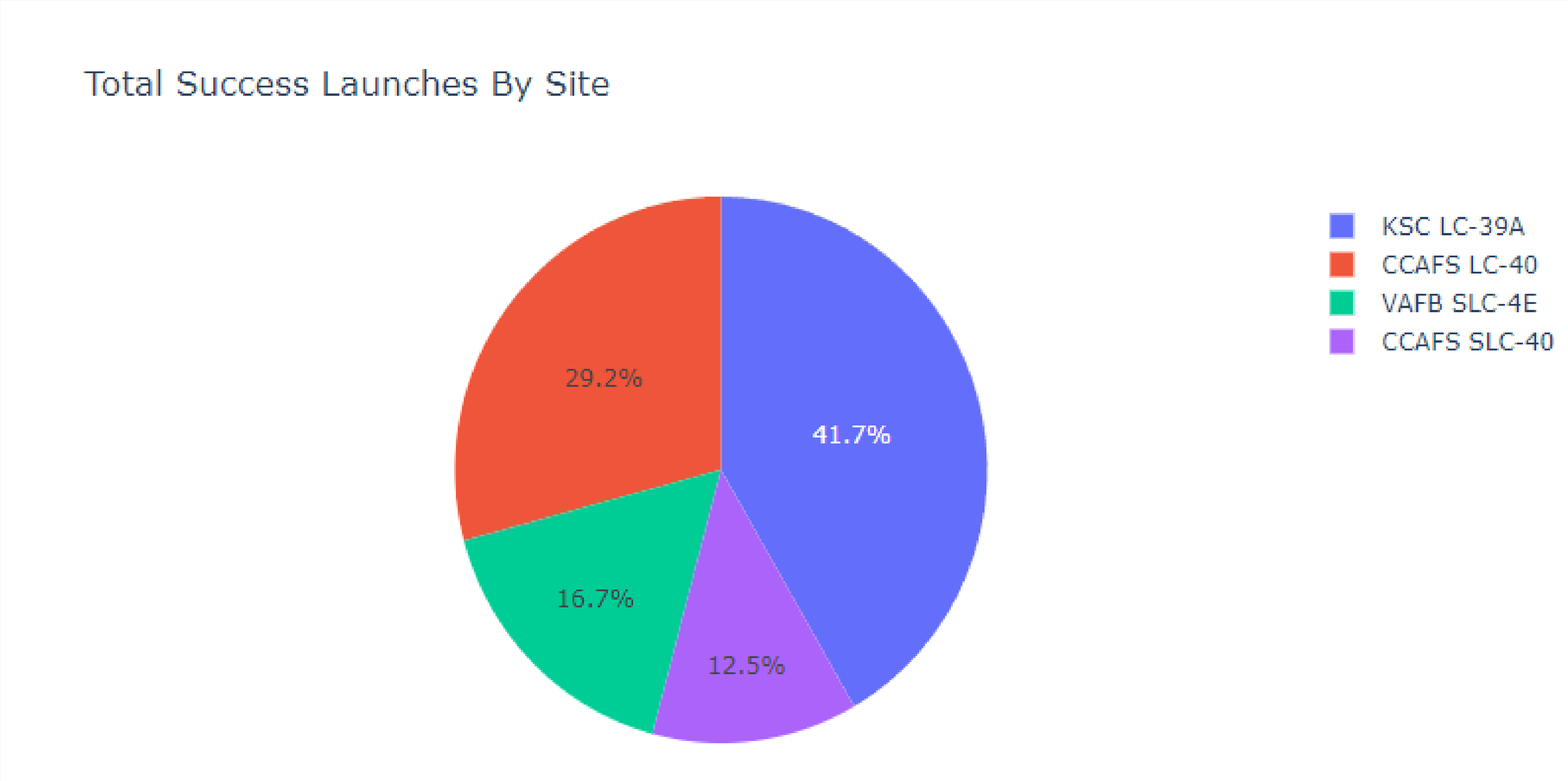


Section 4

Build a Dashboard with Plotly Dash

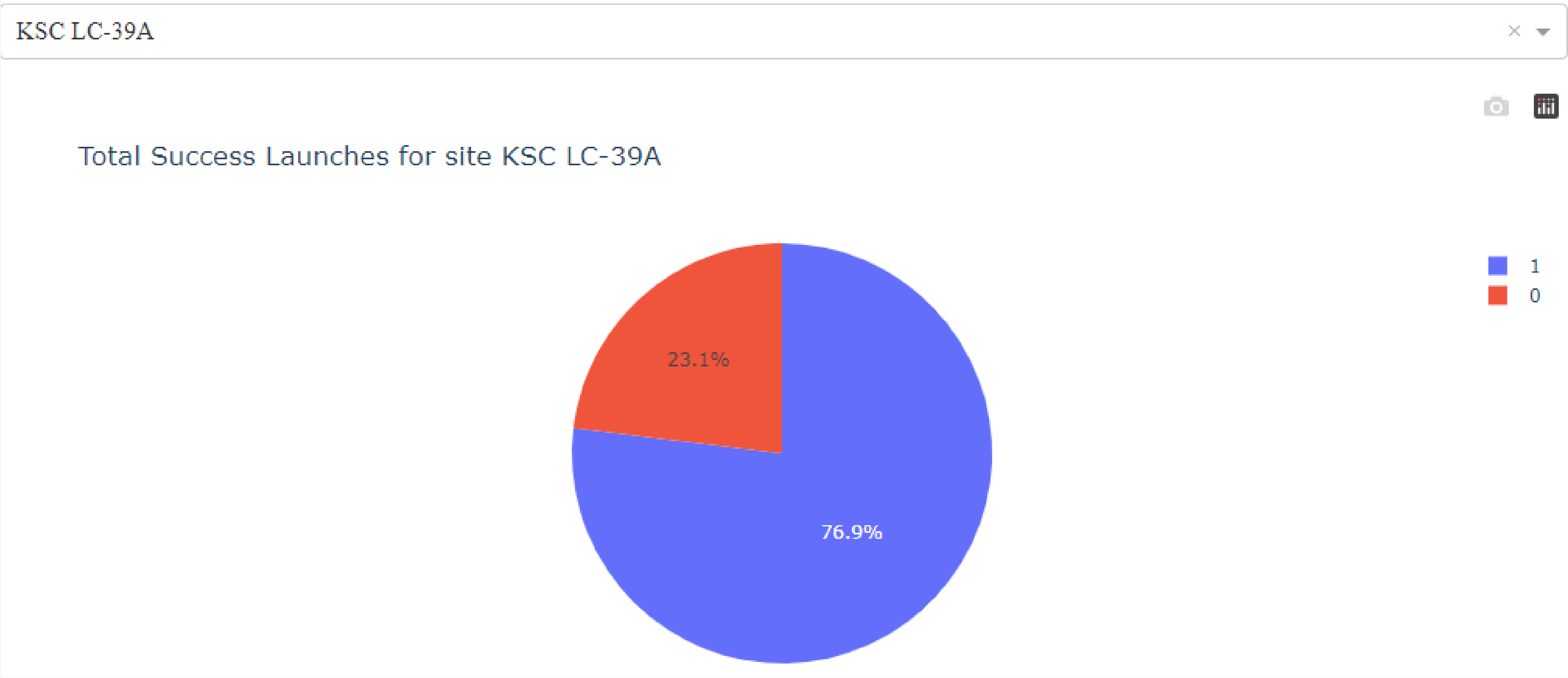
Total Success Launches By Site Pie Chart

KSC LC-39A had the most successful launches, compared to CCAFS SLC-40 which had the least.



KSC LC-39A Success Rate (Highest)

