IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Manel Affi

14th June 2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all result
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

Data collection methodology:

- Data was collected using SpaceX API and web scraping from Wikipedia.

Perform data wrangling

- One-hot encoding was applied 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

- How to build, tune, evaluate classification models

Data Collection

Data collection refers to the systematic process of gathering and acquiring information or data from various sources.

In this project, the data was collected using various methods

- Data collection was done using get request to the SpaceX API.

- Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().

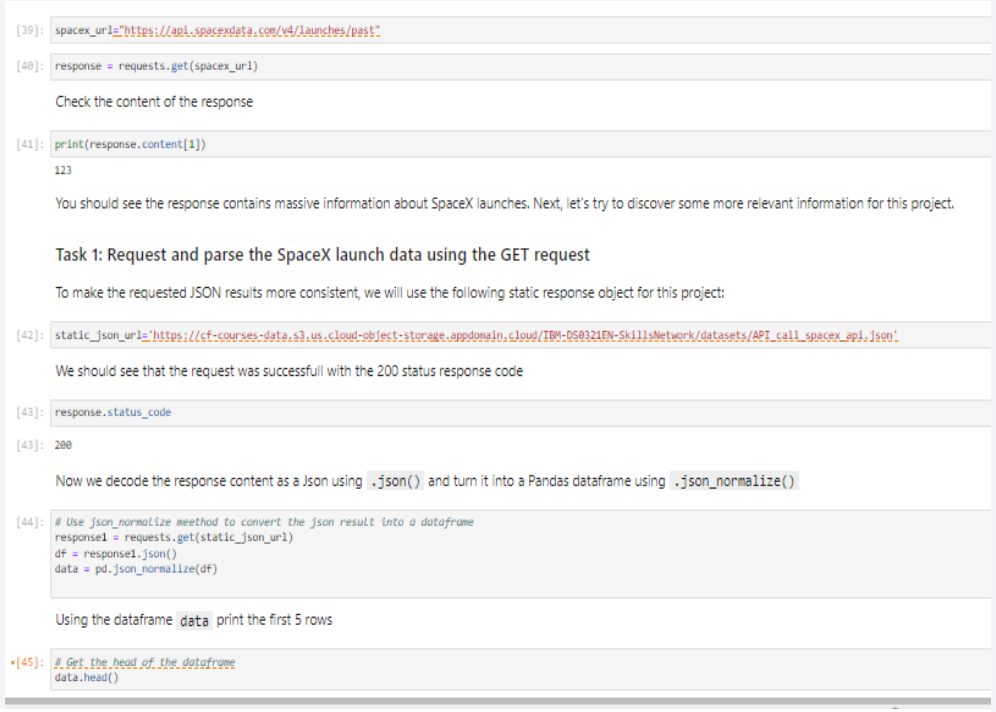
- We then cleaned the data, checked for missing values and fill in missing values where necessary.

- In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.

- The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is: https://github.com/manelAffi/IBM-Data-Science-Capstone-Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

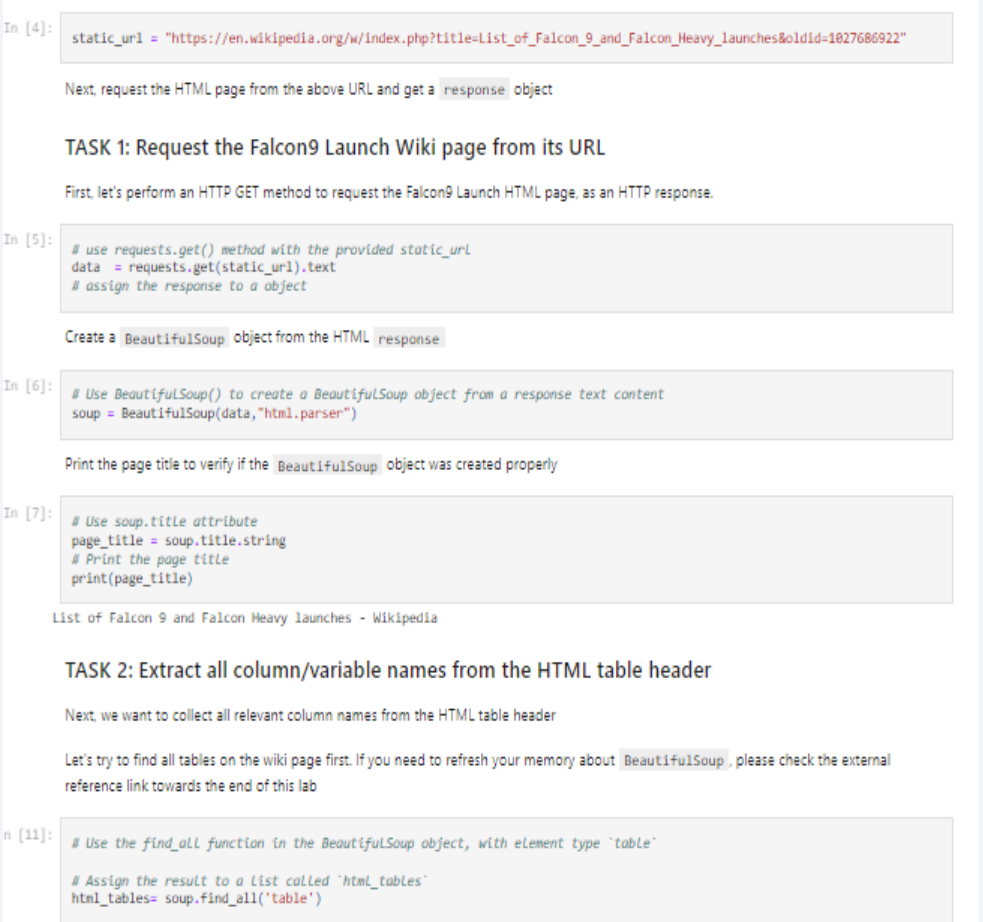


```
[39]: spacex_url="https://api.spacexdata.com/v4/launches/past"

[40]: response = requests.get(spacex_url)

Check the content of the response

[41]: print(response.content[1])

123

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

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:

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

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

[43]: response.status_code

[43]: 200

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

[44]: # Use json_normalize meethod to convert the json result into a dataframe
response1 = requests.get(static_json_url)
df = response1.json()
data = pd.json_normalize(df)

Using the dataframe data print the first 5 rows

[45]: # Get the head of the dataframe
data.head()
```

```
# Calculate the mean value of PayloadMass column
avg_PayloadMass=data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, avg_PayloadMass, inplace=True)
data_falcon9.isnull().sum()
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup.
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is: https://github.com/manelAffi/IBM-Data-Science-Capstone-Project/blob/main/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"
```

Next, request the HTML page from the above URL and get a `response` object

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 [5]:
```
# 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`

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data,"html.parser")
```

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

In [7]:
```
# Use soup.title attribute
page_title = soup.title.string
# Print the page title
print(page_title)
```

```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

n [11]:
```
# 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')
```

Data Wrangling

Data wrangling, also known as data munging or data preprocessing, refers to the process of cleaning, transforming, and preparing raw data for analysis.

- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https:// https://github.com/manelAffi/IBM-Data-Science-Capstone-Project/blob/main/labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]
#Landing_class
```

This variable will represent the classification variable that represents the land successfully; one means the first stage landed Successfully

```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

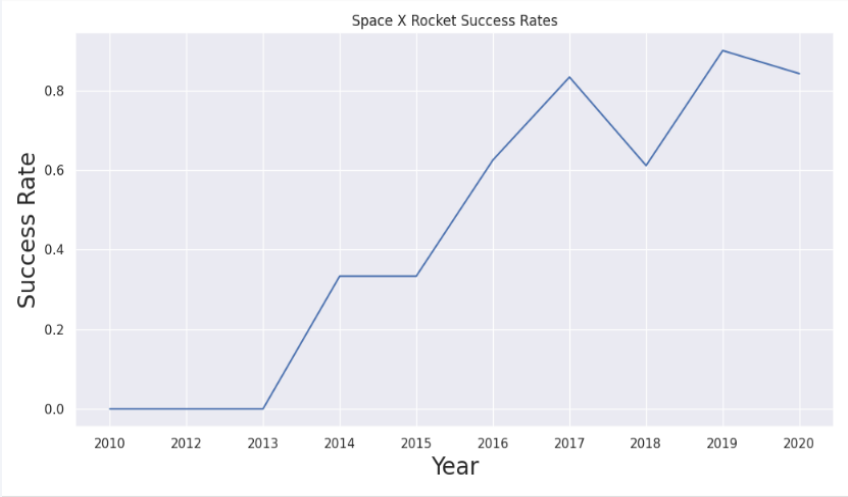
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

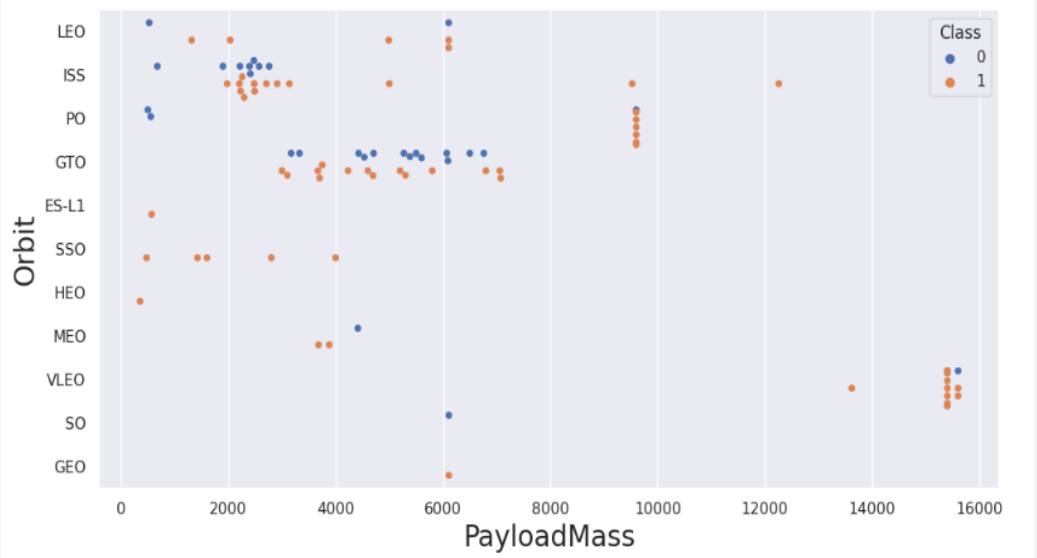
In [9]:
```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

