IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Darren

17th June 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

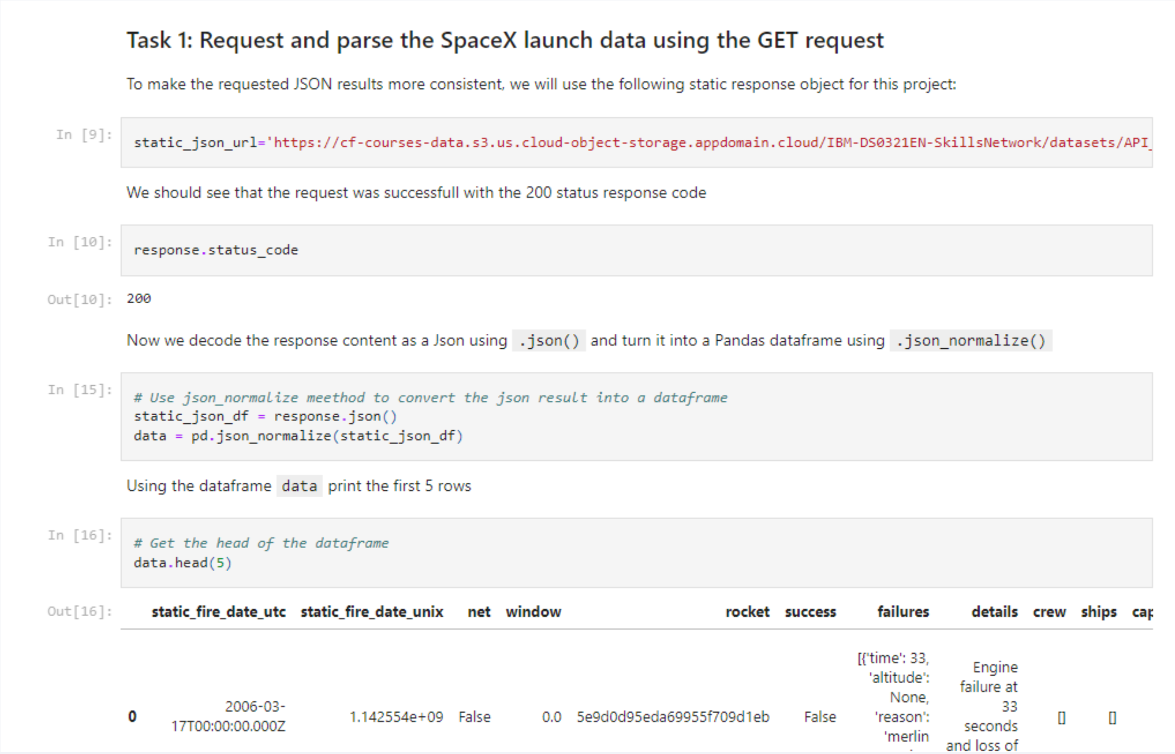
- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- 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.
- The link to the notebook is https://github.com/darrendoang/IBM-Data-Science-Capstone-SpaceX/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:

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_
```

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

In [10]:
```
response.status_code
```

Out[10]: 200

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

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

Using the dataframe `data` print the first 5 rows

In [16]:
```
# Get the head of the dataframe
data.head(5)
```

Out[16]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	cap
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin	Engine failure at 33 seconds and loss of	[]	[]	

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- The link to the notebook is https://github.com/darrendoang/IBM-Data-Science-Capstone-SpaceX/blob/main/jupyter-labs-webscraping.ipynb

1. Apply HTTP Get method to request the Falcon 9 rocket launch page

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

Out[5]:
```
200
```

2. Create a BeautifulSoup object from the HTML response

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

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

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract all column names from the HTML table header

In [10]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

Data Wrangling

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

In [12]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
#landing_class = [x for x in bad_outcomes if df['Outcome'][x] ]

landing_class = []

for key, value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

In [13]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

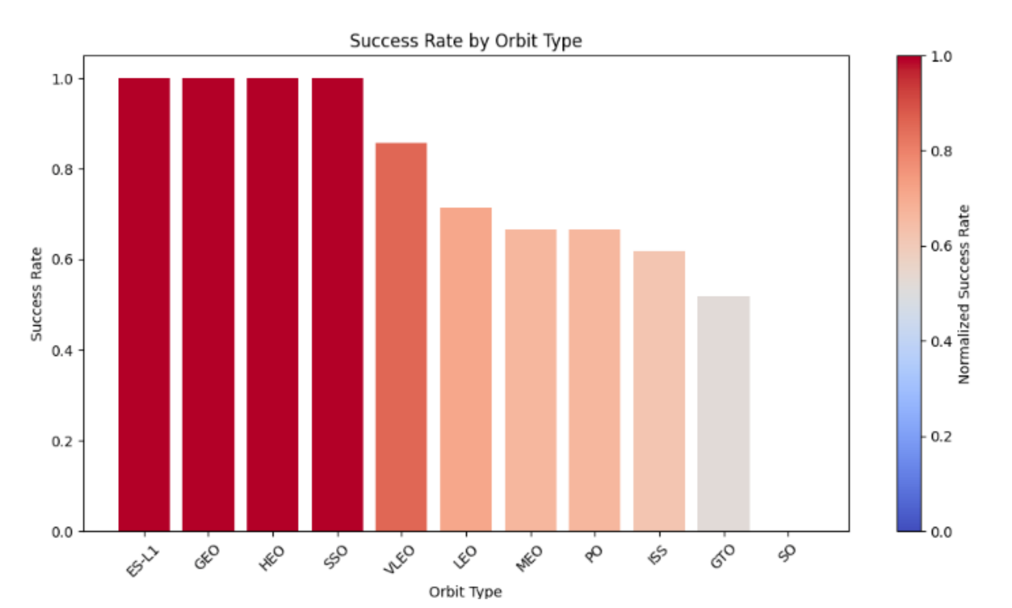
Out[13]:

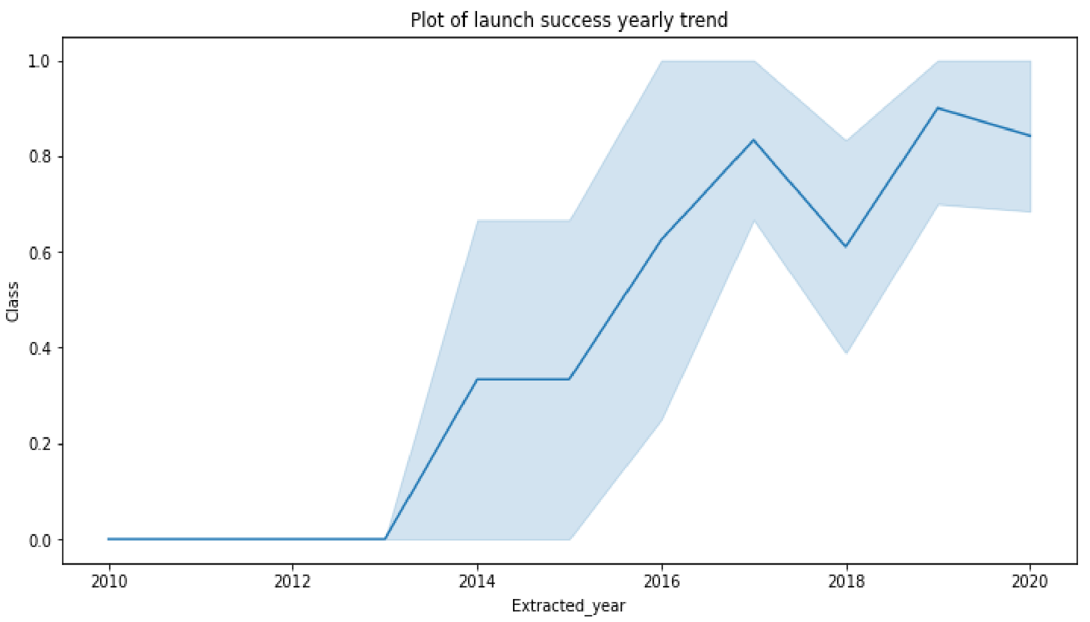
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

