IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Muhib Zaman
Feb 28, 2024

Outline

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

Executive Summary

Summary of methodologies

- Data collection through API and web scraping
- Data wrangling
- Exploratory data analysis (EDA) with data visualization and SQL
- Interactive visuals with Folium and Plotly Dash
- Machine Learning Prediction

Summary of all results

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

Introduction

Project background and context

SpaceX advertises Falcon 9 rocket launches on its website for $62 million, significantly less than other providers whose costs exceed $165 million each. This substantial cost savings is attributed to SpaceX's ability to reuse the first stage of the rocket. Thus, predicting the successful landing of the first stage is crucial in estimating the launch cost. The project aims to develop a machine learning pipeline to predict if the 1st stage will land successfully.

Problems you want to find answers

- What is the impact of impact of Payload Mass, Launch Site, Flight Frequency, and Orbits on first-stage landing success?
- What is the trend of successful landing over time?
- What is best predictive model for successful landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology: Data was collected through a combination of using the SpaceX API and scraping information Wikipedia
- Perform data wrangling: Filtered, handled missing values, and applied one hot-encoding to 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

Data Collection

- **Collected** data using get request from SpaceX API
- **Decoded** the data using .json() and **converted** into a Pandas dataframe using .json_normalize()
- **Collected** additional data points from SpaceX API using custom functions
- **Created** dictionary from data and **created** dataframe from the dictionary
- **Filtered** dataframe to display only Falcon 9 launches
- **Replaced** missing values for PayLoadMass/LandingPad

Data Collection – SpaceX API

- **Collected** data using get request from SpaceX API
- **Decoded** the data using .json() and **Converted** into a Pandas dataframe using .json_normalize()
- **Collected** additional data points from SpaceX API using custom functions
- **Created** dictionary from data and **Created** dataframe from the dictionary
- **Filtered** dataframe to display only Falcon 9 launches and **Replaced** missing values for PayLoadMass/LandingPad

GitHub link:
https://github.com/mzr015/ds_capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

```
# 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
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
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)]
```

```
from pandas import json_normalize

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

# Convert JSON data into a DataFrame
data = json_normalize(json_data)

# Print the dataframe
#print(df)
```

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

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

Data Collection - Scraping

- **Collected** data for Falcon 9 launch using get request from Wikipedia
- **Created** a BeautifulSoup object from the HTML response
- **Extracted** all column from the HTML table header
- **Created** a data frame by parsing the launch HTML tables

GitHub link: https://github.com/mzr015/ds_capstone/blob/main/jupyter-labs-webscraping.ipynb

```
# 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 successful. Response object assigned to the variable 'response'.")
else:
    print("Failed to retrieve page. Status code:", response.status_code)
```

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

# Now 'soup' contains the BeautifulSoup object
print("BeautifulSoup object created successfully.")
```

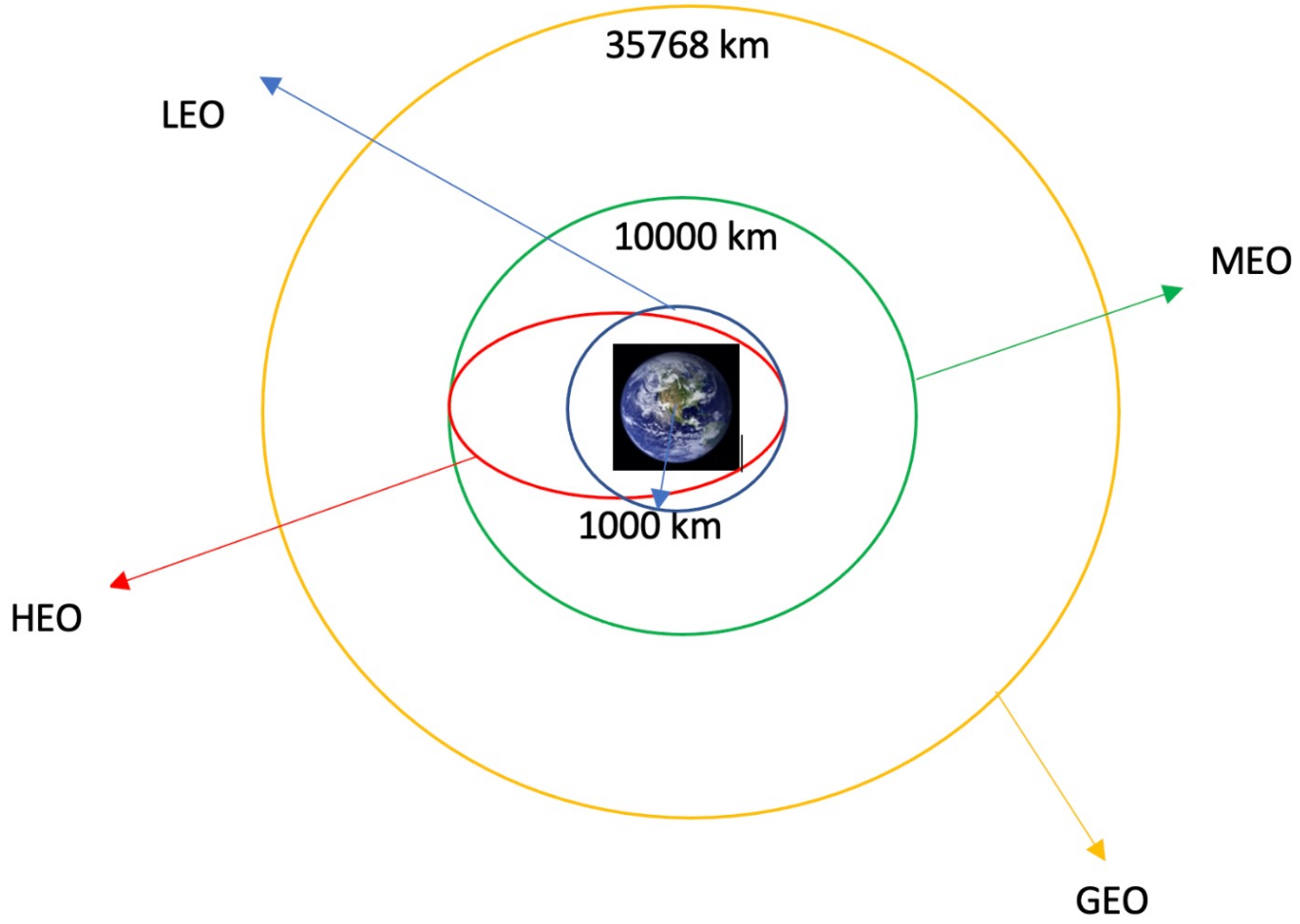
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
th_element = first_launch_table.find_all('th')

# Iterate each th element and apply the provided extract_column_from_header() to get a column name
def extract_column_from_header(th_element):
    return th_element.text.strip()

for th_element in th_elements:
    # Extract the column name
    name = extract_column_from_header(th_element)
    # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
    if name is not None and len(name) > 0:
        column_names.append(name)
```

```
# df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df = pd.DataFrame({key: pd.Series(value, dtype='object') for key, value in launch_dict.items()})
```

Data Wrangling

- **Performed** exploratory data analysis and **determined** training labels
- **Calculated**
  - the number of launches on each site
  - the number and occurrence of each orbit
  - the number and occurrence of mission outcome of the orbits
- **Created** a landing outcome label from Outcome column

