IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Bushra Abdullahi
16/03/2023

Outline

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

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 results

- 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:

- 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

Data was collected through various methods:

Data was obtained via a GET request to the SpaceX API

The response content was decoded into JSON format using the .json() function

The JSON data was converted to a Pandas DataFrame using .json_normalize()

Data cleaning was performed, including identifying and addressing missing values

Web scraping of Falcon 9 launch records was conducted on Wikipedia using BeautifulSoup

The launch data was extracted from an HTML table, parsed, and converted to a Pandas DataFrame for analysis purposes.

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/BushraAbdullahi/DS-final/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

```
[7]  response = requests.get(spacex_url)
                                                                Python
```

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

```
[11] # Use json_normalize meethod to convert the json result into a dataframe
     df = pd.json_normalize(response.json())
                                                                Python
```

+ Code + Markdown

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

```
[36] # Calculate the mean value of PayloadMass column
     mean_payload_mass = data_falcon9['PayloadMass'].mean()

     # Replace the np.nan values with its mean value
     data_falcon9['PayloadMass'].fillna(mean_payload_mass, inplace=True)
     data_falcon9.isnull().sum()
                                                                Python
```

Data Collection - Scraping

We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup

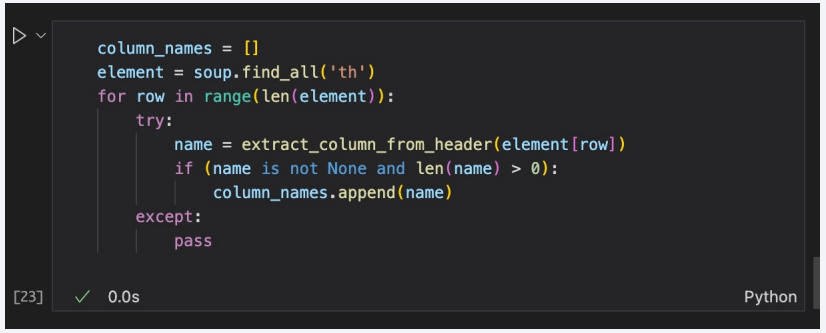
We parsed the table and converted it into a pandas dataframe.

The link to the notebook is https://github.com/BushraAbdullahi/DS-final/blob/main/jupyter-labs-webscraping.ipynb

Apply HTTP GET method to request Falcon 9 rocket launch page:

Create beautiful soup object:

Extract all column names from the HTML table header:



Data Wrangling

- 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://github.com/BushraAbdullahi/DS-final/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

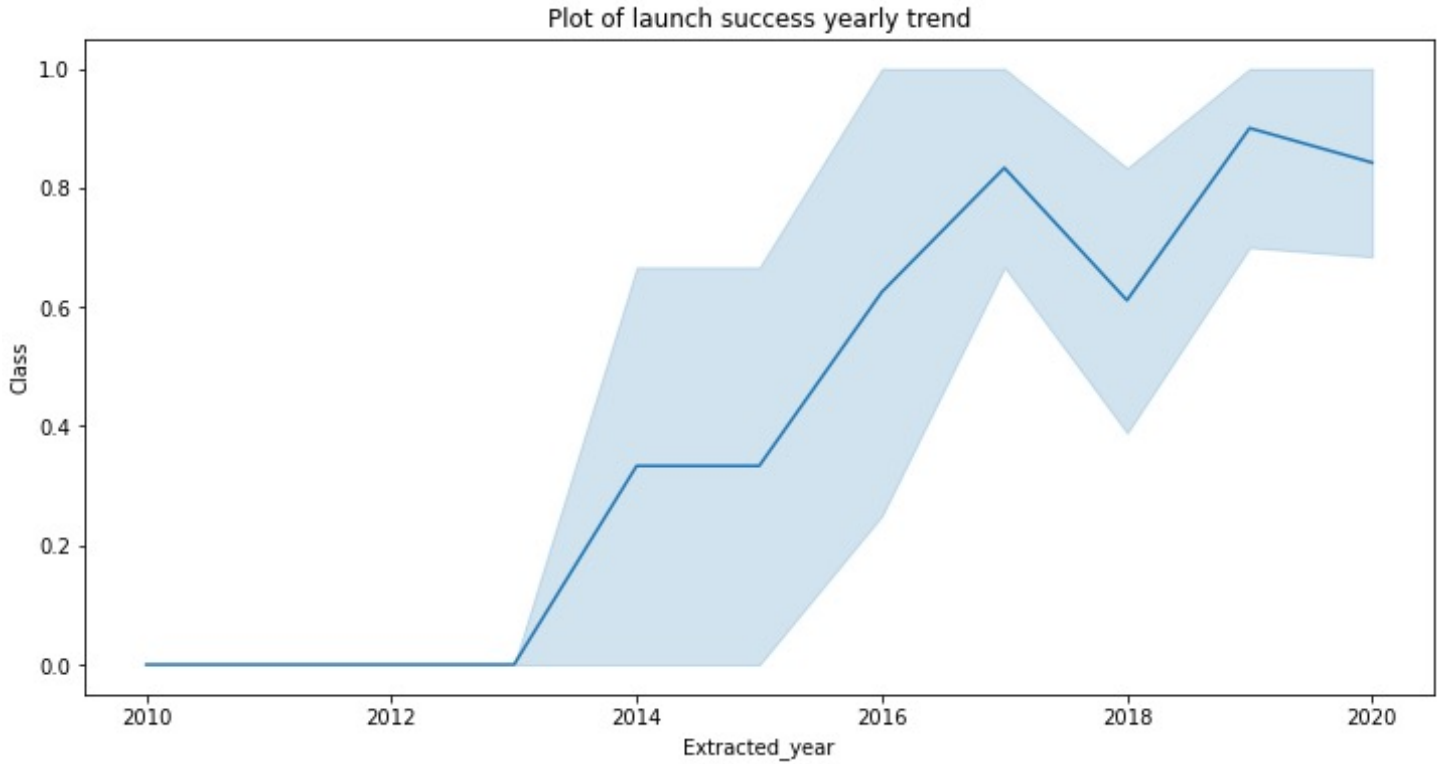
```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

[6] ✓ 0.0s Python

```
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, flight number and orbit type, the launch success yearly trend.
- The link to the notebook is https://github.com/BushraAbdullahi/DS-final/blob/main/jupyter-labs-eda-dataviz.ipynb



EDA with SQL

We loaded the SpaceX dataset into a SQL Lite database without leaving the jupyter notebook.

