IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Anindya Ghosh
January 7, 2025

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection through API
  - Data collection through Web Scrapping
  - Data Wrangling
  - Data analysis with SQL
  - Data analysis with data visualisation
  - Interactive visualization using Folium
  - Machine learning predictions
- Summary of all results
  - Data analysis results
  - Interactive analysis
  - Machine learning prediction results

Introduction

- Project background and context

The Falcon 9 rocket consists of two stages, with the first stage doing most of the work and being significantly larger and more expensive. SpaceX's ability to reuse the first stage contributes to lower launch costs, with Falcon 9 launches priced at $62 million compared to competitors' costs of over $165 million. Hence determining whether we can reuse the first stage will give us help us calculate the total cost for a launch. The rival company Space Y wants to build a ML pipeline to predict whether the first stage can land successfully.

- Problems you want to find answers

a) What factors determine the successful landing of the first stage

b) Ideal conditions necessary for a successful landing

c) Create an interactive viewing platform with different features to visualize the landing outcomes

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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

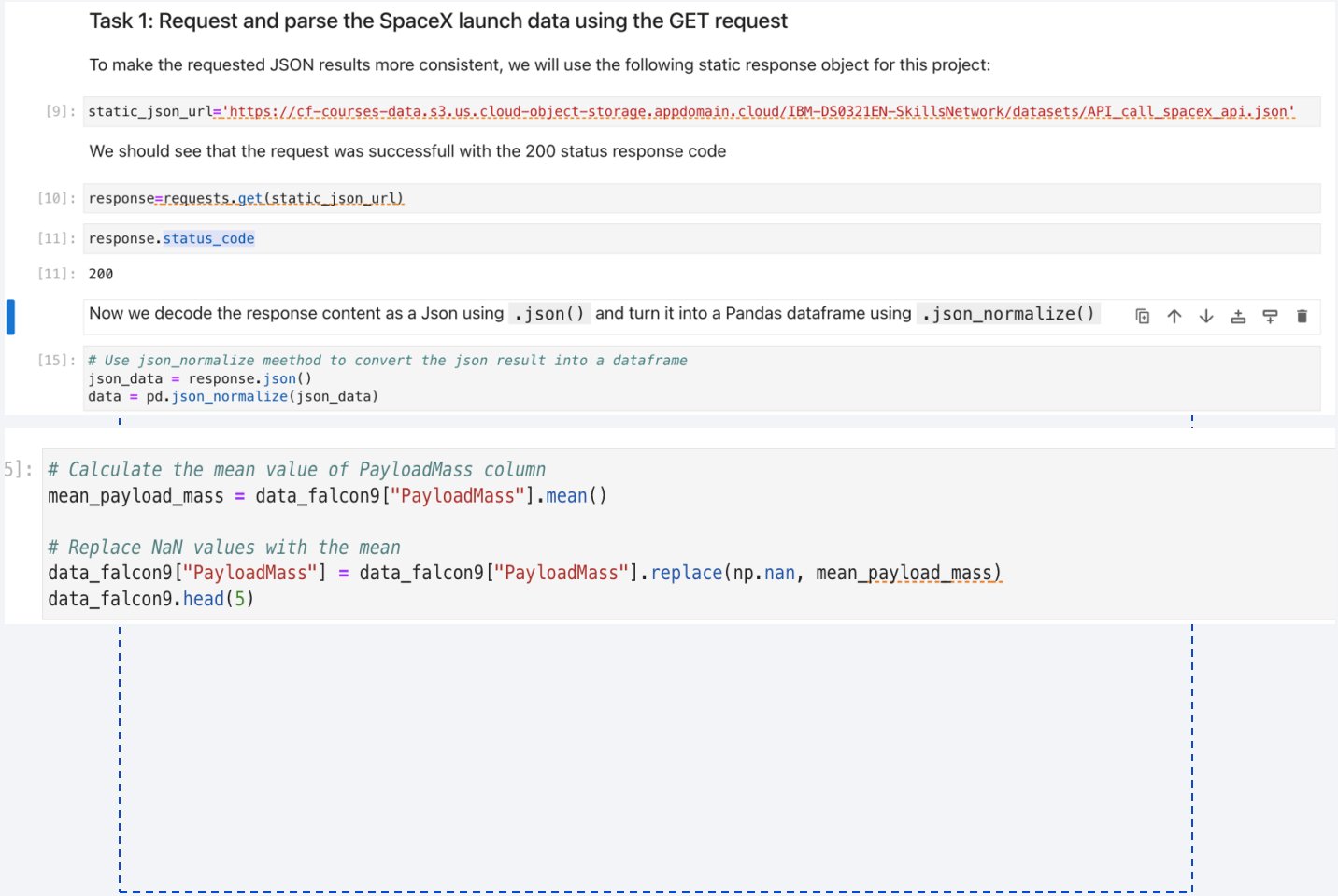
Data Collection

- Describe how data sets were collected.
  - Using the SpaceX API
  - The response is decoded using .json() function and it was normalized to a pandas dataframe using the .json_normalize() function.
  - The data was processed and cleaning removing any Null and missing values.
  - We also retrieved Falcon 9 launch data from Wikipedia using web scrapping and created a BeautifulSoup object to analyze the data.
  - The launch data was extracted as HTML tables which were converted to pandas dataframe

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

- GitHub URL of the completed SpaceX API calls notebook: https://github.com/anghosh/DataSciecne/blob/main/jupyter-labs-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:

```
[9]: 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

```
[10]: response=requests.get(static_json_url)

[11]: response.status_code

[11]: 200
```

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

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

```
5]: # Calculate the mean value of PayloadMass column
mean_payload_mass = data_falcon9["PayloadMass"].mean()

# Replace NaN values with the mean
data_falcon9["PayloadMass"] = data_falcon9["PayloadMass"].replace(np.nan, mean_payload_mass)
data_falcon9.head(5)
```

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

- Add the GitHub URL: https://github.com/anghosh/DataSciecne/blob/main/jupyter-labs-webscraping.ipynb

[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.

[5]:
```
# use requests.get() method with the provided static_url
response = requests.get(static_url)

# assign the response to a object
```

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

[7]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, "html.parser")
```

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

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

[8]:
```
# Use soup.title attribute
print(soup.title.string)
```

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

[11]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
th_elements = first_launch_table.find_all("th")

# Iterate each th element and apply the provided extract_column_from_header() to get a column name

for th in th_elements:
    # Extract the column name using the provided function
    name = extract_column_from_header(th)
------
    # Append to column_names if the name is non-empty
    if name is not None and len(name) > 0:
        column_names.append(name)

# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
```

Check the extracted column names

[12]:
```
print(column_names)
```

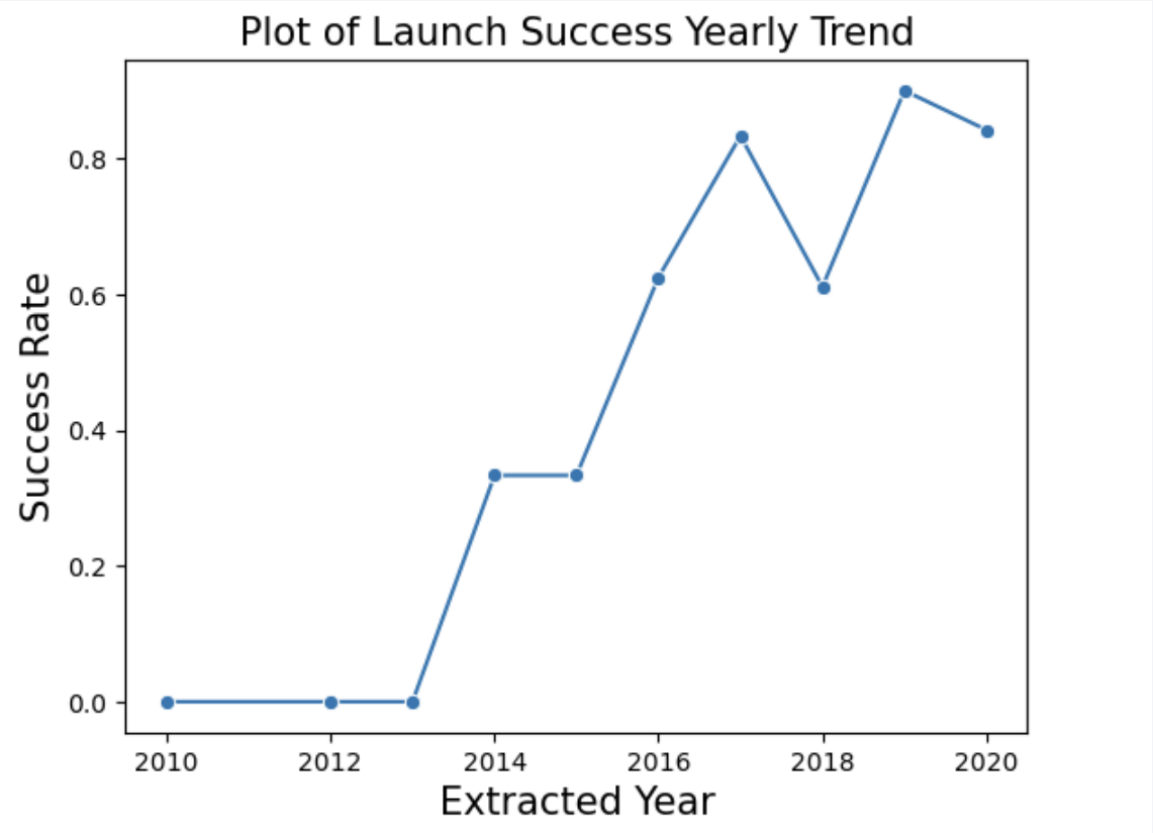
```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

Data Wrangling

- Performed exploratory data analysis and determined the training labels
- Calculated number of launches and occurrences on each site
- Calculated the number and occurrence of mission outcome of the orbits.
- Create a landing outcome label from outcome column.
- Add the GitHub URL: https://github.com/anghosh/DataSciecne/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- The data was investigated by studying the relationship between number of flights vs launch site, payload vs launch site, flight number vs orbit type, success rate of each orbit type, yearly launch success trend.
- Add the GitHub URL: https://github.com/anghosh/DataSciecne/blob/main/edadataviz.ipynb



EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
  - %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE
  - %sql SELECT *FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5
  - %sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_PayloadMass FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCAFS LC-40'
  - %sql SELECT AVG(PAYLOAD_MASS__KG_) AS Avg_PayloadMass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1'
  - %sql SELECT MIN(Date) AS FirstSuccessfull_landing_date FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success (ground pad)'
  - %sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000
  - %sql SELECT COUNT(Mission_Outcome) AS SuccessOutcome FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Success%'
  - %sql SELECT COUNT(Mission_Outcome) AS FailureOutcome FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Failure%'
  - ORDER BY Booster_Version

- Add the GitHub URL: https://github.com/anghosh/DataSciecne/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Added markers on the map pointing out launch sites using markers, circles and lines
- Marked success and failed launches for each site on the map.
- Added feature launch outcomes to point failures and success by assigning class 0 and 1 respectively.
- Colors are used to identify low or high success rates.
- Calculated launch distances from landmarks such as railway station, beach, highways and calculated the distances marked by lines.
- Add the GitHub URL: https://github.com/anghosh/DataSciecne/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

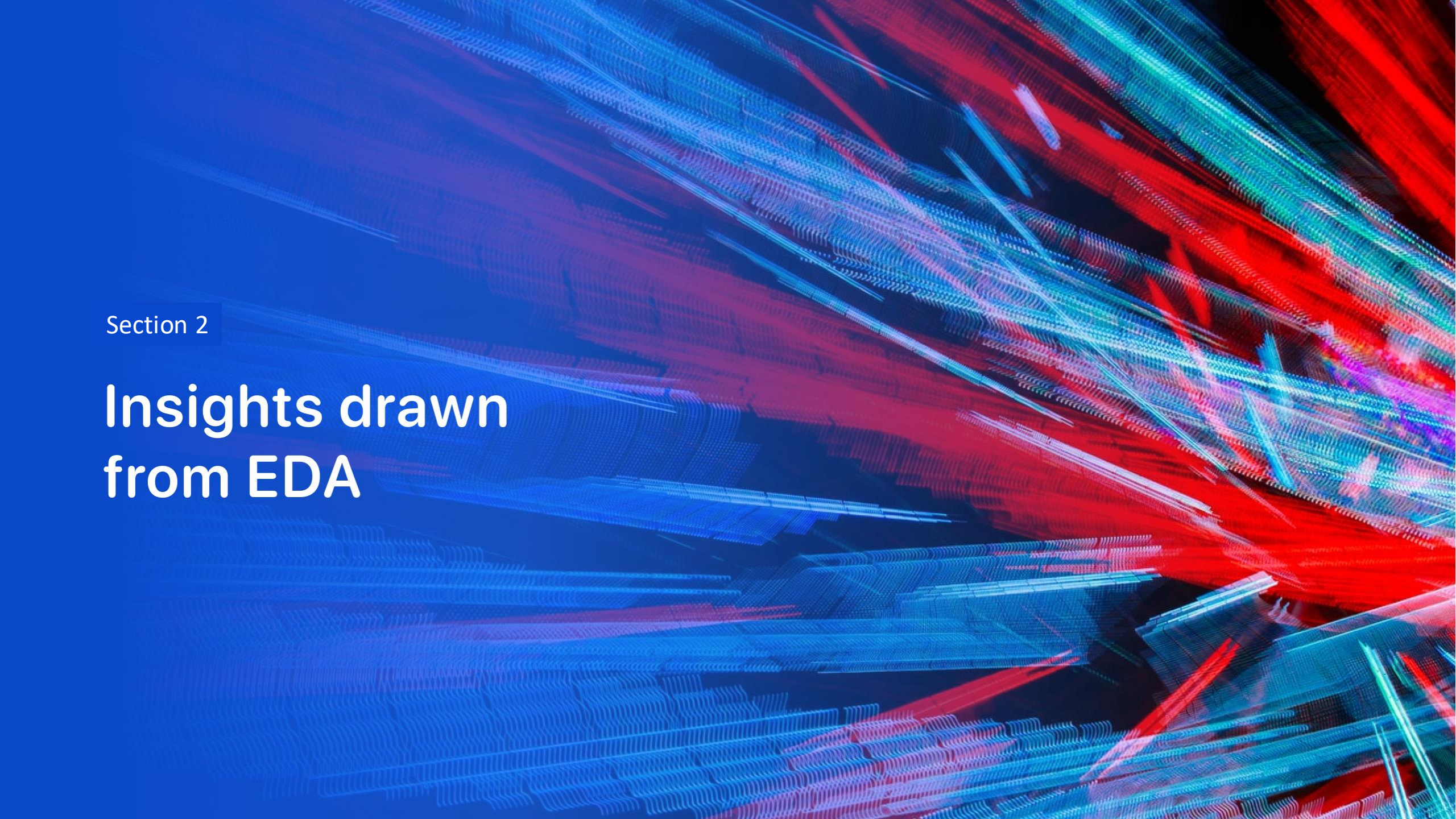
- Built an interactive Dashboard with Plotly Dash.
- Created pie charts showing launch numbers for each site
- Created a scatter plot with payload weight vs success rate for different booster versions.
- The plots and interactions are very useful to interactively play around with different variables and see how the success rate depends on them.
- Add the GitHub URL: https://github.com/anghosh/DataSciecne/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Load the data using pandas, numpy, standardize the data, split it into test and training dataset.
- Built different ML models such as Logistic Regression, SVM, Decision Tree, K-Nearest Neighbors and create a GridSearchCV object for each to find out the best parameters for a particular model.
- Calculated accuracy for each model and is used as the metric to decide which model performs best.
- Among all the tested models the best model is the one with highest accuracy.
- Add the GitHub URL: https://github.com/anghosh/DataSciecne/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