- We performed exploratory data analysis and determined the training labels and calculated the number of launches at each site, and the number and occurrence of each orbits
- We also created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/darrendoang/IBM-Data-Science-Capstone-SpaceX/blob/main/labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.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/darrendoang/IBM-Data-Science-Capstone-SpaceX/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 like first successful landing date and Total payload Mass.
- The link to the notebook is https://github.com/darrendoang/IBM-Data-Science-Capstone-SpaceX/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Total_PayloadMass
45596.0

Out[47]:

FirstSuccessfull_landing_date
22/12/2015

Build an Interactive Map with Folium

- We utilized a folium map to visualize launch sites and their corresponding success or failure outcomes. By assigning a value of 0 for failure and 1 for success, we marked the launch sites with markers, circles, and lines on the map to indicate their respective outcomes. We employed color-labeled marker clusters to identify launch sites with higher success rates. Additionally, we measured the distances between each launch site and its surrounding areas, exploring factors such as proximity to railways, highways, coastlines, and cities.,

- The link to the notebook is https://github.com/darrendoang/IBM-Data-Science-Capstone-SpaceX/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- We developed an interactive dashboard using Plotly Dash. Within the dashboard, we included pie charts that visualize the total number of launches from specific sites. Additionally, we created scatter graphs to examine the relationship between the launch outcome and the payload mass (in kilograms) for various booster versions.