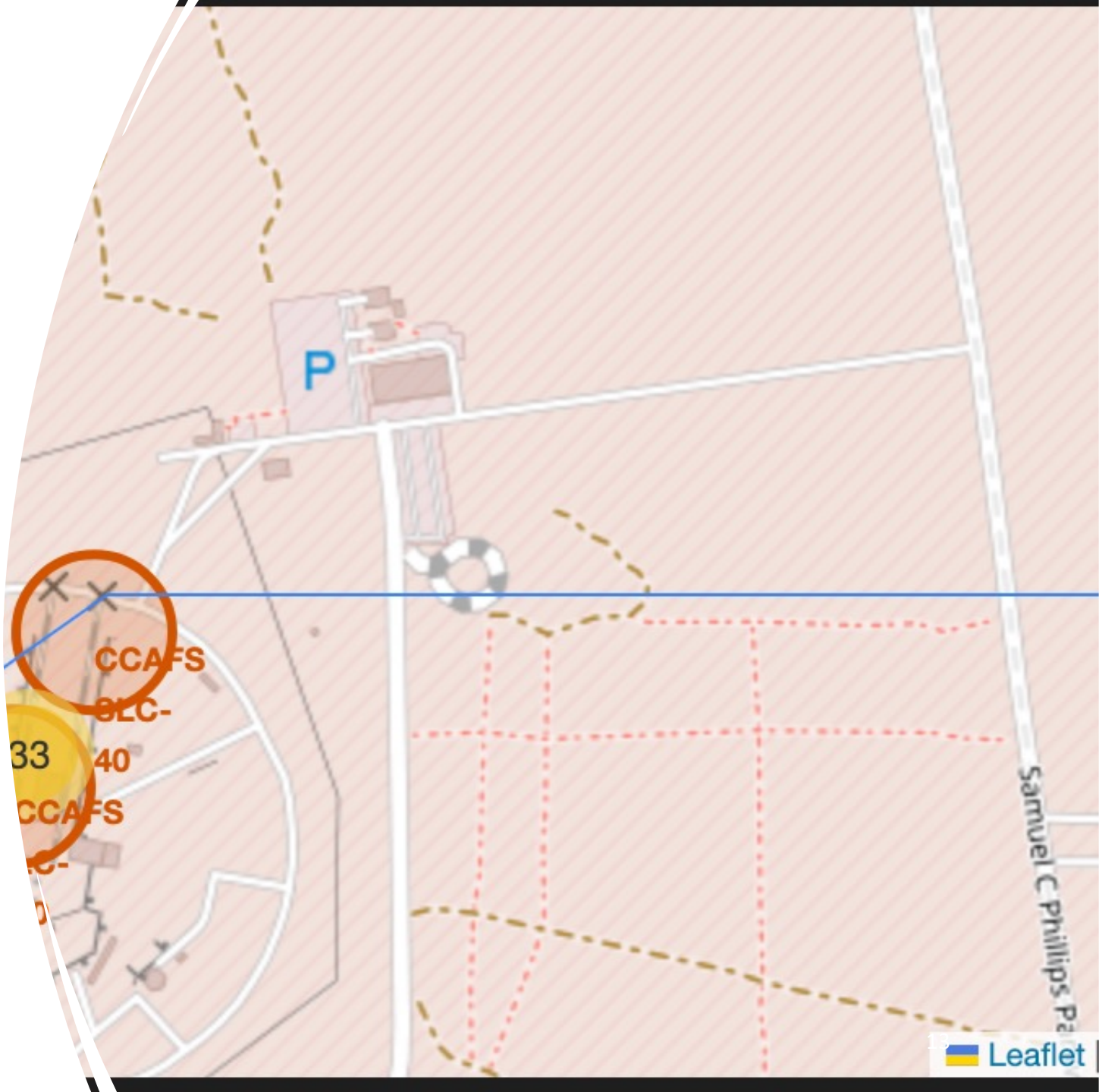
We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:

- 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/BushraAbdullahi/DS-final/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

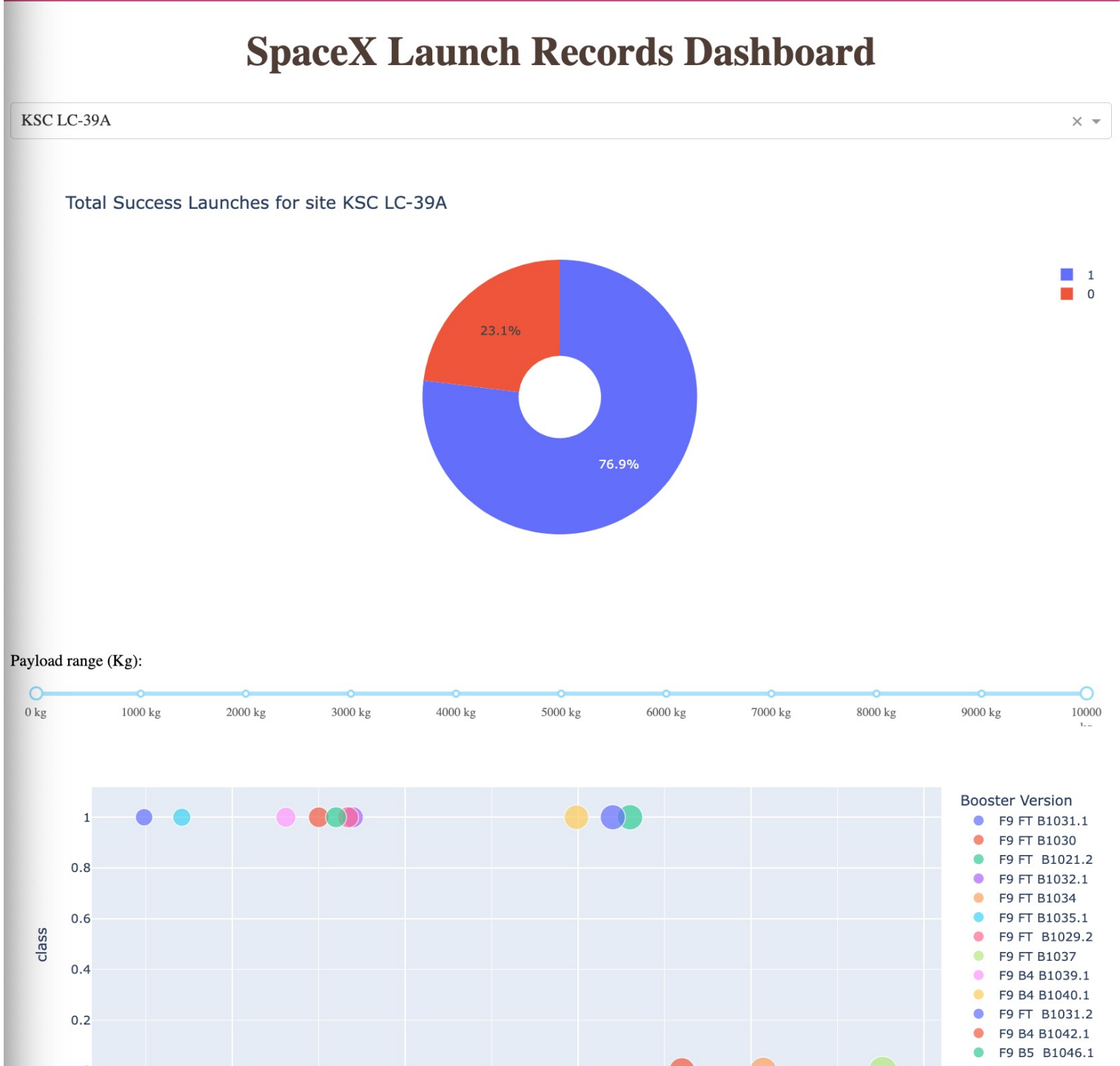
Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
    - Are launch sites near railways, highways and coastlines.
    - Do launch sites keep certain distance away from cities.