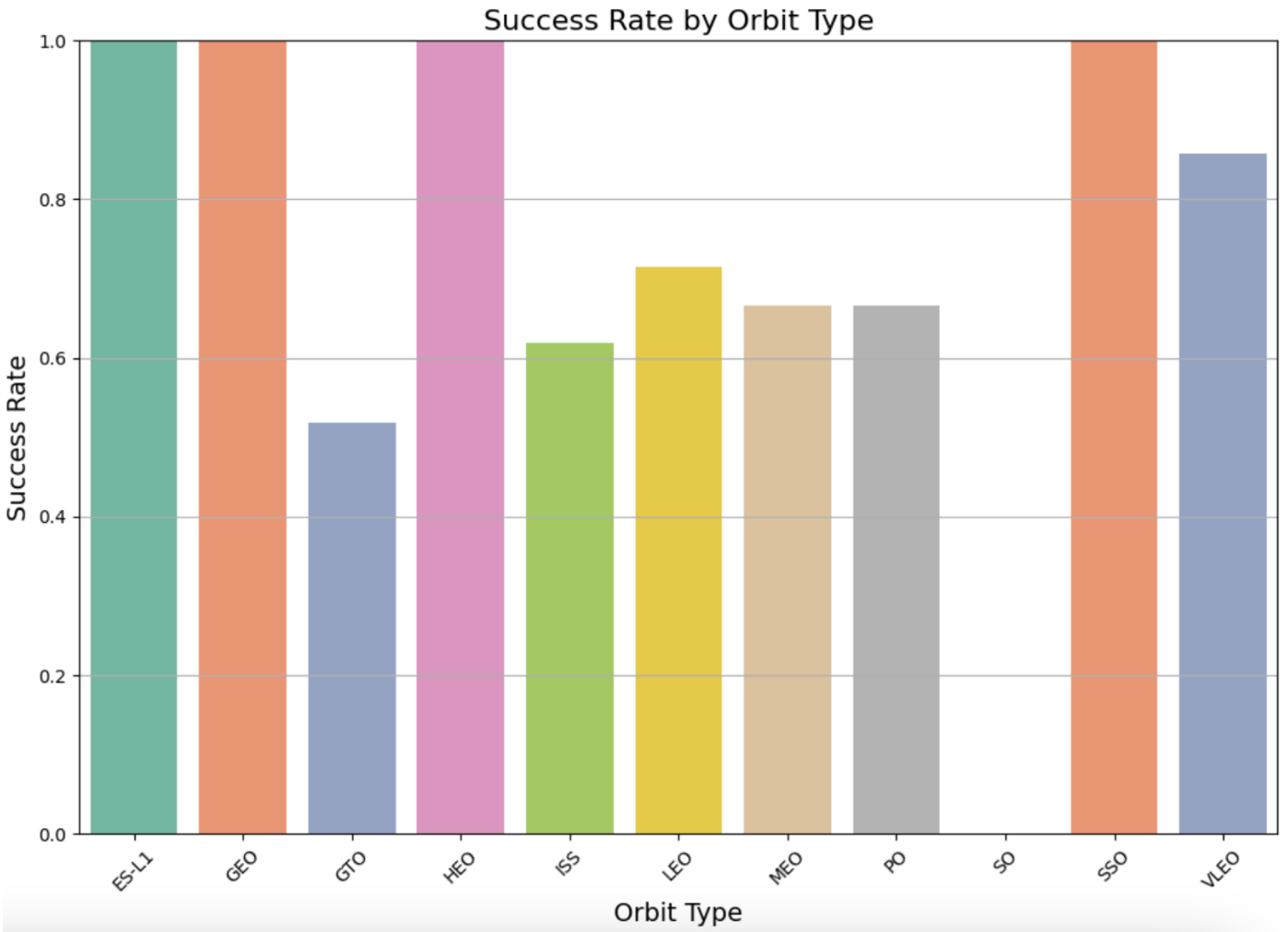
GitHub link:
https://github.com/mzr015/ds_capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



EDA with Data Visualization

- **Visualizations**
- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Success Rate vs. Orbit Type
- Flight Number vs. Orbit Type
- Payload vs. Orbit Type
- Launch Success Yearly Trend

GitHub Link:
https://github.com/mzr015/ds_capstone/blob/main/jupyter-labs-eda-dataviz.ipynb



EDA with SQL

SQL Queries

- All Launch Site Names
- Launch Site Names Begin with 'CCA’
- Total Payload Mass
- Average Payload Mass by F9 v1.1
- First Successful Ground Landing Date
- Successful Drone Ship Landing with Payload between 4000 and 6000
- Total Number of Successful and Failure Mission Outcomes
- Boosters Carried Maximum Payload
- 2015 Launch Records
- Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

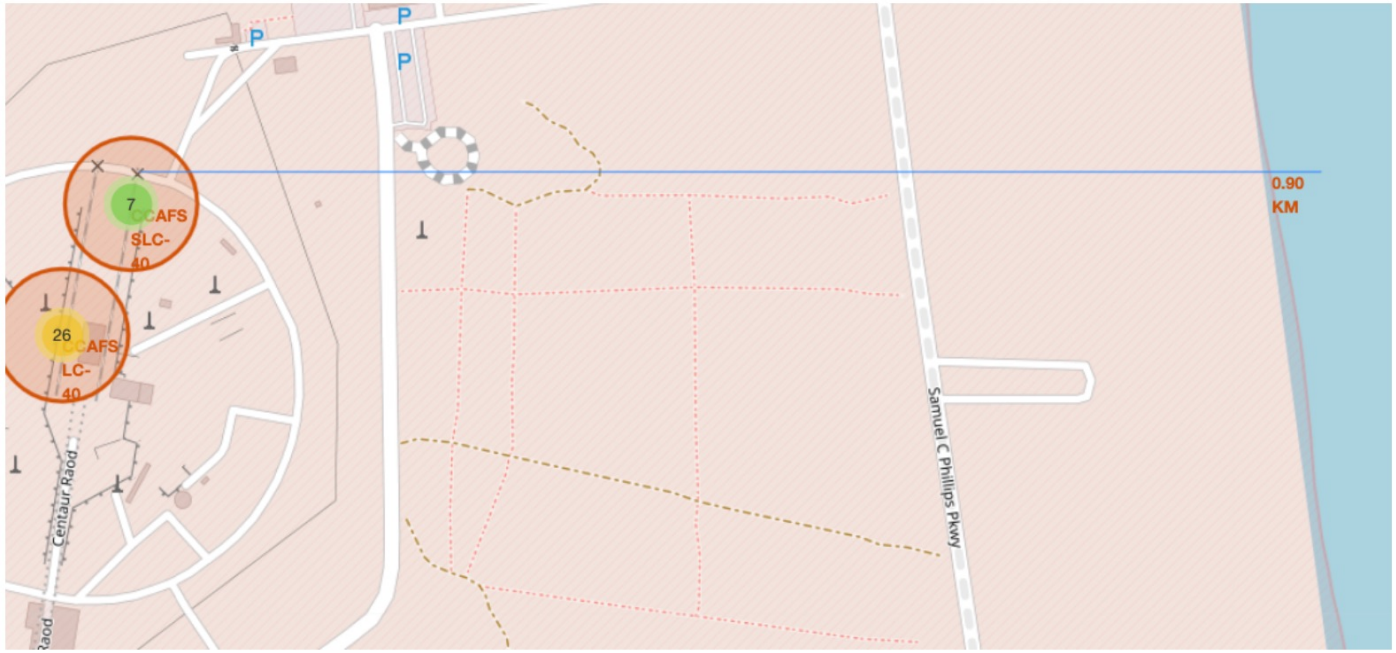
GitHub Link: https://github.com/mzr015/ds_capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

Map objects:

- Markers: To pinpoint each launch site's locations on the map.
- Circles: To highlight each launch site's general areas.
- Marker Clusters: To handle multiple markers with the same coordinates
- Polyline: To represent the distances between launch sites and various proximities such as coastlines.

GitHub Link:
https://github.com/mzr015/ds_capstone/blob/main/lab_jupyter_launch_site_location.ipynb



Build a Dashboard with Plotly Dash

Pie Chart (Success vs. Failure):Displays the total count of successful and failed launches.

- **Interaction**: Users can select a specific launch site from the dropdown menu, and the pie chart dynamically updates to show the success vs. failure counts for the selected site.
- **Purpose**: Provides users with an overview of the success rate for SpaceX launches across different sites.

Scatter Plot (Mission Outcomes vs. Payload Mass): shows the relationship between payload mass and launch outcomes (success or failure).

- **Interaction**: Users can select a launch site from the dropdown menu and adjust the payload range using the slider.
- **Purpose**: This plot allows users to explore how payload mass may affect the success or failure of SpaceX launches at different sites. It also helps identify any correlations between payload mass and mission outcomes.

GitHub Link: https://github.com/mzr015/ds_capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

Data Preparation:

- Loaded and explored the dataset.
- Standardized the features and split the data into training and testing sets.

Model Building:

- Utilized Logistic Regression, SVM, Decision Tree, and KNN algorithms.
- Tuned hyperparameters using GridSearchCV.

Evaluation:

- Calculated accuracies on the test data.
- Visualized performance with confusion matrices.

