IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Arsal Raza

18th November 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:
  - 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 SQL and Visualization.
- Perform interactive visual analytics using Folium and Plotly Dash.
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- 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/Arsal-Raza/SpaceX-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

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

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

Check the content of the response

```
[8]: print(response.content)
```

b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/N •••

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

```
[13]: # 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 that have multiple payloads in a single
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 feature.
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)]
```

Task 3: Dealing with Missing Values

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.

```
[30]: # 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'].replace(np.nan, mean_payload_mass, inplace=True)
```

Data Collection - Scraping

- We applied web scrapping to scrap Falcon 9 launch records with request and BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is https://github.com/Arsal-Raza/SpaceX-Data-Science-Capstone/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
     # assign the response to a object
     response = requests.get(static_url)
```

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

```
[10]: 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:
          name = extract_column_from_header(th)

          # Append the Non-empty column name to the list called column_names
          if name is not None and len(name) > 0:
              column_names.append(name)
```

Check the extracted column names

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

```
[15]: df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

We can now export it to a **CSV** for the next section, but to make the answers consistent and in case you have difficulties finishing this lab.

Following labs will be using a provided dataset to make each lab independent.

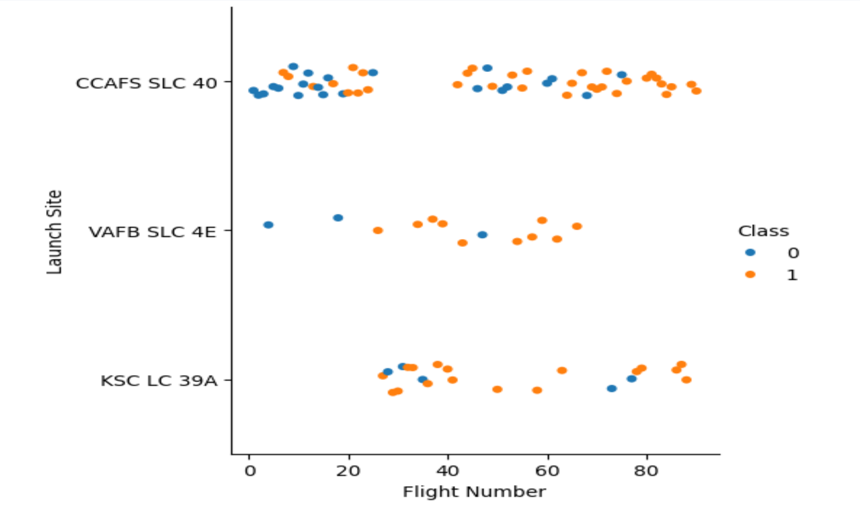
```
[16]: df.to_csv('spacex_web_scraped.csv', index=False)
```

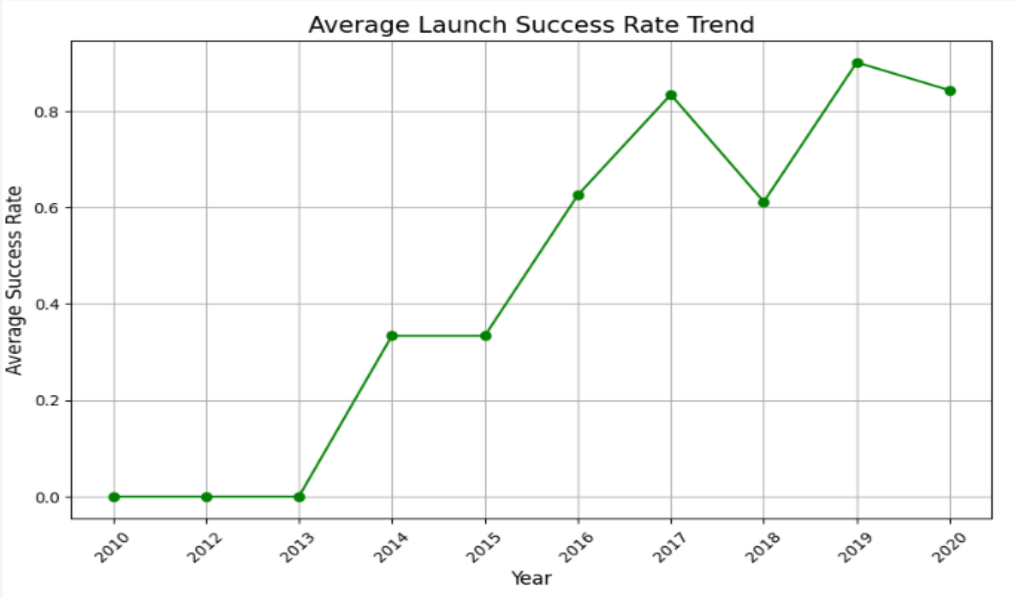
Data Wrangling

- 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/Arsal-Raza/SpaceX-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

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/Arsal-Raza/SpaceX-Data-Science-Capstone/blob/main/jupyter-labs-eda-dataviz.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/Arsal-Raza/SpaceX-Data-Science-Capstone/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.
- The link to the notebook is https://github.com/Arsal-Raza/SpaceX-Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

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.
- We have also added dropdown to select launch site and a rangeslider to select min or max of payload in dashboard.
- The link to the notebook is https://github.com/Arsal-Raza/SpaceX-Data-Science-Capstone/blob/main/spacex_dash_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/Arsal-Raza/SpaceX-Data-Science-Capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