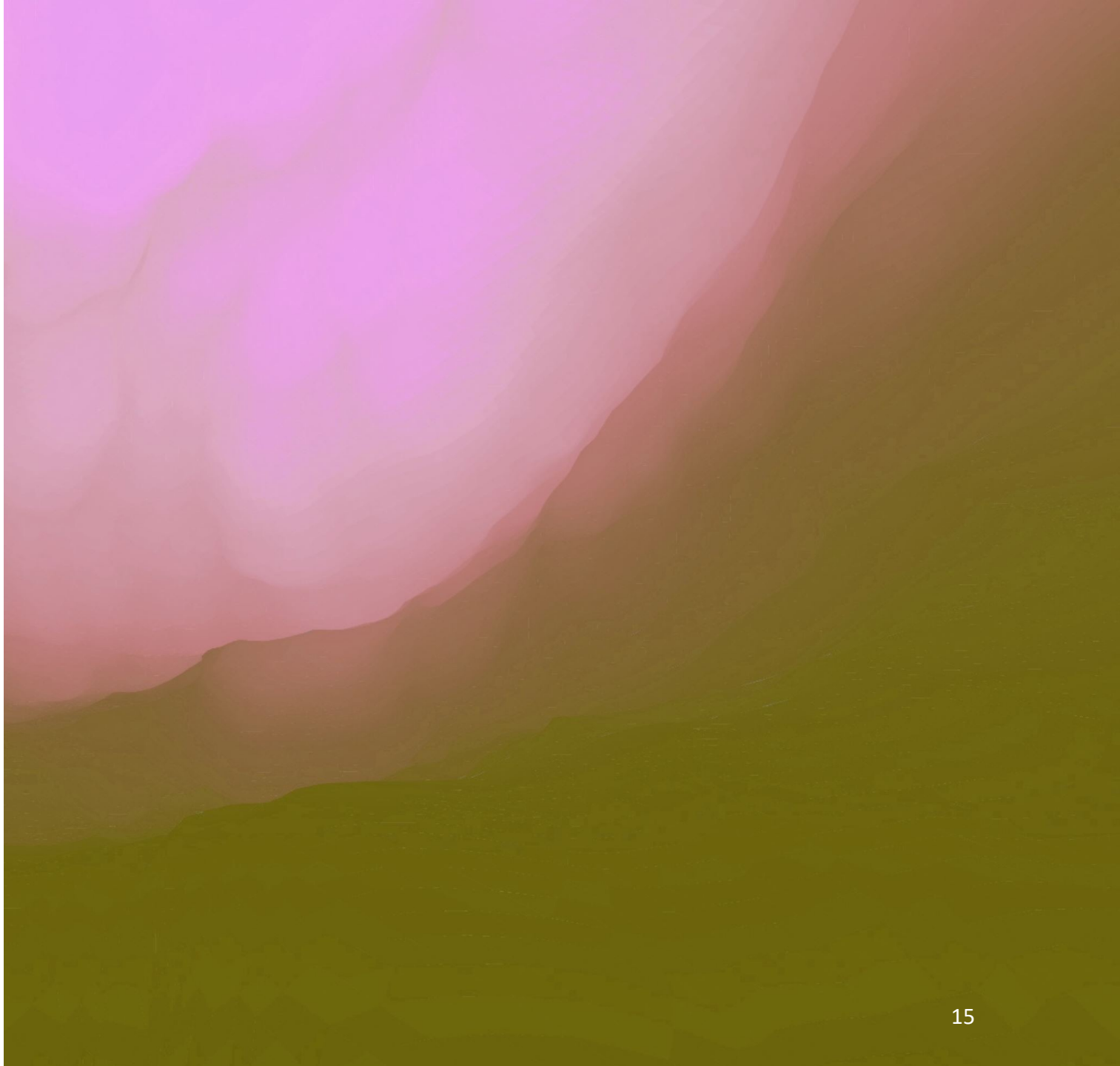
Build a Dashboard with Plotly Dash

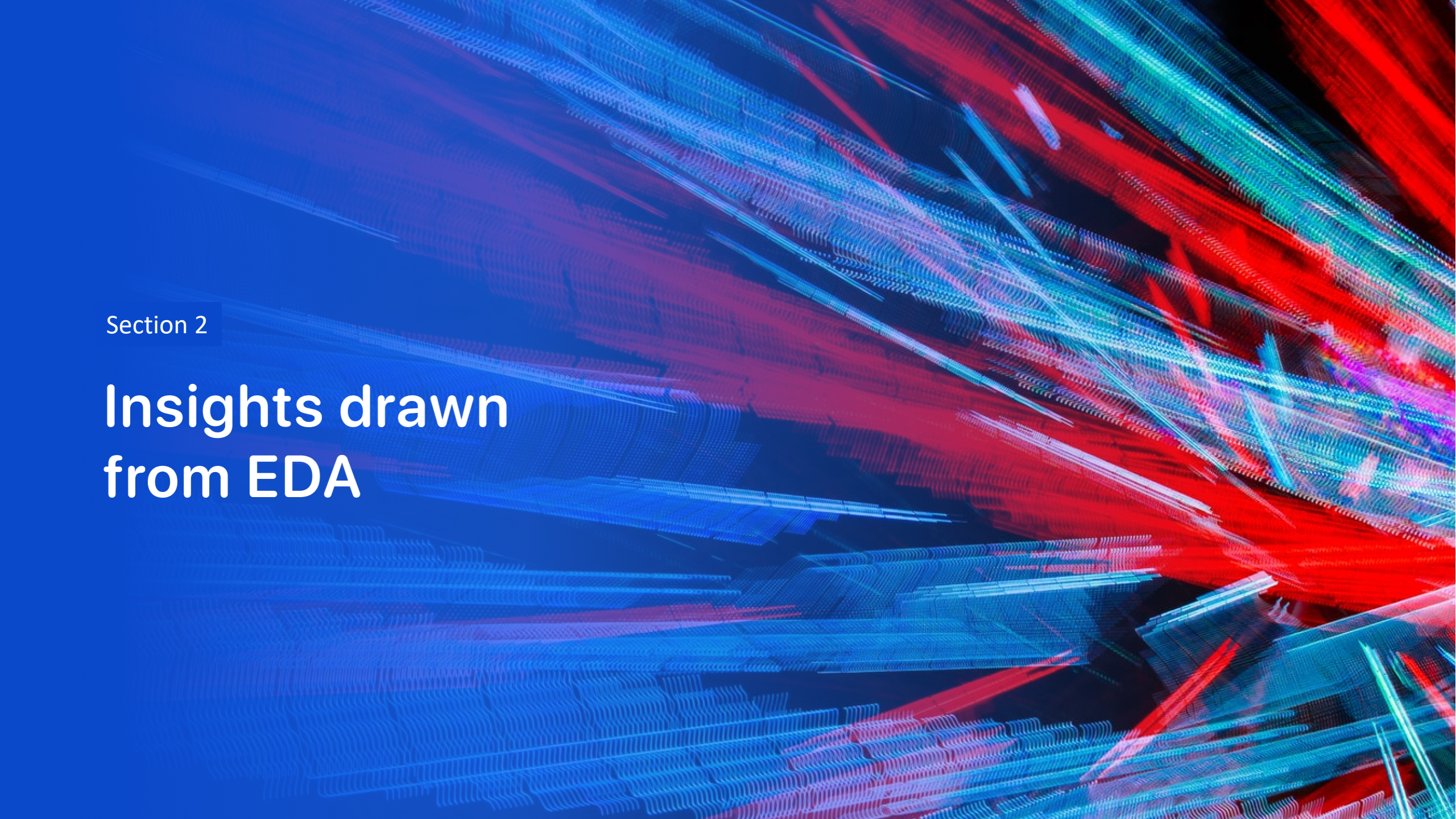
- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is https://github.com/BushraAbdullahi/DS-final/blob/main/app.py

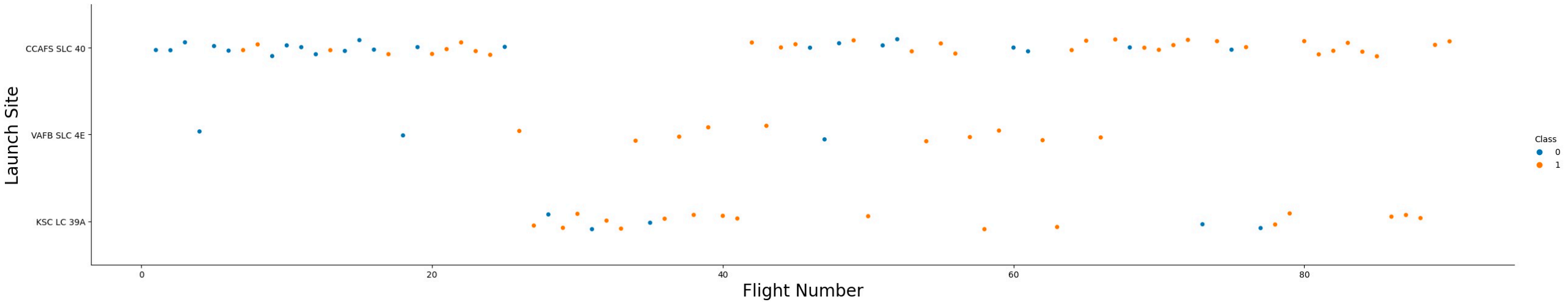


Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is https://github.com/BushraAbdullahi/DS-final/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

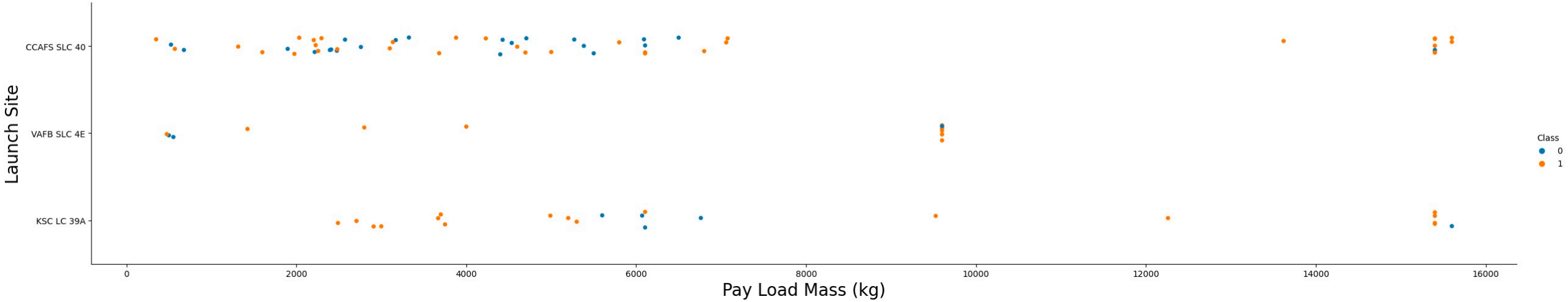



Section 2
Insights drawn from EDA



Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.

