Winning Space Race with Data Science

Fridoom P. C. Koridama
June, 21 2024

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

    SpaceX offers Falcon 9 rocket launches at $62 million, significantly lower than the $165 million charged by other providers. This cost efficiency is largely due to SpaceX's ability to reuse the first stage of the rocket. Predicting the successful landing of this first stage helps estimate launch costs more accurately, which is valuable for competitors. The goal of this project is to create a machine learning model that predicts the successful landing of the Falcon 9's first stage.

- Problems you want to find answers

    - **Key Factors** : Identify factors affecting landing success
    - **Feature Interactions** : Understand how features interact to affect success rate
    - **Operational Condition** : Determine necessary conditions for successful landings

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 and formatting.
- GitHub link to the notebook: https://github.com/Fridoom14/Space-X-Falcon-9-First-Stage-Landing-Success/blob/main/1.%20Data%20Collection%20and%20Data%20Wrangling/jupyter-labs-spacex-data-collection-api.ipynb

1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

```
In [12]: # Use json_normalize method to convert the json result into a dataframe

         # decode response content as json
         static_json_df = res.json()
```

```
In [13]: # apply json_normalize
         data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

```
In [30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]

         df_rows = pd.DataFrame(rows)
         df_rows = df_rows.replace(np.nan, PayloadMass)

         data_falcon9['PayloadMass'][0] = df_rows.values
         data_falcon9
```

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.
- GitHub link to the notebook: https://github.com/Fridoom14/Space-X-Falcon-9-First-Stage-Landing-Success/blob/main/1.%20Data%20Collection%20and%20Data%20Wrangling/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

[11]:
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

[11]: {'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}

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`:

[12]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class = [1 if outcome not in bad_outcomes else 0 for outcome in df['Outcome']]
```

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

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

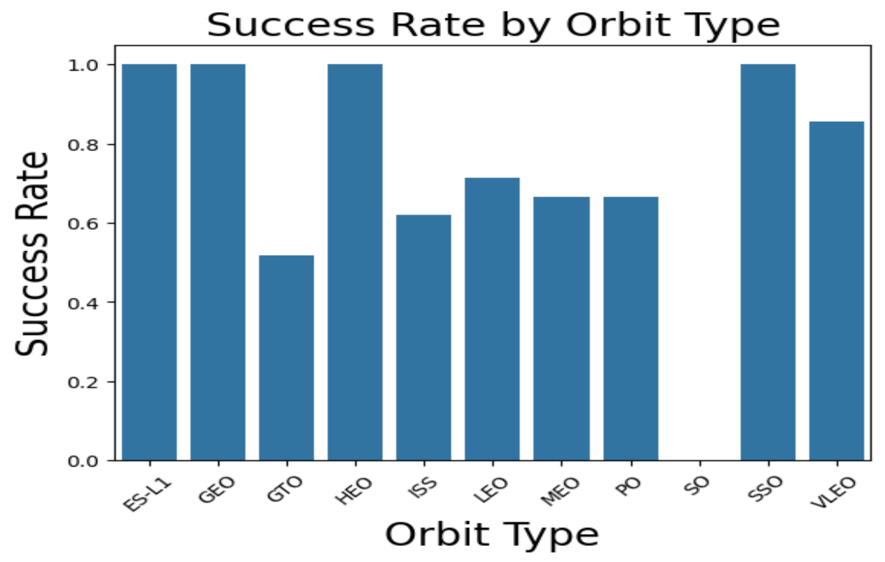
[13]:

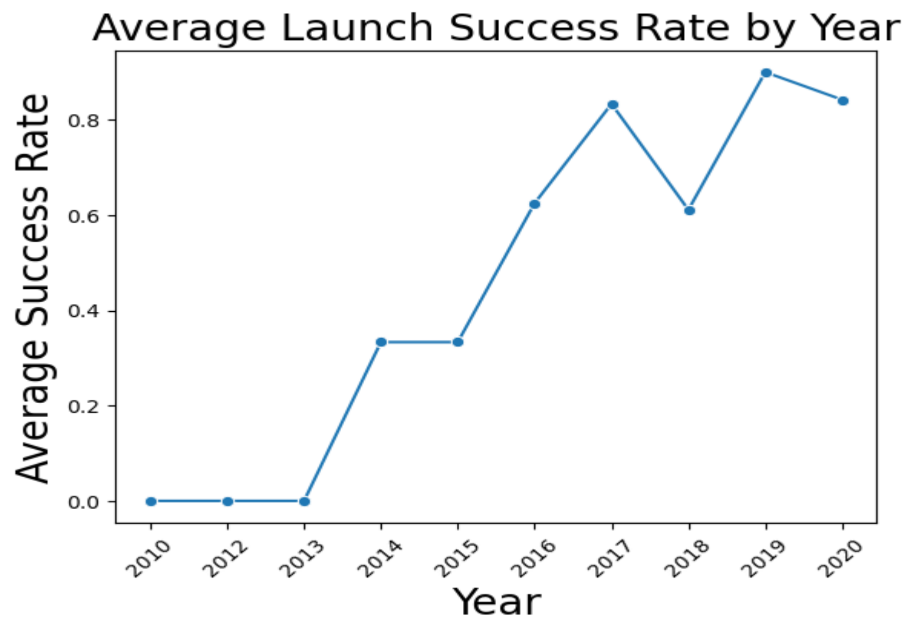
	Class
0	0
1	0
2	0
3	0
4	0
5	0

- 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.
- GitHub link to the notebook: https://github.com/Fridoom14/Space-X-Falcon-9-First-Stage-Landing-Success/blob/main/1.%20Data%20Collection%20and%20Data%20Wrangling/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.





- GitHub link to the notebook: https://github.com/Fridoom14/Space-X-Falcon-9-First-Stage-Landing-Success/blob/main/2.%20Exploratory%20Data%20Analysis/edadataviz.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.
- GitHub link to the notebook: https://github.com/Fridoom14/Space-X-Falcon-9-First-Stage-Landing-Success/blob/main/2.%20Exploratory%20Data%20Analysis/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.
- GitHub link to the notebook: https://github.com/Fridoom14/Space-X-Falcon-9-First-Stage-Landing-Success/blob/main/3.%20Visualization%20and%20Dashboard%20Analysis/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.
- GitHub link to the notebook: https://github.com/Fridoom14/Space-X-Falcon-9-First-Stage-Landing-Success/blob/main/3.%20Visualization%20and%20Dashboard%20Analysis/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.
- GitHub link to the notebook: https://github.com/Fridoom14/Space-X-Falcon-9-First-Stage-Landing-Success/blob/main/4.%20Machine%20Learnng/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

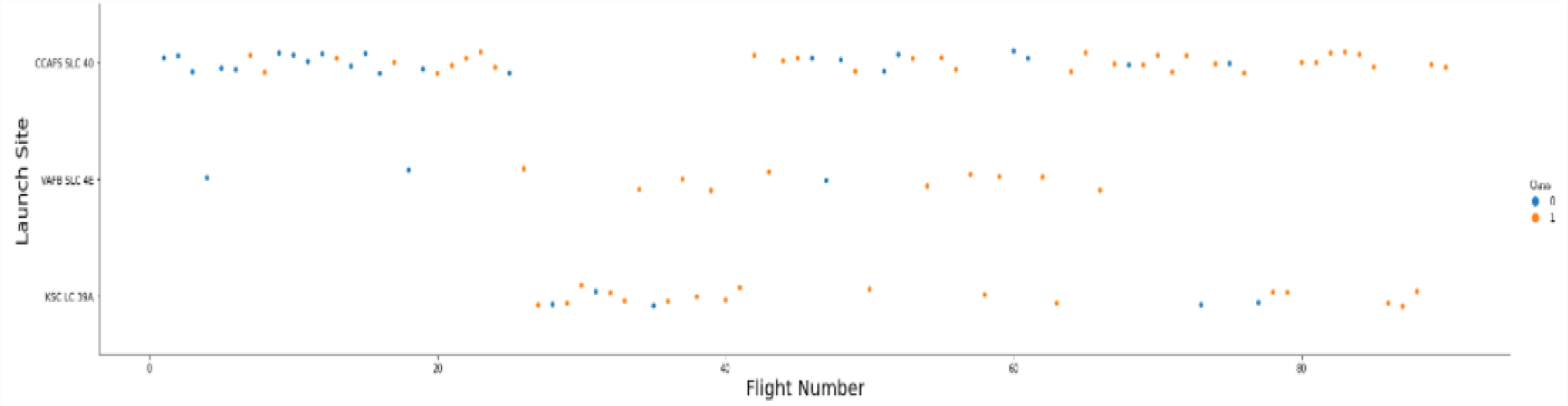
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