Selection:

- Identified the best model based on max accuracy

GitHub Link: https://github.com/mzr015/ds_capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.ipynb

Results

Exploratory data analysis results

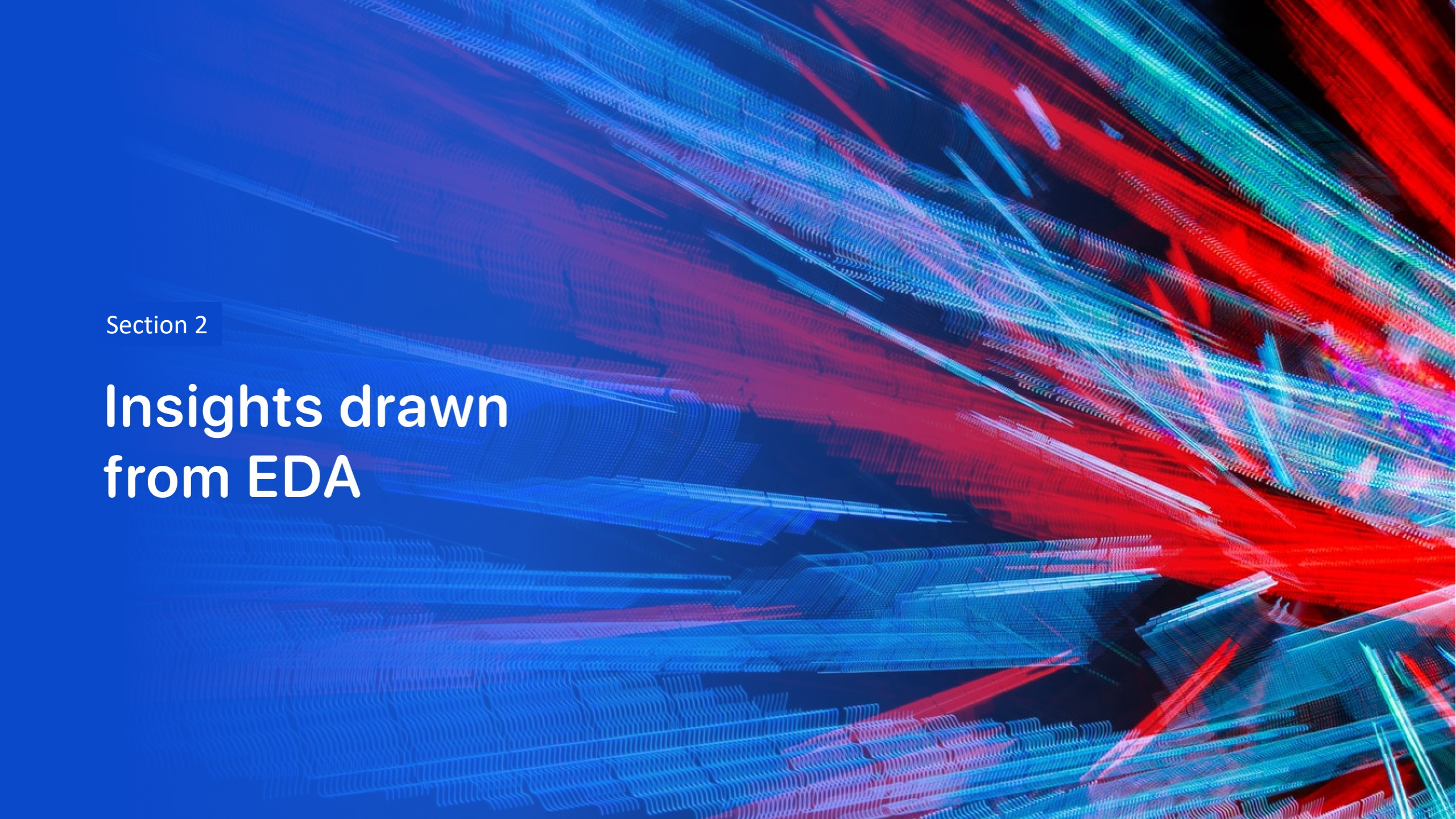
- Success rate has increased over time
- Orbits ES-L1, GEO, HEO, SSO have a 100% success rate
- KSC LC 39A is most successful launching site

Interactive analytics demo in screenshots

- All launch sites are close to coastlines
- They are generally far away from cities, highways, and other infrastructures

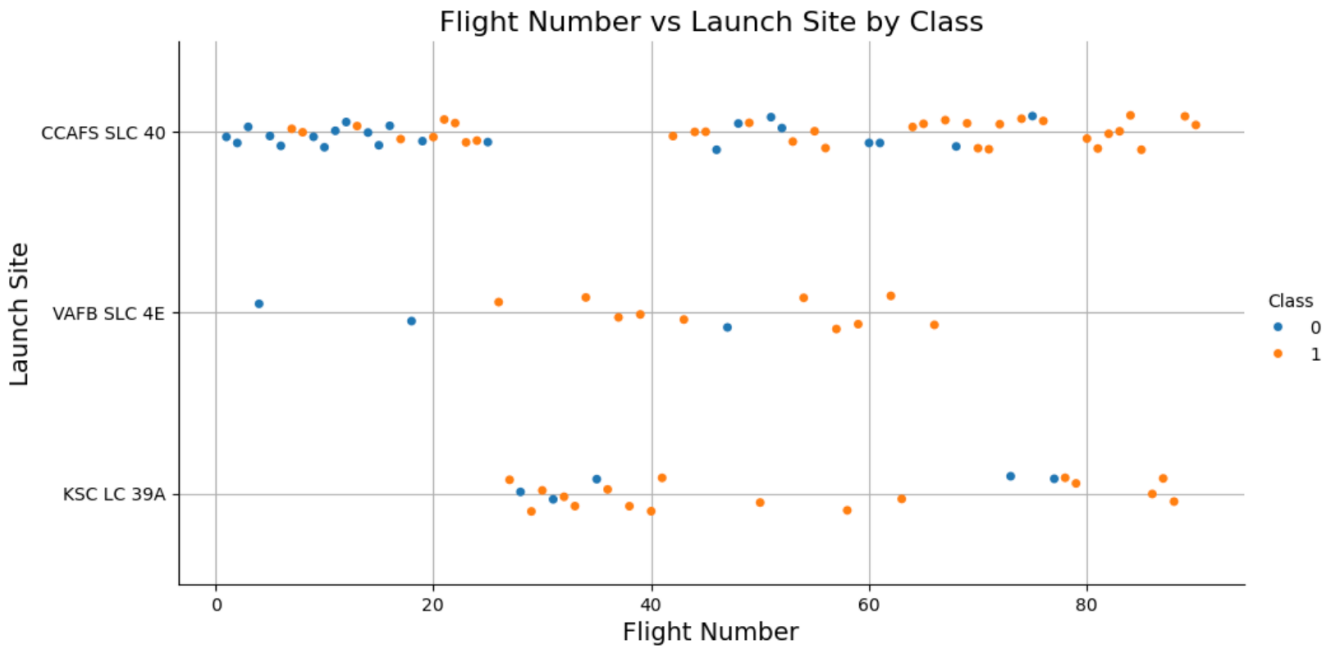
Predictive analysis results

- Decision Tree Classifier is the best predictive model (score: 0.88571)


Section 2
Insights drawn from EDA

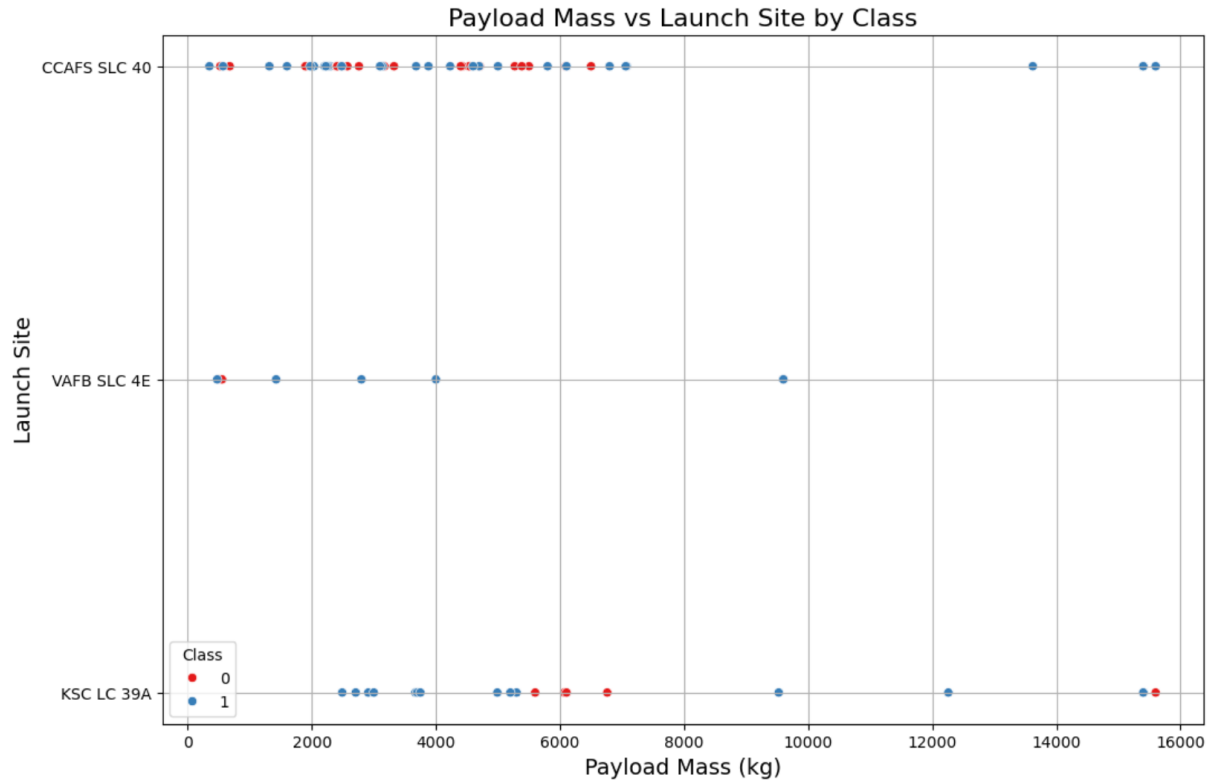
Flight Number vs. Launch Site

The analysis revealed that launch sites with higher flight frequencies tend to exhibit higher success rates.