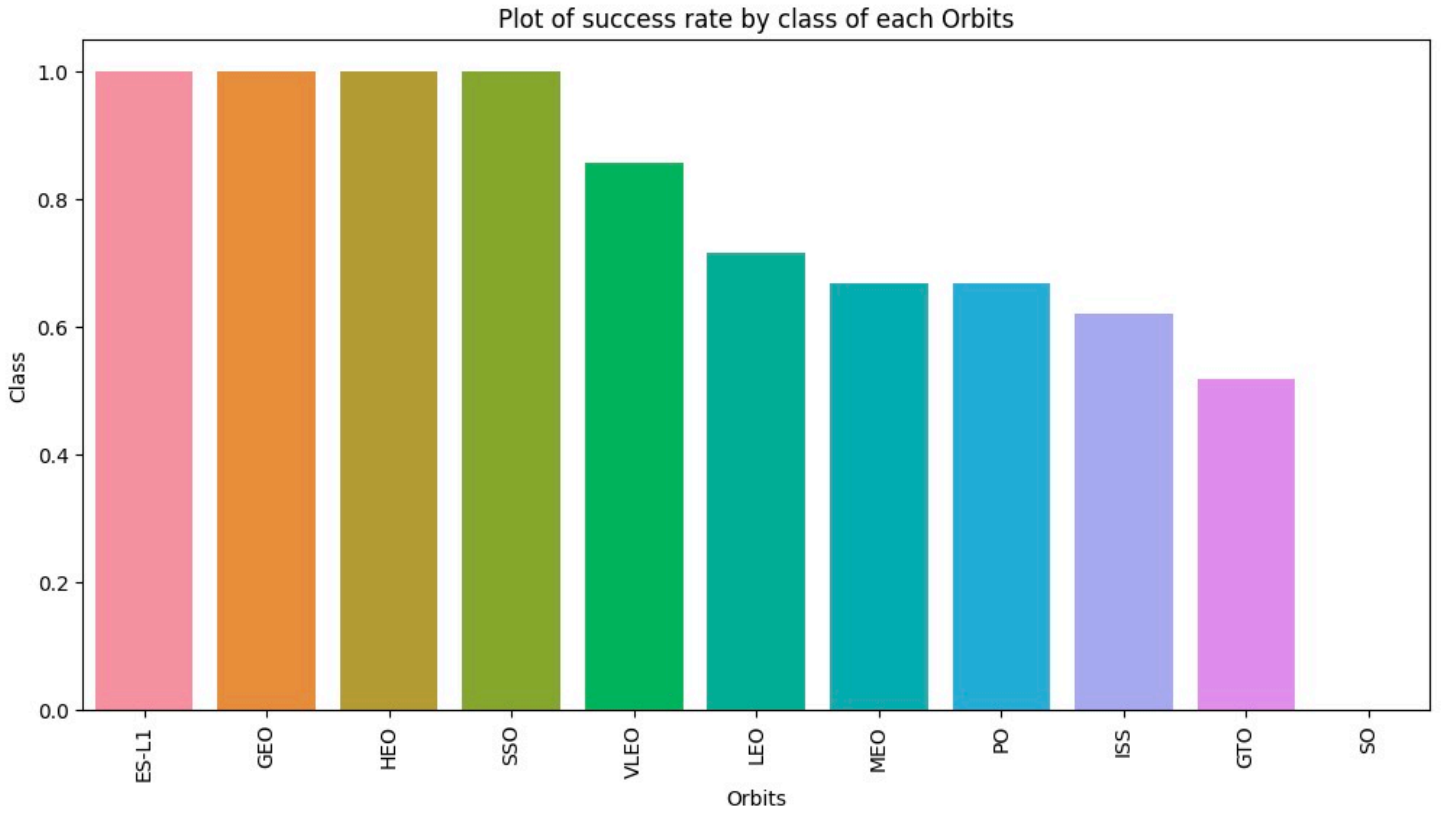


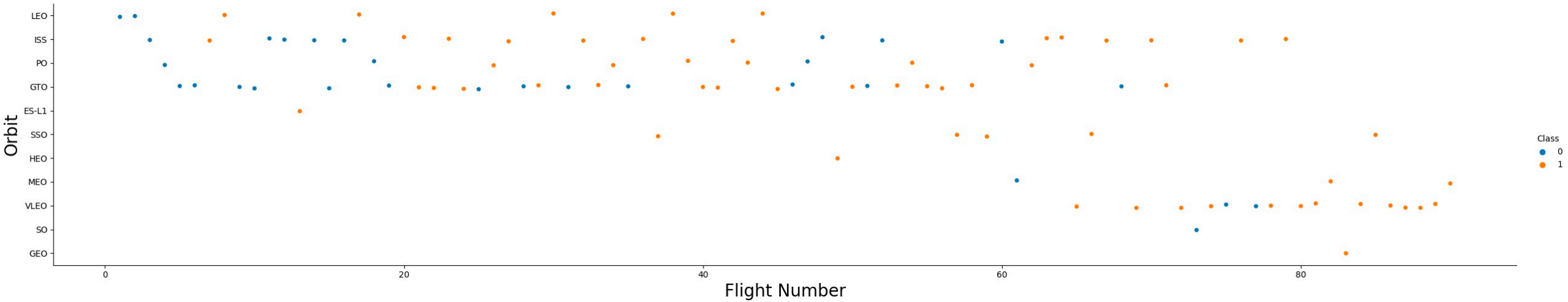
Payload vs. Launch Site

- The greater the payload mass for launch site CCAFS SLC 40, the higher the success rate for the rocket

Success Rate vs. Orbit Type

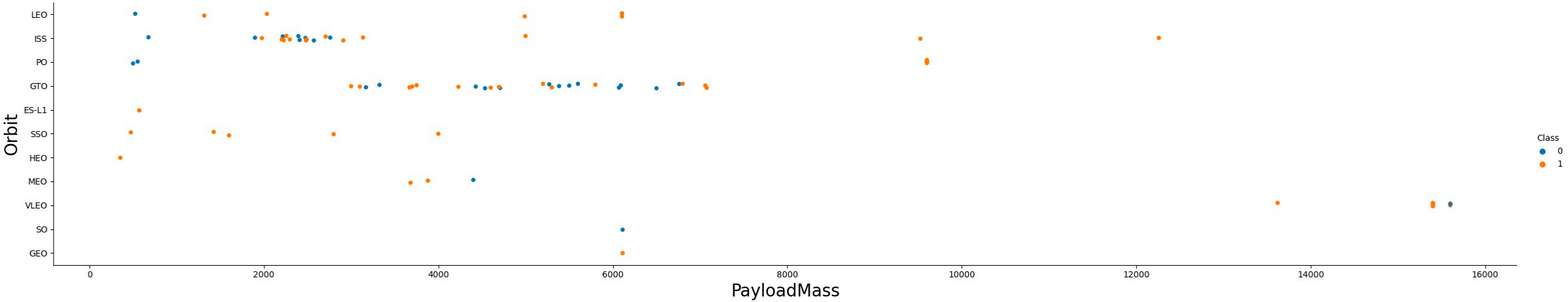
- From the bar chart, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.