- The link to the notebook is https://github.com/darrendoang/IBM-Data-Science-Capstone-SpaceX/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- We imported the necessary libraries such as NumPy and Pandas to load and preprocess the data. After loading the data, we performed data transformations and split it into training and testing sets.
- Next, we constructed various machine learning models and utilized GridSearchCV to tune different hyperparameters. We used accuracy as the evaluation metric for our models. To enhance the performance of the models, we applied feature engineering techniques and fine-tuned the algorithms.
- Finally, by evaluating the models based on their accuracy, we identified the best-performing classification model among the ones we built. The link to the notebook is https://github.com/darrendoang/IBM-Data-Science-Capstone-SpaceX/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

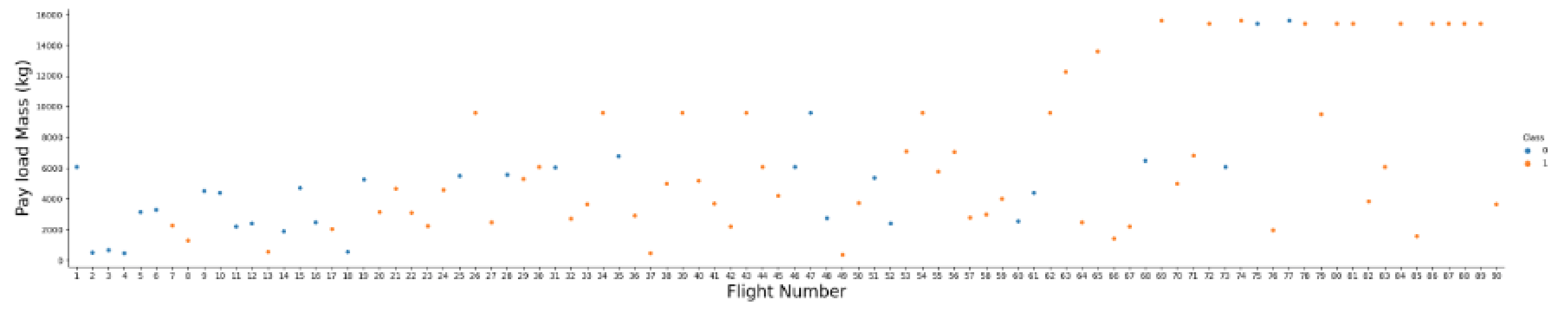
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

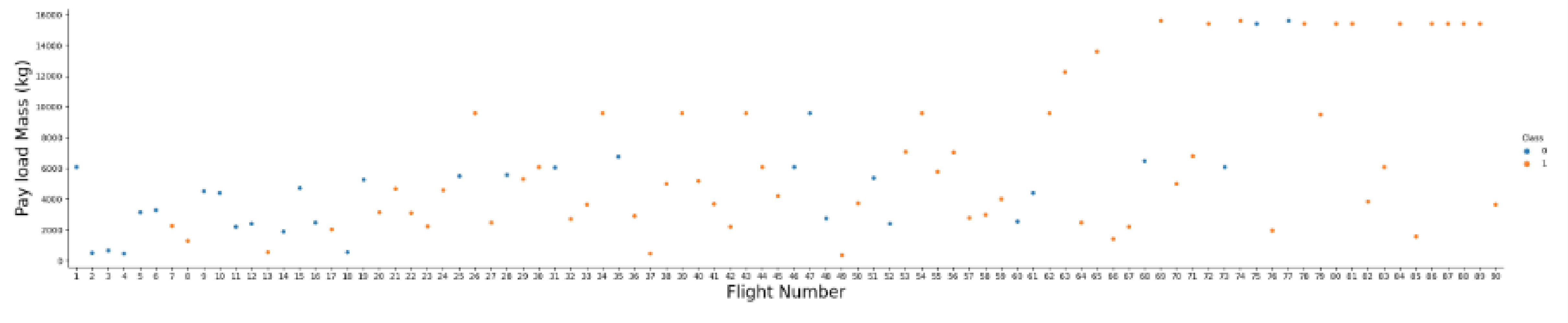
Flight Number vs. Launch Site

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



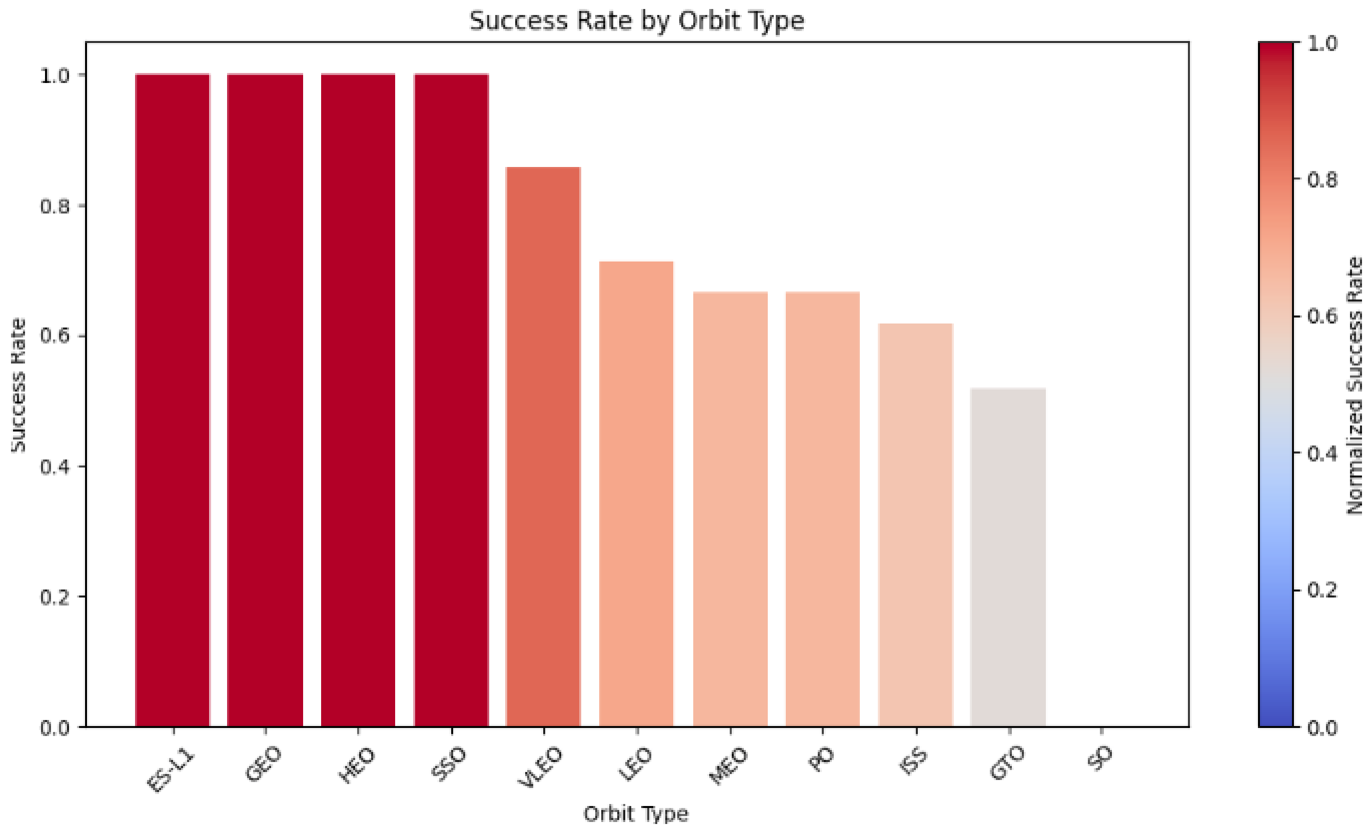
Flight Number vs. Payload Mass

- We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.



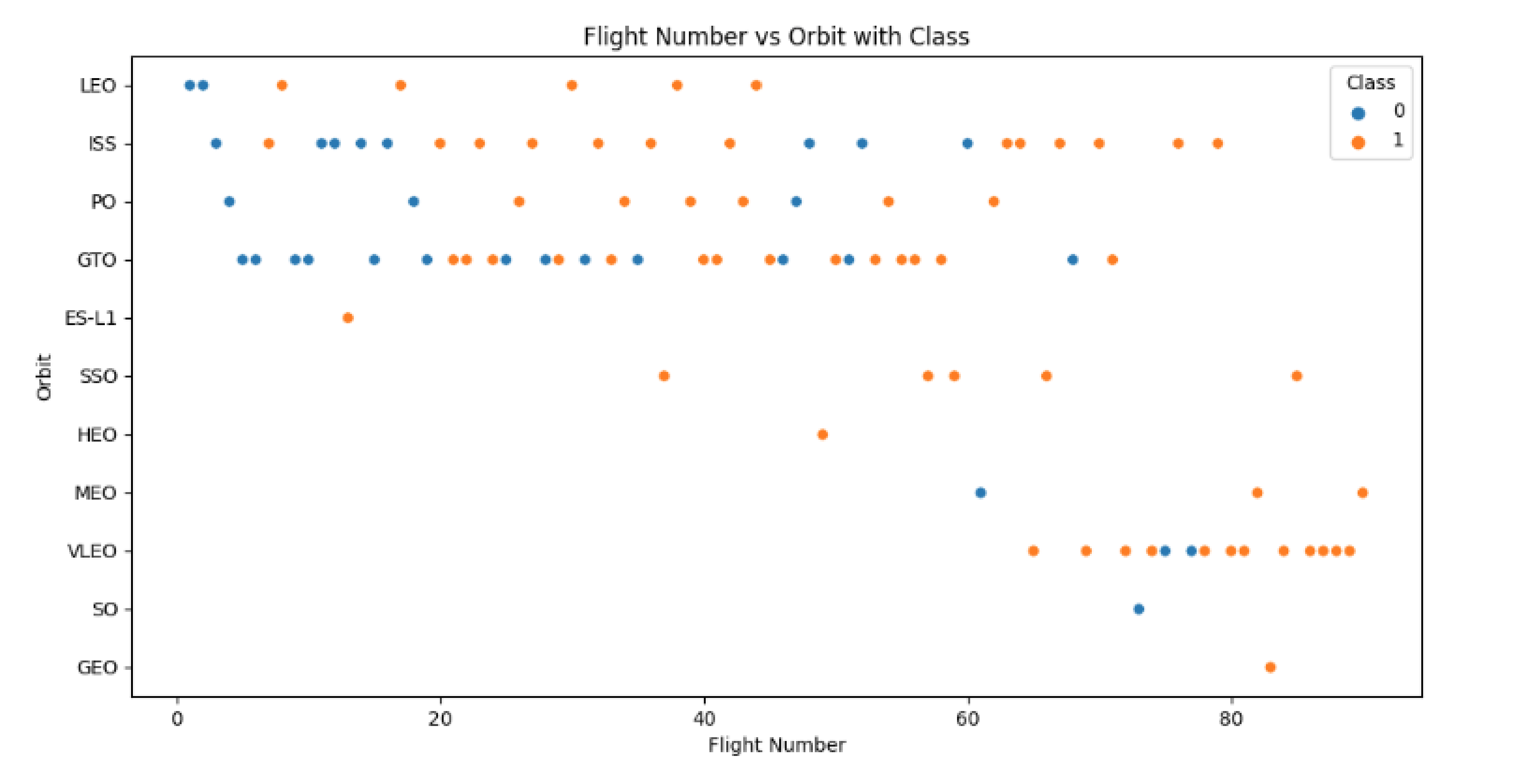
Success Rate vs. Orbit Type

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



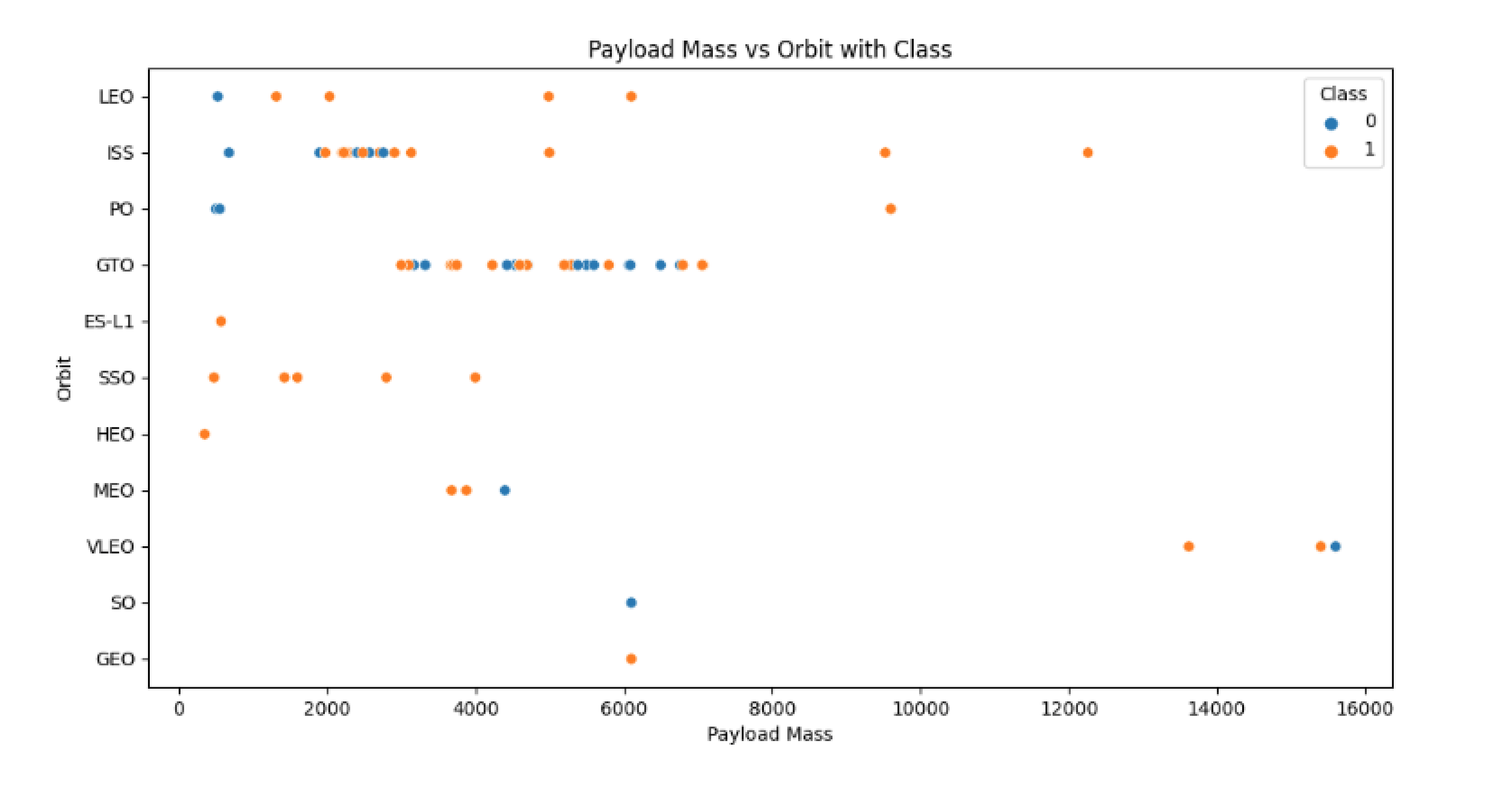
Flight Number vs. Orbit Type

- 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.



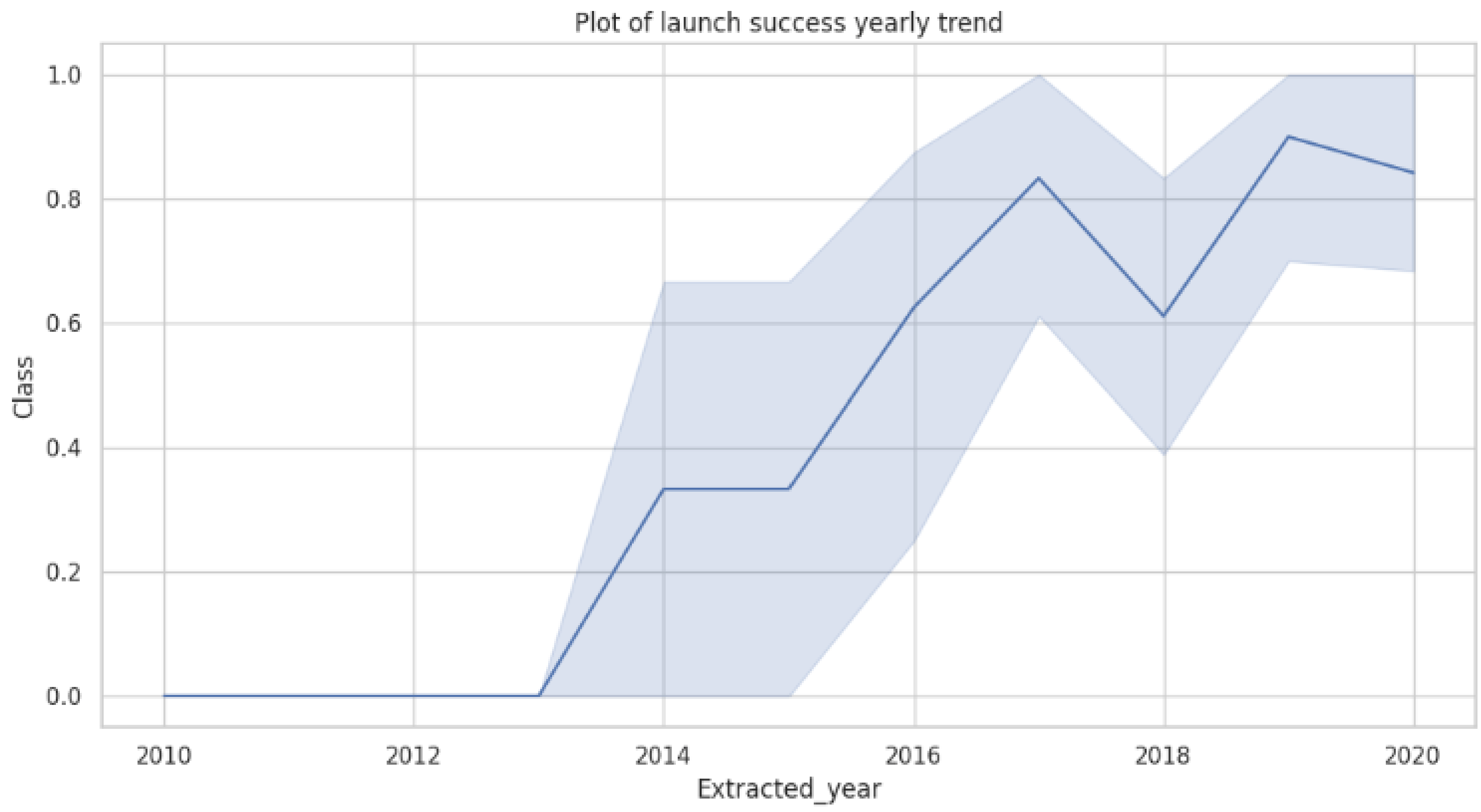
Payload vs. Orbit Type

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



Launch Success Yearly Trend

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



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