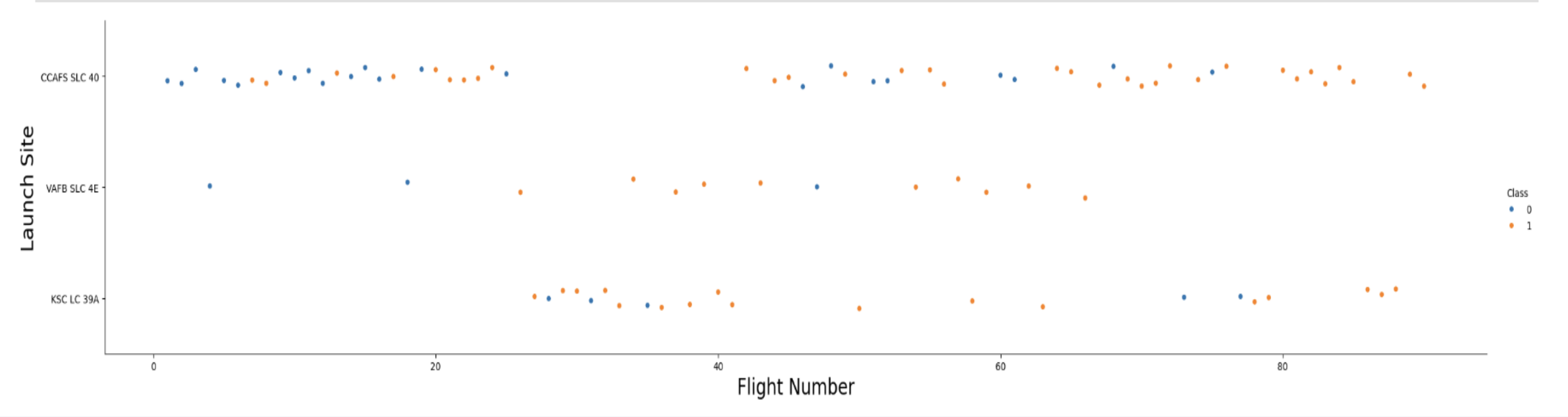
Results

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

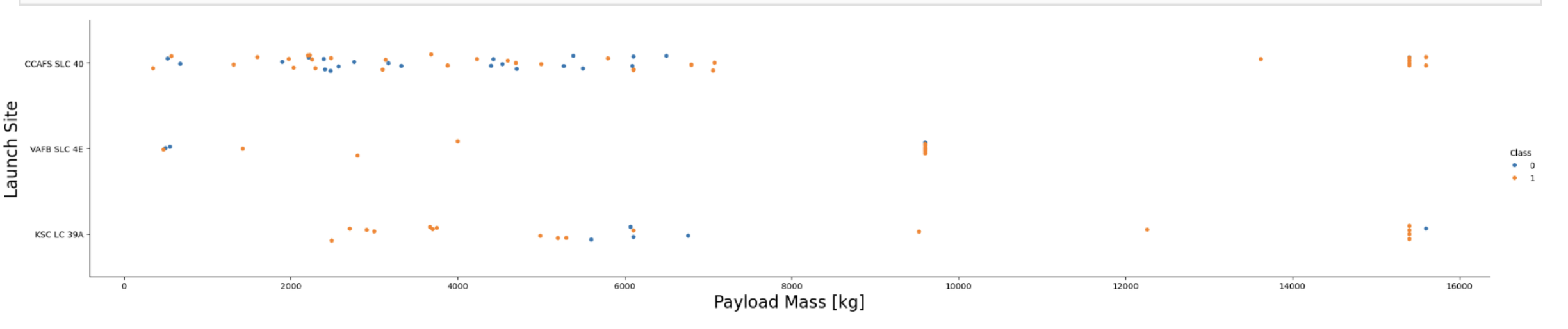
- Show a scatter plot of Flight Number vs. Launch Site



The success rate of a launch site increases with increasing number of flights

Payload vs. Launch Site

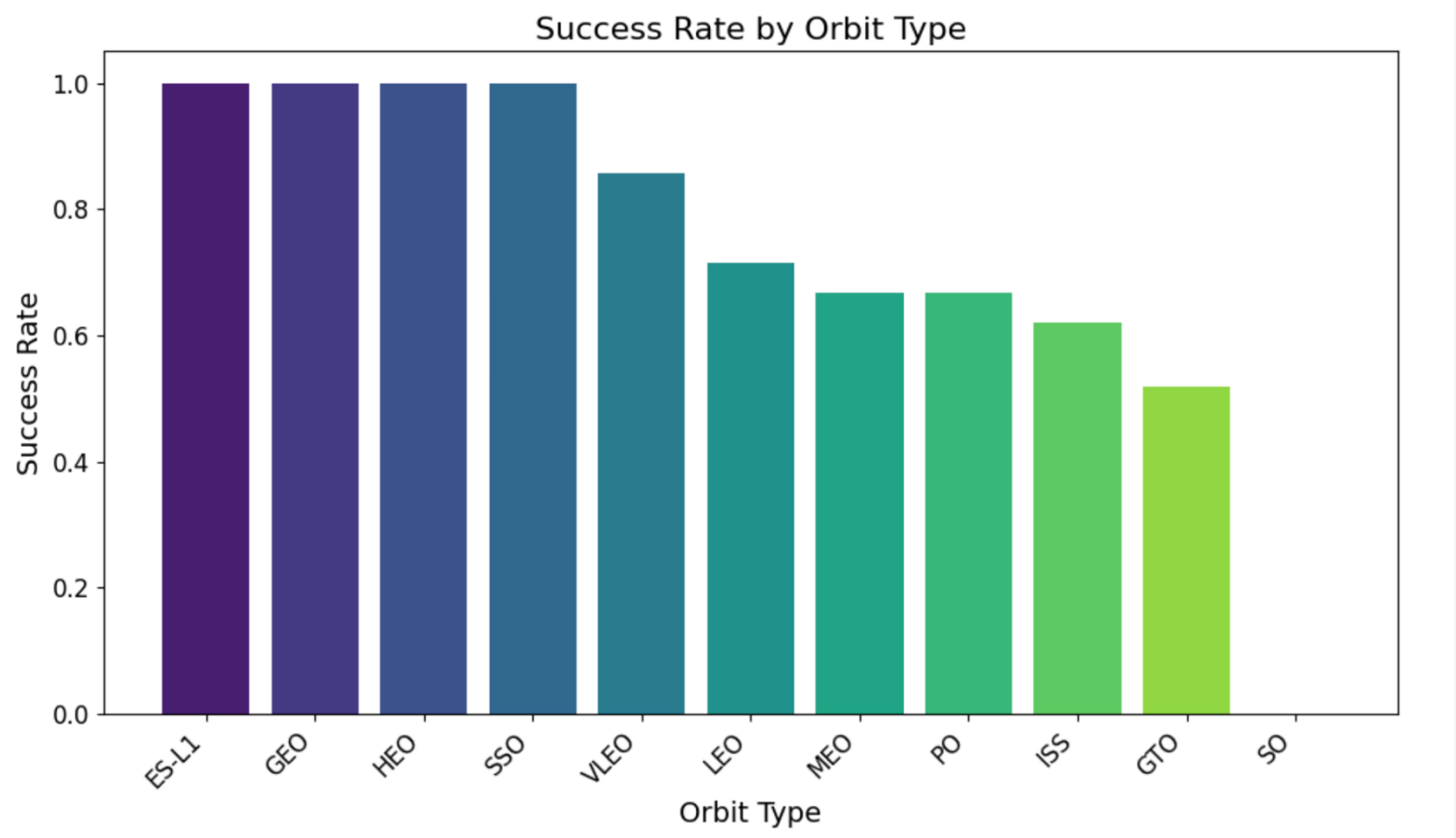
- Show a scatter plot of Payload vs. Launch Site