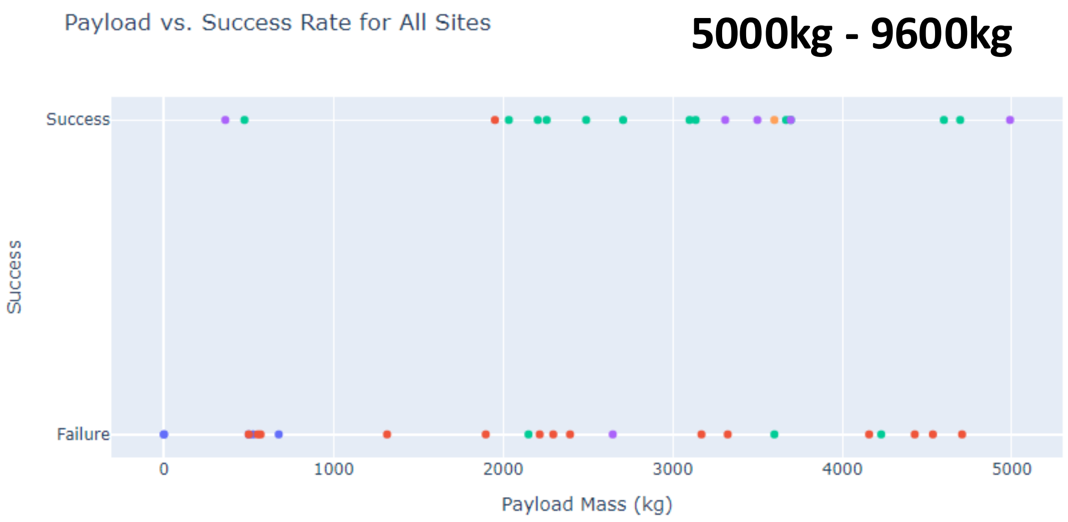
This launch site produced an astounding 76.9% success rate with only a 23.1% fail rate.



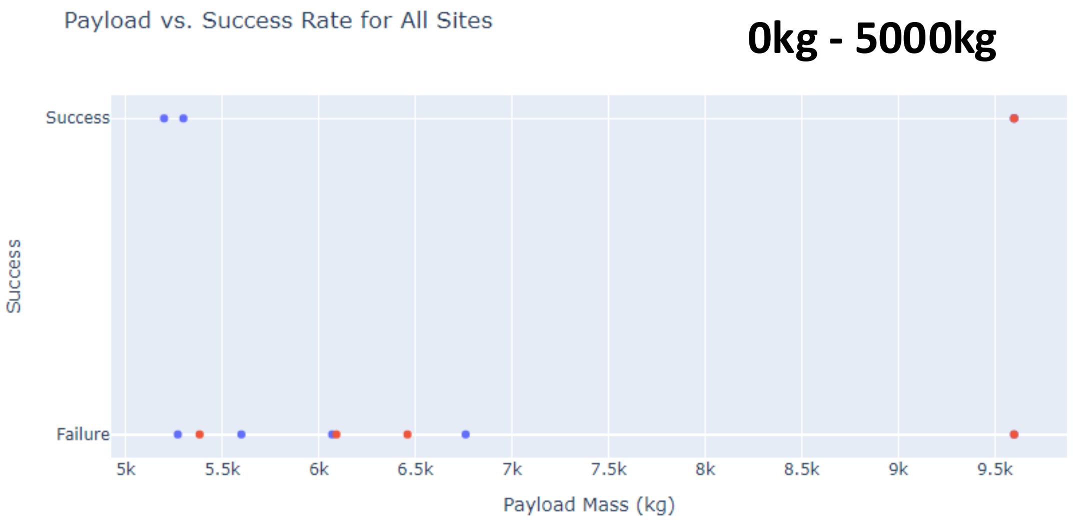
Payload VS Launch Outcome Scatter Plot

We can see the success rate for high weighted payloads is higher than the low weighted payloads.

5000kg - 9600kg



0kg - 5000kg



Section 5

Predictive Analysis (Classification)

Classification Accuracy

The code performs the following actions:

- Computes Test Accuracies
- Stores Test Accuracies in a Dictionary
- Finds the Best Performing Model
- Prints the Results

The output of the code indicates:

- The best performing method is **Logistic Regression**.
- The test accuracy of the best performing method is approximately **0.8333**.