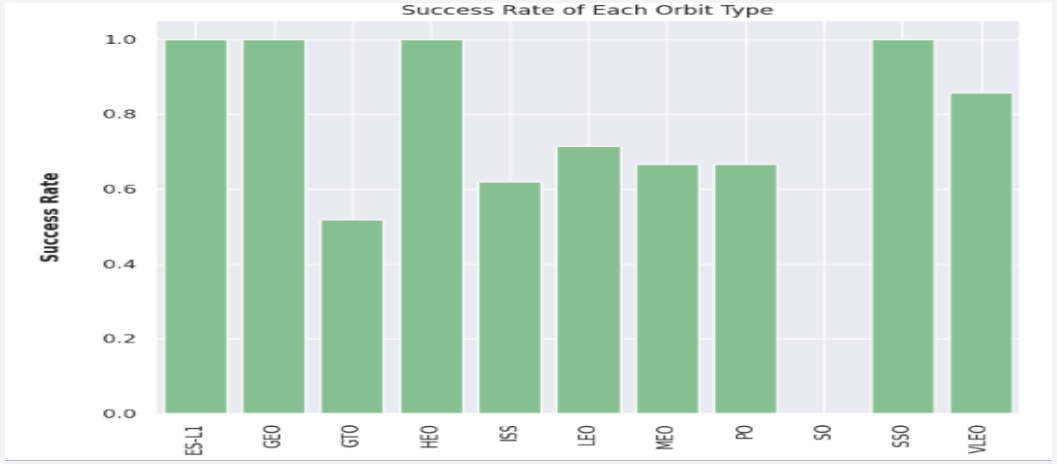
Out[9]:
```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

EDA with Data Visualization

We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, Payload mass and orbit type, the launch success yearly







The link to the notebook is https://github.com/manelAffi/IBM-Data-Science-Capstone-Project/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.

The link to the notebook is https://github.com/manelAffi/IBM-Data-Science-Capstone-Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We added markers, circles, and lines to the folium map to represent all the launch sites and indicate the success or failure of launches for each site.
- By assigning the launch outcomes (success or failure) to class 0 and 1, we were able to color-code the markers on the map accordingly.
- Using the color-labeled marker clusters, we were able to identify launch sites that have a relatively high success rate.
- We calculated the distances between each launch site and its neighboring sites, allowing us to answer questions such as the proximity of different launch sites to each other.

-Are launch sites near railways, highways and coastlines.

-Do launch sites keep certain distance away from cities.

The link to the notebook is https://github.com/manelAffi/IBM-Data-Science-Capstone-Project/blob/main/Foluim_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- We have developed an interactive dashboard using Plotly Dash, where we incorporated various visualizations:
- The dashboard includes pie charts that showcase the total launches for specific sites.