Flight Number vs. Orbit Type

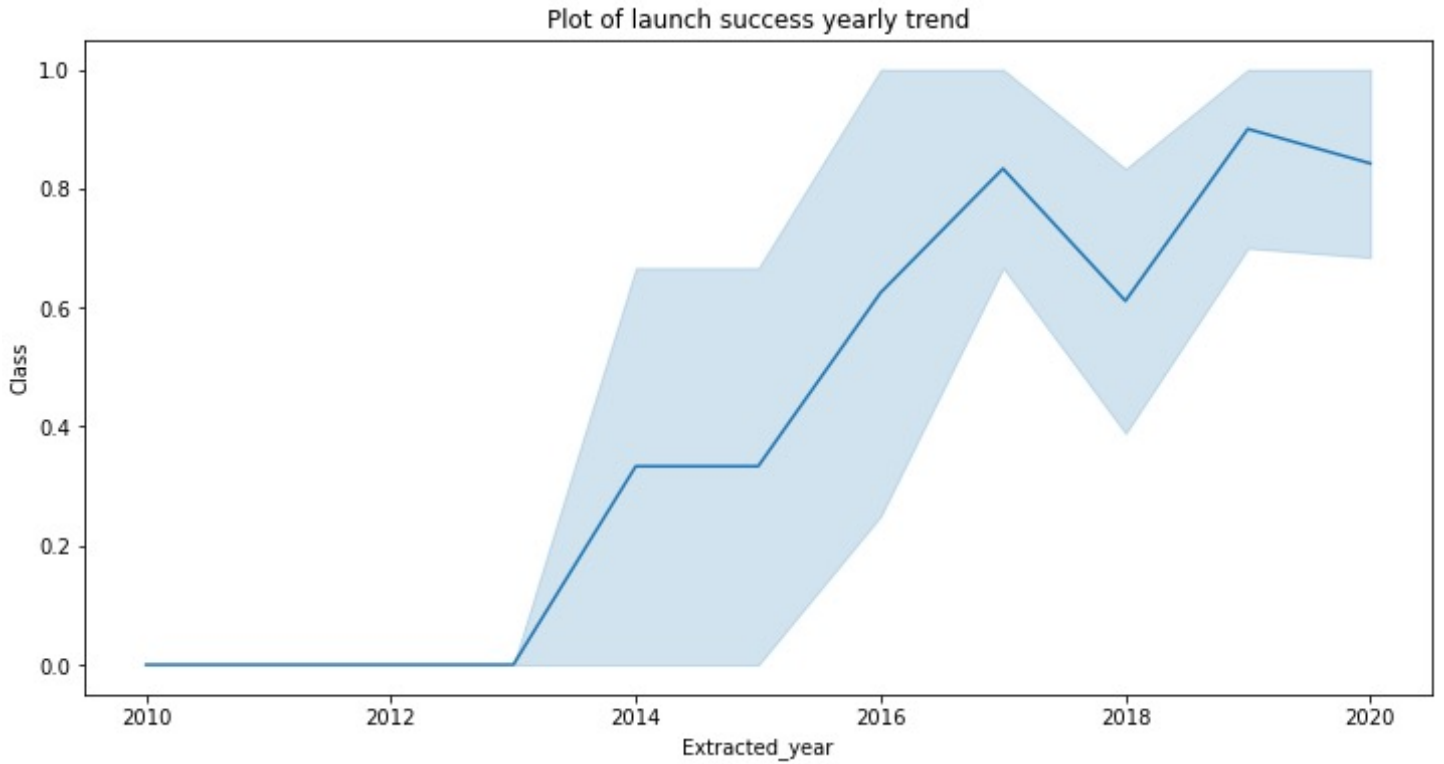
- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.

Launch Success Yearly Trend



- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.

All Launch Site Names

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

```
task_1 = '''
        SELECT DISTINCT Launch_Site
        FROM SPACEXTBL
'''
results = pd.read_sql_query(task_1, con)
results
```

72] ✓ 0.1s Python

	Launch_Site
0	CCAFS LC-40
1	VAFB SLC-4E
2	KSC LC-39A
3	CCAFS SLC-40

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

Launch Site Names Begin with ‘CCA’

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

```
task_2 = '''
        SELECT *
        FROM SPACEXTBL
        WHERE Launch_Site LIKE 'CCA%'
        LIMIT 5
'''
results = pd.read_sql_query(task_2, con)
results
```

✓ 0.1s Python

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__
0	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	
1	2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	
3	2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	
4	2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	

- We used the query above to display 5 records where launch sites begin with `CCA`

Total Payload Mass

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

```
task_3 = '''
        SELECT SUM(PAYLOAD_MASS__KG_) AS Total_PayloadMass
        FROM SPACEXTBL
        WHERE Customer LIKE 'NASA (CRS)'
        '''
results = pd.read_sql_query(task_3, con)
results
```

✓ 0.0s Pyth

	Total_PayloadMass
0	45596

- We calculated the total payload carried by boosters from NASA as 45596 using the query below

Average Payload Mass by F9 v1.1

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

```
task_4 = '''
        SELECT AVG(PAYLOAD_MASS__KG_) AS Avg_PayloadMass
        FROM SPACEXTBL
        WHERE Booster_Version = 'F9 v1.1'
        '''
results = pd.read_sql_query(task_4, con)
results
```

✓ 0.0s

	Avg_PayloadMass
0	2928.4

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

First Successful Ground Landing Date

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SPACEXTBL
        WHERE Landing_Outcome LIKE 'Success (ground pad)';
        '''
results = pd.read_sql_query(task_5, con)
results
```

6] ✓ 0.0s Python

	FirstSuccessfull_landing_date
0	2015-12-22

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
task_6 = '''
        SELECT Booster_Version
        FROM SPACEXTBL
        WHERE Landing_Outcome = 'Success (drone ship)'
            AND PAYLOAD_MASS__KG_ > 4000
            AND PAYLOAD_MASS__KG_ < 6000
        '''
results = pd.read_sql_query(task_6, con)
results
```

30] ✓ 0.0s Python

	Booster_Version
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

- We used wildcard like '%' to filter for **WHERE** MissionOutcome was a success or a failure.

List the total number of successful and failure mission outcomes

```
task_7a = '''
        SELECT COUNT(Mission_Outcome) AS Success_Outcome
        FROM SPACEXTBL
        WHERE Mission_Outcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(Mission_Outcome) AS Failure_Outcome
        FROM SPACEXTBL
        WHERE Mission_Outcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
results = pd.read_sql_query(task_7a, con)
display(results)

print('The total number of failed mission outcome is:')
results = pd.read_sql_query(task_7b, con)
display(results)
```

✓ 0.0s

```
The total number of successful mission outcome is:
```

	Success_Outcome
0	100

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
task_8 = '''
        SELECT Booster_Version, PAYLOAD_MASS__KG_
        FROM SPACEXTBL
        WHERE PAYLOAD_MASS__KG_ = (
                                SELECT MAX(PAYLOAD_MASS__KG_)
                                FROM SPACEXTBL
                                )
        ORDER BY Booster_Version
        '''
results = pd.read_sql_query(task_8, con)
results
```

4] ✓ 0.0s Python

	Booster_Version	PAYLOAD_MASS__KG_
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600

2015 Launch Records

- We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
task_9 = '''
        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'
        '''
results = pd.read_sql_query(task_9, con)
results
```

5] ✓ 0.0s Python

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

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

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

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

```
task_10 = '''
        SELECT Landing_Outcome, COUNT(Landing_Outcome)
        FROM SPACEXTBL
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY Landing_Outcome
        ORDER BY COUNT(Landing_Outcome) DESC
        '''
results = pd.read_sql_query(task_10, con)
results
```

✓ 0.0s Python

	Landing_Outcome	COUNT(Landing_Outcome)
0	No attempt	10
1	Success (drone ship)	5
2	Failure (drone ship)	5
3	Success (ground pad)	3
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Failure (parachute)	2
7	Precluded (drone ship)	1