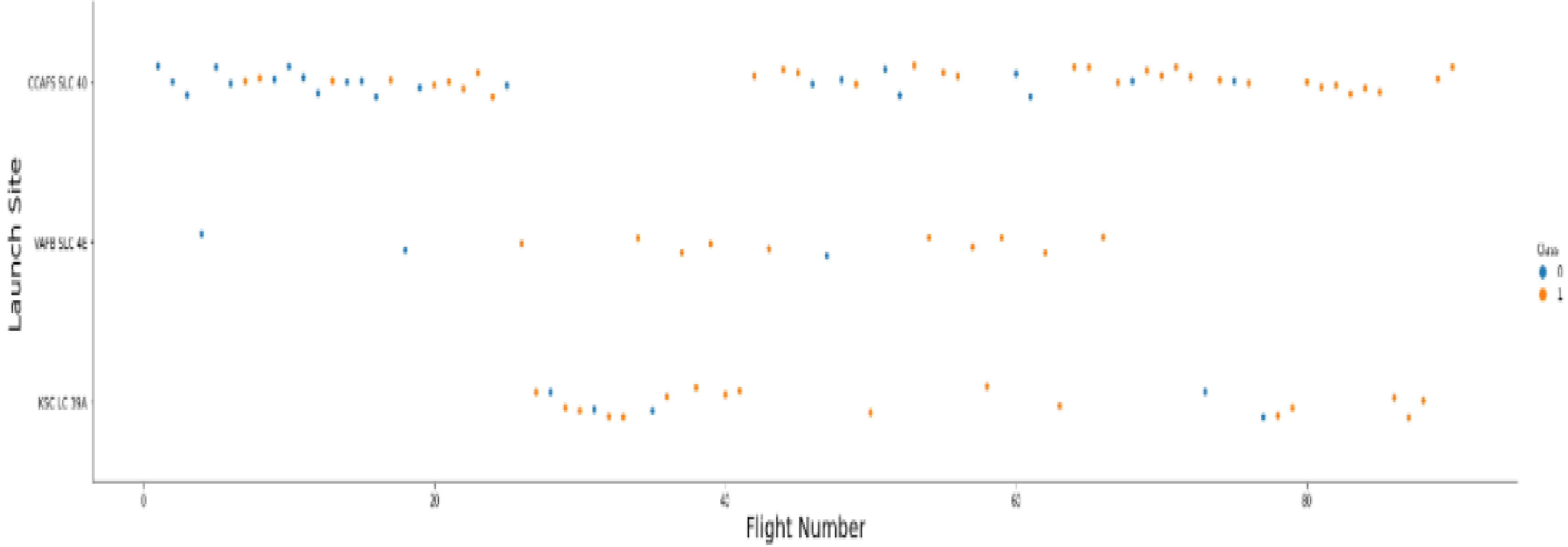
Flight Number vs. Launch Site

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



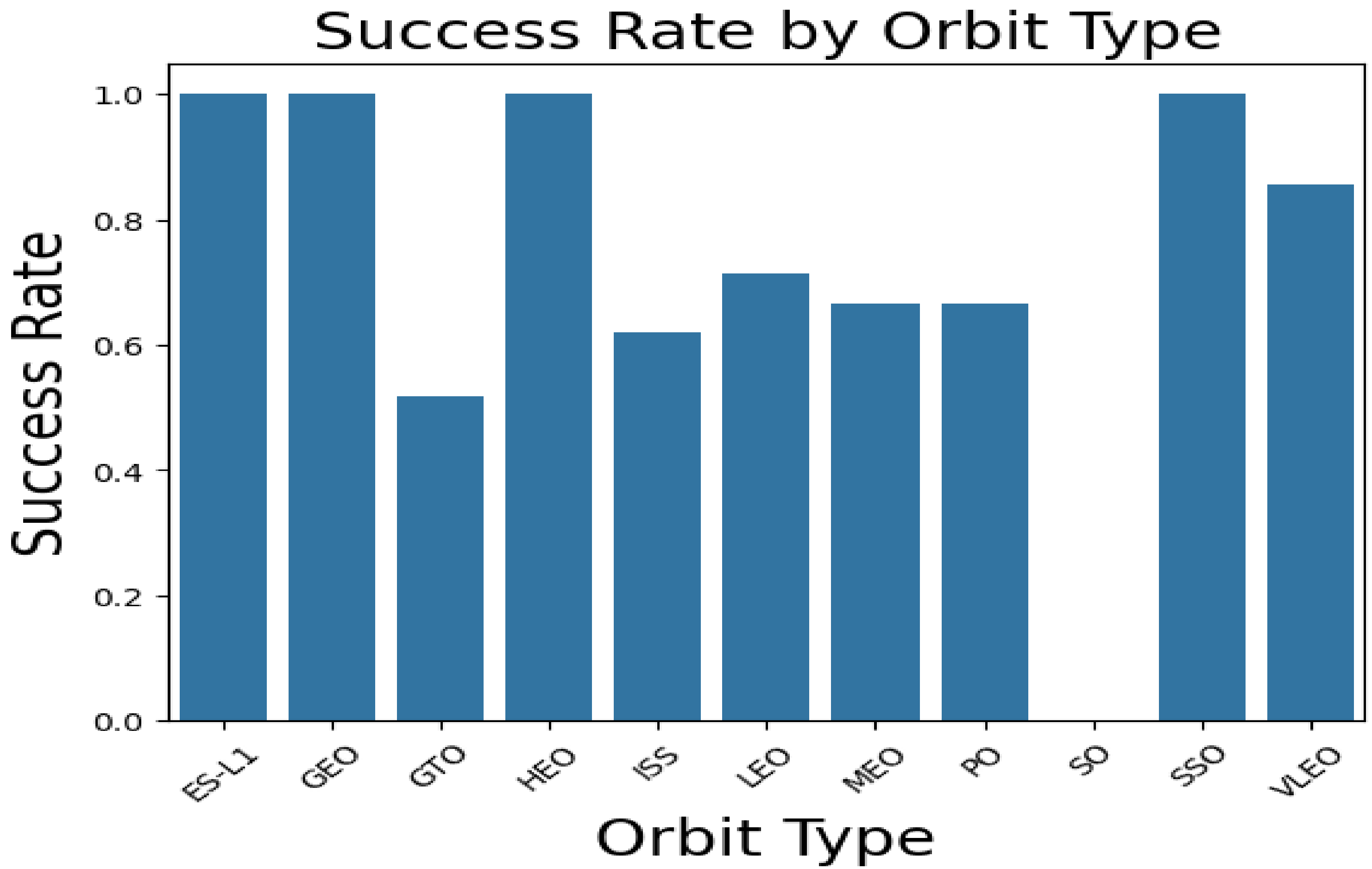
Payload vs. Launch Site

- The greater Payload Mass for Launch Site (CCAFS SLC 40), the higher success rate for the rockets.