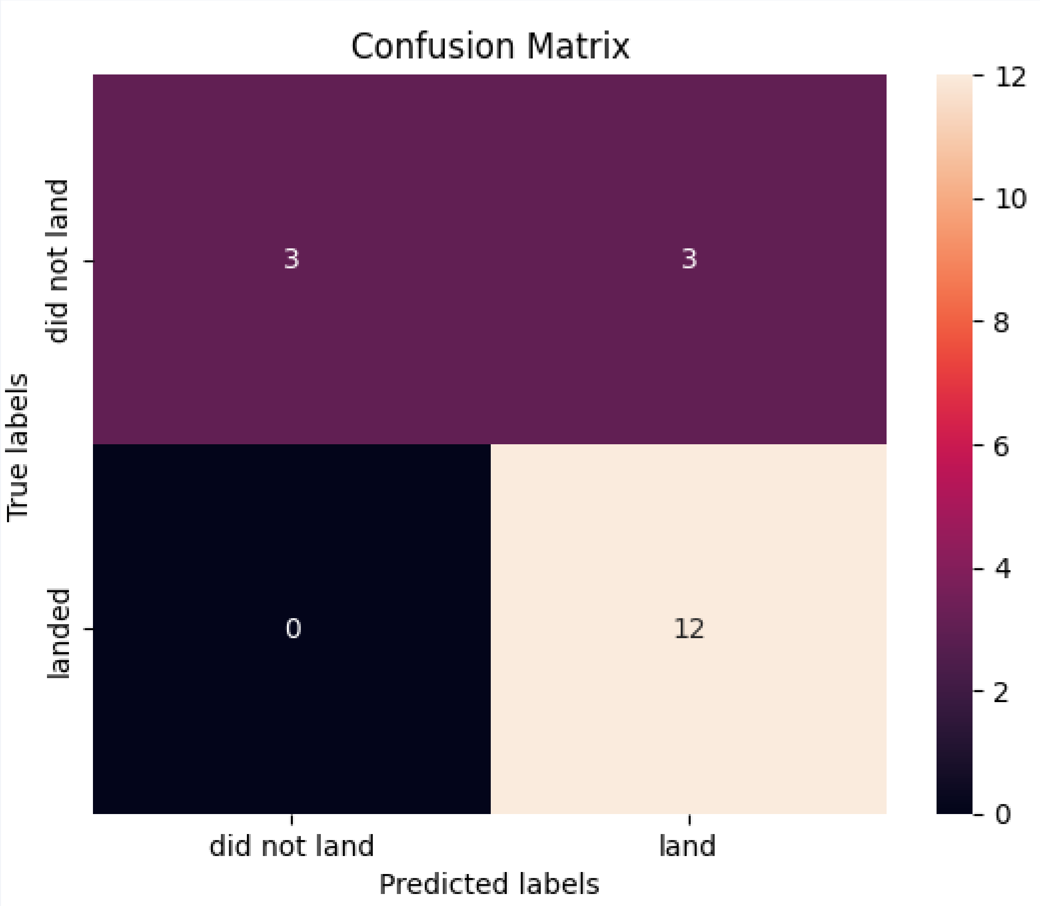
In [39]:

```
logreg_test_accuracy = logreg_cv.score(X_test, Y_test)
svm_test_accuracy = svm_cv.score(X_test, Y_test)
tree_test_accuracy = tree_cv.score(X_test, Y_test)
knn_test_accuracy = knn_cv.score(X_test, Y_test)

# Store test accuracies in a dictionary
test_accuracies = {
    'Logistic Regression': logreg_test_accuracy,
    'SVM': svm_test_accuracy,
    'Decision Tree': tree_test_accuracy,
    'KNN': knn_test_accuracy
}

# Find the method with the highest test accuracy
best_method = max(test_accuracies, key=test_accuracies.get)
best_accuracy = test_accuracies[best_method]

print("Best performing method:", best_method)
print("Test accuracy:", best_accuracy)
```

```
Best performing method: Logistic Regression
Test accuracy: 0.8333333333333334
```

Confusion Matrix

- The model has a high recall, meaning it correctly identifies almost all the "landed" cases, but has a moderate precision due to some false positives.



Conclusions

- Our conclusion is that a launch site's success rate increases with the number of flights conducted there.
- From 2013 to 2020, the launch success rate increased.
- The most successful orbits were ES-L1, GEO, HEO, SSO, and VLEO.
- The most successful launches of any facility were at KSC LC-39A.
- The most effective machine learning approach for this task is the logistic regression.

Thank you!