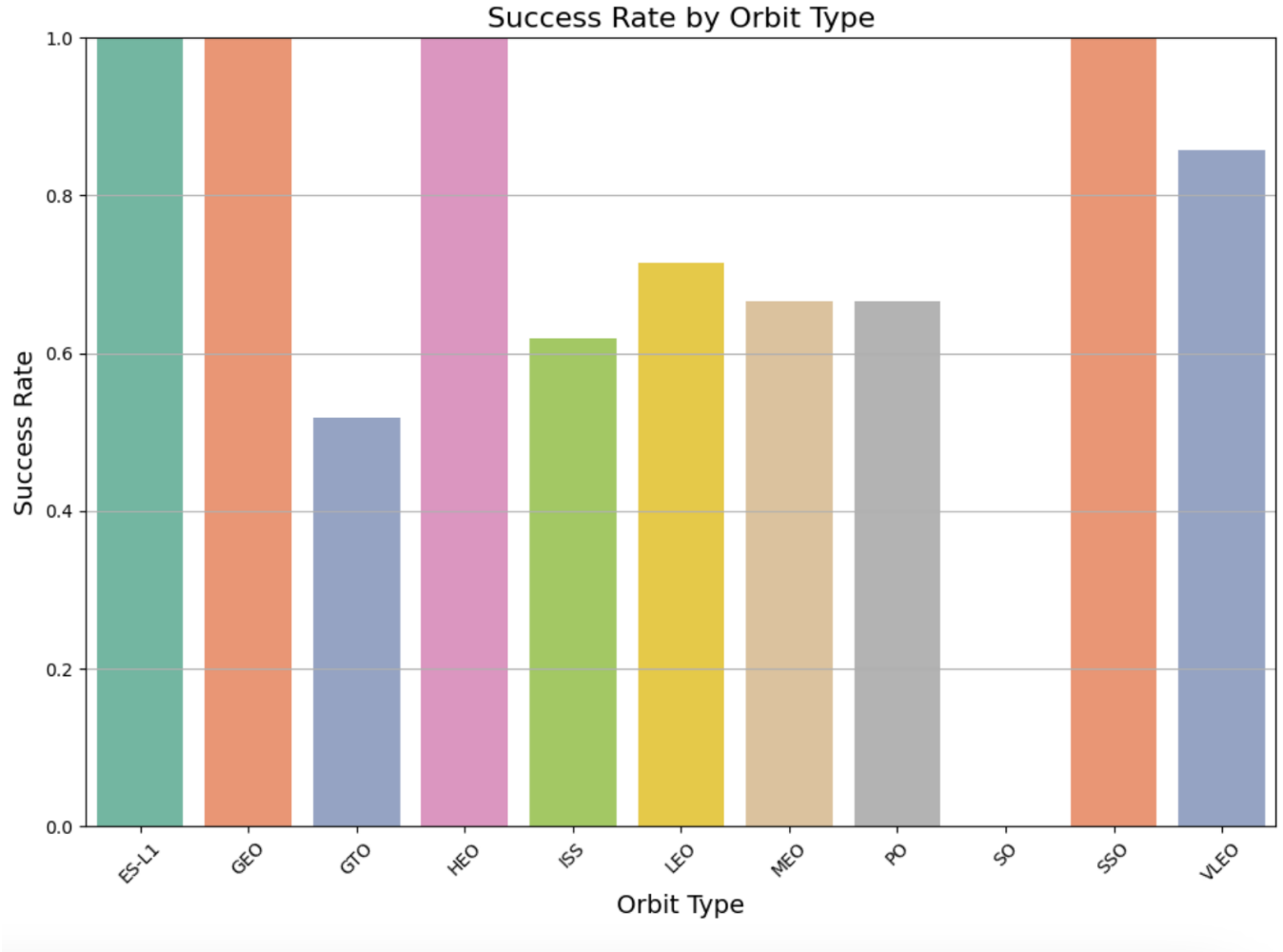
Payload vs. Launch Site

The higher the payload mass (kg), the higher the success rate



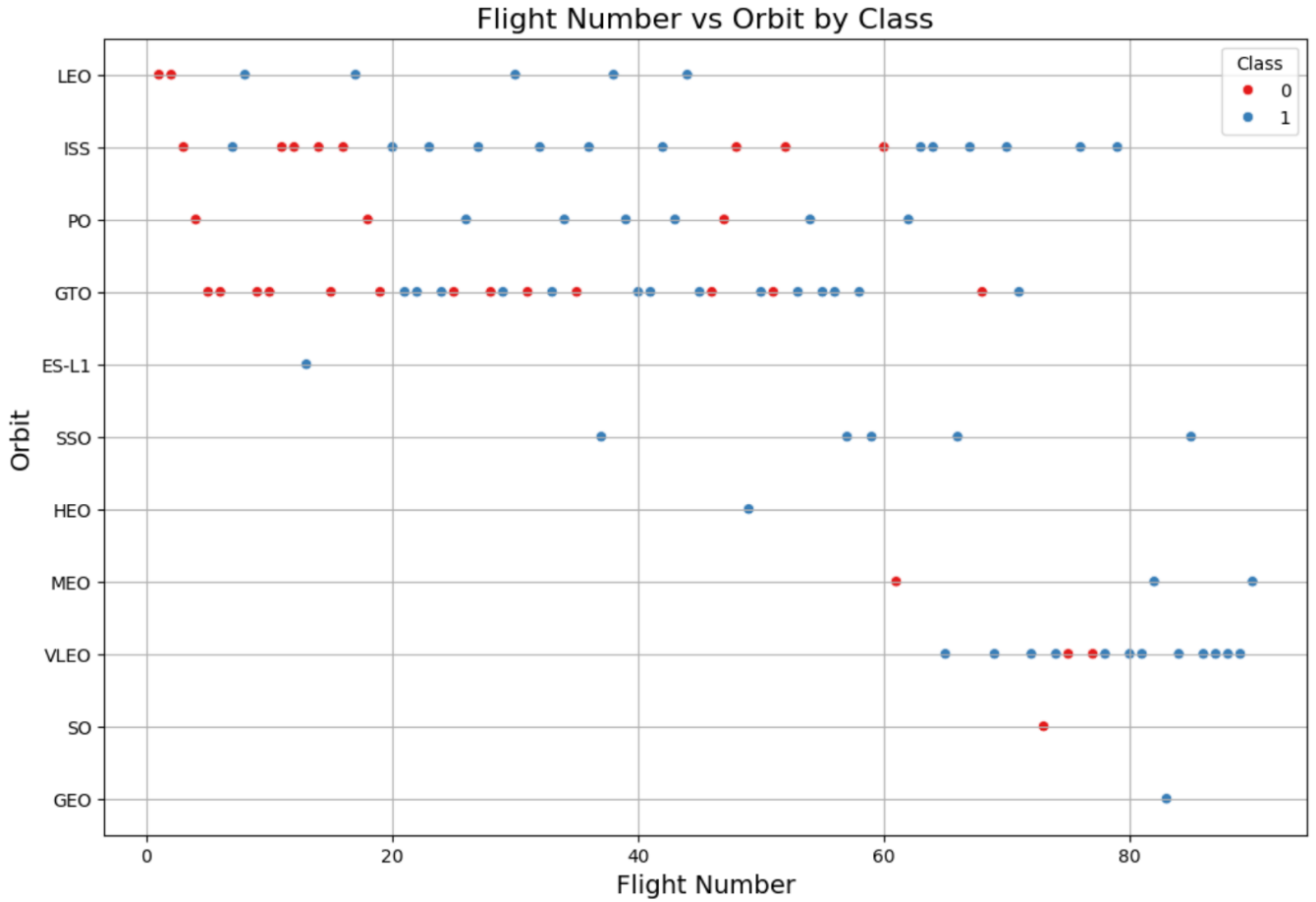
Success Rate vs. Orbit Type

- Orbits **ES-L1**, **GEO**, **HEO**, **SSO** have a 100% success rate
- Orbit **SO** has 0% success rate
- The rest have success rates between 50%-85%