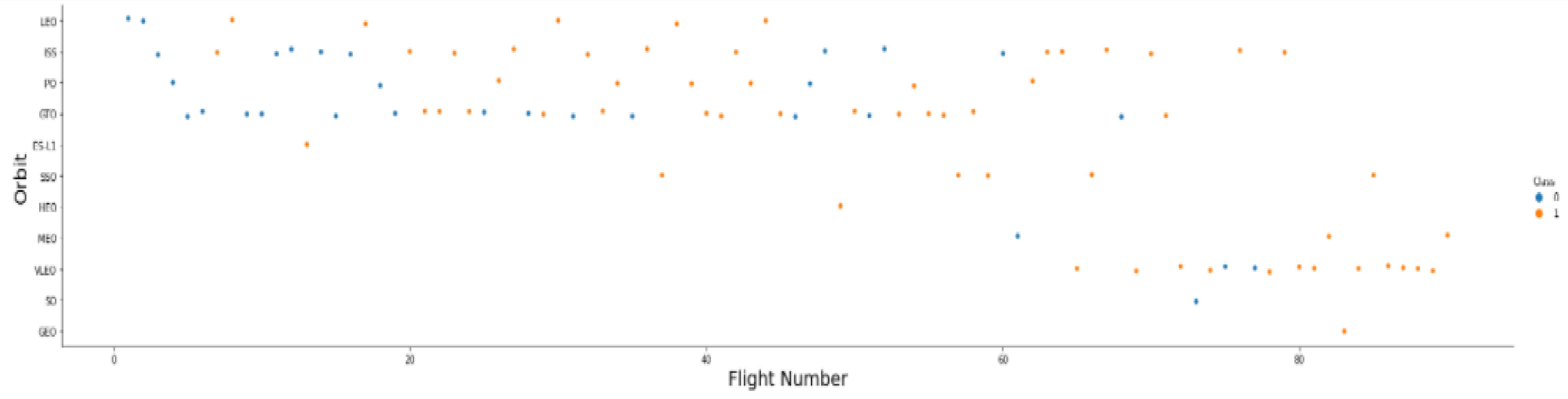
Success Rate vs. Orbit Type

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



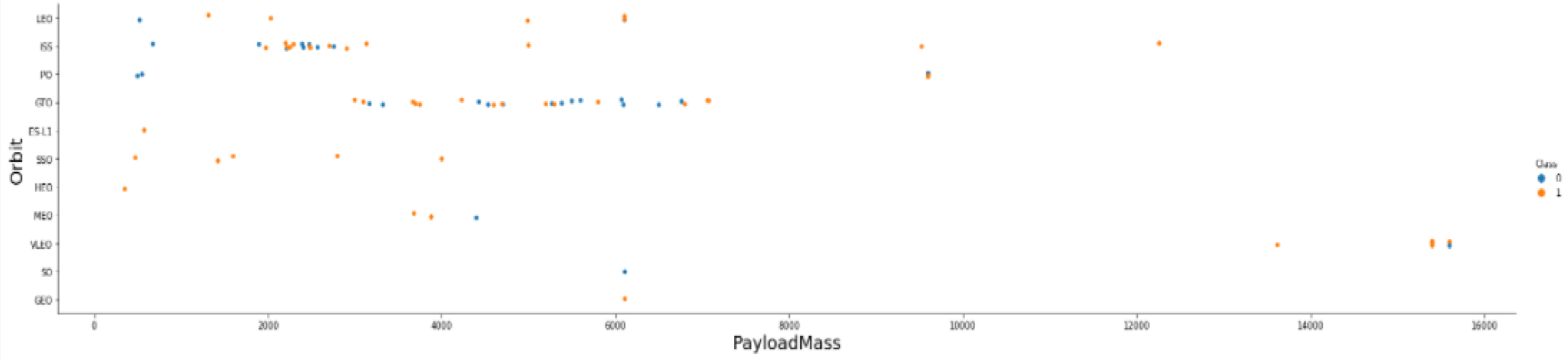
Flight Number vs. Orbit Type