- Additionally, a scatter plot was implemented to depict the relationship between launch outcomes and payload mass (in kilograms) across different booster versions.
- The link to the notebook is https://github.com/manelAffi/IBM-Data-Science-Capstone-Project/blob/main/spacex_dash_app.py

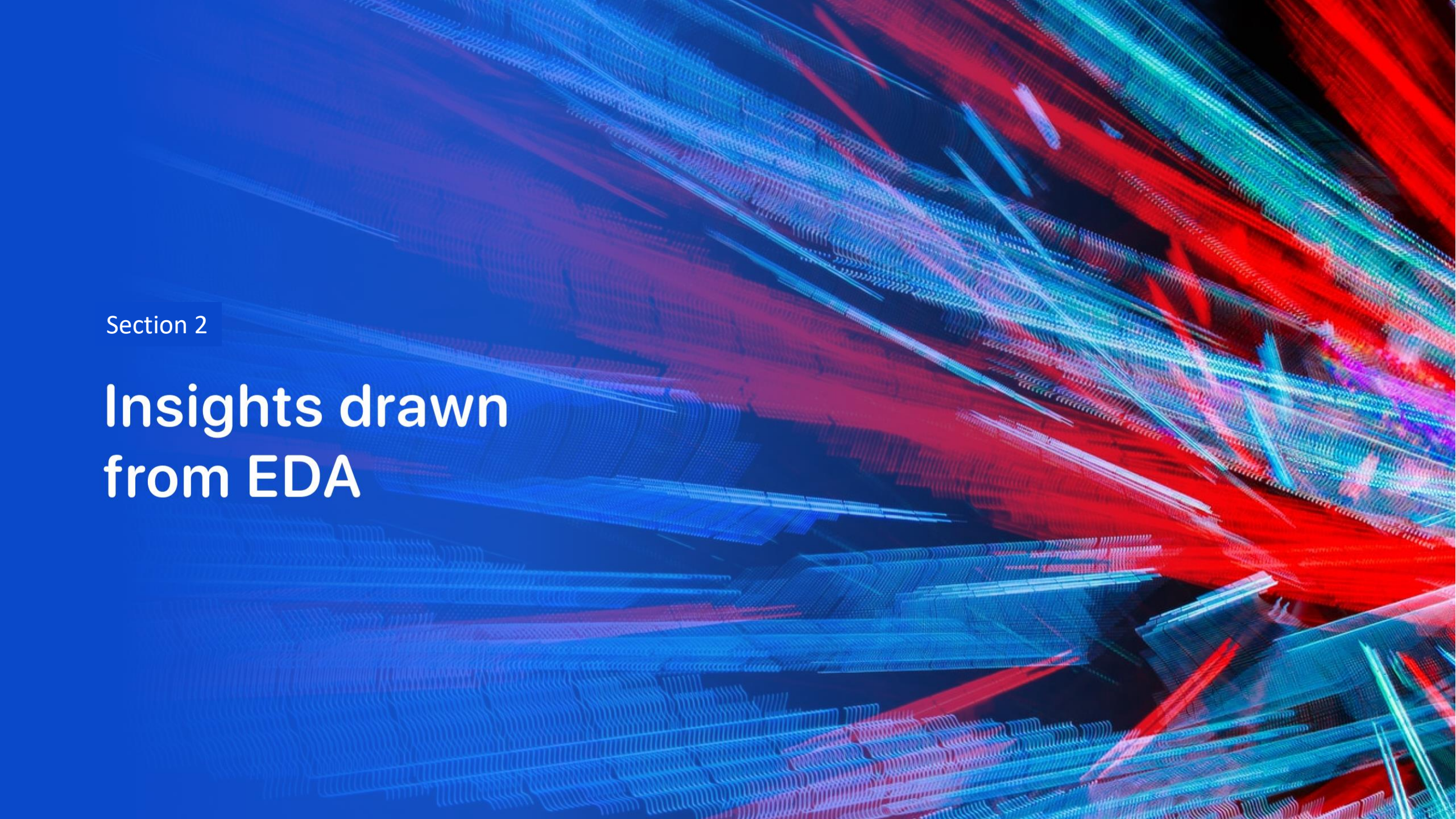
Predictive Analysis (Classification)

1. Data loading and preprocessing: We utilized numpy and pandas libraries to load the data and apply necessary transformations.
2. Train-test split: The dataset was divided into training and testing sets.
3. Model construction: We developed multiple machine learning models.
4. Hyperparameter tuning: GridSearchCV was employed to optimize the hyperparameters of the models.
5. Performance evaluation: Accuracy was used as the evaluation metric for our models.
6. Feature engineering: We enhanced the model's performance by employing feature engineering techniques.
7. Algorithm tuning: Additional adjustments were made to the algorithms' parameters.
8. Identification of the best model: We determined the best performing classification model based on the accuracy metric.

- The link to the notebook is https://github.com/manelAffi/IBM-Data-Science-Capstone-Project/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

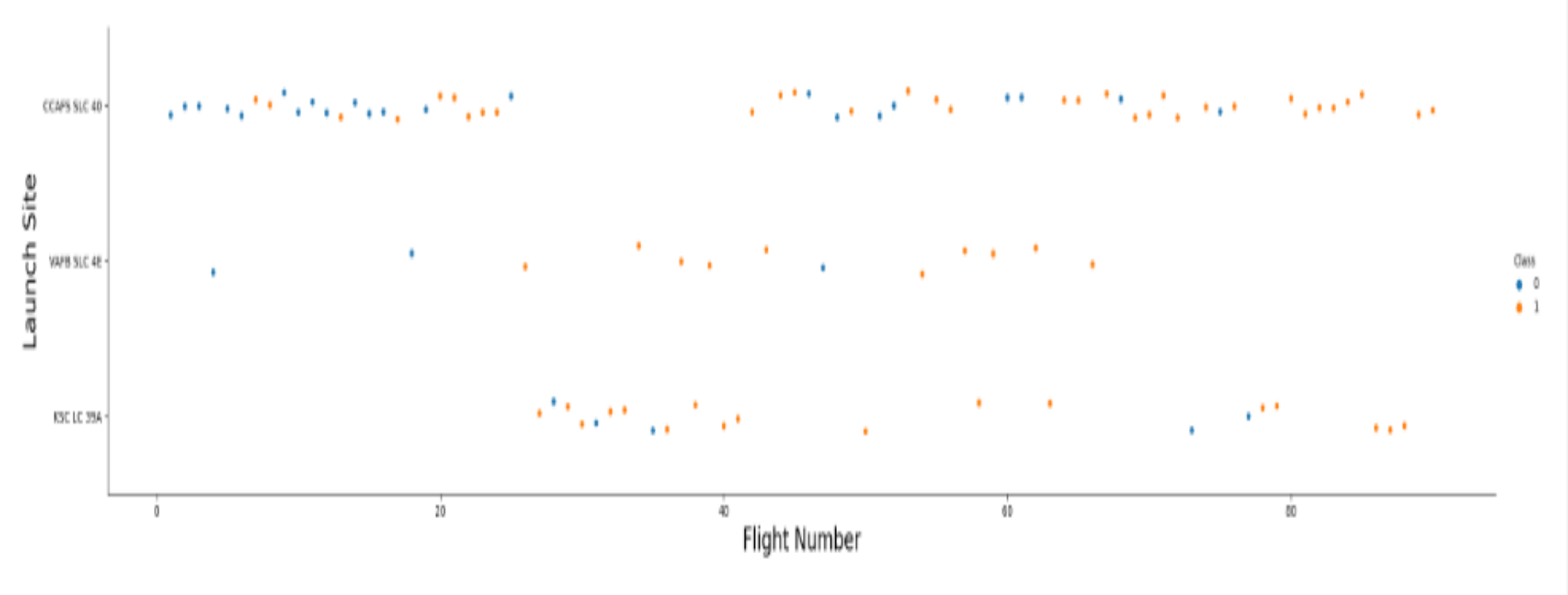
Results

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


Section 2
Insights drawn from EDA

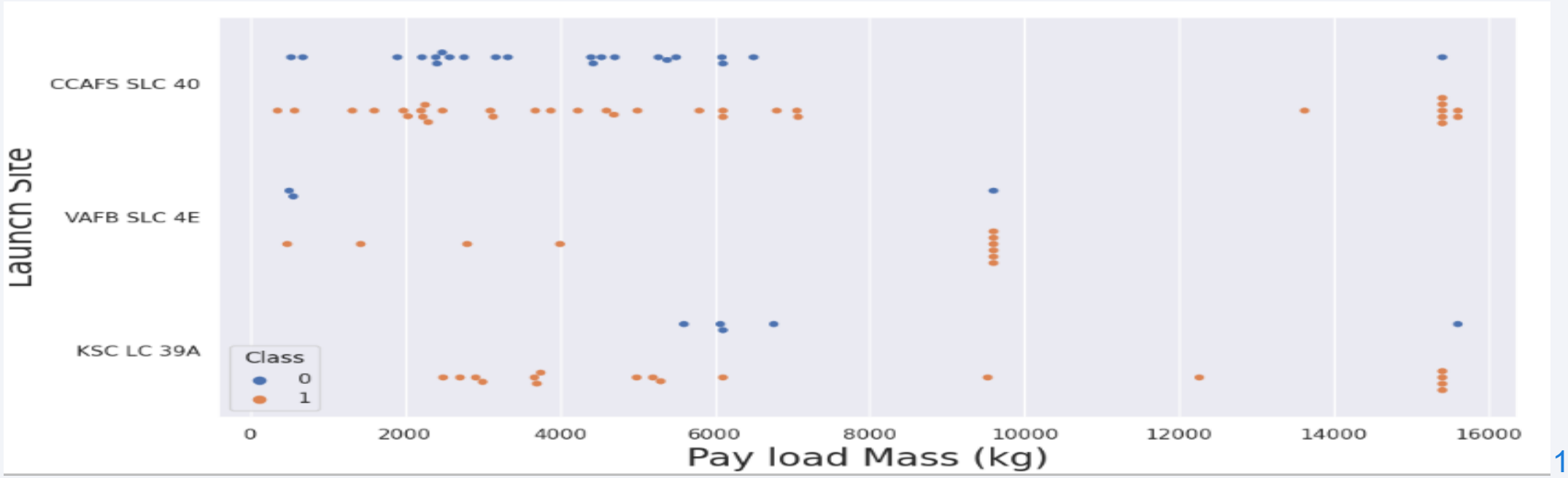
Flight Number vs. Launch Site

Based on the plot, we observed an increasing success rate at a launch site starting from flight number 40.