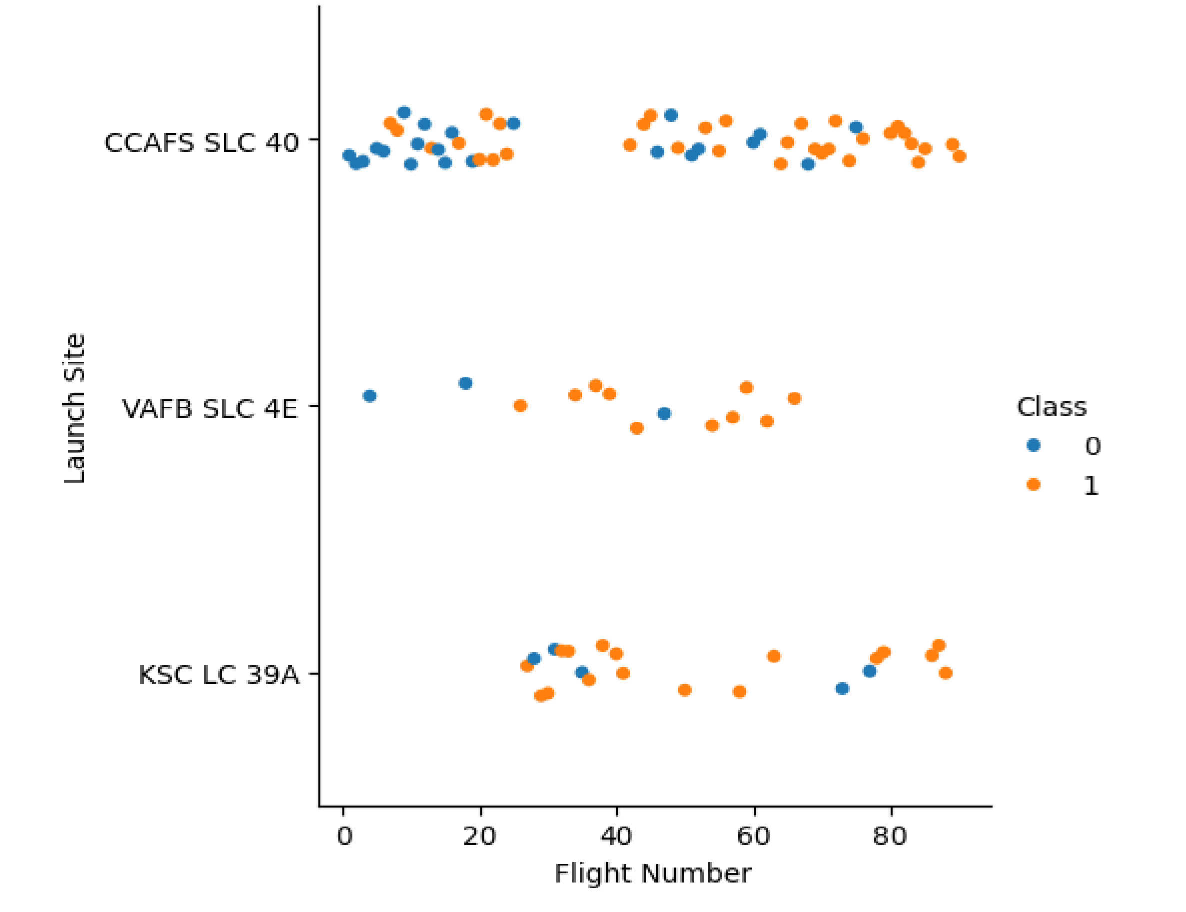
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