- For CCAFS SLC 40 site success rate increases with heavier payload mass.
- VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000)

Success Rate vs. Orbit Type

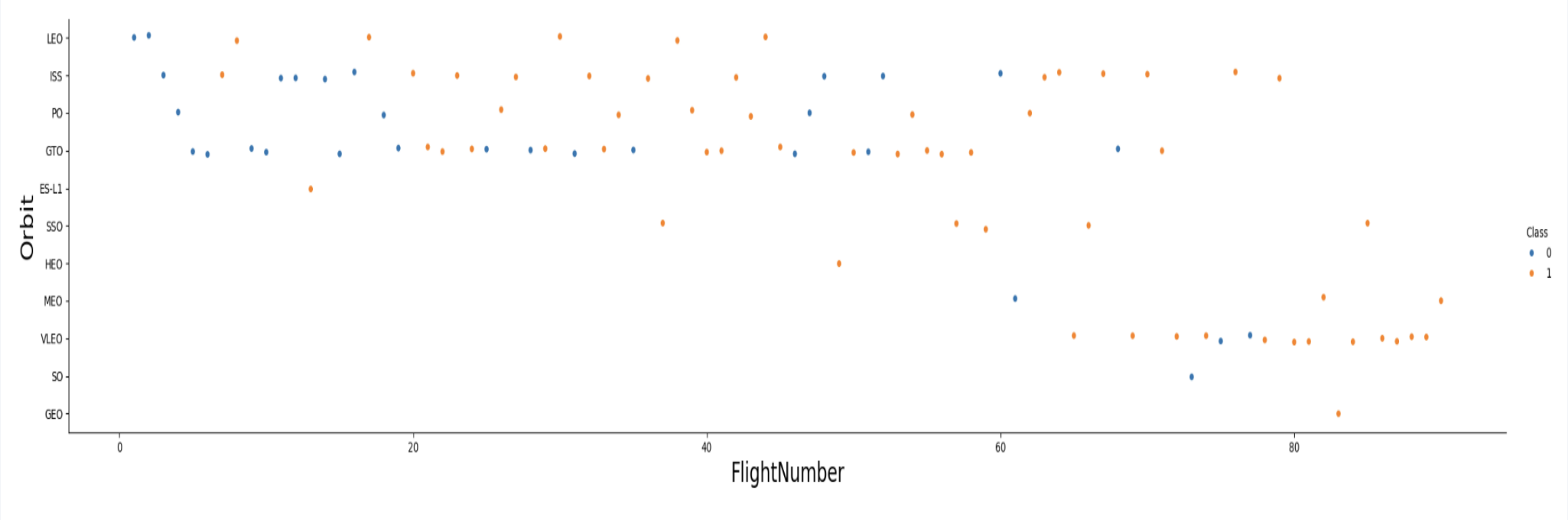
- Show a bar chart for the success rate of each orbit type



- ES-L1, GEO, HEO, SSO launch sites have the highest success rates

Flight Number vs. Orbit Type

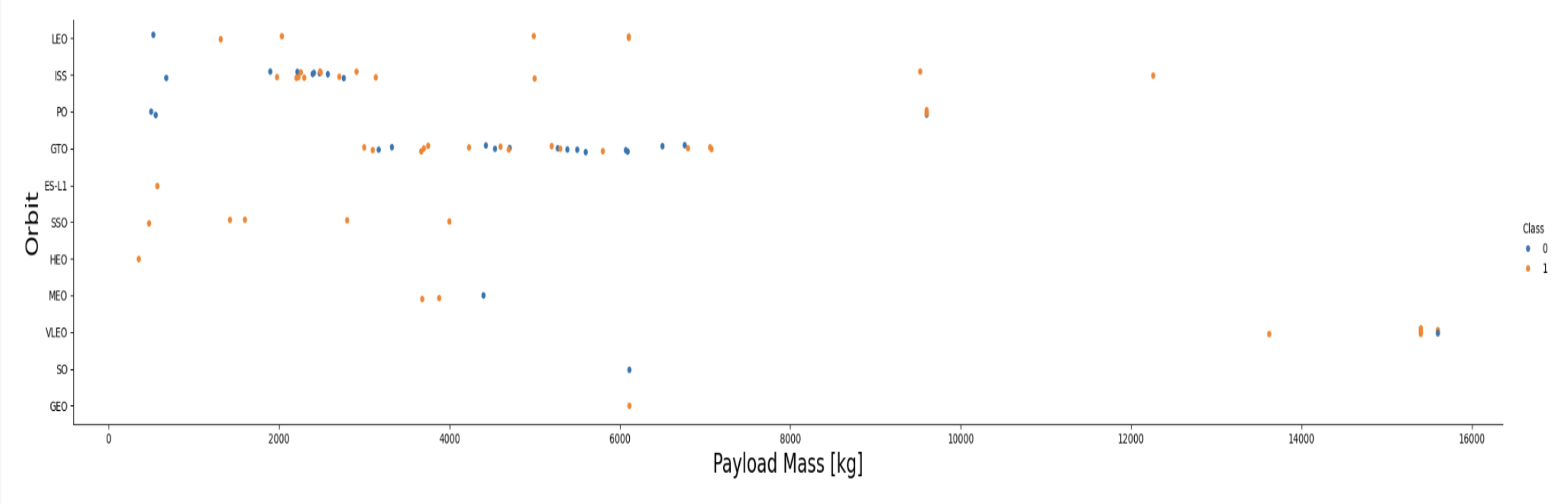
- Show a scatter point of Flight number vs. Orbit type



In LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

Payload vs. Orbit Type

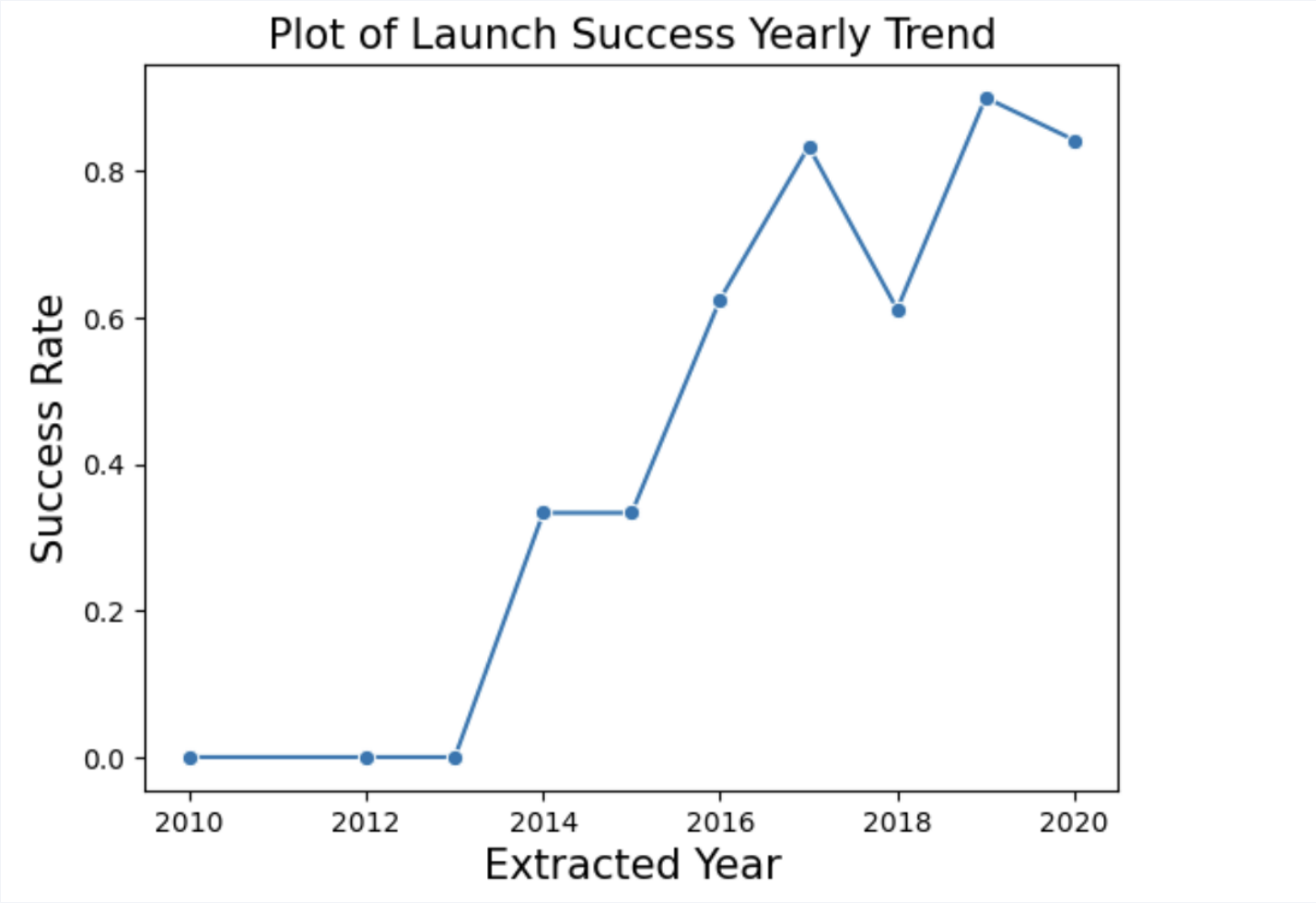
- Show a scatter point of payload vs. orbit type