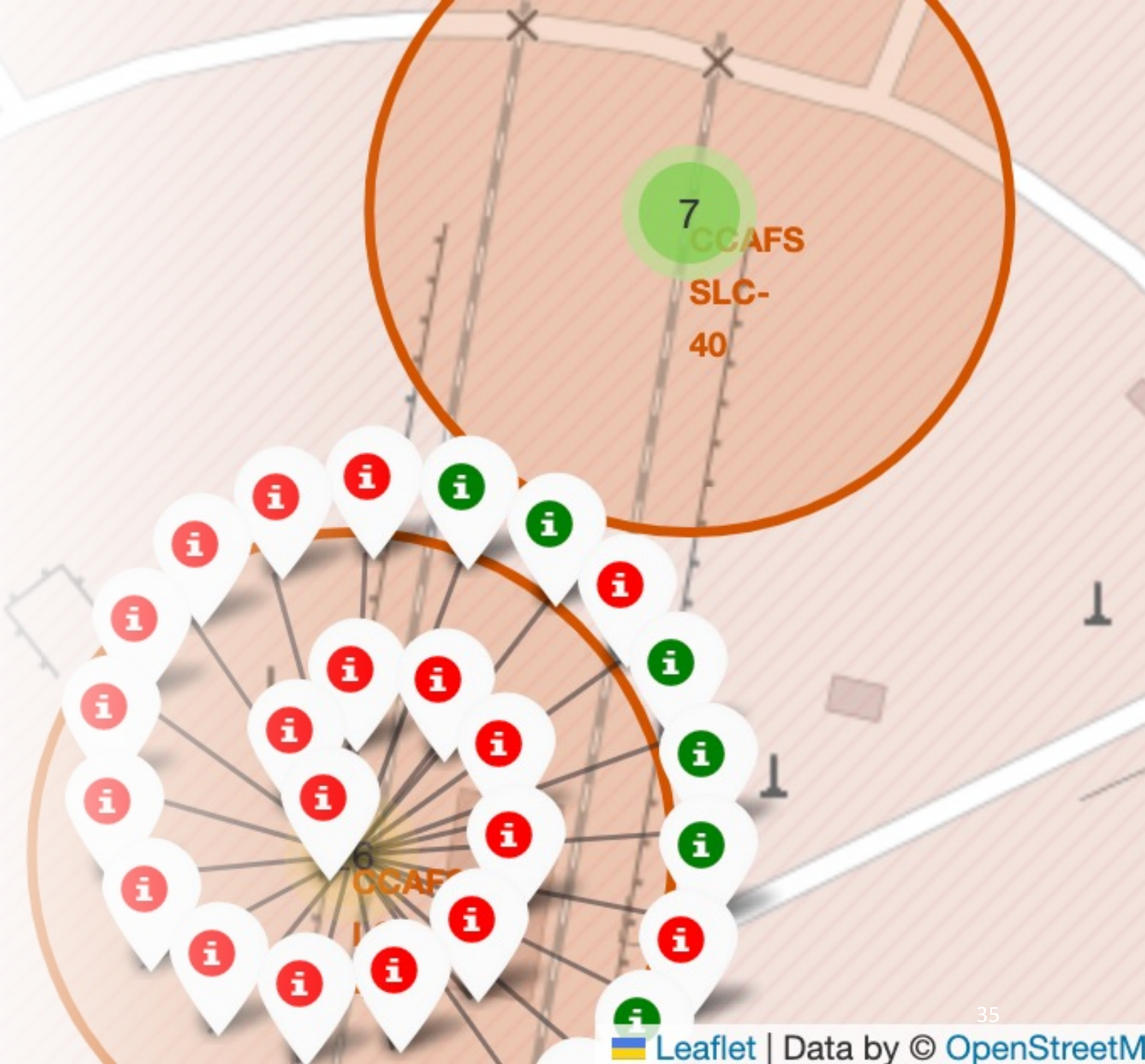

Section 3
Launch Sites
Proximities Analysis

All Launch Site Markers

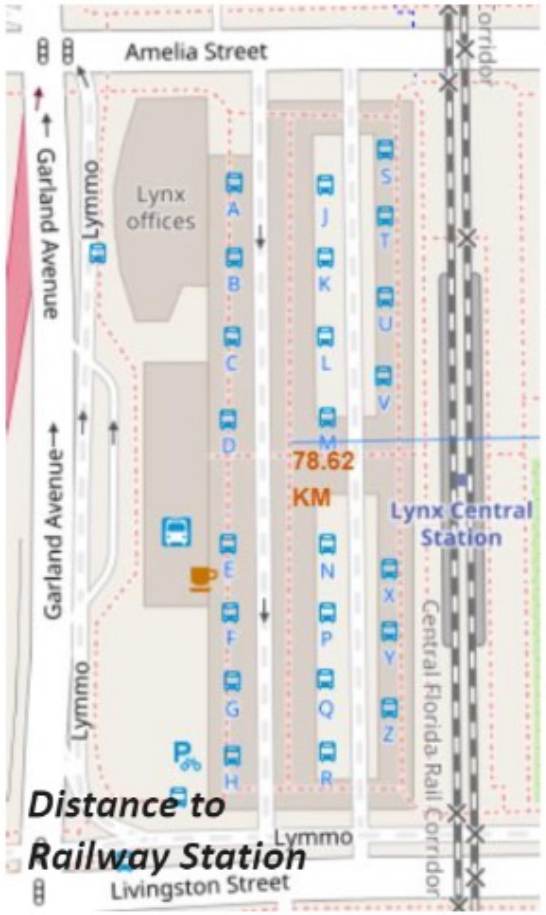
- We can see that the SpaceX launch sites are located in the East and West coast of the United States of America

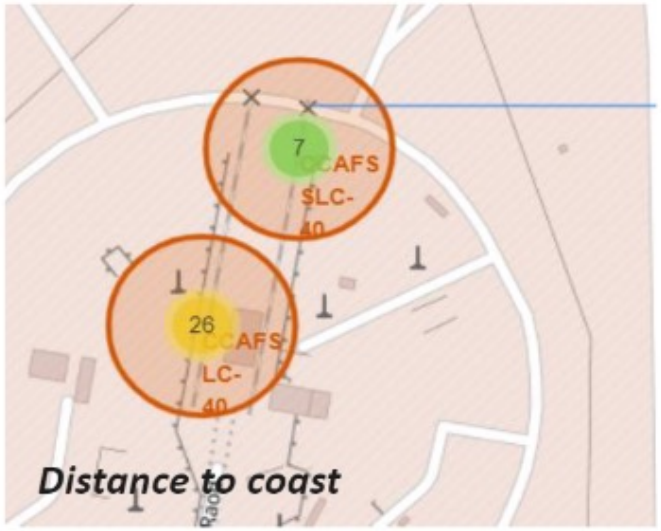
Launch Sites with Color Labels

- Florida launch sites showing green markers for successful launches and red for failures