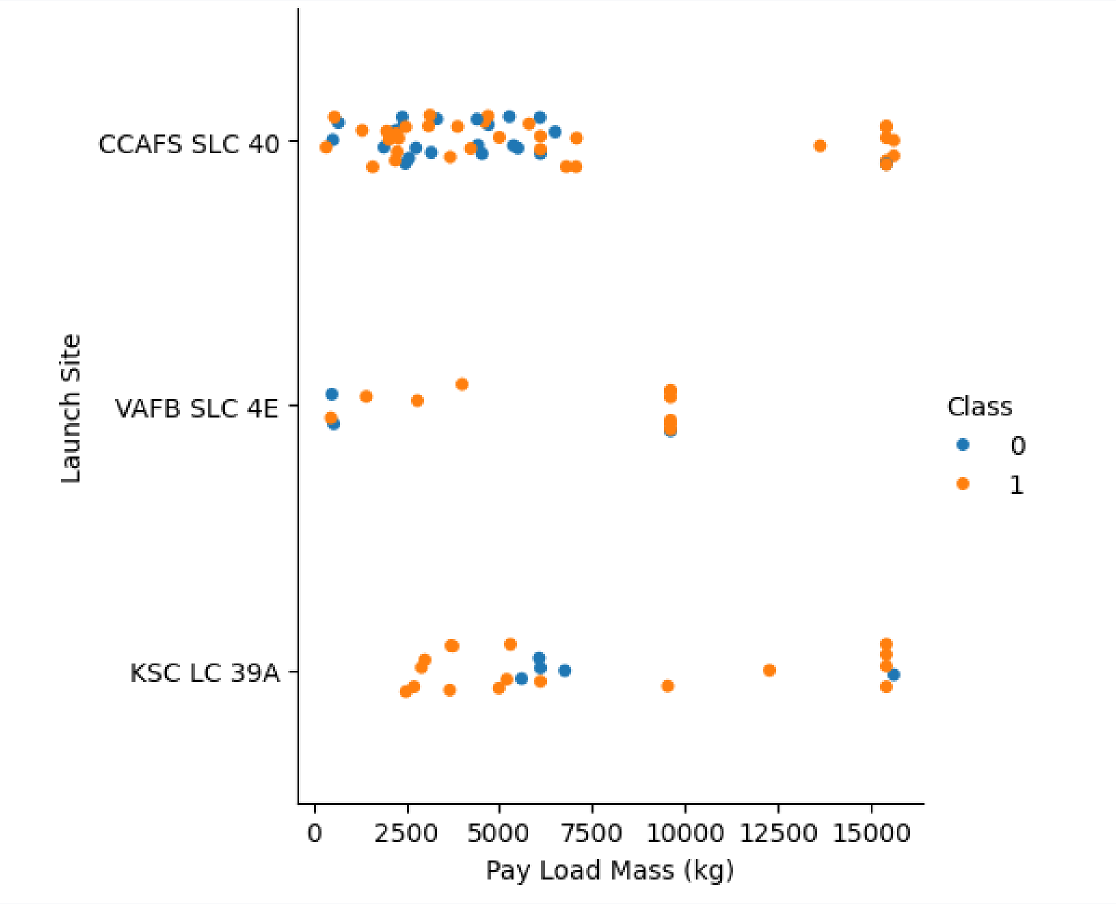
Flight Number vs. Launch Site

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



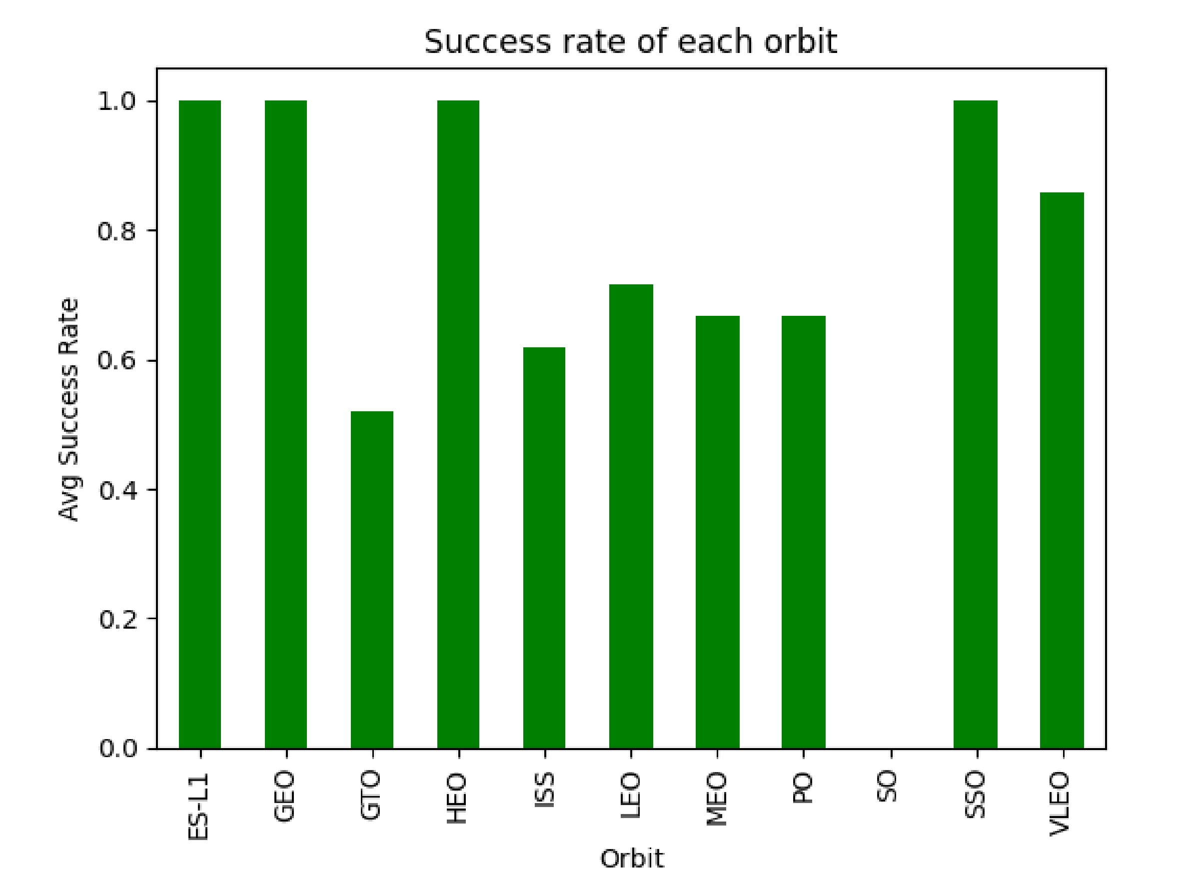
Payload vs. Launch Site

- We observe that in Payload Vs. Launch Site scatter point chart, VAFB-SLC launch site there are no rockets launched for heavypayload mass(greater than 10000).
- Also greater the payload mass for launch site the higher the success rate for rocket.