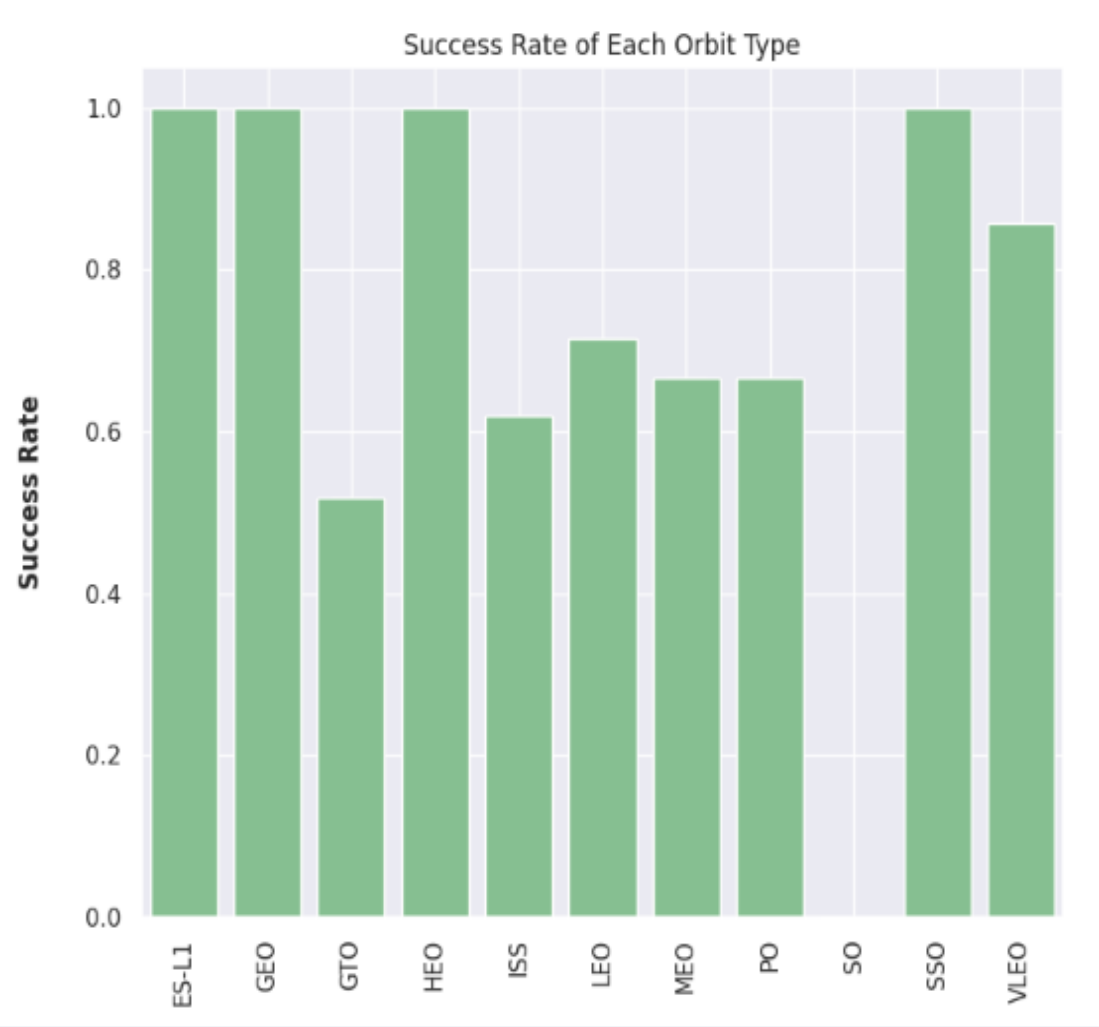


Payload vs. Launch Site

Based on the plot analysis, it is evident that no rockets were launched with a heavy payload mass (greater than 10000) at the VAFB-SLC launch site. However, for CCAFS LC-40 and KSC LC-39A, the success rate demonstrates an upward trend when the payload mass exceeds 10000.



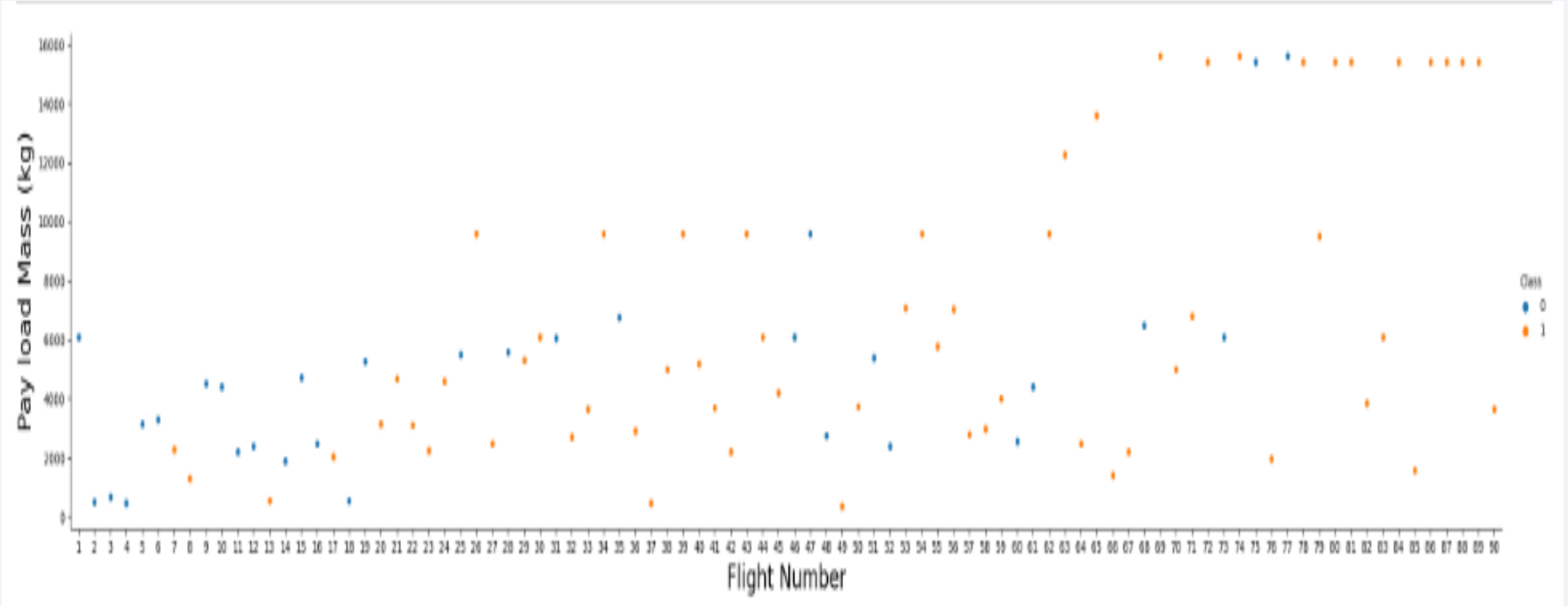
Success Rate vs. Orbit Type



Based on the plot analysis, it is clear that ES-L1, GEO, HEO, and SSO orbits exhibit optimal success rates, while the success rate for SO orbit is zero.

FlightNumber vs. PayloadMass

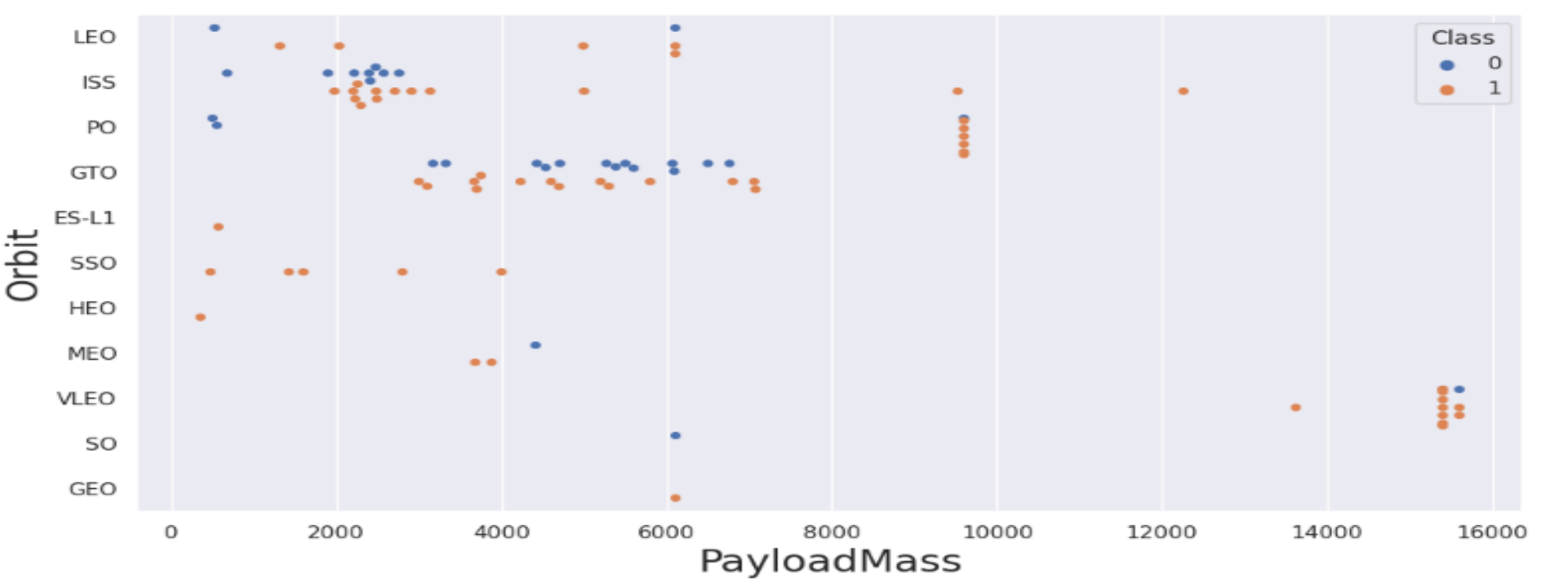
Based on the plot, we observed a positive correlation between a payload mass exceeding 7000 and an increasing success rate.



Flight Number vs. Orbit Type

The plot provided illustrates the relationship between Flight Number and Orbit type. It can be observed that in the LEO orbit, the success rate is influenced by the number of flights. However, in the GTO orbit, there appears to be no correlation between the flight number and the orbit's success rate.

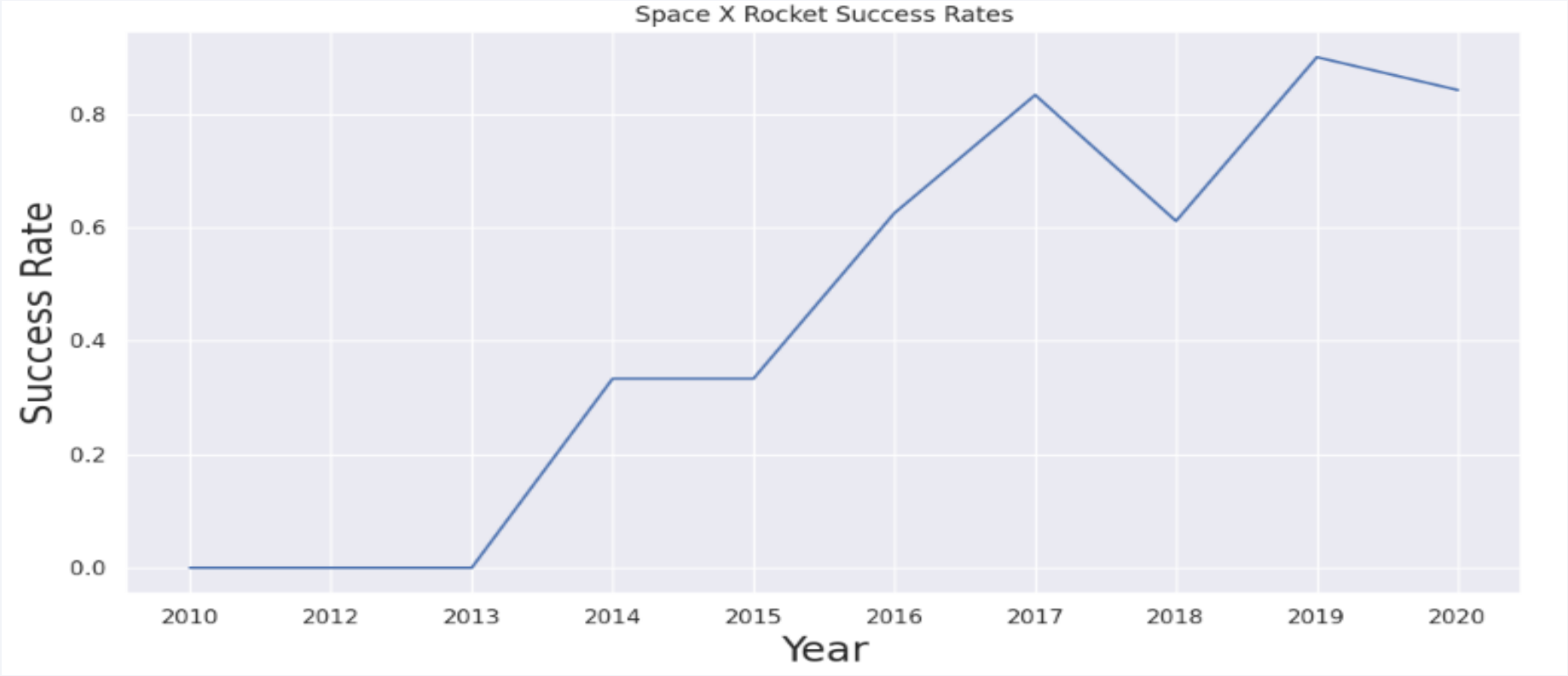
Payload vs. Orbit Type



From the observations, it can be concluded that heavy payloads tend to have a higher success rate for landing in the PO, LEO and ISS orbits.

Launch Success Yearly Trend

Based on the plot, it is evident that the success rate has consistently increased from 2013 until 2020.



All Launch Site Names

- Description:

We used the key word DISTINCT to show only unique launch sites from the SpaceX data.

- SQL Query