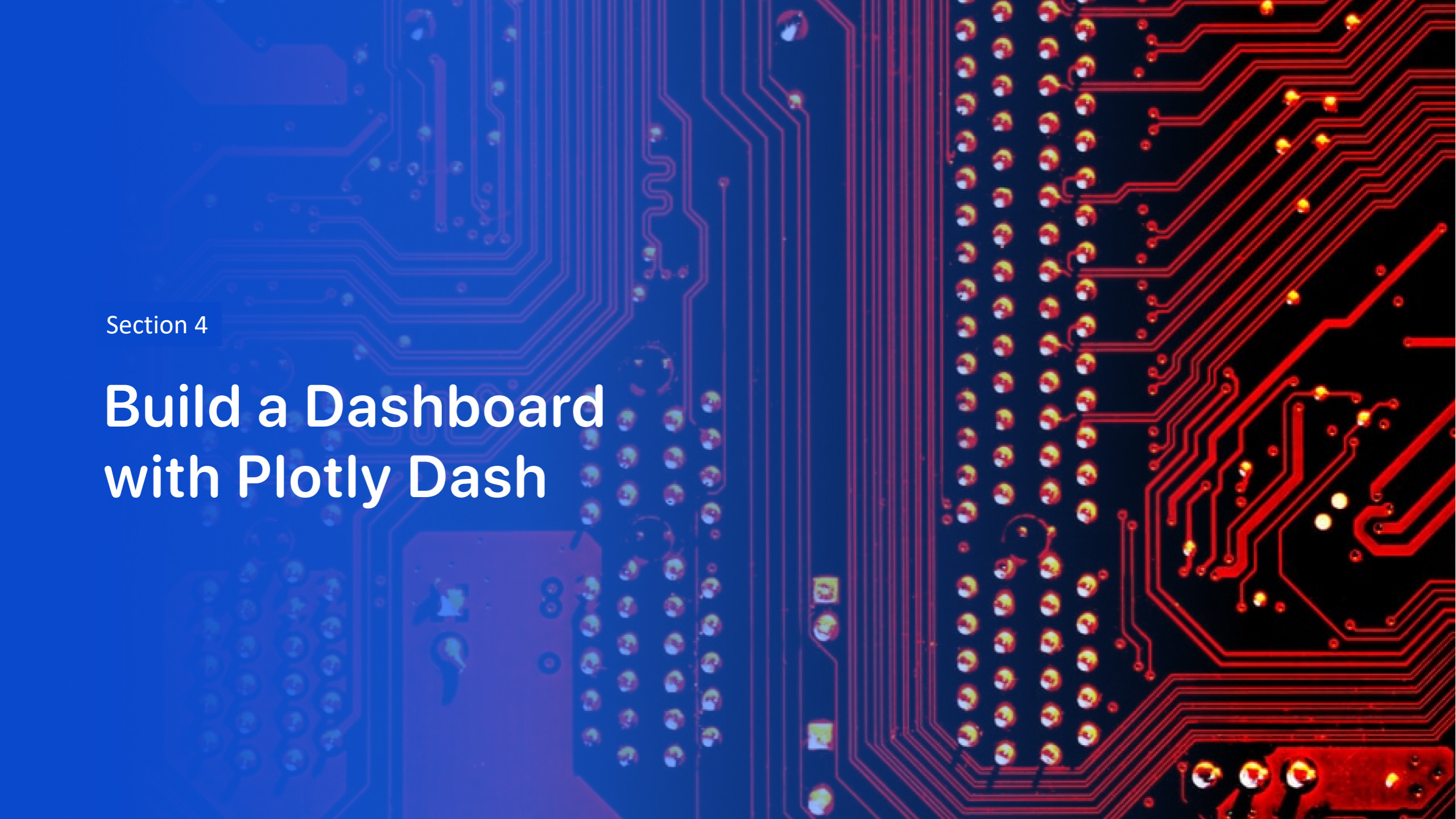


Launch Site distance to landmarks



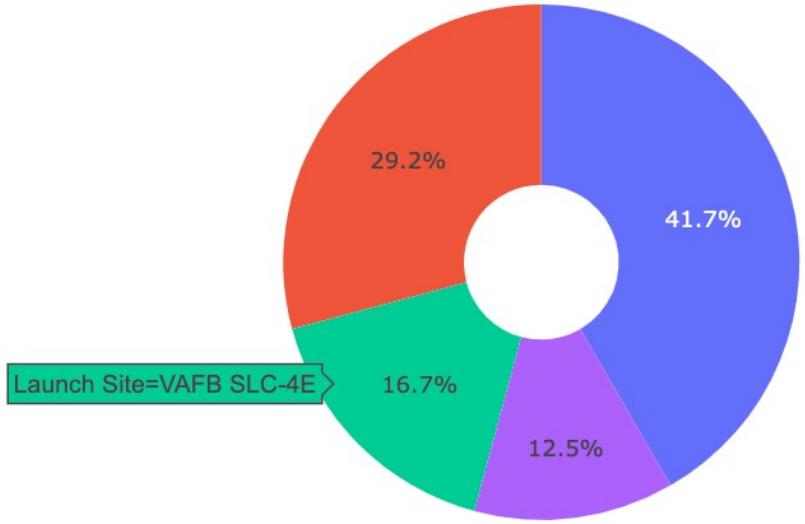


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes


Section 4
Build a Dashboard with Plotly Dash

SpaceX Launch Records Dashboard

Success Launches By all sites

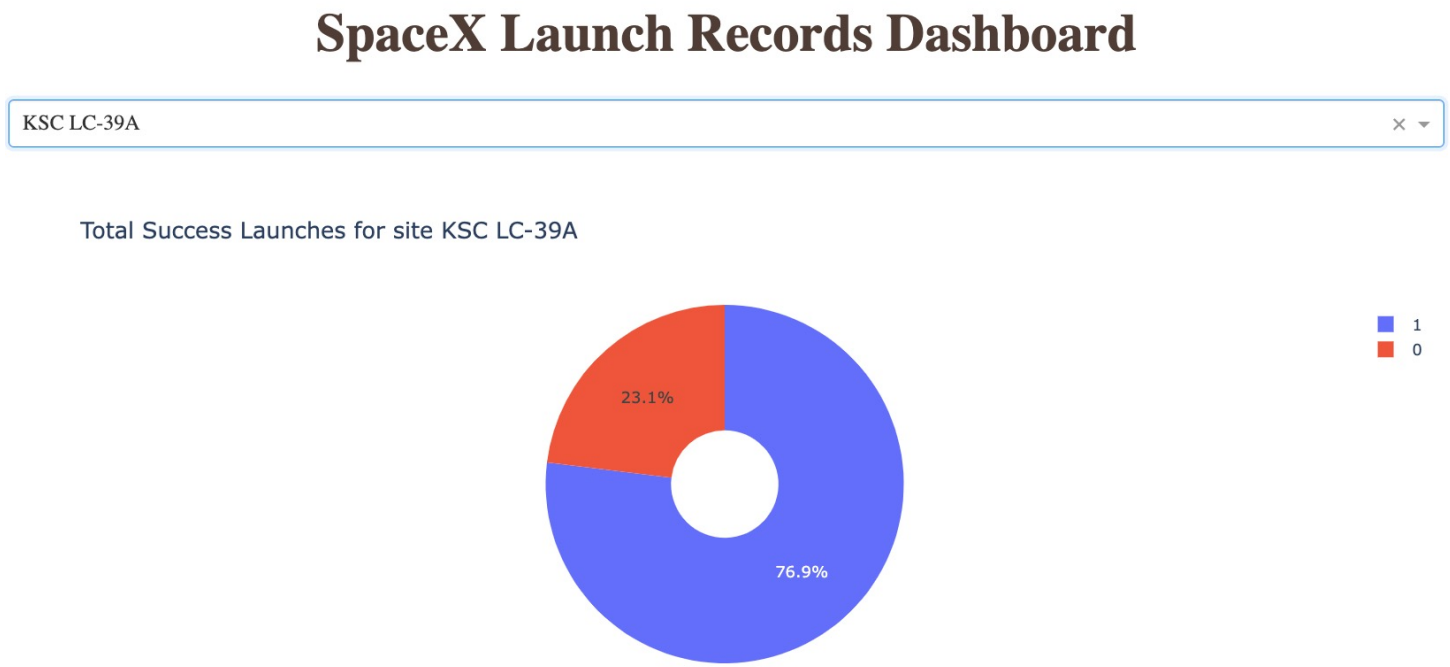


The Success Percentage Achieved by Each Launch Site

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

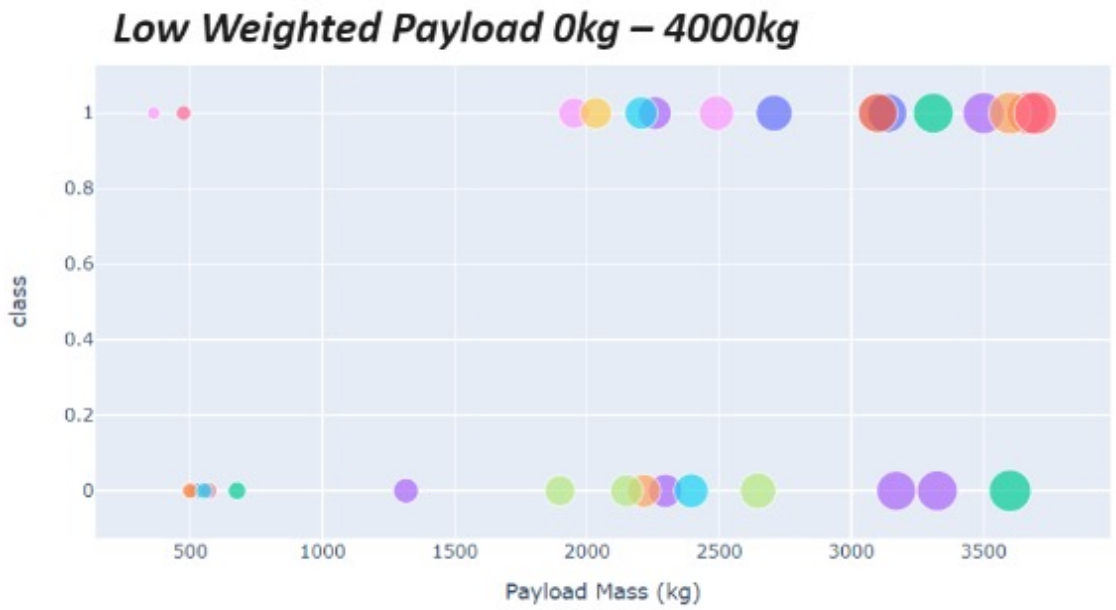
Highest Launch Success Ratio

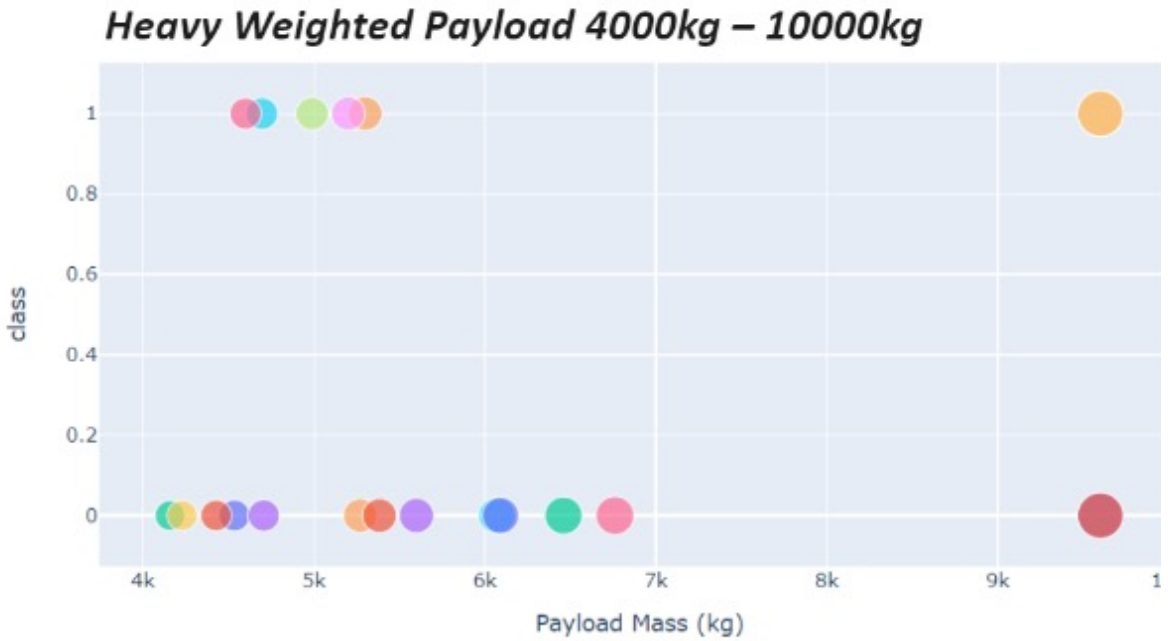
- KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate



Payload vs Launch Outcome for all Sites, with Different Payload Selected in the Range Slider