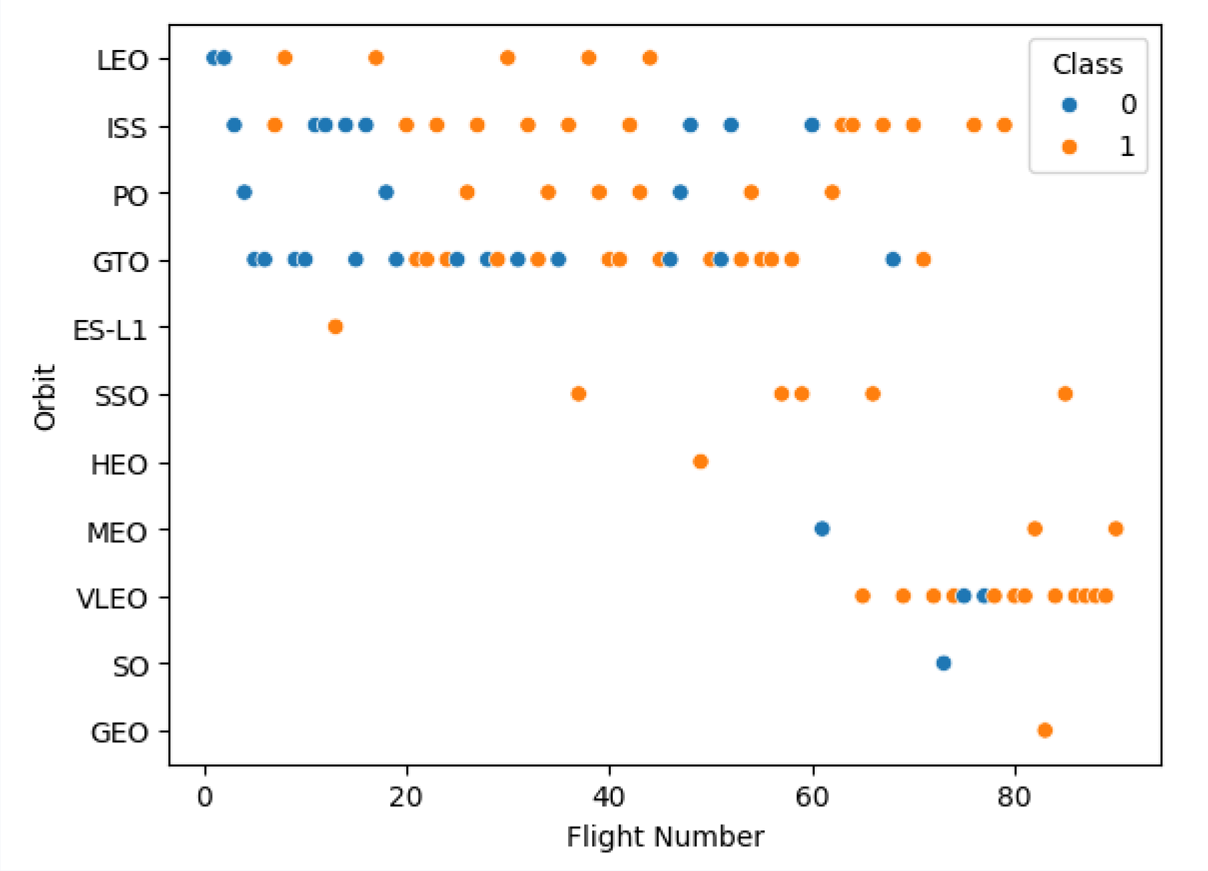
Success Rate vs. Orbit Type

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



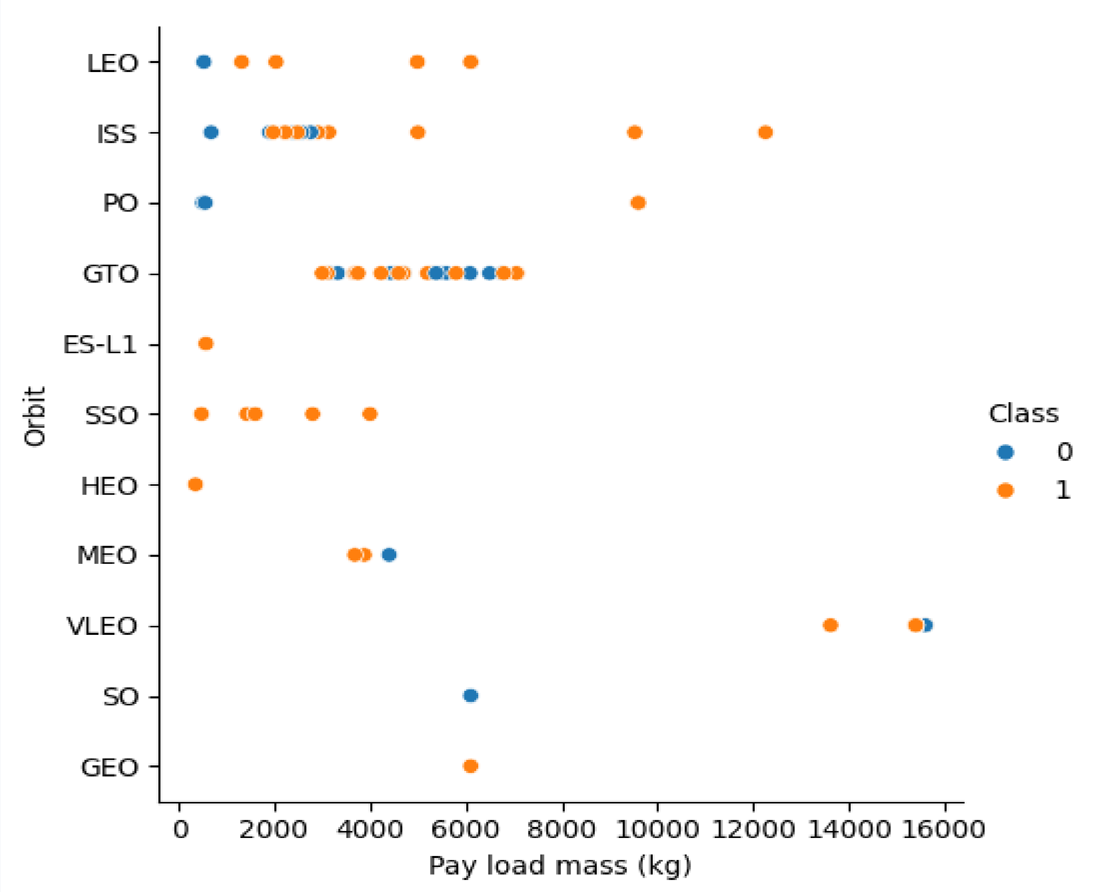
Flight Number vs. Orbit Type

- 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.



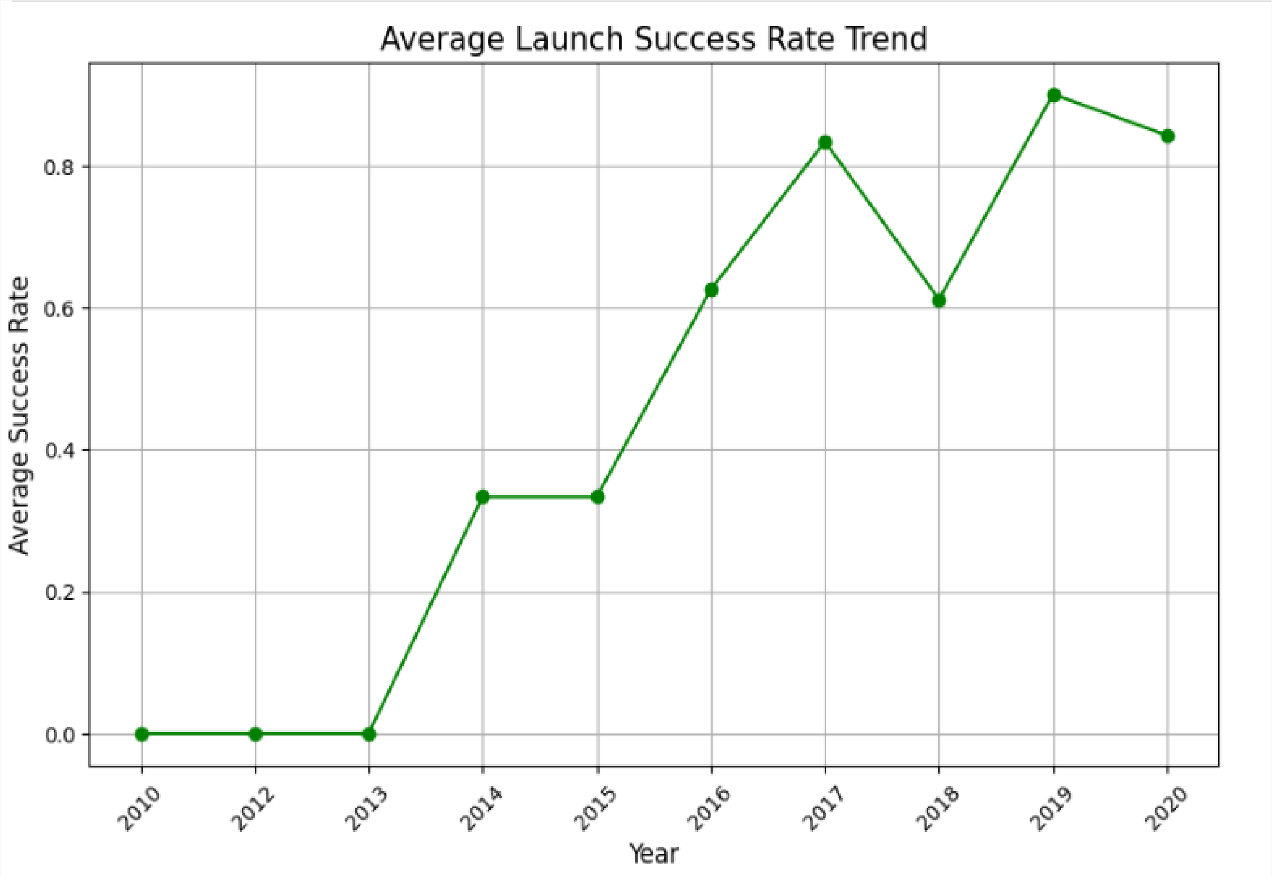
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.
- But for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there.



Launch Success Yearly Trend

- From the plot, we can observe that the success rate since 2013 kept increasing till 2017 (stable in 2014) and after 2015 it started increasing.