```
%%sql SELECT DISTINCT "Launch_site"
FROM SPACEXTBL;
```

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

Launch Site Names Begin with 'CCA'

- Description:

We executed the aforementioned query to retrieve a set of five records where the launch sites start with the prefix "CCA".

- SQL Query

```
%%sql SELECT *
FROM  SPACEXTBL
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Description:

We calculated the total payload carried by boosters from NASA as 45596.

- SQL Query

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

total_payload_mass
45596.0

Average Payload Mass by F9 v1.1

- Description:

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

- SQL Query

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

average_payload_mass
2928.4

First Successful Ground Landing Date

- Description:

We observed that the dates of the first successful landing outcome on ground pad was 01/08/2018

- SQL Query

In [22]:
```
%%sql
SELECT MIN(date) AS first_successful_landing_date
FROM SPACEXTBL
WHERE landing_outcome = 'Success (ground pad)';
```

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

Out[22]:

first_successful_landing_date
01/08/2018

Successful Drone Ship Landing with Payload between 4000 and 6000

- Description:

Successful Drone Ship Landing with Payload between 4000 and 6000.

- SQL Query

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

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

Total Number of Successful and Failure Mission Outcomes

- Description:

Total Number of Successful and Failure Mission Outcomes

- SQL Query

```
%%sql SELECT COUNT(Mission_Outcome) AS SuccessOutcome
        FROM SPACEXTBL
        WHERE Mission_Outcome LIKE 'Success%'
```

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

SuccessOutcome
100

```
%%sql SELECT COUNT(Mission_Outcome) AS FailureOutcome
        FROM SPACEXTBL
        WHERE Mission_Outcome LIKE 'Failure%'
```

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

FailureOutcome
1

Boosters Carried Maximum Payload

- Description:
- Boosters Carried Maximum Payload.
- Using a sub-query within the WHERE clause and the MAX() function, we were able to identify the booster that has carried the highest payload.

- SQL Query

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

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

- Description:

- 2015 Launch Records
- By utilizing a combination of the WHERE clause, LIKE, AND, and BETWEEN conditions, we applied filtering to extract failed landing outcomes in drone ships, corresponding booster versions, and launch site names specifically for the year 2015.