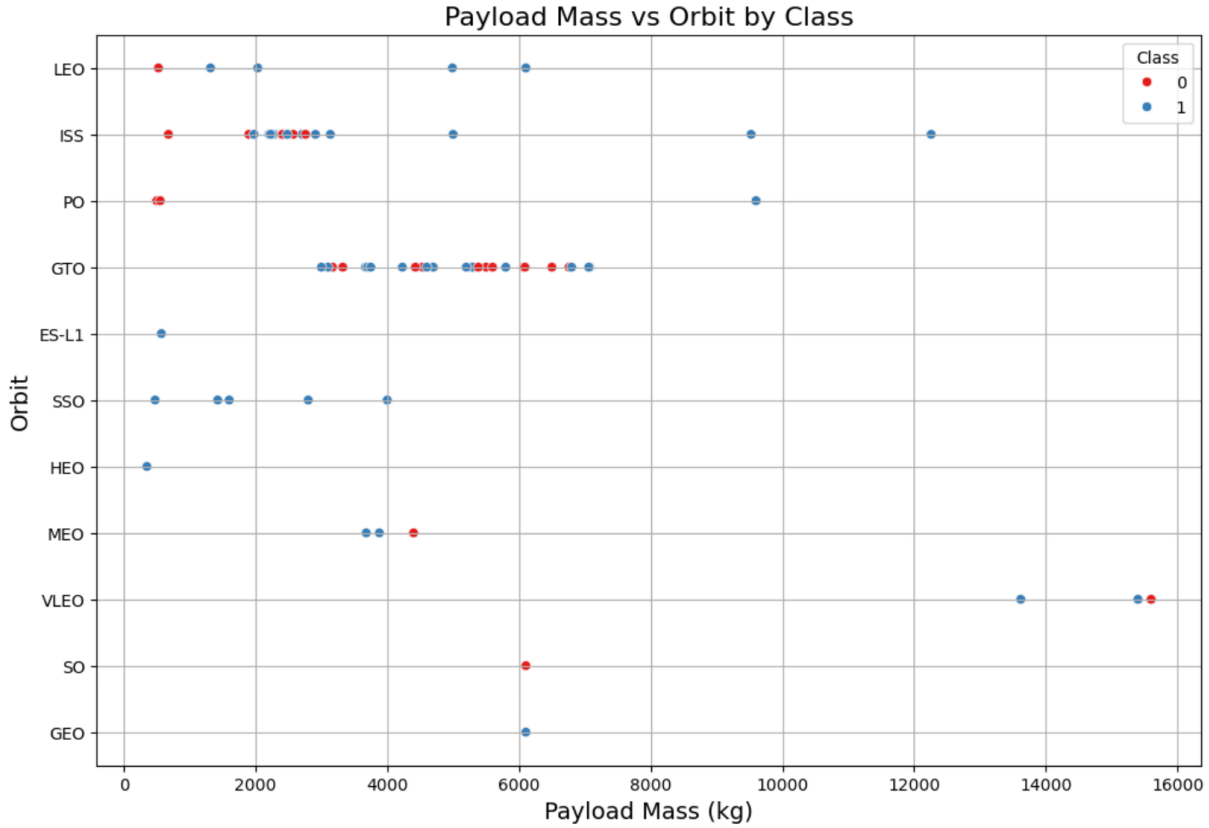
Flight Number vs. Orbit Type

The success rate typically rises with the number of flights in each orbit. In **LEO**, success correlates with the frequency of flights. However, **GTO** does not follow this trend.



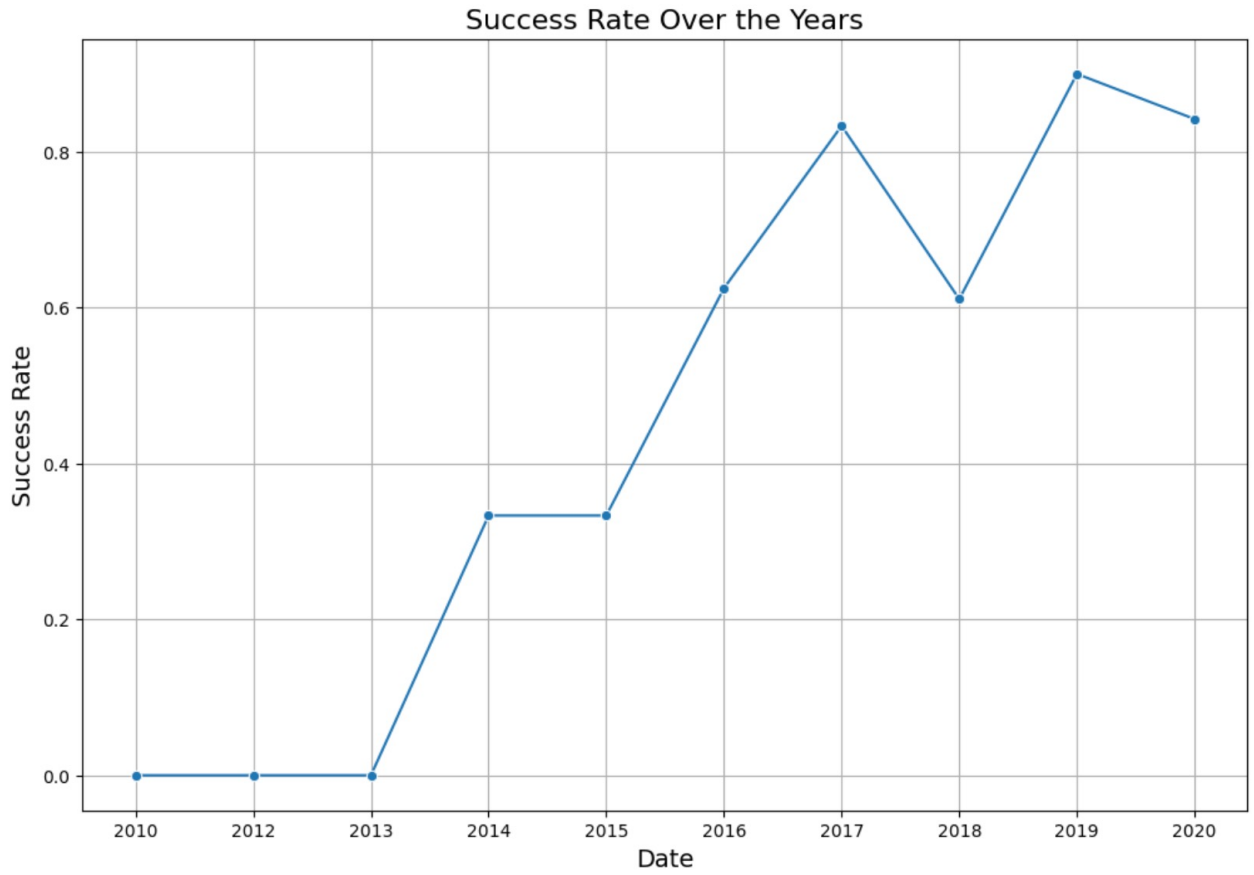
Payload vs. Orbit Type

Heavier payloads result in more successful landings for **PO**, **LEO**, and **ISS** orbits.



Launch Success Yearly Trend

Success rate has steadily increased from **2013** to **2020**



All Launch Site Names

Used the query to find all unique launch site names

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

```
%sql SELECT distinct("Launch_Site") FROM SPACEXTABLE
```

```
 * 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'

Used the query to find all launch site names that begins with ‘CCA’ and limited the result to 5

Display 5 records where launch sites begin with the string 'CCA'

```
%sql SELECT * FROM SPACEXTABLE  \
     WHERE Launch_Site like ('CCA%') \
     LIMIT 5

#%sql SELECT distinct("Landing_Outcome") FROM SPACEXTABLE
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (p
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 (p
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

Total Payload Mass

Used the query to compute the total payload carried by boosters from NASA, resulting in a total of **45,596**

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as totalPayloMass \
     FROM SPACEXTABLE \
     WHERE Customer like 'NASA (CRS)'
```

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

totalPayloMass
45596

Average Payload Mass by F9 v1.1

Used the query to compute the average payload carried by booster version F9 v1.1, resulting in a total of **2928.4**

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as avgPayloMass \
     FROM SPACEXTABLE \
     WHERE Booster_Version = 'F9 v1.1'
```

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

avgPayloMass
2928.4

First Successful Ground Landing Date

Used the query to identify the date of the first successful ground landing, which occurred on **December 22nd, 2015**.

```
%sql SELECT min("Date") FROM SPACEXTABLE \
     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

Used the query to identify successful landing with payload mass greater than 4000 but less than 6000

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

```
 * 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

Used the query to calculated the total number of successful (**100**) and failure (**1**) mission outcomes

```
%sql SELECT \
        CASE \
            WHEN "Mission_Outcome" like 'Success%' THEN 'Success' \
            WHEN "Mission_Outcome" like 'Failure%' THEN 'Failure' \
            ELSE "Mission_Outcome" \
        END AS "Outcome_Group", \
        COUNT(*) AS "TotalMissions" \
      FROM SPACEXTABLE \
      GROUP BY "Outcome_Group"
```

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

Outcome_Group	TotalMissions
Failure	1
Success	100

Boosters Carried Maximum Payload

Used the query to identify the boosters that have carried the maximum payload

```
%sql SELECT Booster_Version, PAYLOAD_MASS__KG_ \
     FROM SPACEXTABLE \
     WHERE PAYLOAD_MASS__KG_ = (SELECT max("PAYLOAD_MASS__KG_") FROM SPACEXTABLE)
```