All Launch Site Names

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

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

In [8]:
```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;
```

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

Out[8]:

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

Launch Site Names Begin with 'CCA'

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

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

In [9]:
```
%sql SELECT "Launch_Site" FROM SPACEXTABLE WHERE "Launch_Site" LIKE '%CCA%' LIMIT 5;
```

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

Out[9]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

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

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

In [10]:
```
%sql SELECT SUM("PAYLOAD_MASS__KG_"),Customer FROM SPACEXTABLE WHERE Customer='NASA (CRS)';
```

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

Out[10]:

SUM("PAYLOAD_MASS__KG_")	Customer
45596	NASA (CRS)

Average Payload Mass by F9 v1.1

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

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

In [11]:
```
%sql SELECT AVG("PAYLOAD_MASS__KG_"),"Booster_Version" FROM SPACEXTABLE WHERE "Booster_Version" ='F9 v1.1';
```

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

Out[11]:

AVG("PAYLOAD_MASS__KG_")	Booster_Version
2928.4	F9 v1.1

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015 by using min function.

In [12]:

```
%sql SELECT MIN(Date),"Landing_Outcome" FROM SPACEXTABLE WHERE "Landing_Outcome" ='Success (ground pad)';
```

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

Out[12]:

MIN(Date)	Landing_Outcome
2015-12-22	Success (ground pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

- 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

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

```
In [13]: %sql SELECT Payload,"PAYLOAD_MASS__KG_","Landing_Outcome" FROM SPACEXTABLE WHERE "Landing_Outcome" ='Success (drone ship)' A
```

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

Out[13]:

Payload	PAYLOAD_MASS__KG_	Landing_Outcome
JCSAT-14	4696	Success (drone ship)
JCSAT-16	4600	Success (drone ship)
SES-10	5300	Success (drone ship)
SES-11 / EchoStar 105	5200	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- We used wildcard like ‘%’ to filter for **WHERE** Mission Outcome was a success or a failure.

List the total number of successful and failure mission outcomes

```
In [14]: %sql SELECT "Mission_Outcome", COUNT(*) as "Count" FROM SPACEXTABLE WHERE "Mission_Outcome" = 'Success' OR "Mission_Outcome'

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

Out[14]:

Mission_Outcome	Count
Success	98

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

In [15]:
```
%sql SELECT DISTINCT "Booster_Version" FROM SPACEXTABLE WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SI
```

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

Out[15]:

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

- 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

```
In [16]: %sql SELECT substr("Date", 6, 2) as Month, "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE substr
```

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

Out[16]:

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

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.

```
In [17]: %sql SELECT "Landing_Outcome", COUNT(*) as "Count" FROM SPACEXTABLE WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' GROU