- 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.



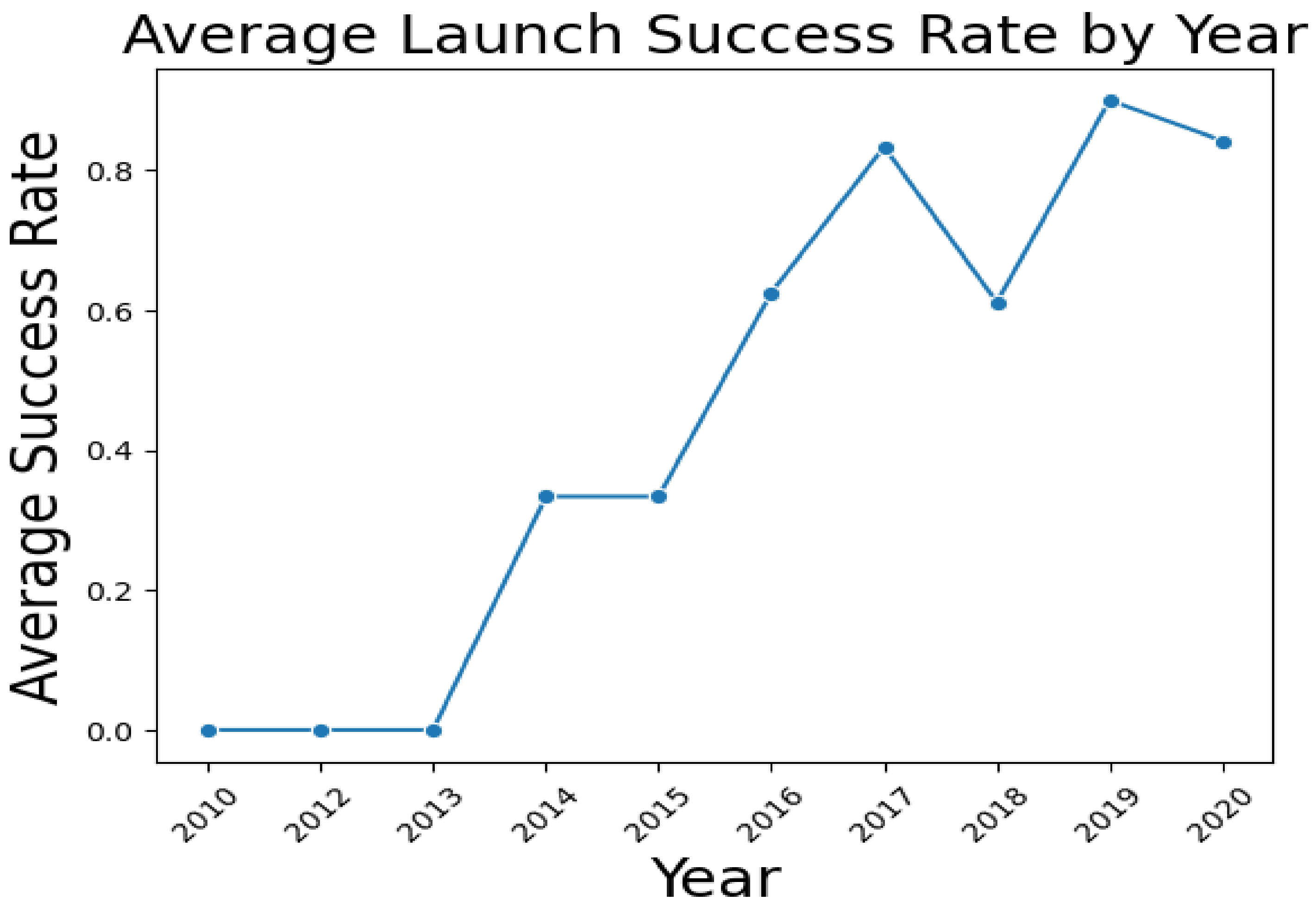
Payload vs. Orbit Type

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



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.