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

Used the query to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

```
%sql 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", \
    Landing_Outcome, \
    Booster_Version, \
    Launch_Site \
FROM SPACEXTABLE \
WHERE substr("Date", 0, 5) = '2015' \
AND Landing_Outcome = 'Failure (drone ship)'
```

```
 * 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

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

Used the query to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20 and then ranked by count

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

#%sql select payload_mass__kg_ from SPACEXTBL where payload_m
```

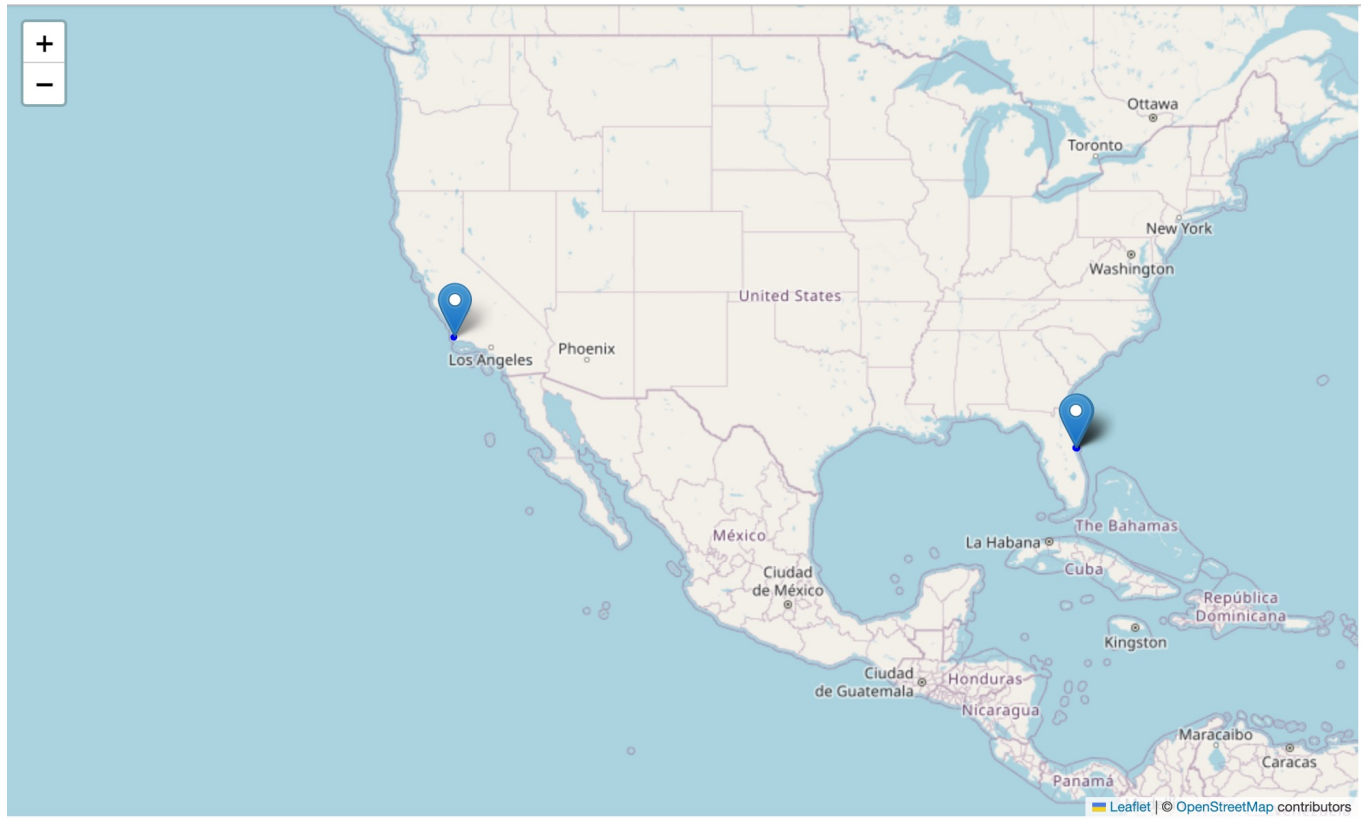
```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	OutcomeCount
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

Launch Sites

The launch sites are situated on the coast of the United States, with one in California and the other in Florida.



Launch Outcomes

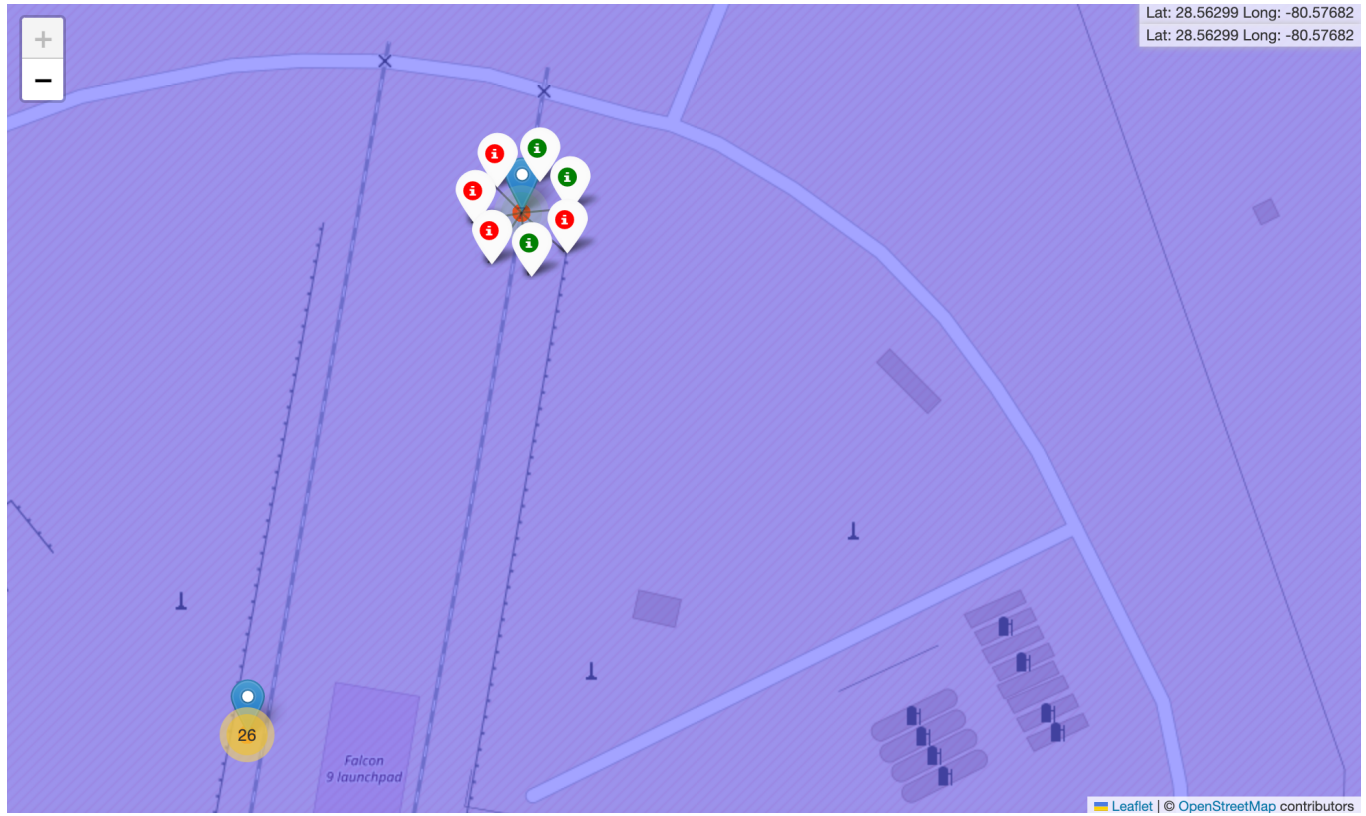
- **Green**: Successful launches
- **Red**: Failed Launches