```
%%sql SELECT DISTINCT LAUNCH_SITE
        FROM SPACEXTBL ;
```

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

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

Launch Site Names Begin with 'CCA'

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

In [23]:

```
%%sql
SELECT * from SPACEXTBL where Launch_Site LIKE 'CCA%' limit 5;
```

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

Out[23]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (paracl
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 (paracl
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 att
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No att
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No att

Total Payload Mass

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

Task 3

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

```
%%sql  SELECT SUM(PAYLOAD_MASS__KG_) AS Total_PayloadMass
        FROM SpaceXTBL
        WHERE Customer LIKE 'NASA (CRS)'
```

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

Total_PayloadMass
45596.0

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

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

Out[30]:

Avg_PayloadMass
2928.4

First Successful Ground Landing Date

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

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

Hint:Use min function

In [47]:

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

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

Out[47]:

FirstSuccessfull_landing_date
22/12/2015

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of the boosters which have success in drone ship and have

[49]:
```
%%sql         SELECT Booster_Version
        FROM SpaceXTBL
        WHERE Landing_Outcome = 'Success (drone ship)'
            AND Payload_Mass__KG_ > 4000
            AND Payload_Mass__KG_ < 6000
```

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

t[49]:

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

```
In [58]: %%sql SELECT Mission_Outcome, COUNT(*) AS Total
         FROM SpaceXTBL
         GROUP BY Mission_Outcome;
```

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

Out[58]:

Mission_Outcome	Total
None	898
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- We used Groupby to filter the result grouped by the Mission_Outcome wether the mission was successful or failure

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.

Task 8

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

```
60]:
%%sql SELECT Booster_Version
FROM SpaceXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SpaceXTBL
    Order By Booster_Version
);
```

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

60]:

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** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

```
In [76]:  %%sql        SELECT Booster_Version, Launch_Site, Landing_Outcome , Date
                FROM SpaceXTBL
                WHERE Landing_Outcome LIKE 'Failure (drone ship)'
                    AND Date LIKE '%2015'

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

Out[76]:

Booster_Version	Launch_Site	Landing_Outcome	Date
F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)	01/10/2015
F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)	14/04/2015

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

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

In [79]:
```
%%sql SELECT Landing_Outcome, COUNT(*) AS Count_Successful
FROM SpaceXTBL
WHERE Date BETWEEN '04/06/2010' AND '20/03/2017'
GROUP BY Landing_Outcome
ORDER BY Count_Successful DESC;
```

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

Out[79]:

Landing_Outcome	Count_Successful
Success	20
No attempt	9
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

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.