With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend

- Show a line chart of yearly average success rate



The success rate since 2013 kept increasing till 2020.

All Launch Site Names

```
[26]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

[26]:

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

Using DISTINCT keyword to show unique launch sites.

Launch Site Names Begin with 'CCA'

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

[27]:
```
%sql SELECT *FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

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

[27]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
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 (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Using sql query using WHERE clause and LIKE, LIMIT functionality to show records of launch sites starting with CCA.

Total Payload Mass

```
[30]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_PayloadMass FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCAFS LC-40'
 * sqlite:///my_data1.db
Done.
[30]: Total_PayloadMass
------------------------
                 67363
```

- Using SUM function and WHERE clause to calculate palyload carried by NASA booster (CCAFS LC-40)

Average Payload Mass by F9 v1.1

```
[33]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AS Avg_PayloadMass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1'

 * sqlite:///my_data1.db
Done.

[33]: Avg_PayloadMass
------------------
           2928.4
```

- Using sql query with the AVG function and WHERE clause to find payload mass carried by booster version F9 v1.1.

First Successful Ground Landing Date

```
%sql SELECT MIN(Date) AS FirstSuccessfull_landing_date FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success (ground pad)'
```

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

FirstSuccessfull_landing_date
2015-12-22

Using sql your query with WHERE clause and LIKE function to get fist successful landing date on ground pad.

Successful Drone Ship Landing with Payload between 4000 and 6000

[32]:
```
%%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.
```

[32]:

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

Using WHERE clause to first select Landing outcome,
then using AND to select the payload mass ranging between 4000-6000 kg for all boosters.

Total Number of Successful and Failure Mission Outcomes

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

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

SuccessOutcome
100

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

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

FailureOutcome
1

- Used wildcard % to select mission outcomes

Boosters Carried Maximum Payload

```
41]: %%sql SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (
                                    SELECT MAX(PAYLOAD_MASS__KG_)
                                    FROM SPACEXTABLE
                                    )
            ORDER BY Booster_Version
 * sqlite:///my_data1.db
Done.
```

41]:

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

Using WHERE clause and MAX function to list names of the booster which have carried the maximum payload mass

2015 Launch Records

```
%%sql SELECT Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTABLE 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
F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

Using combinations of WHERE, Like, AND, BETWEEN to select landing outcome for year 2015.

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

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%%sql SELECT Landing_Outcome, COUNT(Landing_Outcome)
        FROM SPACEXTABLE
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY Landing_Outcome
        ORDER BY COUNT(Landing_Outcome) DESC
```

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

Landing_Outcome	COUNT(Landing_Outcome)
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

- Selected landing outcomes and COUNT of landing outcomes from the table and used WHERE clause to select a date range.
- Then used GROUP BY clause to group by the landing outcome and ORDER BY clause to sort the landing outcome in DESC order.

Section 3

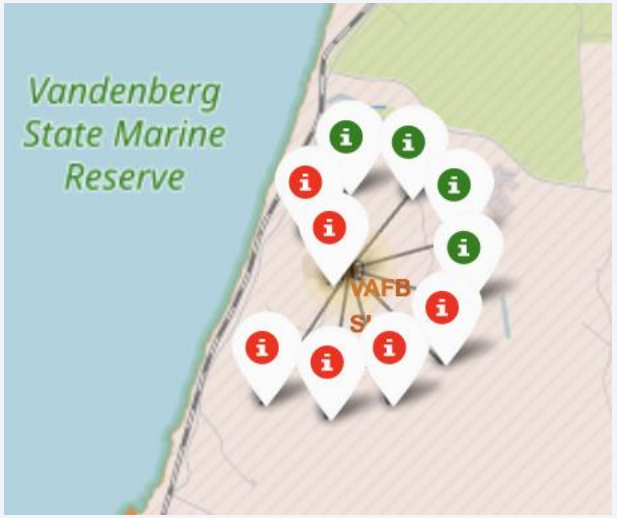
Launch Sites Proximities Analysis

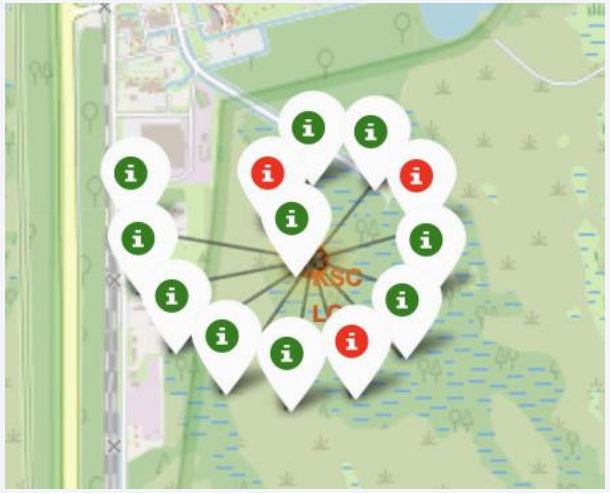
Folium Map Launch Sites

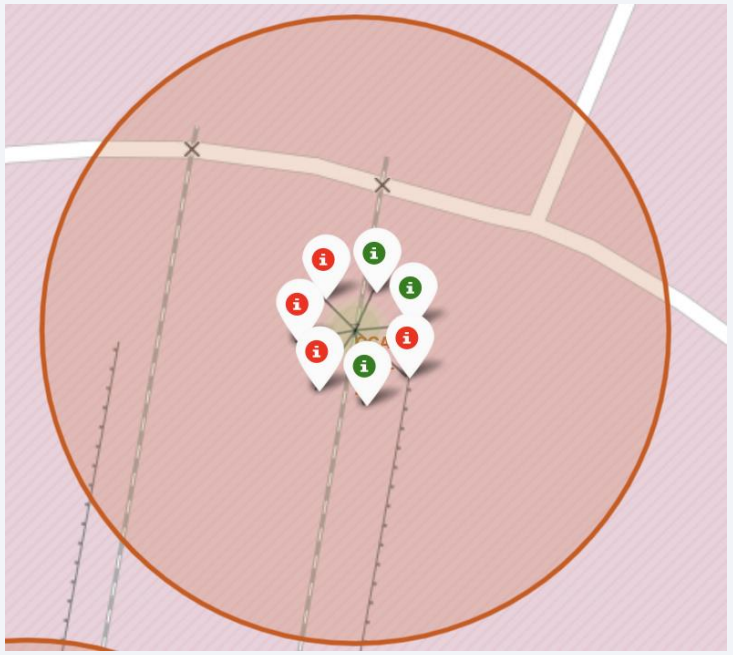
- Launch sites

- The Launch sites are marked in red on a global map.
- The sites are in proximity of sea.