- SQL Query

```
%%sql  SELECT Booster_Version, Launch_Site, Landing_Outcome
        FROM SPACEXTBL
        WHERE Landing_Outcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'

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

Booster_Version	Launch_Site	Landing_Outcome

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

- Description:

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

- SQL Query

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

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

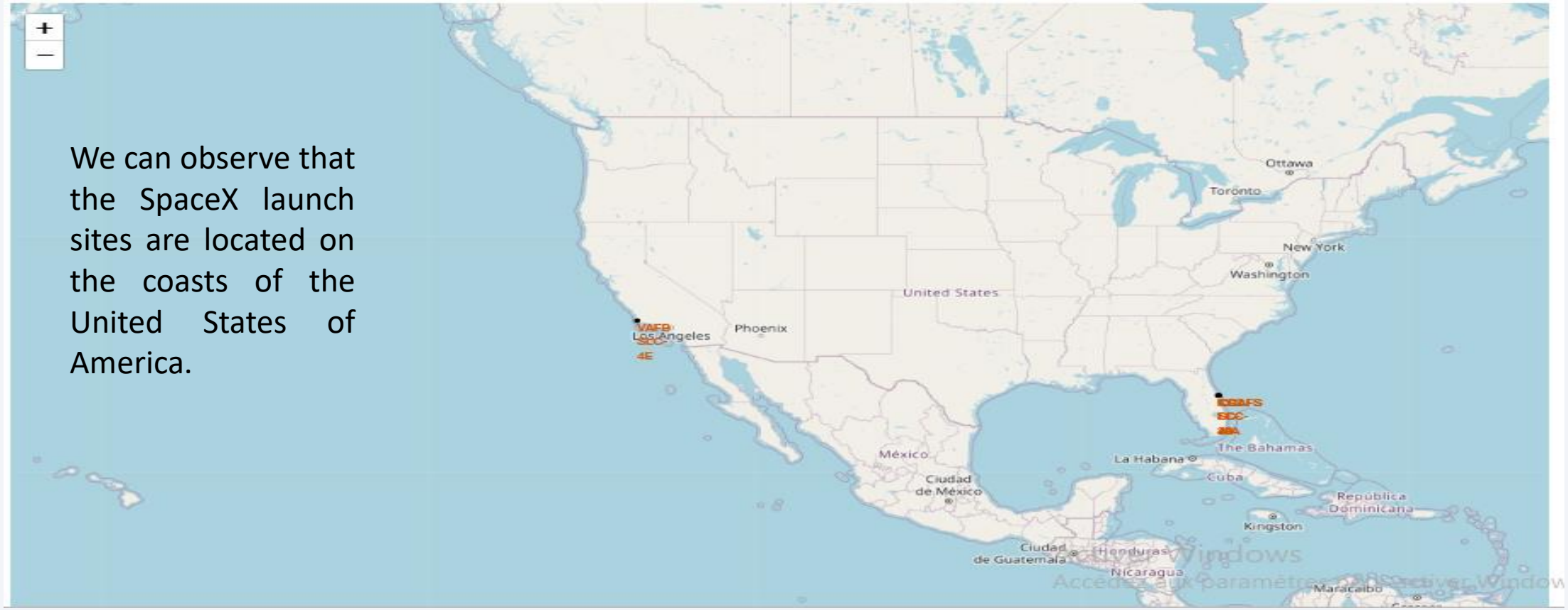
Landing_Outcome	count_successful_outcomes

Section 3

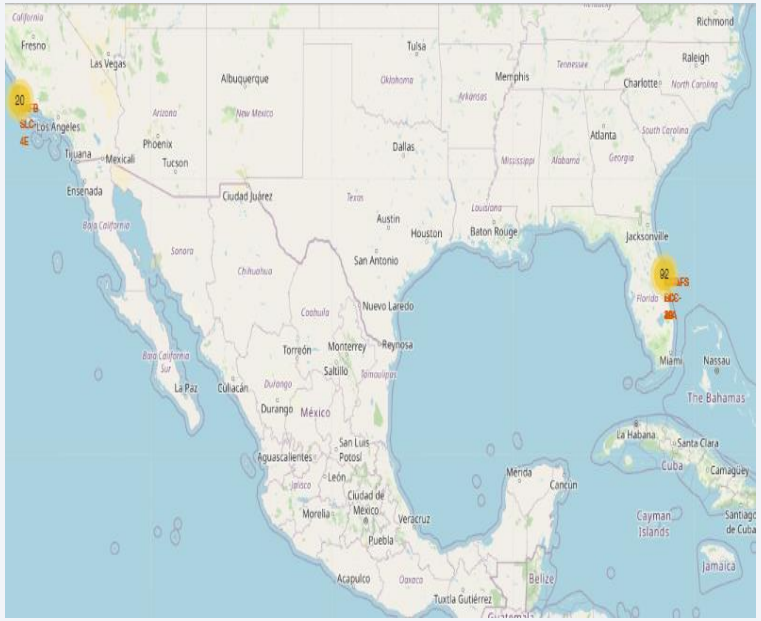
Launch Sites Proximities Analysis

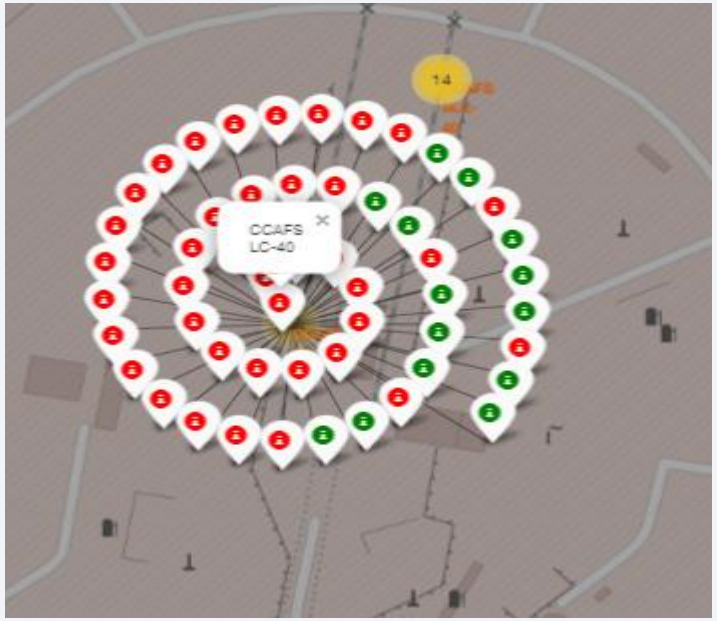
All launch sites global map markers

We can observe that the SpaceX launch sites are located on the coasts of the United States of America.



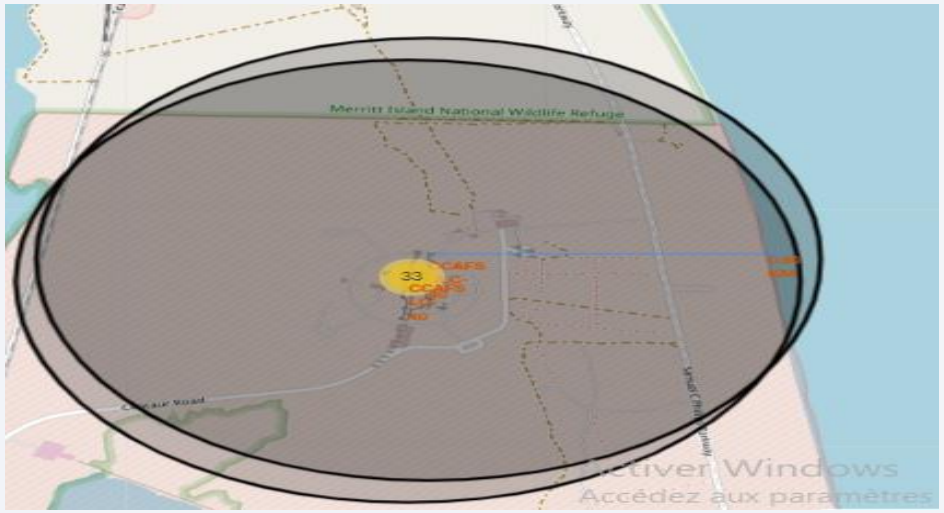
Markers showing launch sites with color labels



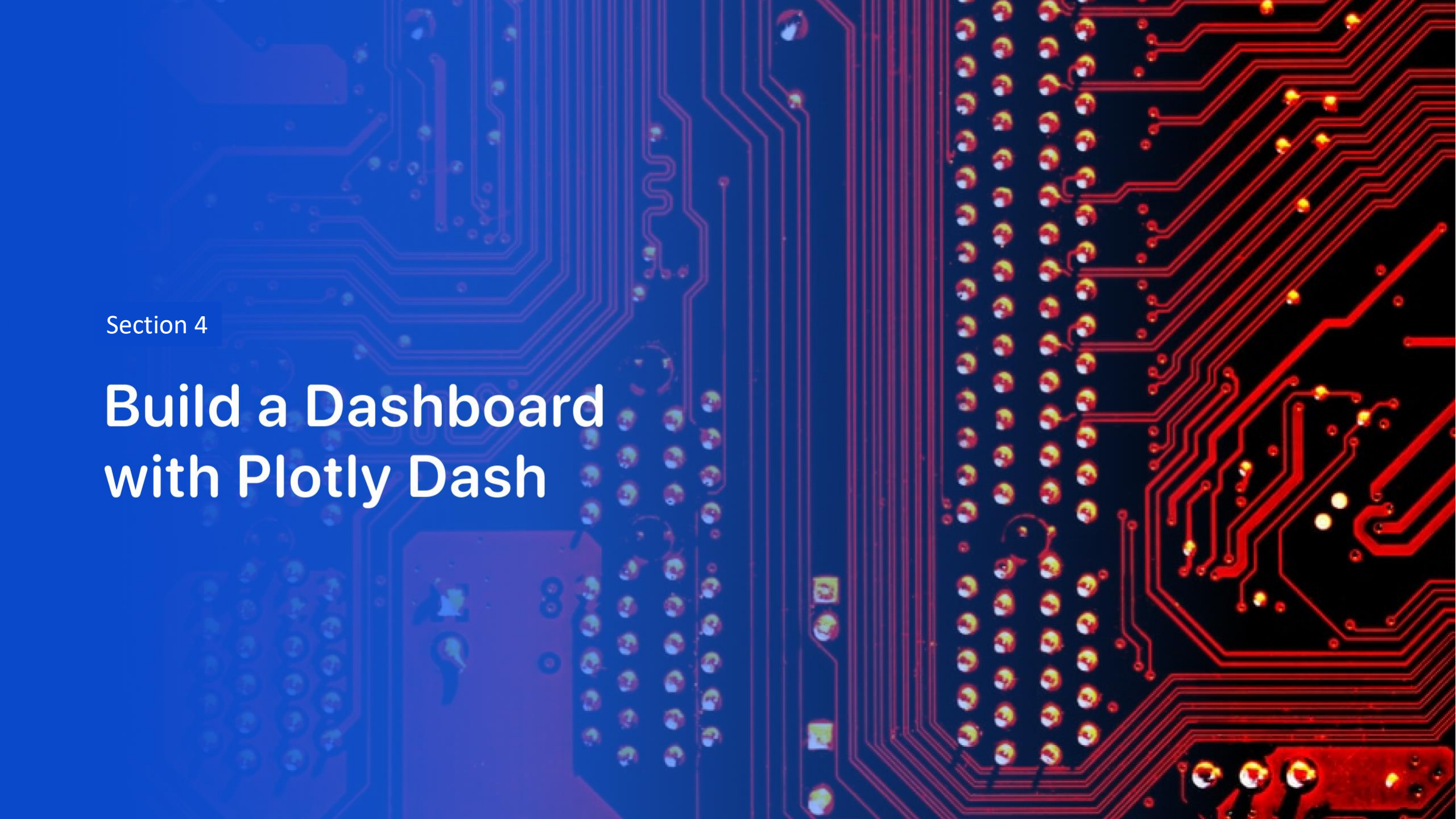


In this Folium map, successful launches are represented by green markers, while failures are indicated by red markers.

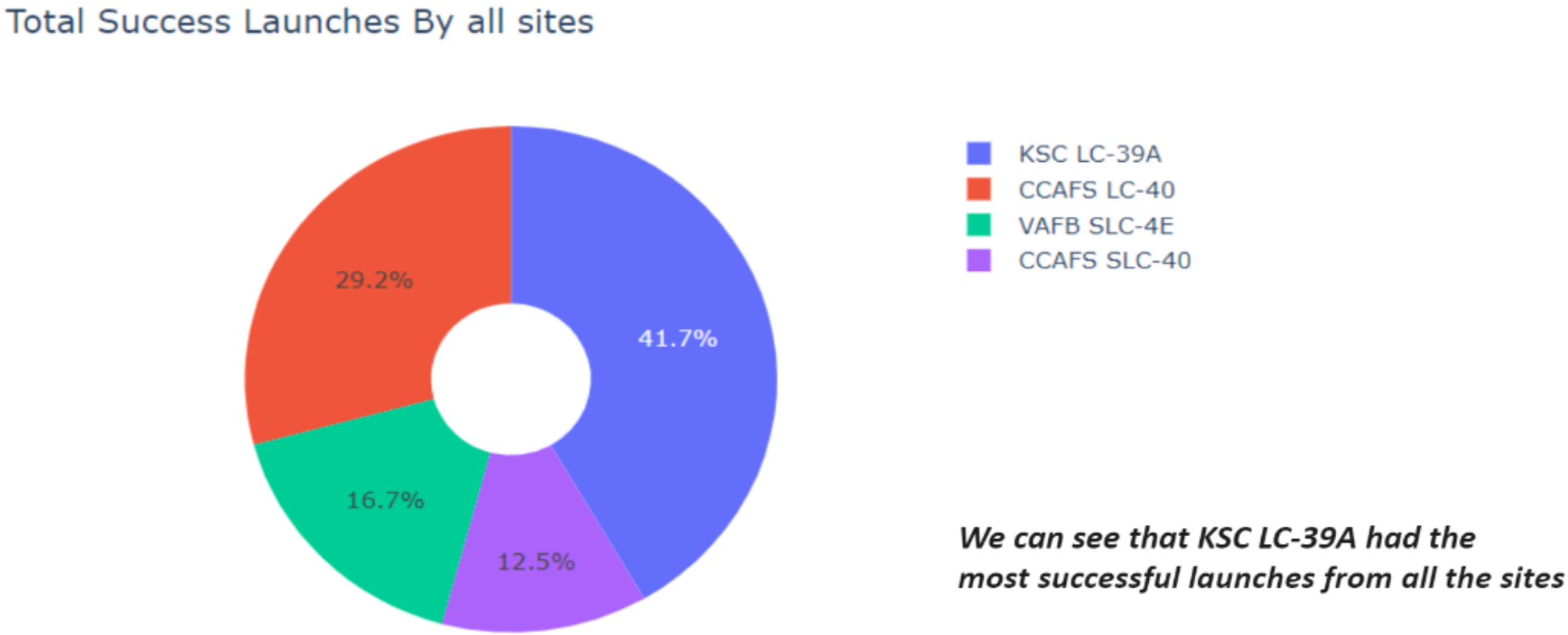
Launch Site distance to landmarks



- Launch sites are not situated in close proximity to railways.