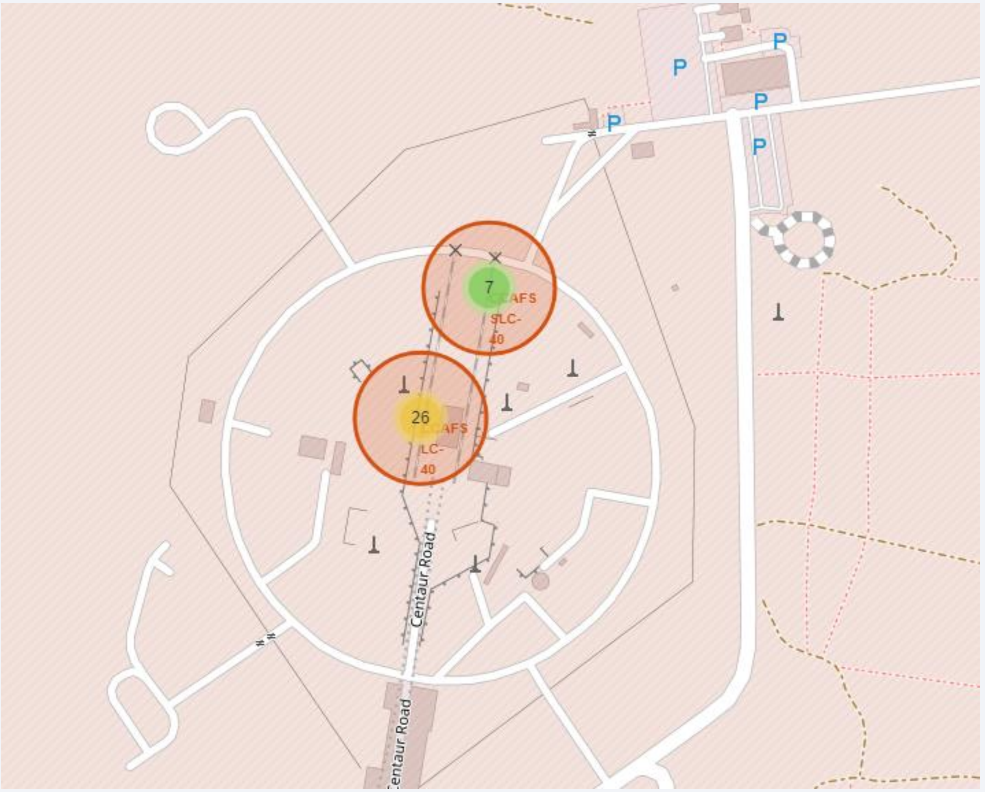
Section 4

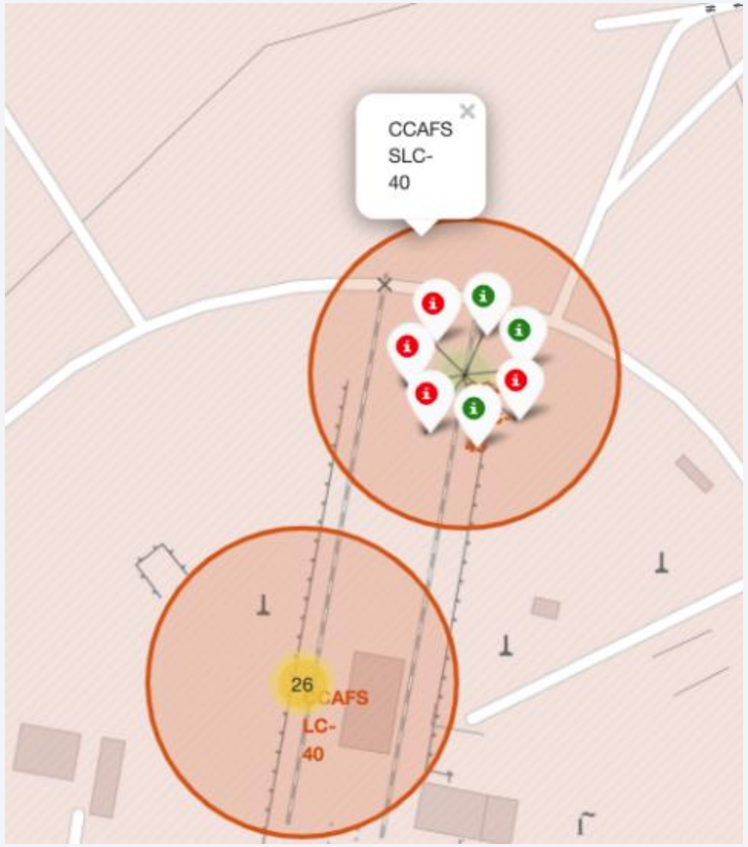
Launch Sites Proximities Analysis

All launch sites global map markers

As we can see the launch sites are in the USA coasts : Florida and California

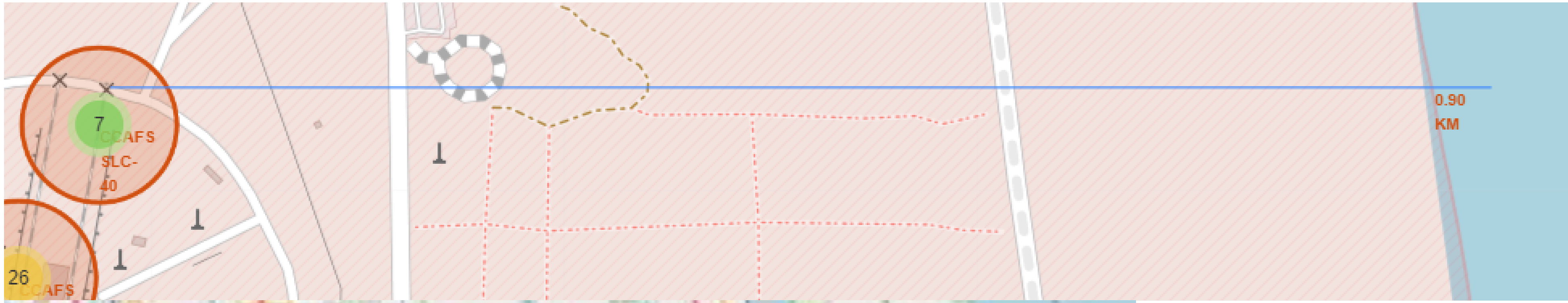
Markers showing launch sites with color labels





Green marker = Success | Red Marker = Failures

Launch Site distance to landmarks

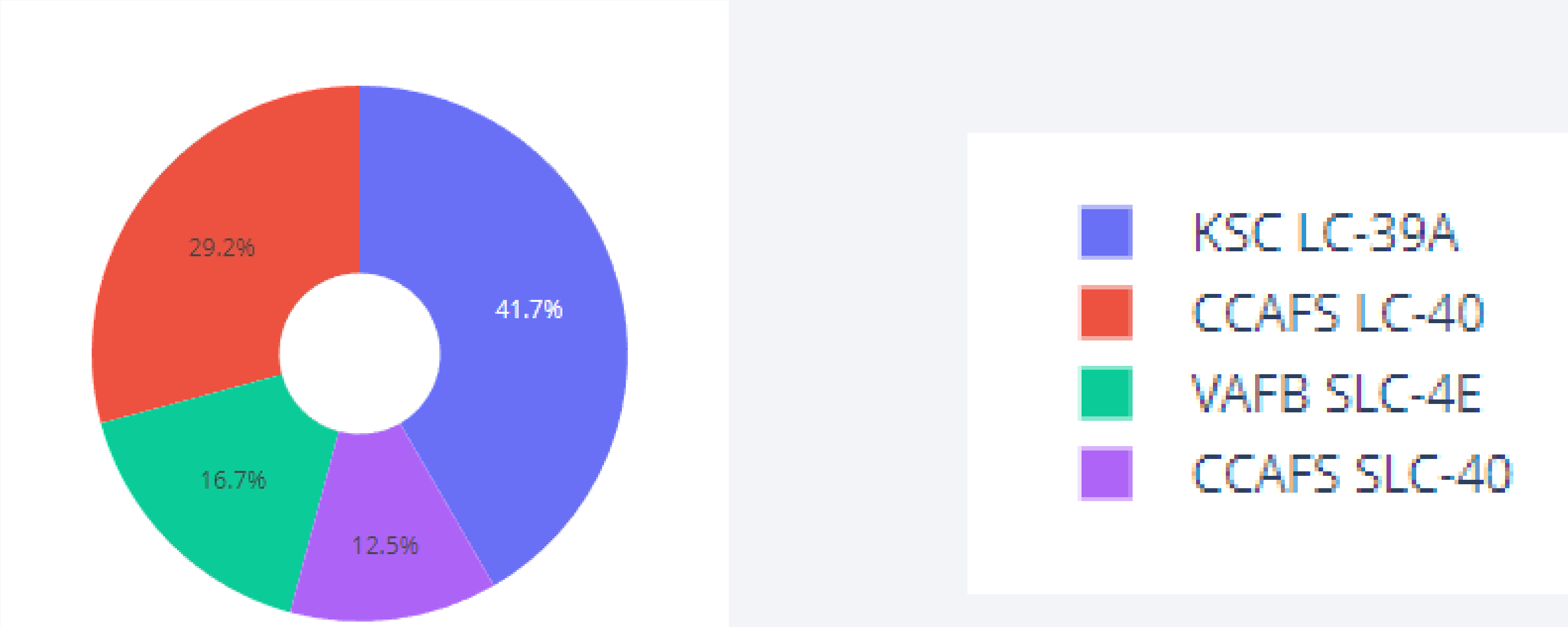


- Are launch sites in close proximity to railways?
- Are launch sites in close proximity to highways?
- Are launch sites in close proximity to coastline?
- Do launch sites keep certain distance away from cities