Task 1

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

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

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

[9]:

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

Launch Site Names Begin with 'CCA'

Task 2

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

[10]:
```
%%sql
SELECT * FROM SPACEXTABLE
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
```

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

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 48213 using the query below

Task 3

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

[12]:

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) as Total_Payload_Mass
FROM SPACEXTABLE
WHERE "Customer" LIKE '%NASA (CRS)%';
```

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

[12]:

Total_Payload_Mass
48213

Average Payload Mass by F9 v1.1

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

Task 4

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

[16]:
```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) as Average_Payload_Mass
FROM SPACEXTABLE
WHERE "Booster_Version" = 'F9 v1.1';
```

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

[16]:

Average_Payload_Mass
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

Task 5

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

Hint:Use min function

[20]:
```
%%sql
SELECT MIN(Date) as First_Successful_Landing
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE '%Success (ground pad)%';
```

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

[20]:

First_Successful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

```
[21]: %%sql
SELECT Booster_Version as Boosters_Names
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE '%Success (drone ship)%'
AND "PAYLOAD_MASS__KG_" > 4000
AND "PAYLOAD_MASS__KG_" < 6000;

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

[21]:

Boosters_Names
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

```
[24]: %%sql
SELECT "Mission_Outcome", COUNT(*) as Outcome_Count
FROM SPACEXTABLE
GROUP BY "Mission_Outcome";

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

[24]:

Mission_Outcome	Outcome_Count
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- We used **COUNT** to count the numbers and **GROUPBY** to grouping the count.

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.

```
[25]: %%sql
SELECT "Booster_Version"
FROM SPACEXTABLE
WHERE "PAYLOAD_MASS__KG_" = (
    SELECT MAX("PAYLOAD_MASS__KG_")
    FROM SPACEXTABLE
);

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

[25]:

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5

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

```
[26]: %%sql
SELECT
    substr(Date, 6, 2) AS month,
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM
    SPACEXTABLE
WHERE
    substr(Date, 0, 5) = '2015'
    AND "Landing_Outcome" LIKE 'Failure%Drone%'

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

[26]:

month	Landing_Outcome	Booster_Version	Launch_Site
01	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