- Launch sites are not situated in close proximity to highways.
- Launch sites are in close proximity to coastlines.
- Launch sites are deliberately located at a certain distance away from cities.

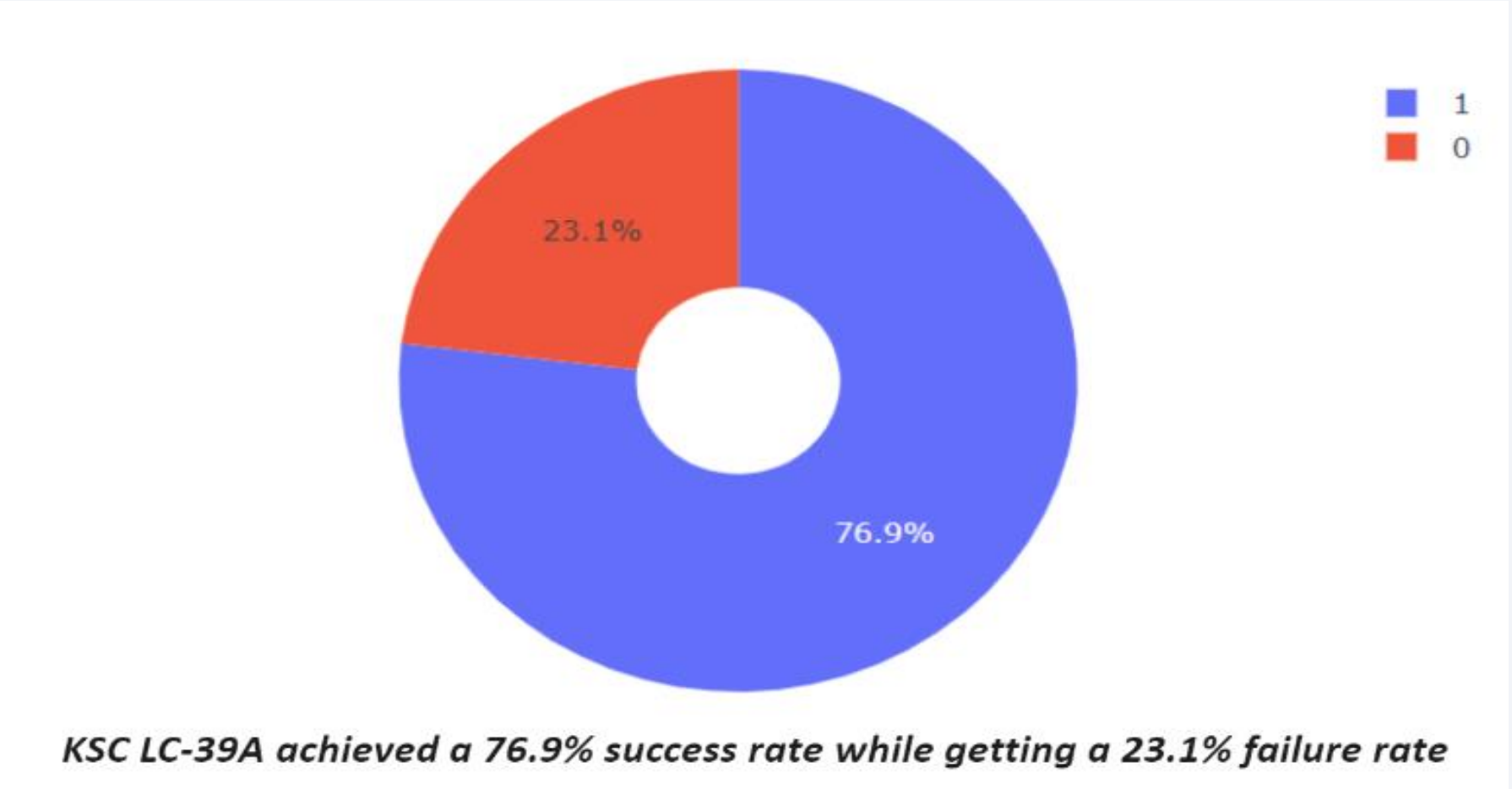

Section 4
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by all launch sites



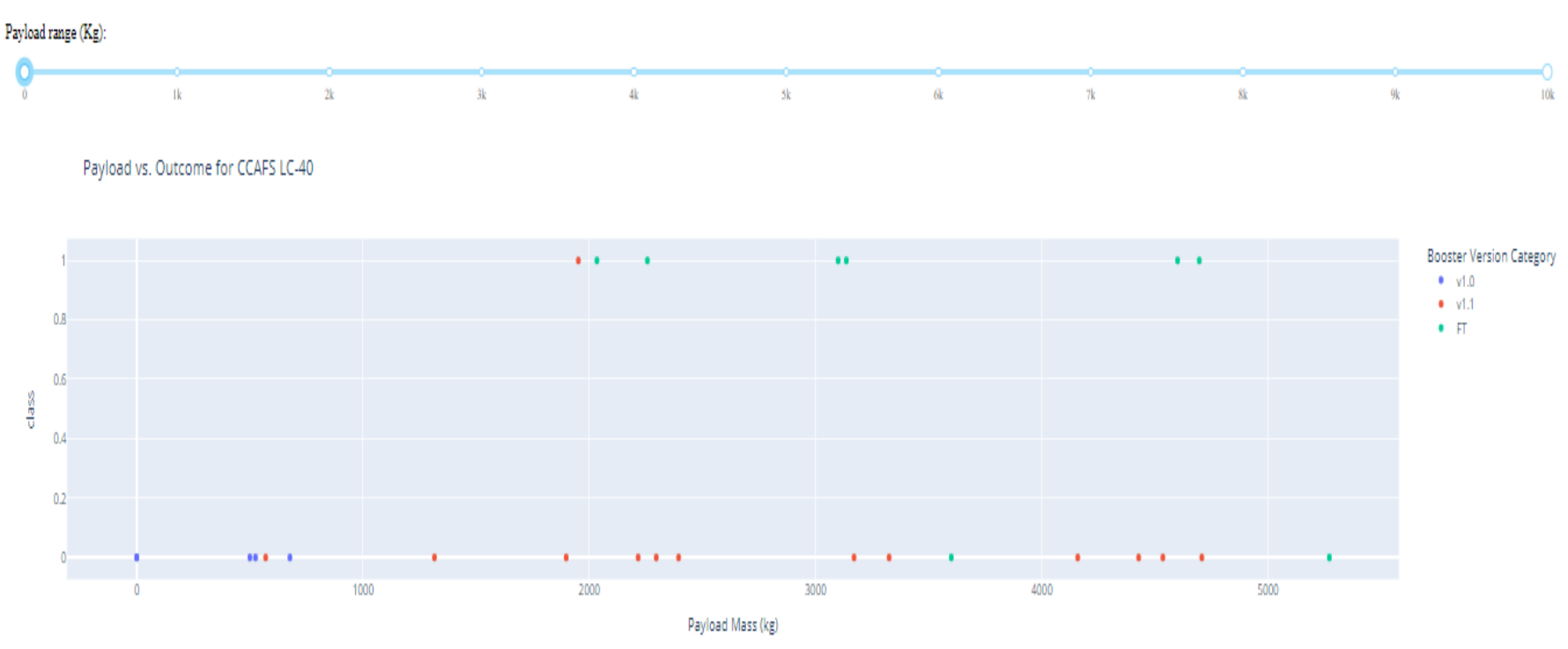
We can see that KSC LC-39A had the most successful launches from all the sites

Pie chart showing the Launch site with the highest launch success ratio



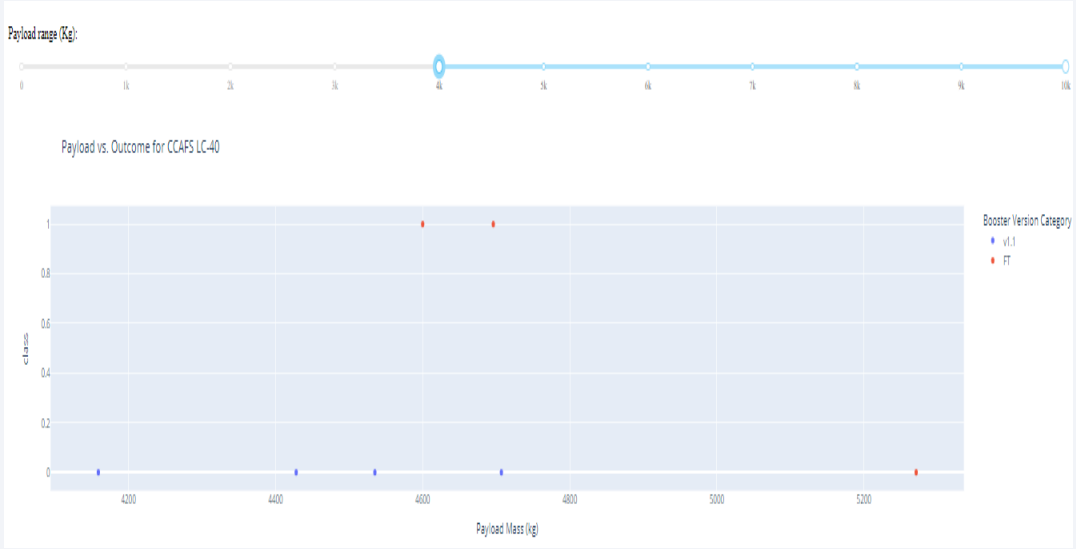
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

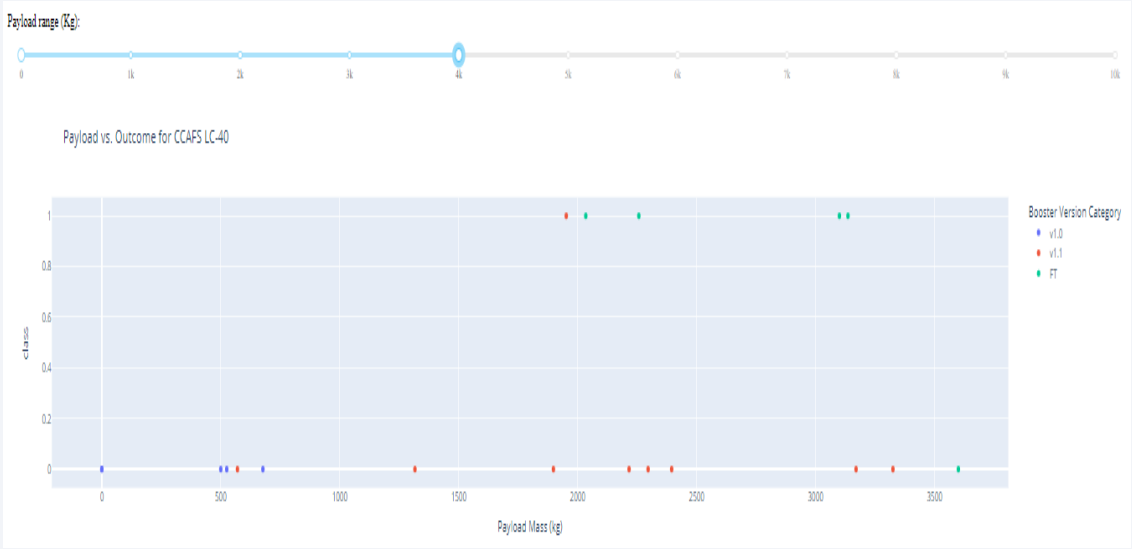
Scatter plot of Payload vs Launch Outcome for CCAFS LC-40 site



For payload masses between 2000 and 5000, the observation reveals that the FT booster version demonstrates the highest success rate, whereas the v1.1 and v1.0 booster version exhibit the lowest success rate.

Scatter plot of Payload vs Launch Outcome for CCAFS LC-40 site, with different payload selected in the range slider





The success rates of low-weighted payloads are observed to be higher compared to heavy-weighted payloads.

Section 5

Predictive Analysis (Classification)

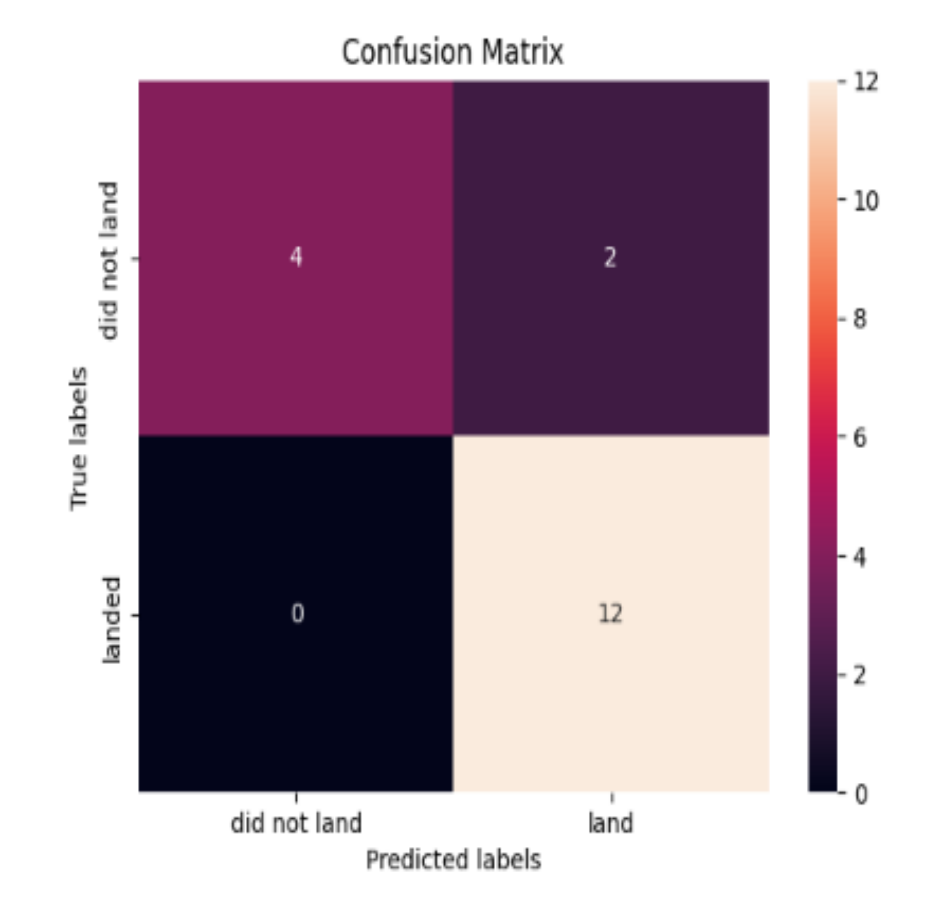
Classification Accuracy

```
models = {'DecisionTree':tree_cv_.best_score_,
          'KNeighbors':knn_cv.best_score_,

              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv_.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.875
Best params is : {'criterion': 'gini', 'max_depth': 14, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 5, 'splitter': 'random'}
```


The decision tree classifier is the model with the highest classification accuracy.

Confusion Matrix



The confusion matrix for the decision tree classifier demonstrates its ability to differentiate between the various classes. However, a significant issue arises in the form of false positives, where the classifier wrongly identifies unsuccessful landings as successful ones.

Conclusions

Based on our analysis, we can draw the following conclusions:

- There is a positive correlation between the number of flights at a launch site and its success rate.
- The launch success rate has shown a consistent upward trend from 2013 to 2020.
- Orbits such as ES-L1, GEO, HEO, SSO, and VLEO have demonstrated the highest success rates.
- Among all the launch sites, KSC LC-39A stands out with the highest number of successful launches.
- After evaluating multiple machine learning algorithms, the Decision tree classifier has proven to be the most effective for this task.

Thank you!