Section 5

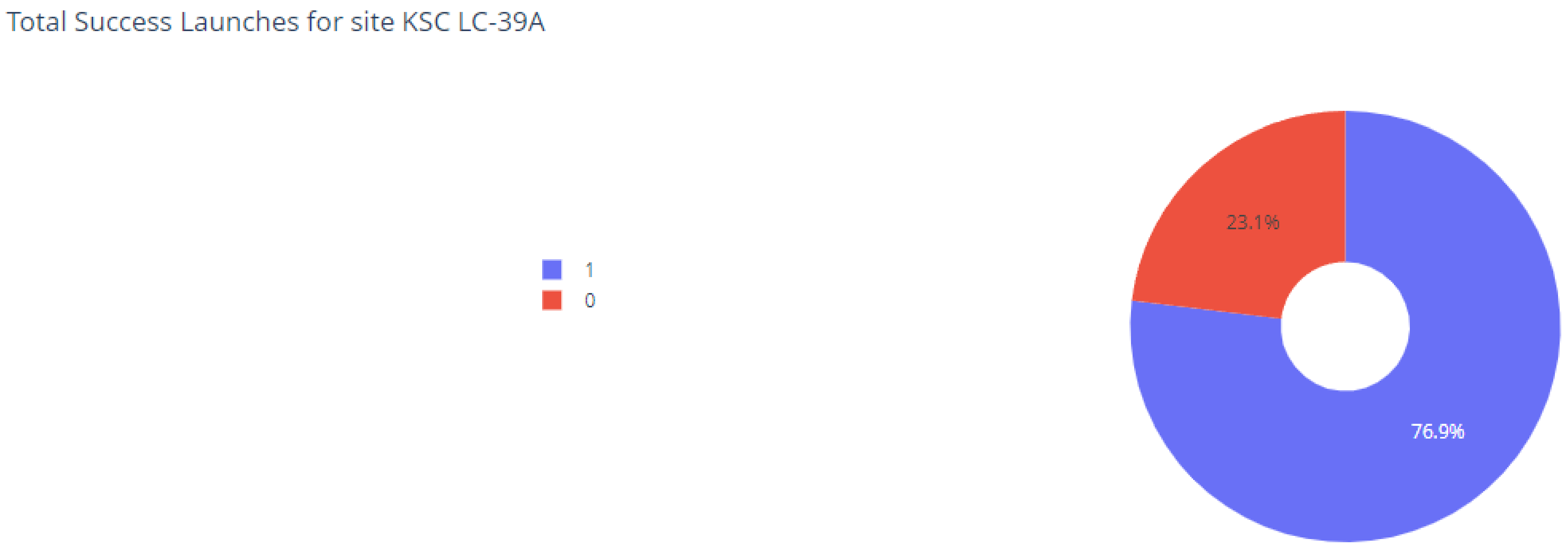
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



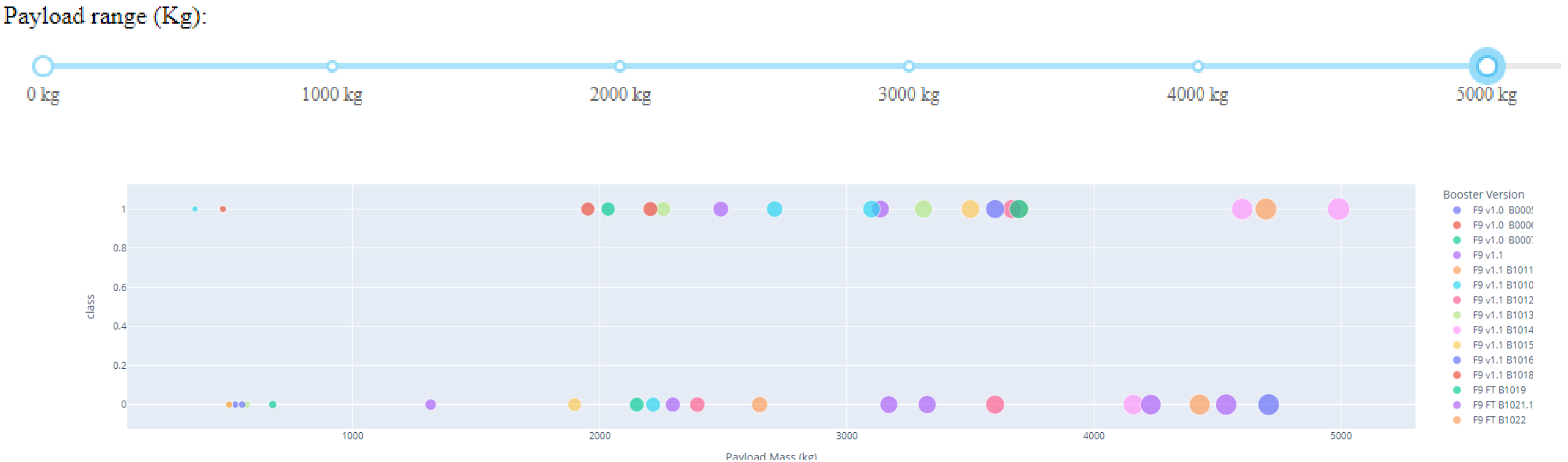
From the chart, KSC LC-39A had the most successful launches

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



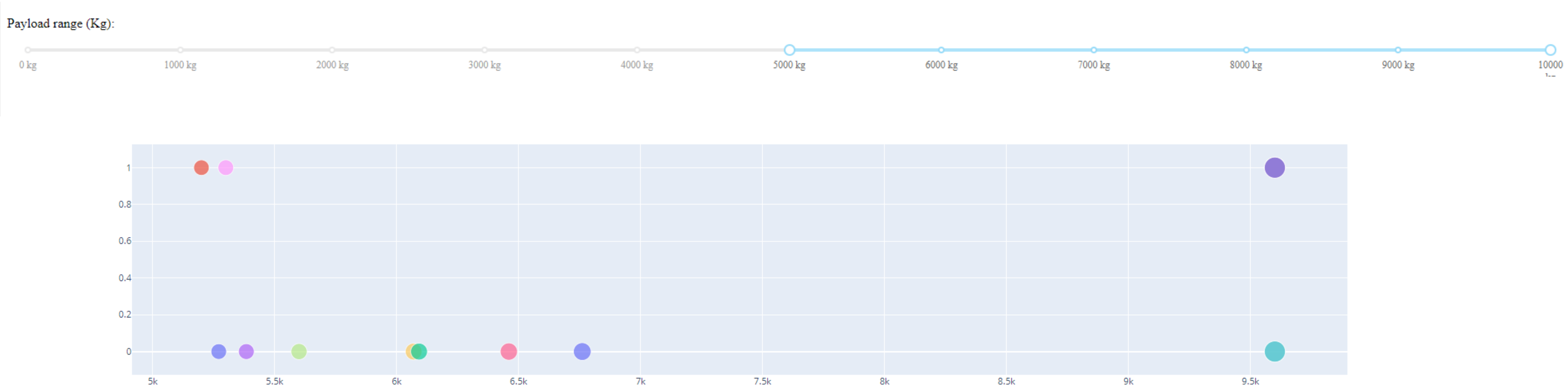
From the chart, KSC LC-39A had 76,9% successful launch rate and 23.1% failure launch rate