```
[27]: %%sql
SELECT
    "Landing_Outcome",
    COUNT("Landing_Outcome") AS outcome_count
FROM
    SPACEXTABLE
WHERE
    Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY
    "Landing_Outcome"
ORDER BY
    outcome_count DESC
```

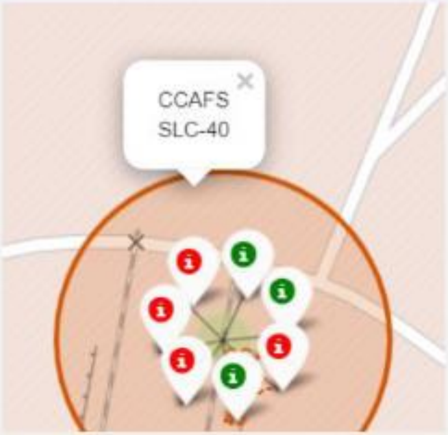
- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing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

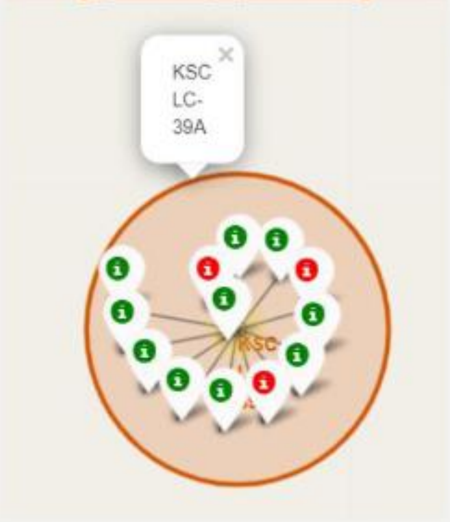
Section 4