 * sqlite:///my_data1.db
Done.
Out[17]:
```

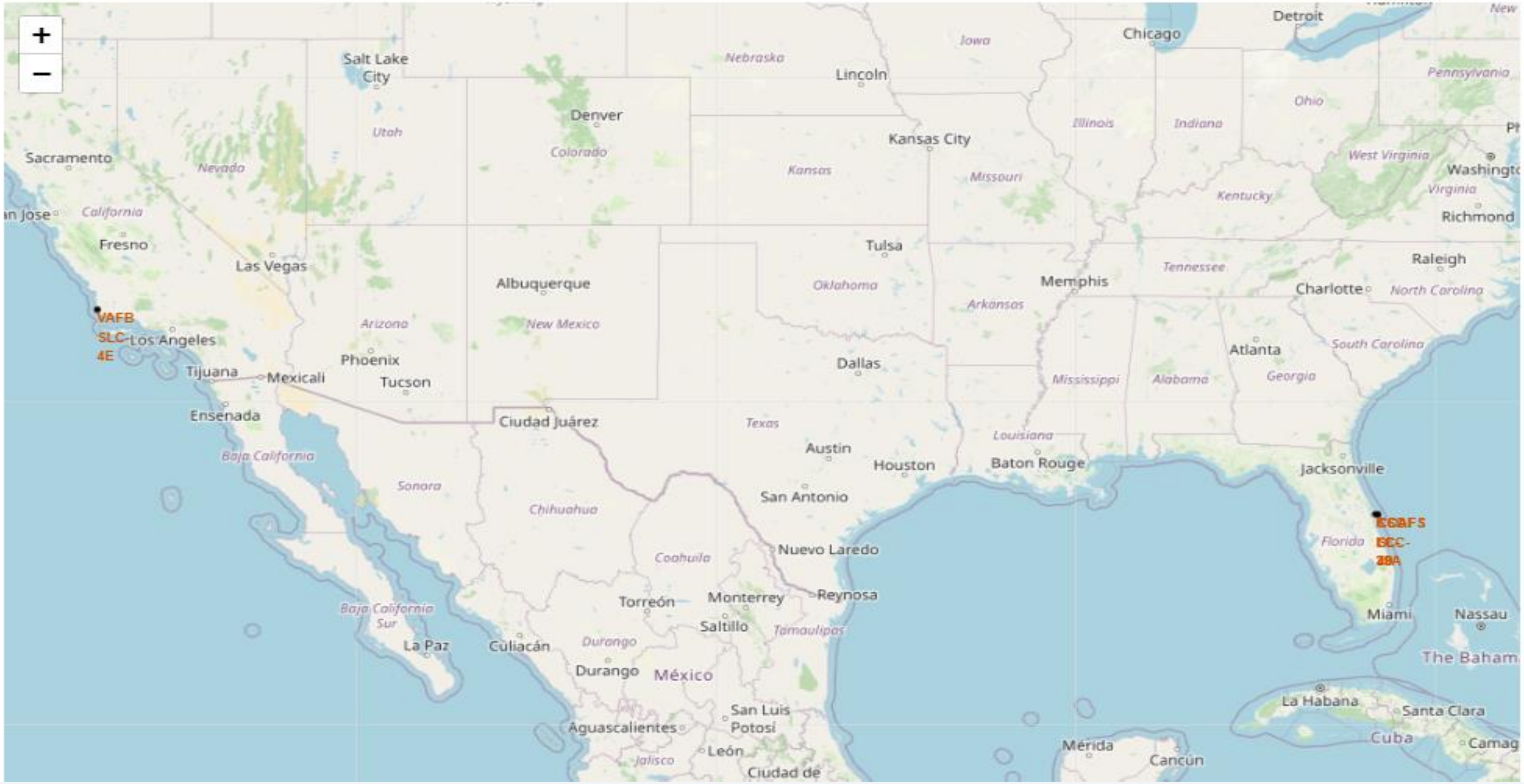
Landing_Outcome	Count
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

Section 3

Launch Sites Proximities Analysis

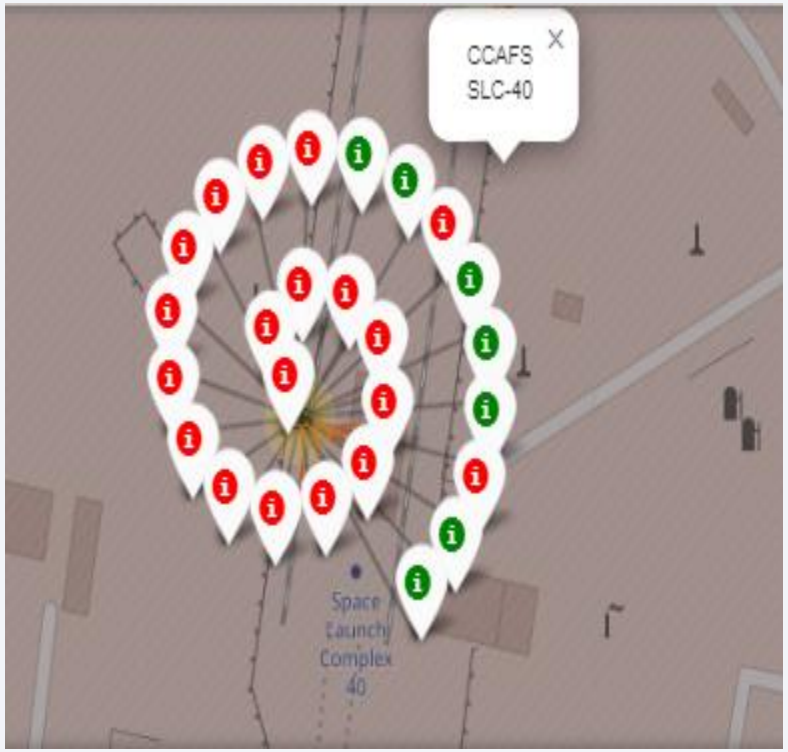
All launch sites global map markers

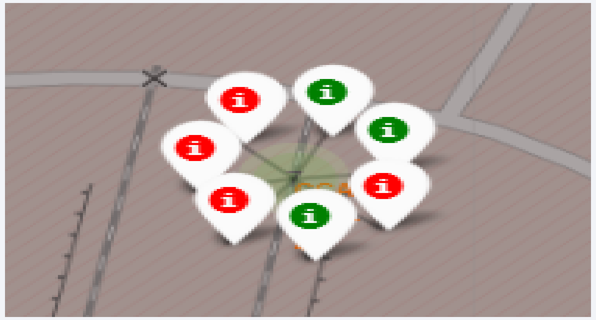
[8]:



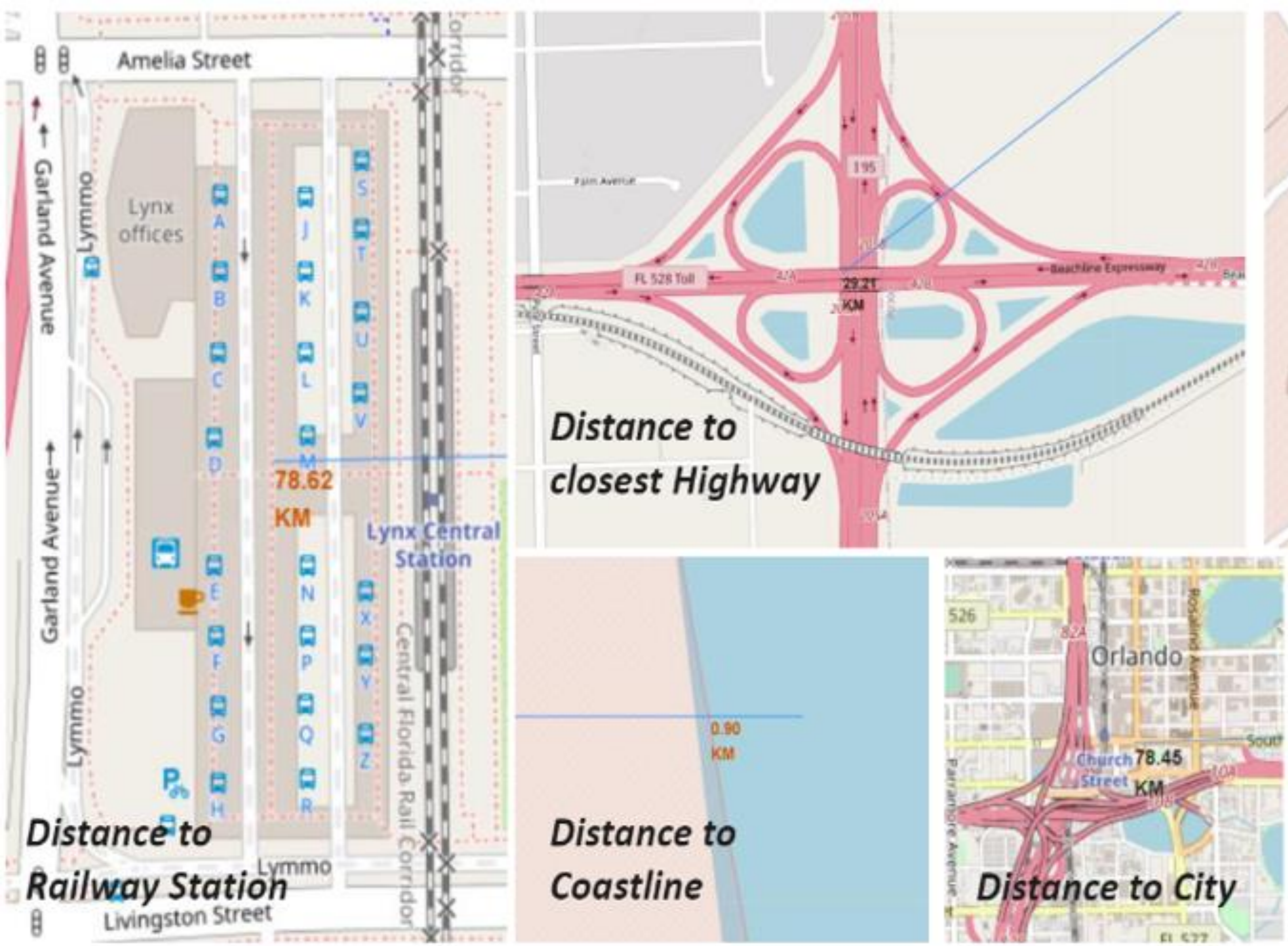
Markers showing launch sites with color labels

Green Markers show successful launches and Red Markers show failure on each site.





Launch Site distance to landmarks



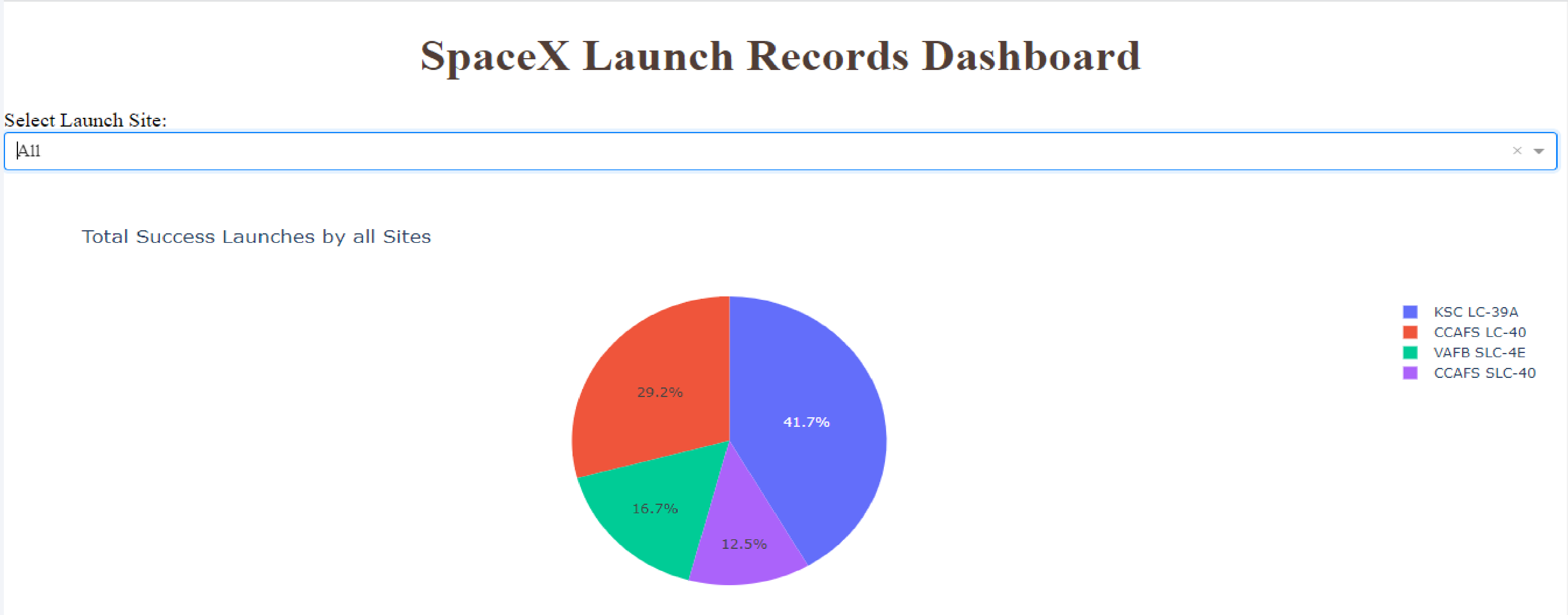
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

Build a Dashboard with Plotly Dash

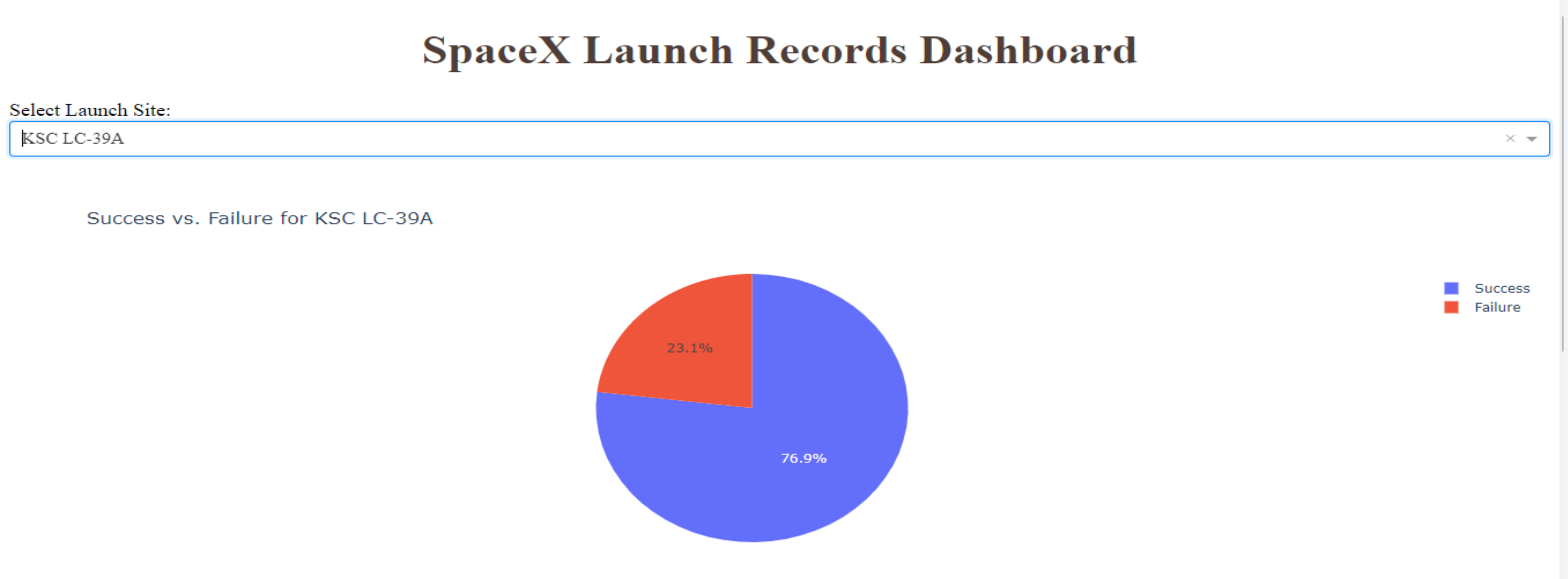
Pie chart showing the success percentage achieved by each launch site

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



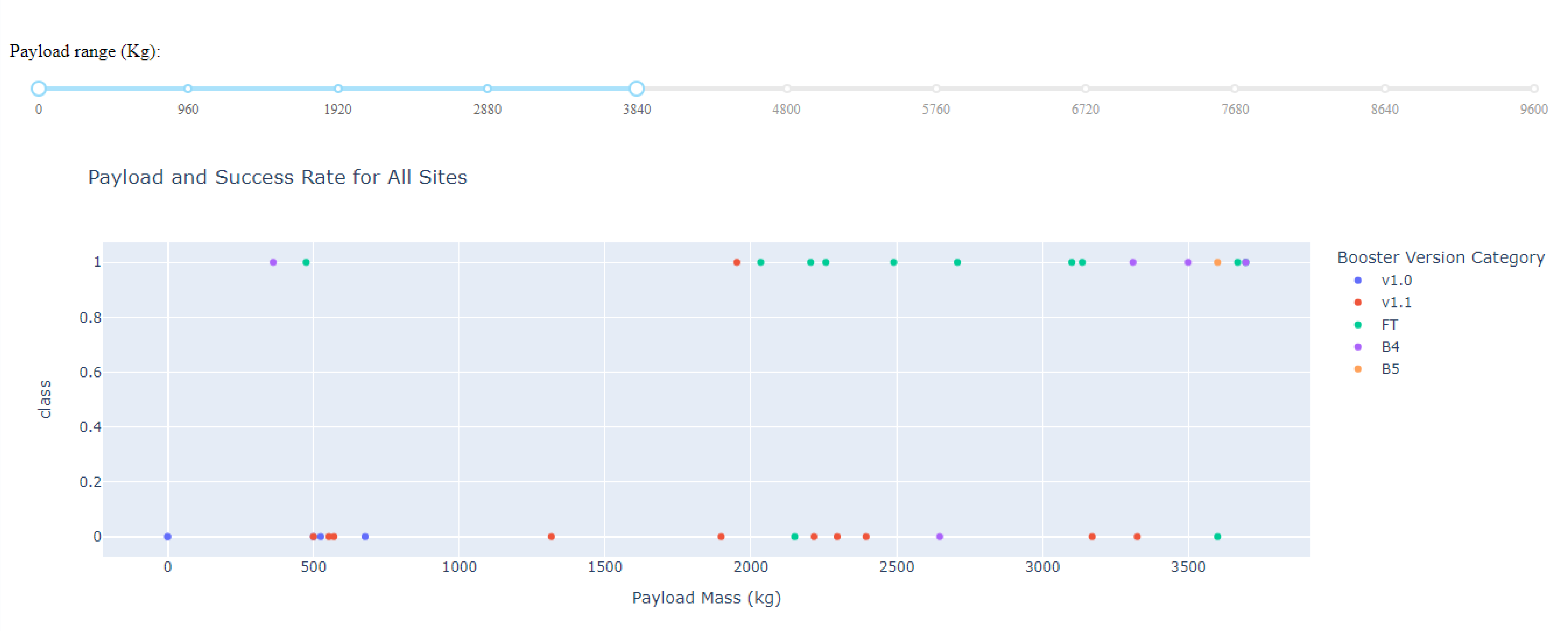