CCAFS SLC-40: 3 successful launches out of 7

CCAFS LC-40: 7 successful launches out of 26

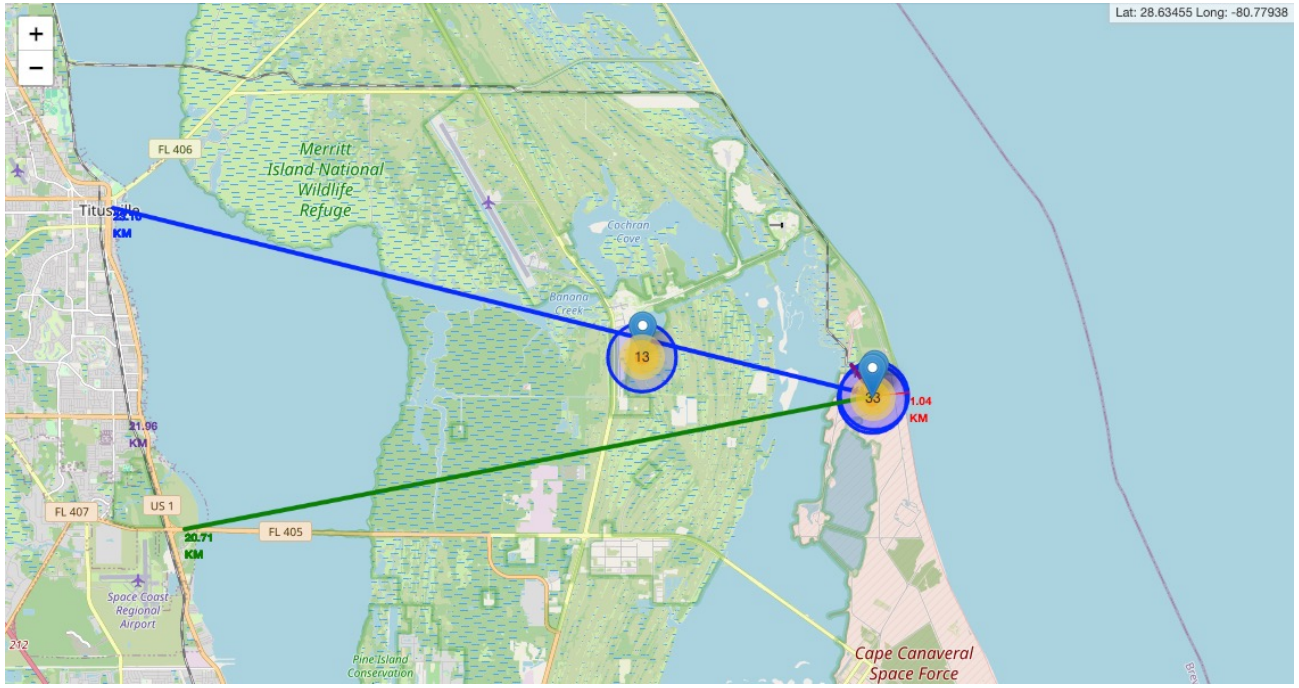
KSC LC-39A: 10 successful launches out of 13

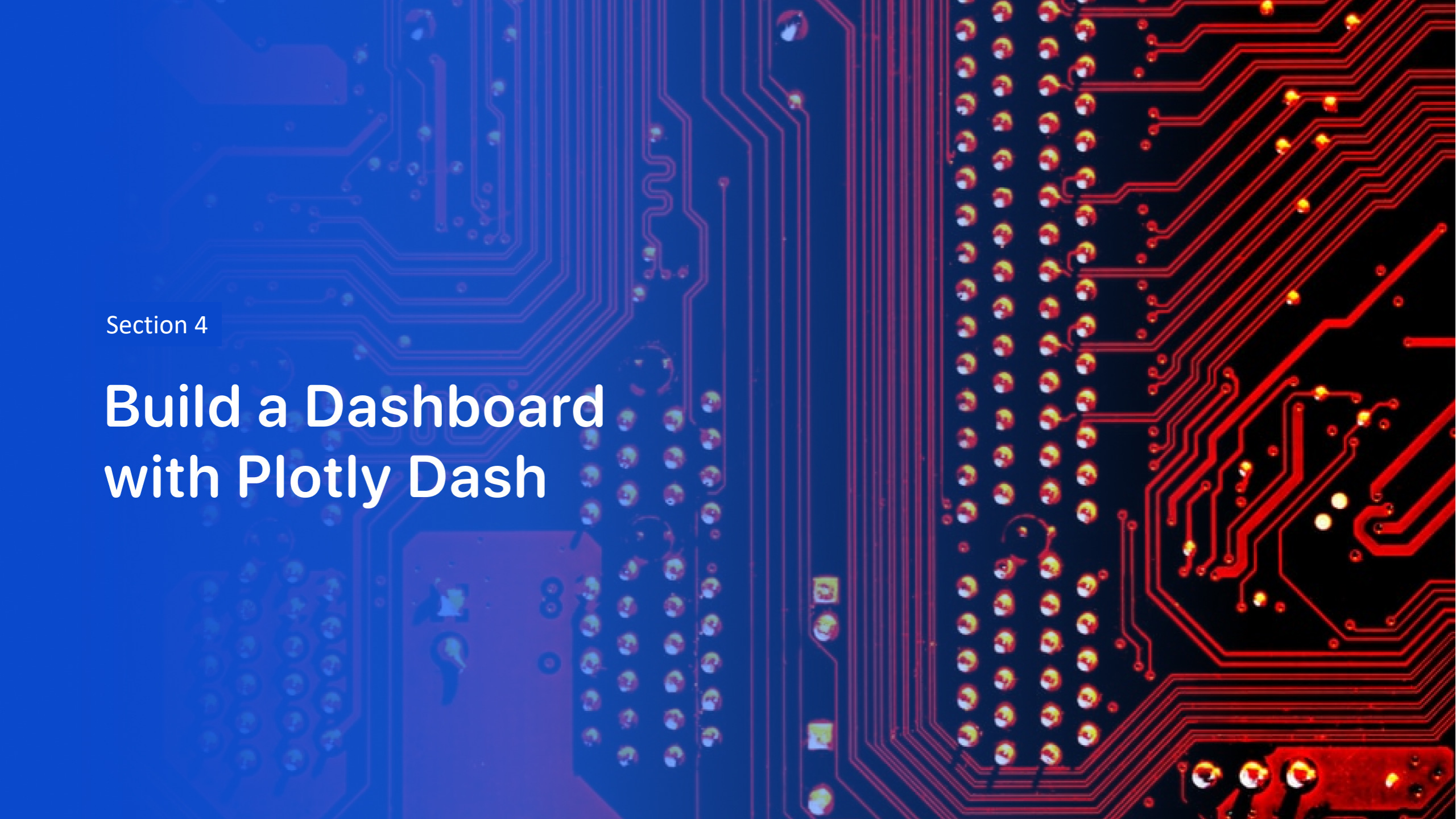
VAFB SLC-4E: 4 successful launches out of 10



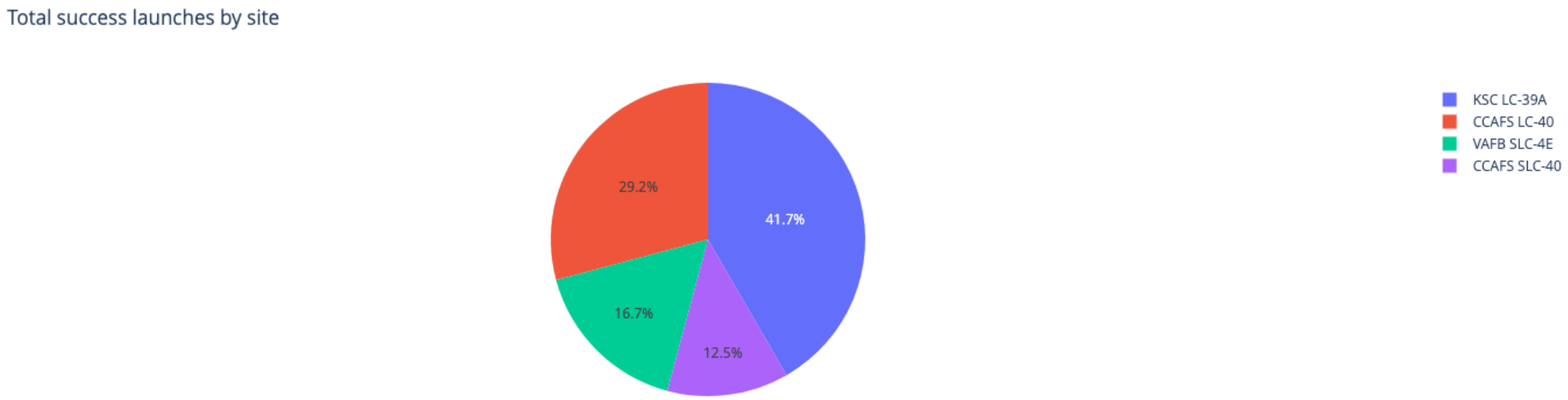
Distance to Proximities

- Distance to closest coastline: 1.04 KM
- Distance to closest city: 23.10 KM
- Distance to closest highway: 20.71 KM
- Distance to closest railway: 1.11 KM



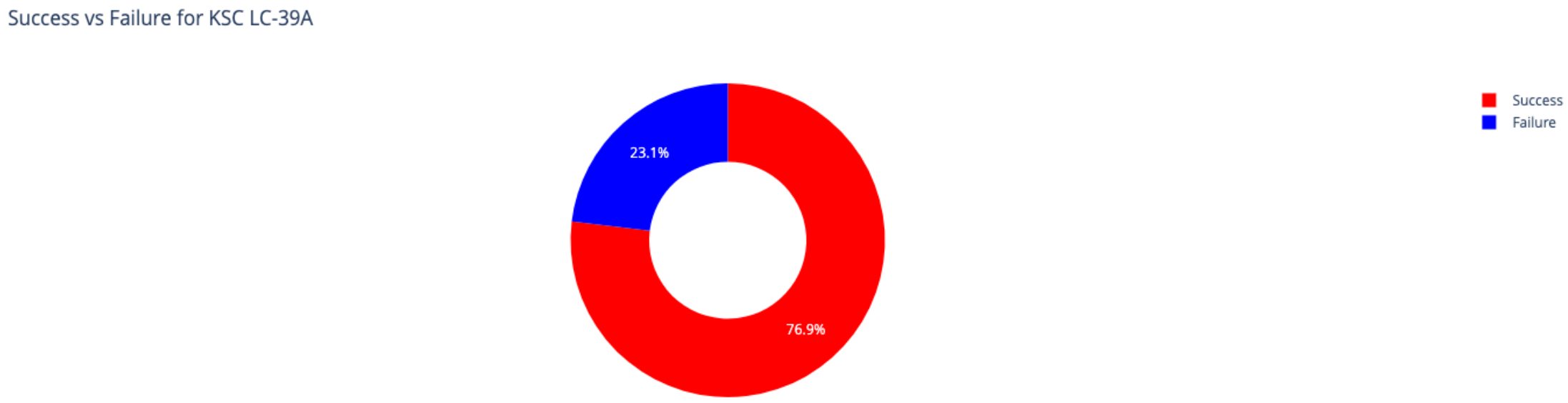

Section 4
Build a Dashboard with Plotly Dash

Launch Success by Site



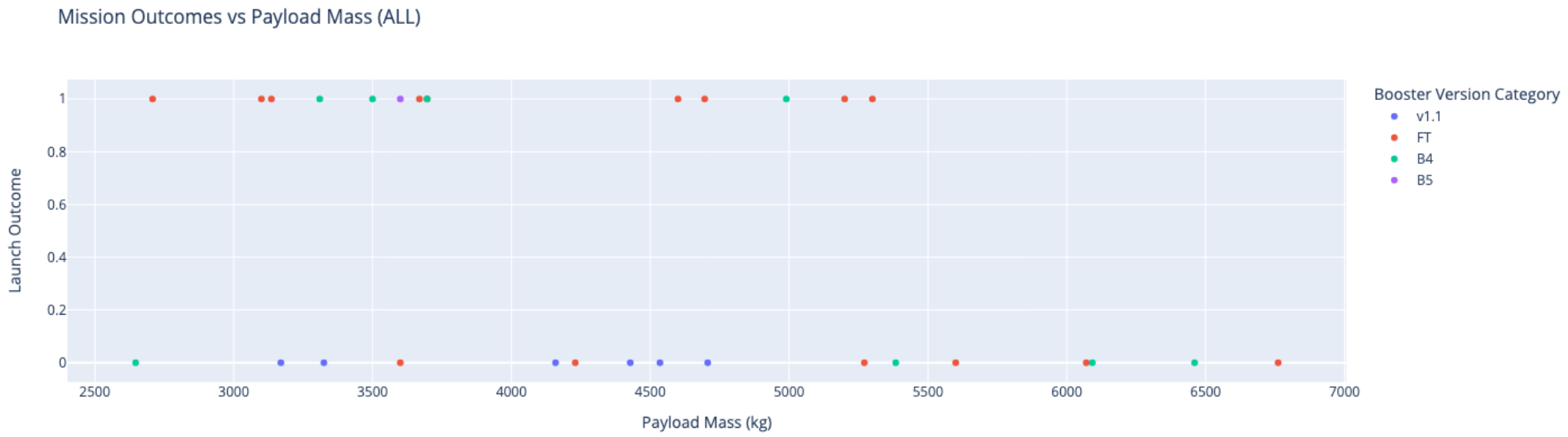
KSC LC-39A has the highest success rate (41.7%) among all launch sites

Success Ratio of KSC LC-39A



KSC LC-39A has a success ratio of 76.9%

Payload Mass vs. Launch Outcome



Payloads with lower weight (2000 kg to 5000 kg) has a higher success rate compared to heavier payloads.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

According to this analysis, **Decision Tree Classifier** is the best model (with a score of 0.88571) for prediction