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



The success rate for low weight payloads is higher than heavy weight payloads

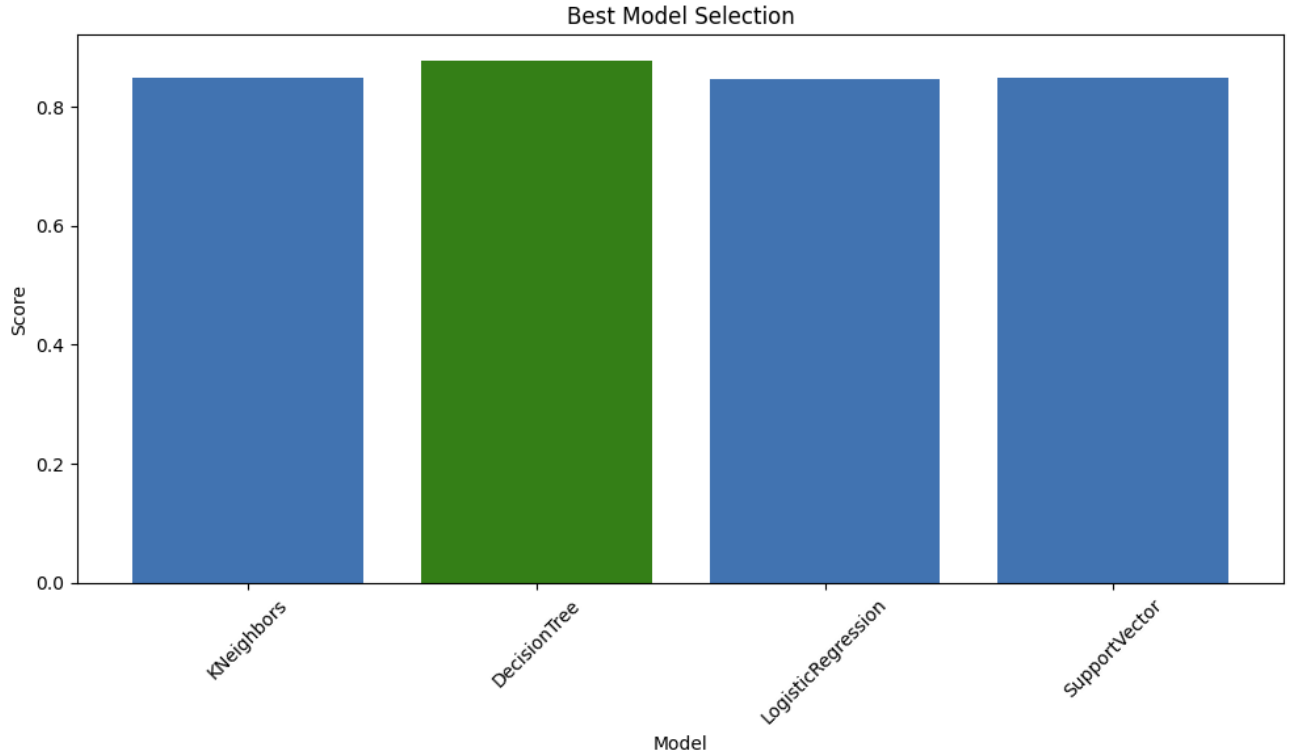
Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



The success rate for low weight payloads is higher than heavy weight payloads

Section 6

Predictive Analysis (Classification)

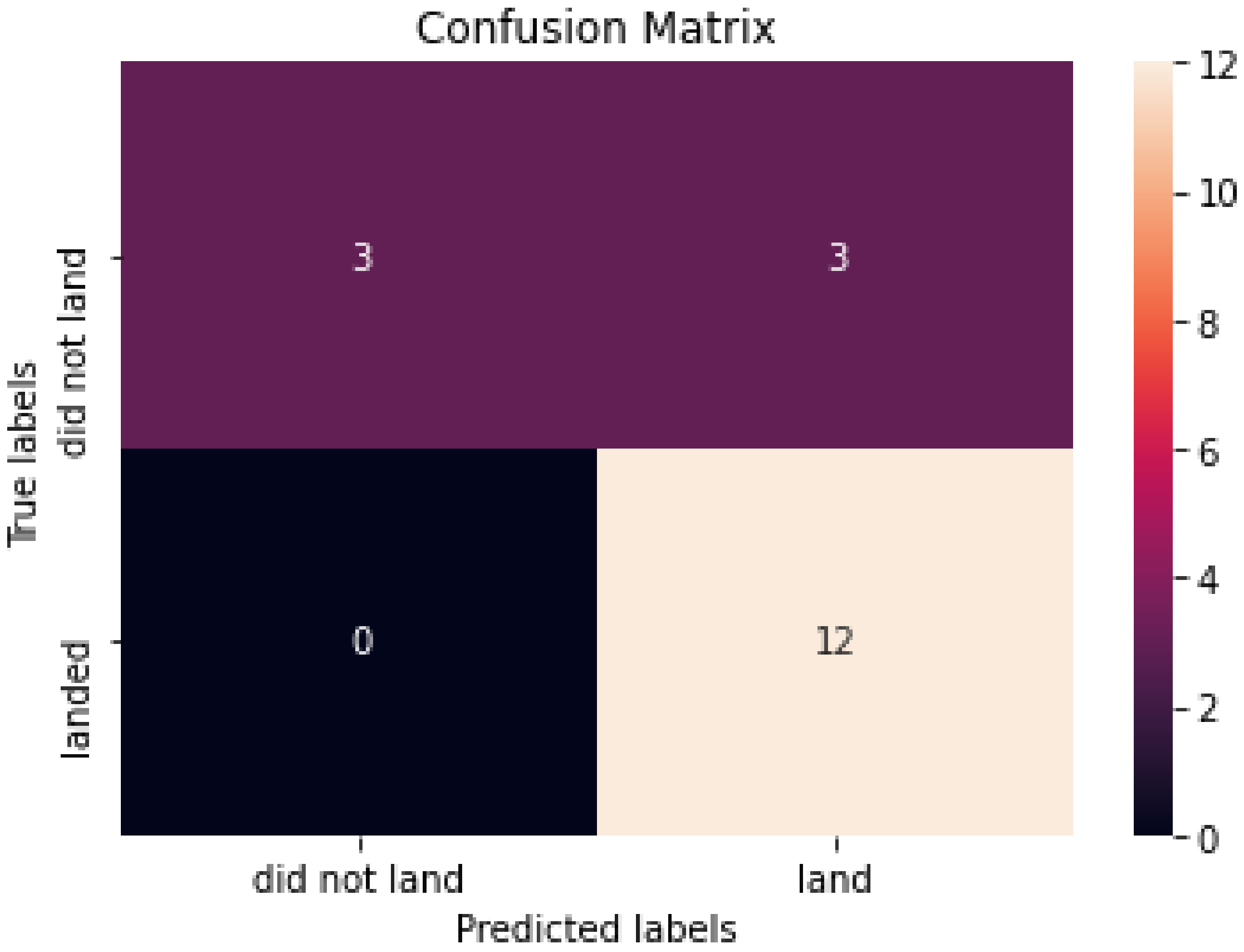


Classification Accuracy

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

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!