Pie chart showing the Launch site with the highest launch success ratio

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



Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

- Low weighted payload 0 Kg - 3880 Kg



- Heavy weighted payload 3880 Kg – 9600 Kg

- We can see that success rate for low weighted payloads is higher than heavy weighted payloads.

Section 5

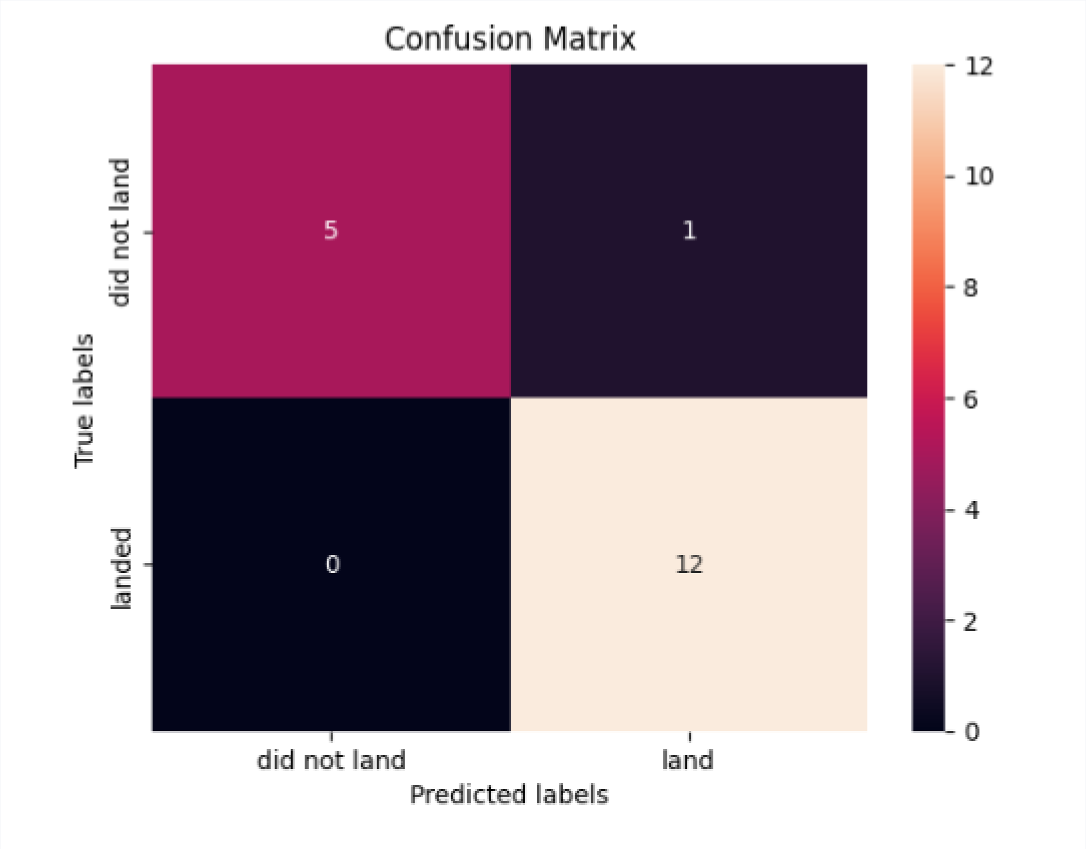
Predictive Analysis (Classification)

Classification Accuracy ➡ The decision tree classifier is the model with the highest classification accuracy

```
[32]: #Lets see who perform best in term of best score of model

# Accuracy for Logistic Regression
logreg_accuracy = logreg_cv.best_score_

# Accuracy for Support Vector Machine
svm_accuracy = svm_cv.best_score_

# Accuracy for Decision Tree
tree_accuracy = tree_cv.best_score_

# Accuracy for Decision Tree
knn_accuracy = knn_cv.best_score_

# Creating a dictionary to map the model names to their respective scores
model_scores = {'Logistic Regression': logreg_accuracy,
                'Support Vector Machine': svm_accuracy,
                'Decision Tree': tree_accuracy,
                'K-Nearest Neighbors': knn_accuracy}

# Finding the model with the highest score
best_model = max(model_scores, key=model_scores.get)
best_score = model_scores[best_model]
best_parameter = tree_cv.best_params_

# Printing the best performing model and its score
print(f"The best performing model is {best_model} with a score of {best_score}.")
print(best_parameter)

The best performing model is Decision Tree with a score of 0.8777777777777779.
{'criterion': 'gini', 'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'best'}
```

Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The small problem is the false positive i.e one 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!