Folium Map Launch Sites Marker cluster



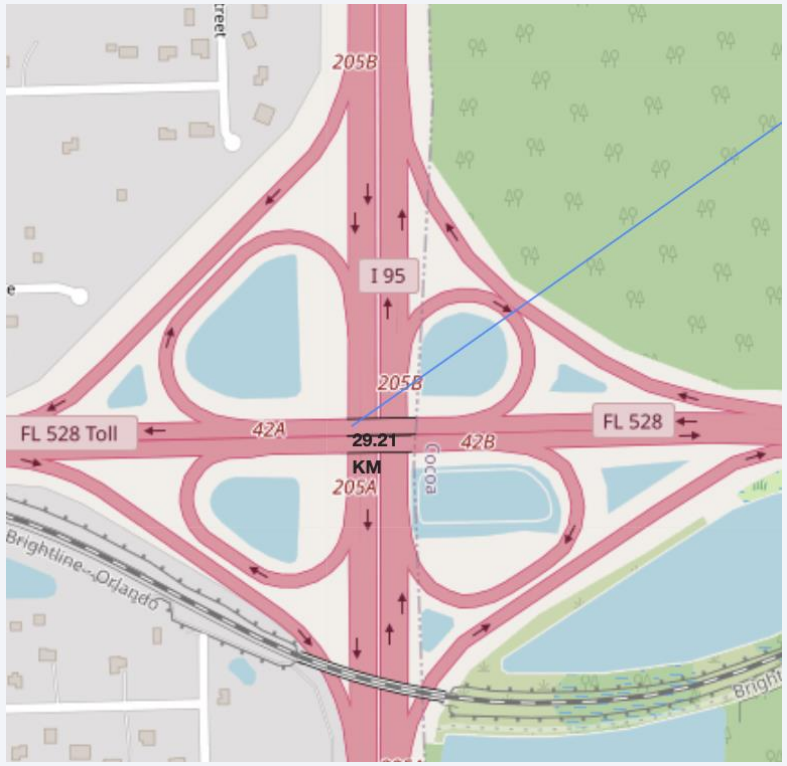




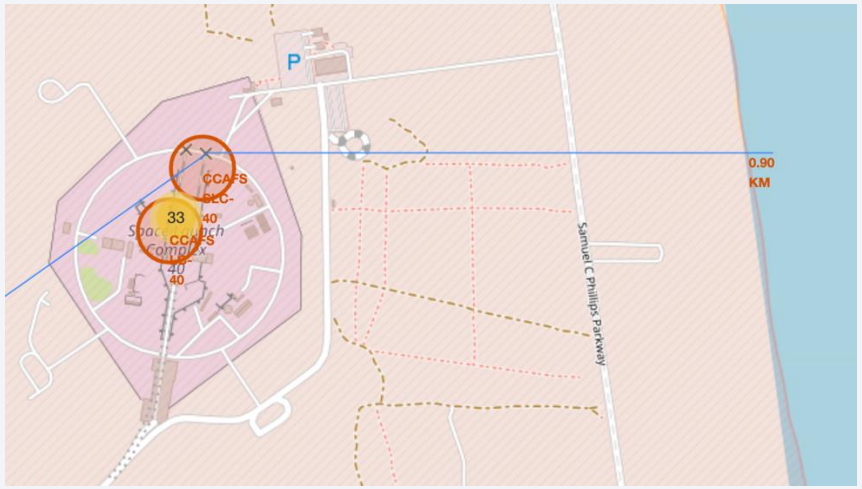
- Red markers represent fail launch sites
- Green markers represent successful launch sites