```
models = {'Logistic Regression': logreg_cv.best_score_,
              'Support Vector Machine': svm_cv.best_score_,
              'Decision Tree Classifier': tree_cv.best_score_,
              'K-Nearest Neighbors': knn_cv.best_score_}

# Find the method with the highest accuracy
best_method = max(models, key=models.get)

#print which is the best model with relevant score
print('Best model is', best_method, 'with a score of', models[best_method])

if best_method == 'Logistic Regression':
    print('Best param is :', logreg_cv.best_params_)
if best_method == 'Support Vector Machine':
    print('Best param is :', svm_cv.best_params_)
if best_method == 'Decision Tree Classifier':
    print('Best param is :', tree_cv.best_params_)
if best_method == 'K-Nearest Neighbors':
    print('Best param is :', knn_cv.best_params_)
```

```
Best model is Decision Tree Classifier with a score of 0.8857142857142858
Best param is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_
split': 10, 'splitter': 'best'}
```

Confusion Matrix

Confusion Matrix Outputs:

- 10 True positive
- 1 True negative
- 5 False positive
- 2 False Negative

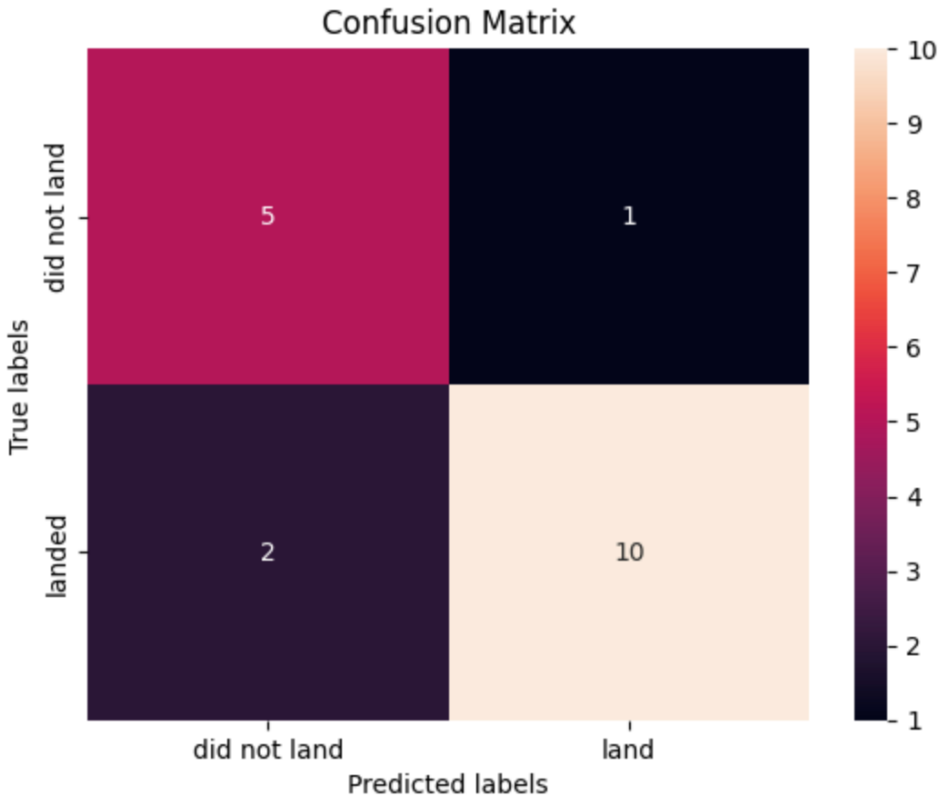
Precision = TP / (TP + FP) = 10 / 15 = 0.67

Recall = TP / (TP + FN) = 10 / 12 = 0.83

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)= 2 * (.67 * .83) / (.67 + .83) = .74

Accuracy = (TP + TN) / (TP + TN + FP + FN) = .61

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- **Model Performance**: The models exhibited similar performance on the test set, with the decision tree model showing slight superiority.
- **Coastal Location**: All launch sites are located in close proximity to coastlines, facilitating easier access and logistics for launching.
- **Launch Success Trend**: There is a noticeable increase in launch success rates since 2013.
- **KSC LC-39A**: Among all launch sites, KSC LC-39A stands out with the highest success rate. Notably, it boasts a 100% success rate for launches with payloads less than 5,500 kg.
- **Orbital Success**: Certain orbits, such as ES-L1, GEO, HEO, and SSO, have achieved a 100% success rate, indicating robust performance in reaching these specific orbital destinations.

Thank you!