- We can see the success rates for low weighted payloads is higher than the heavy weighted payloads





Section 5

Predictive Analysis (Classification)

Classification Accuracy

Find the method performs best:

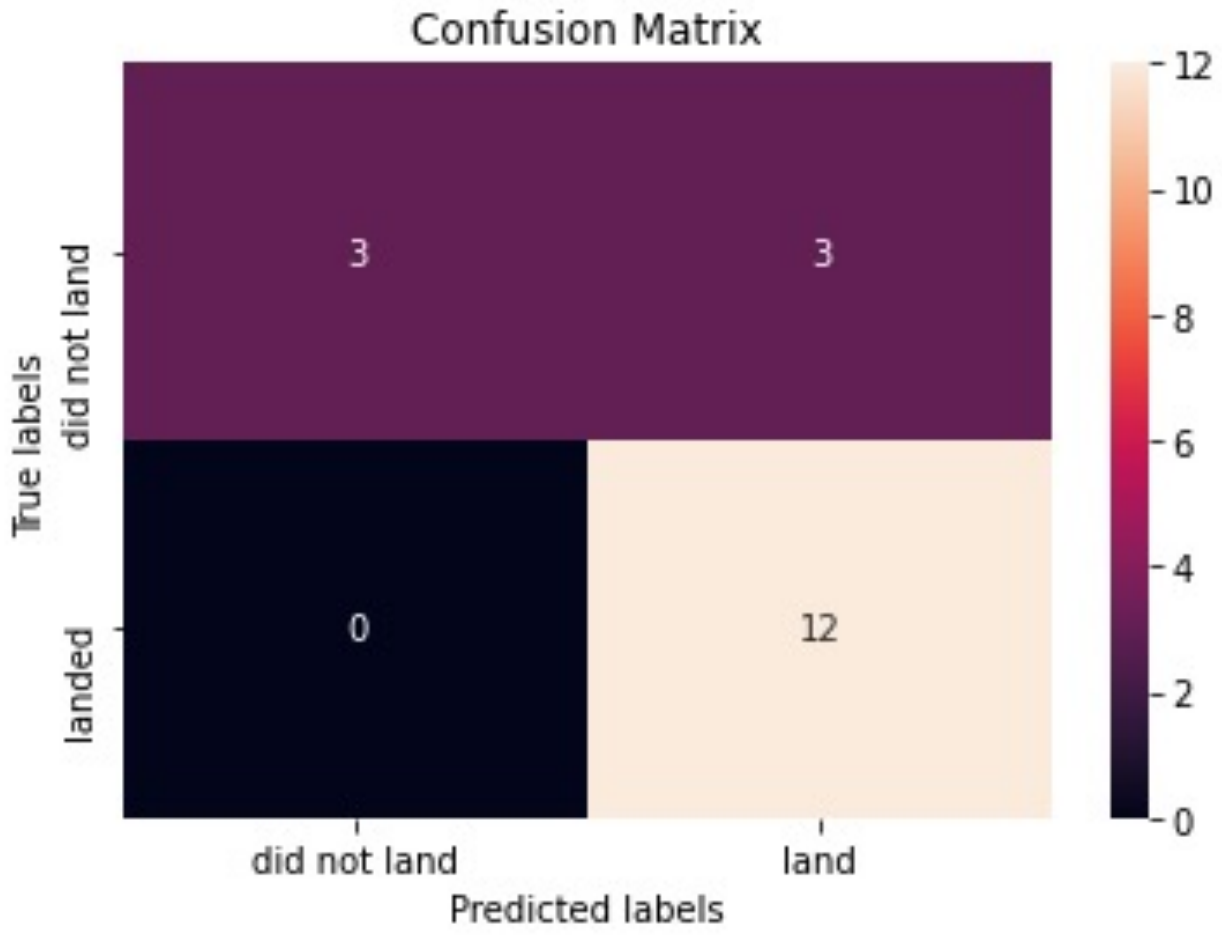
```
models = {'KNeighbors':knn_cv.best_score_,
          'DecisionTree':tree_cv.best_score_,
          'LogisticRegression':logreg_cv.best_score_,
          'SupportVector': svm_cv.best_score_}

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_)
```

7] ✓ 0.0s Python

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix



- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.

Conclusions

- We can conclude that:
- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!