Folium Map Proximities

Nearest Highway



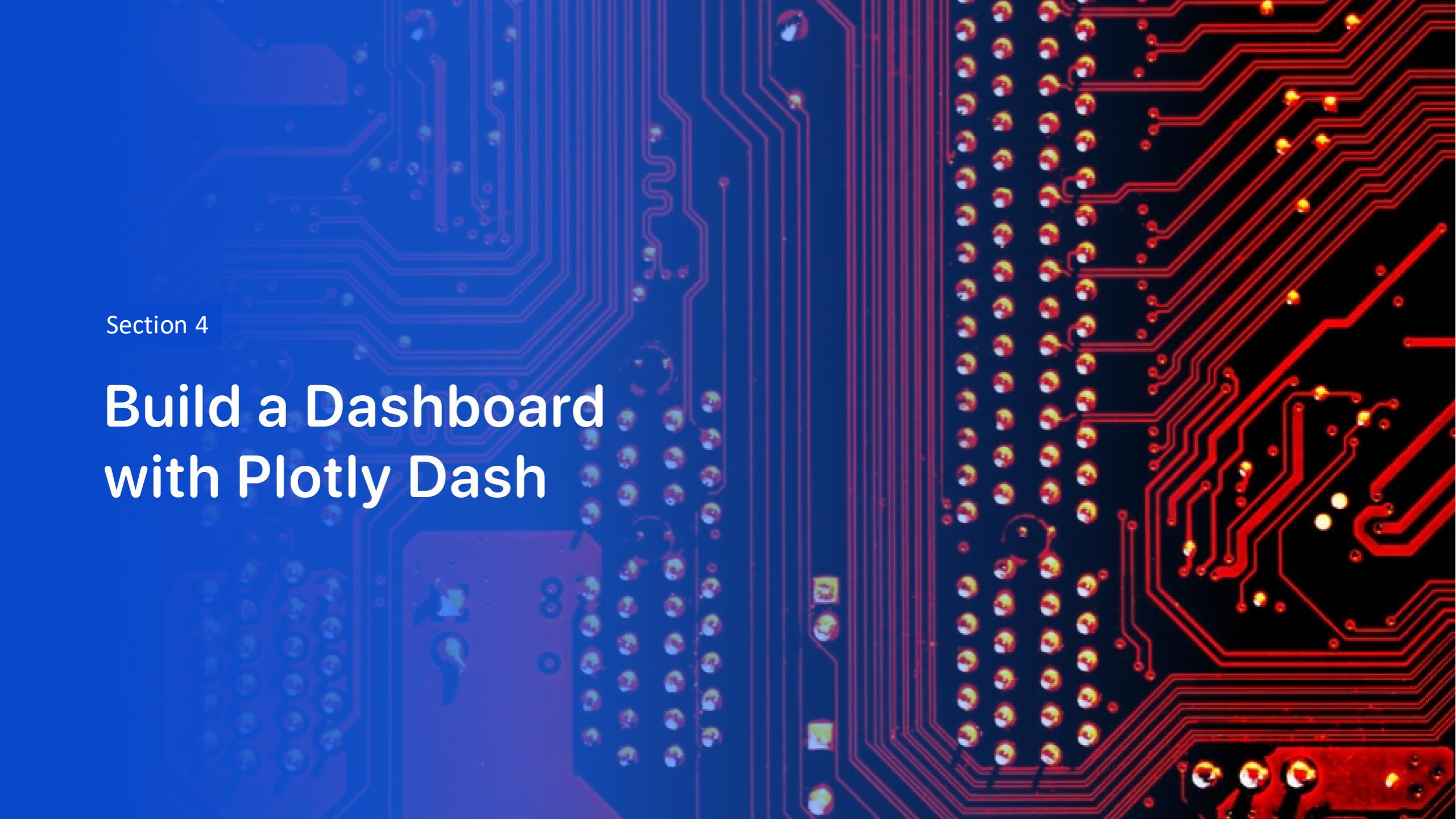
Nearest Coast line



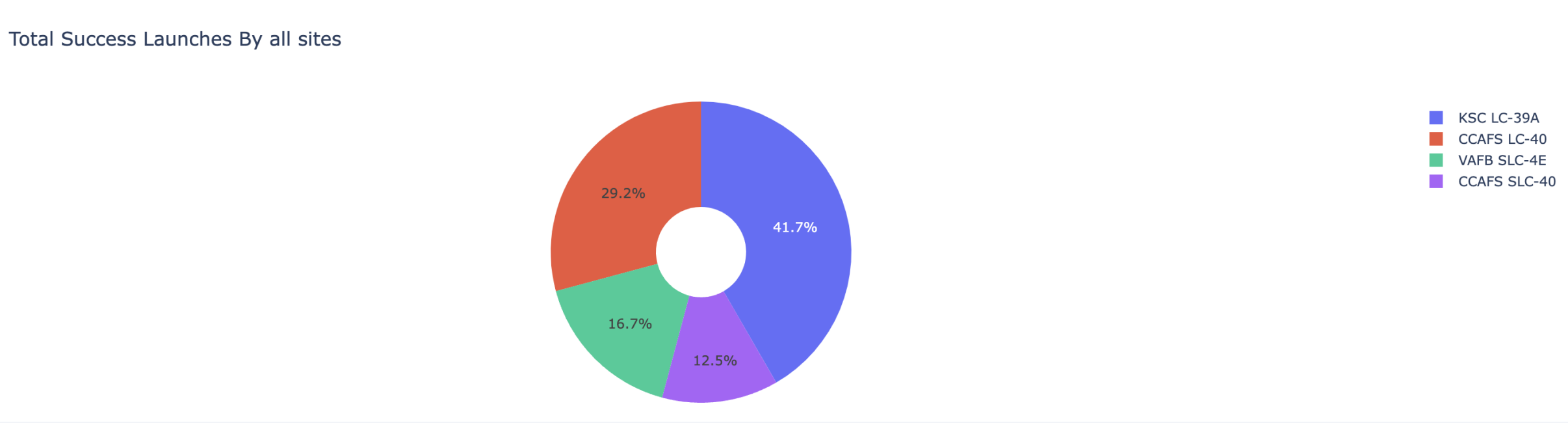
Distance to city

Railways Coast line

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

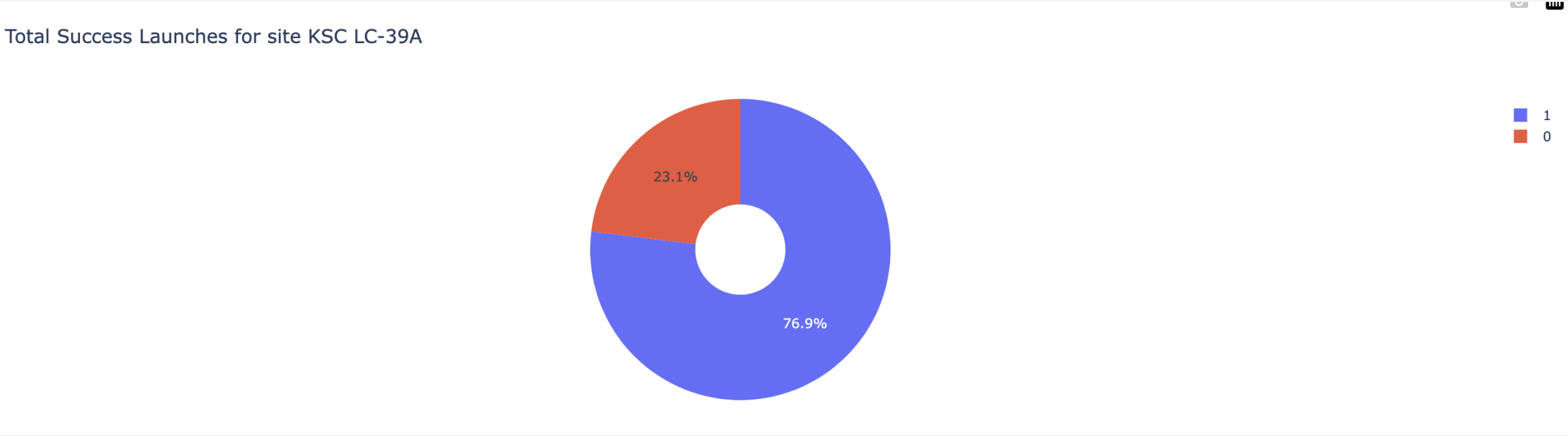

Section 4
Build a Dashboard with Plotly Dash

Launch Success Pie chart All sites



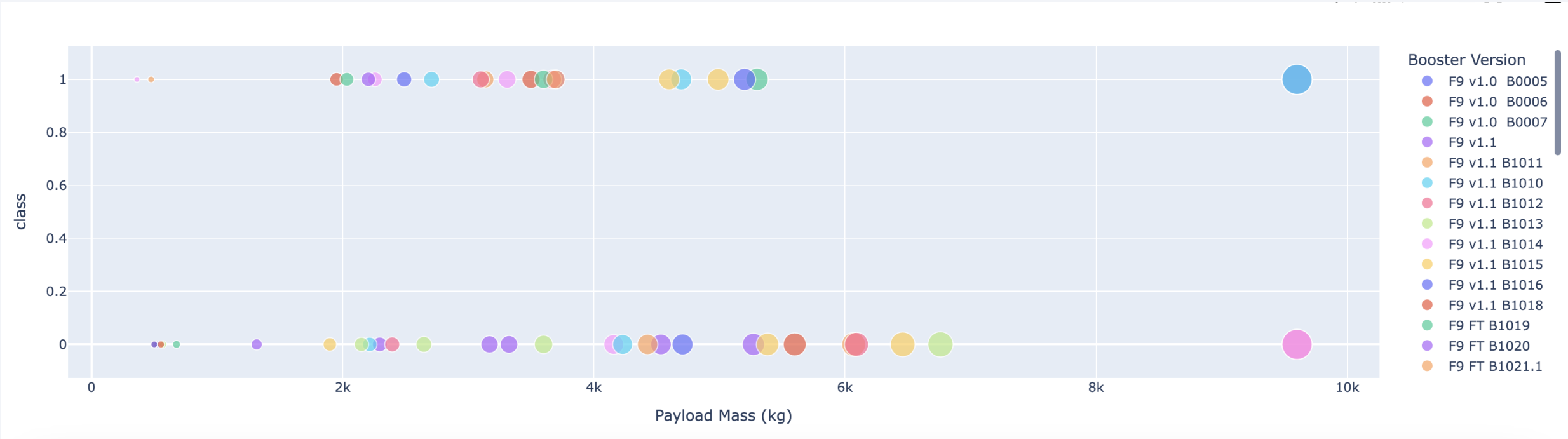
KSC LC-39A has the highest launch success rate.

Highest Launch success ratio



- KSC LC-39A has the highest launch success ratio of 76.9%

Payload vs Launch Outcome

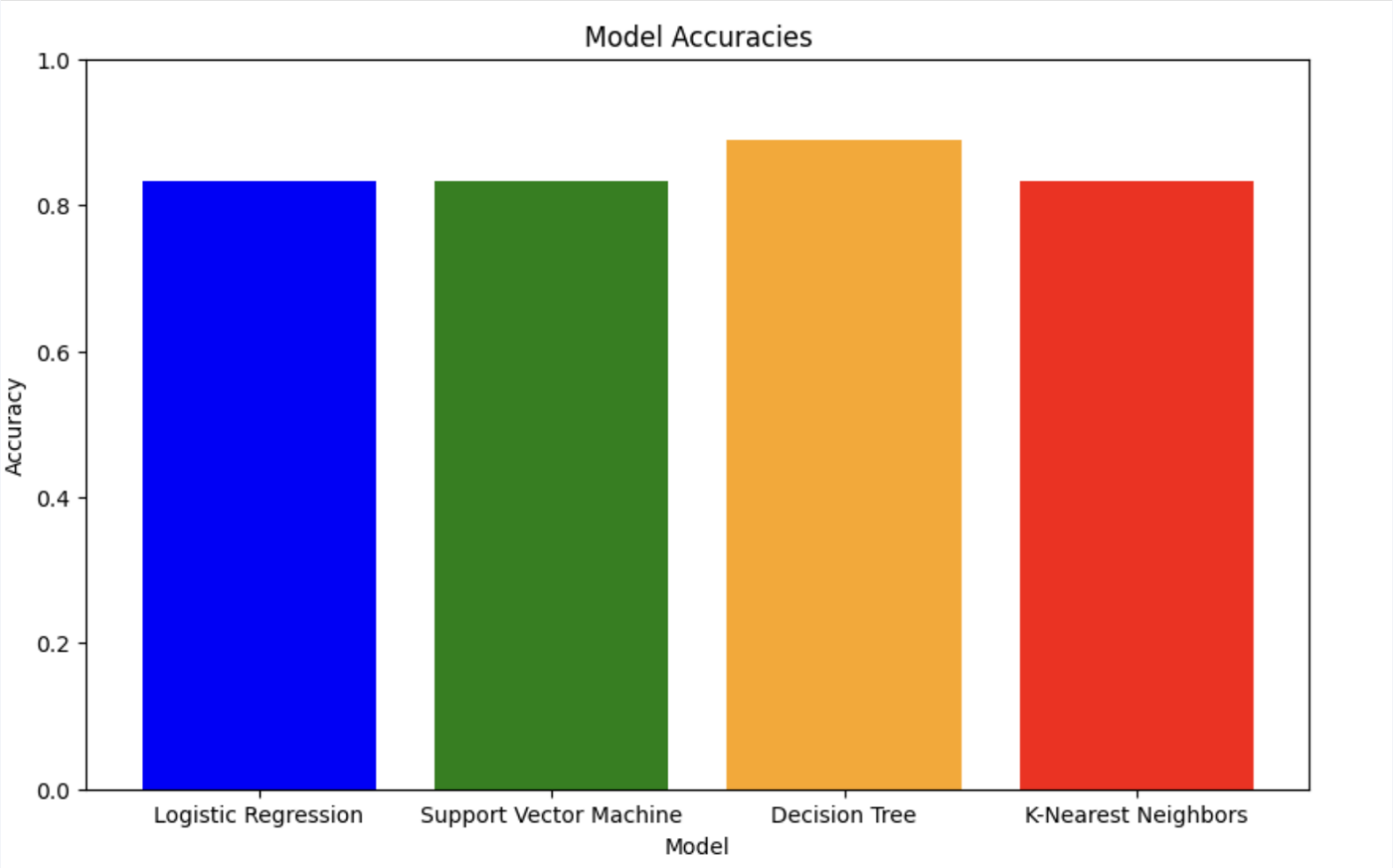


- Payload mass between 2000 to 5300 kg has the highest success rate.
- F9 B4 B1041.1 is the booster with highest success rate with heaviest payload weight.

Section 5

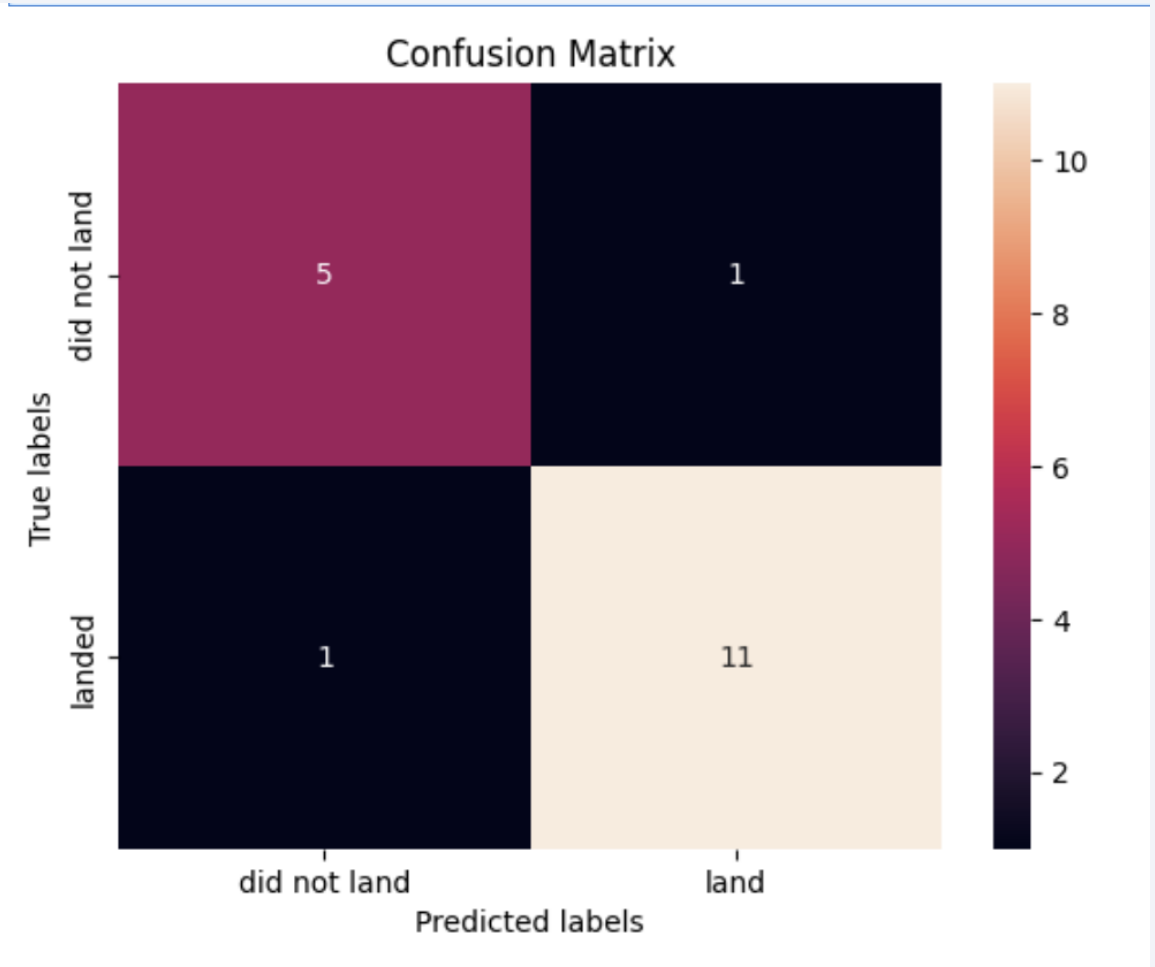
Predictive Analysis (Classification)

Classification Accuracy



- Decision tree ML method has the highest accuracy.

Confusion Matrix



- Confusion matrix of the Decision tree ML method.
- True Postive - 11 (True label is landed, Predicted label is also landed)
- False Postive - 1 (True label is not landed, Predicted label is landed)

Conclusions

- The success rate of stage 1 increases with higher number of flights.
- ES-L1, GEO, HEO, SSO launch sites have the highest success rates.
- Success rate increased since 2013 to 2020.
- KSC LC-39A has the highest success rate among all the launch sites with 76.9% success rate.
- The Decision Tree ML algorithm is the best to create the pipeline to analyze success rate for Space Y project.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!