Launch Sites Proximities Analysis

All launch sites global map markers

Markers showing launch sites with color labels





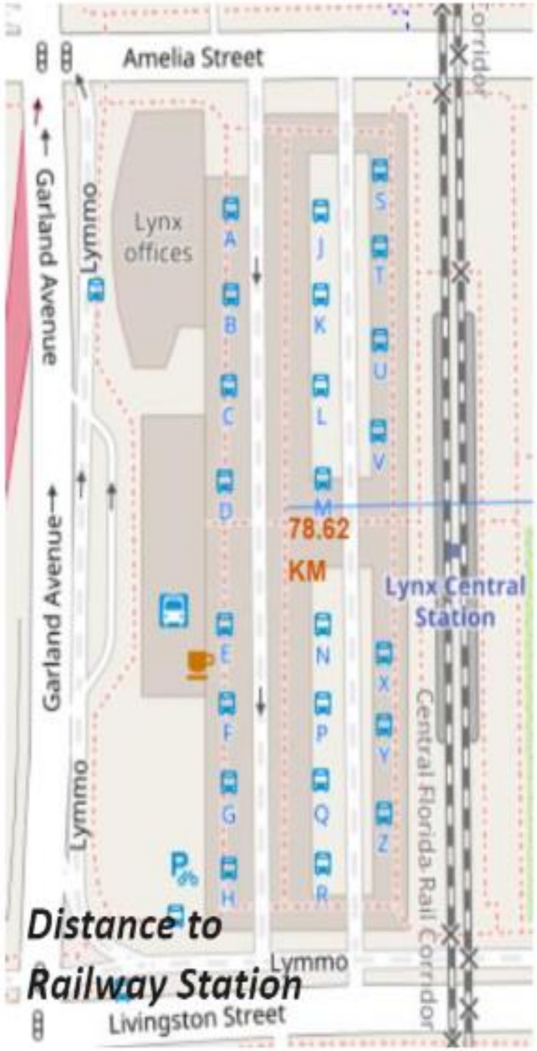
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

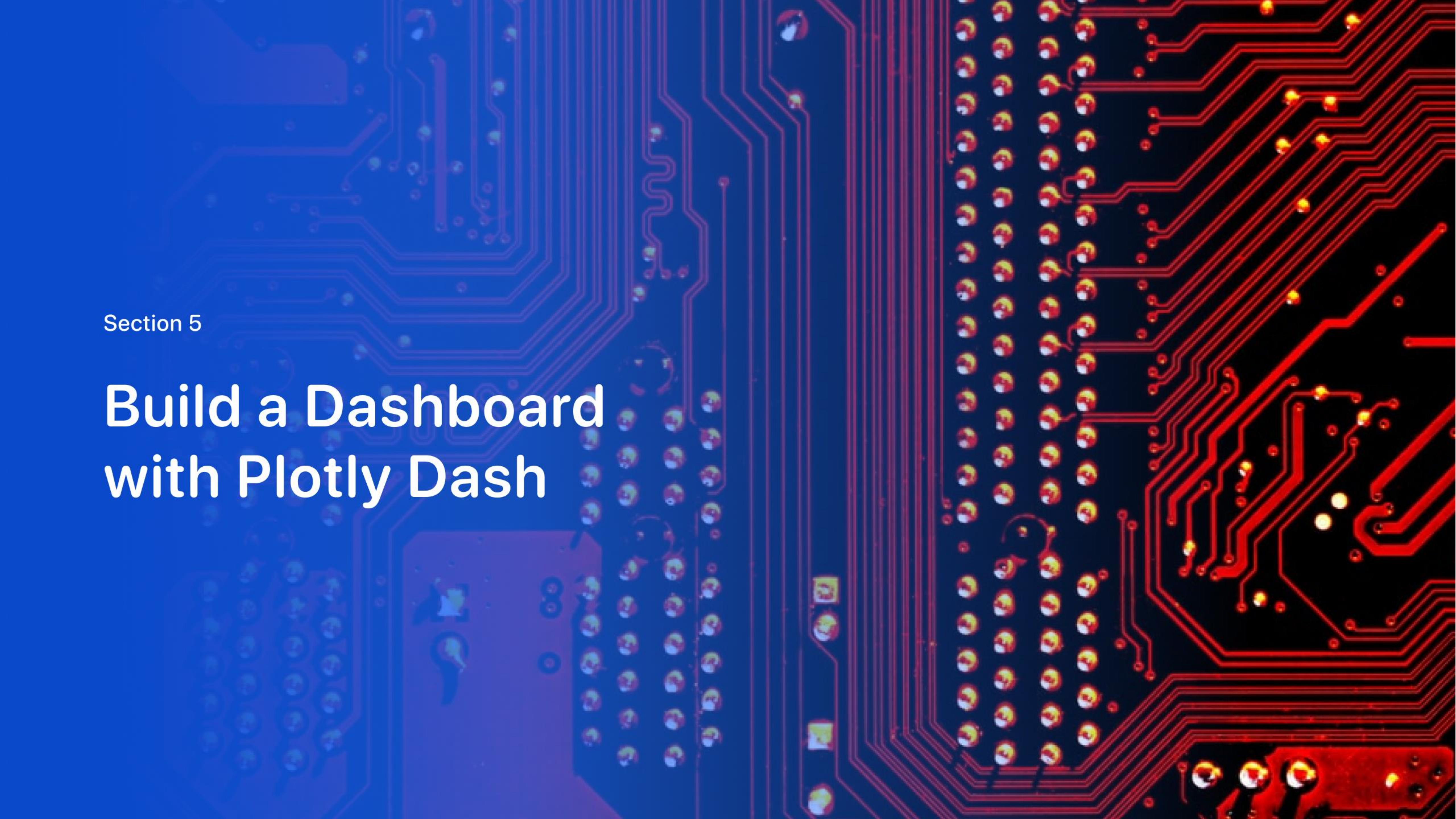
California Launch Site

37

Launch Site distance to landmarks



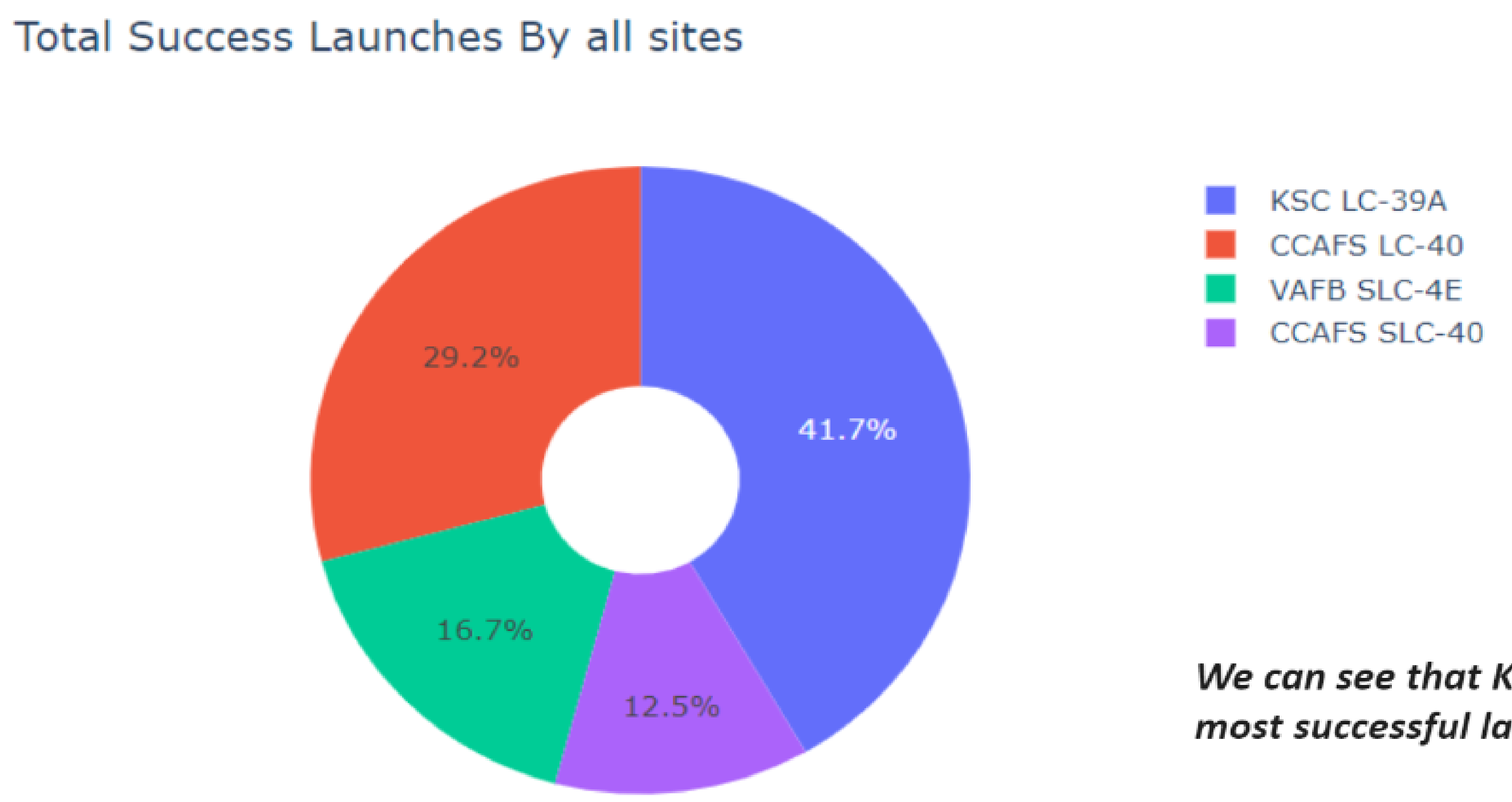
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes



Section 5

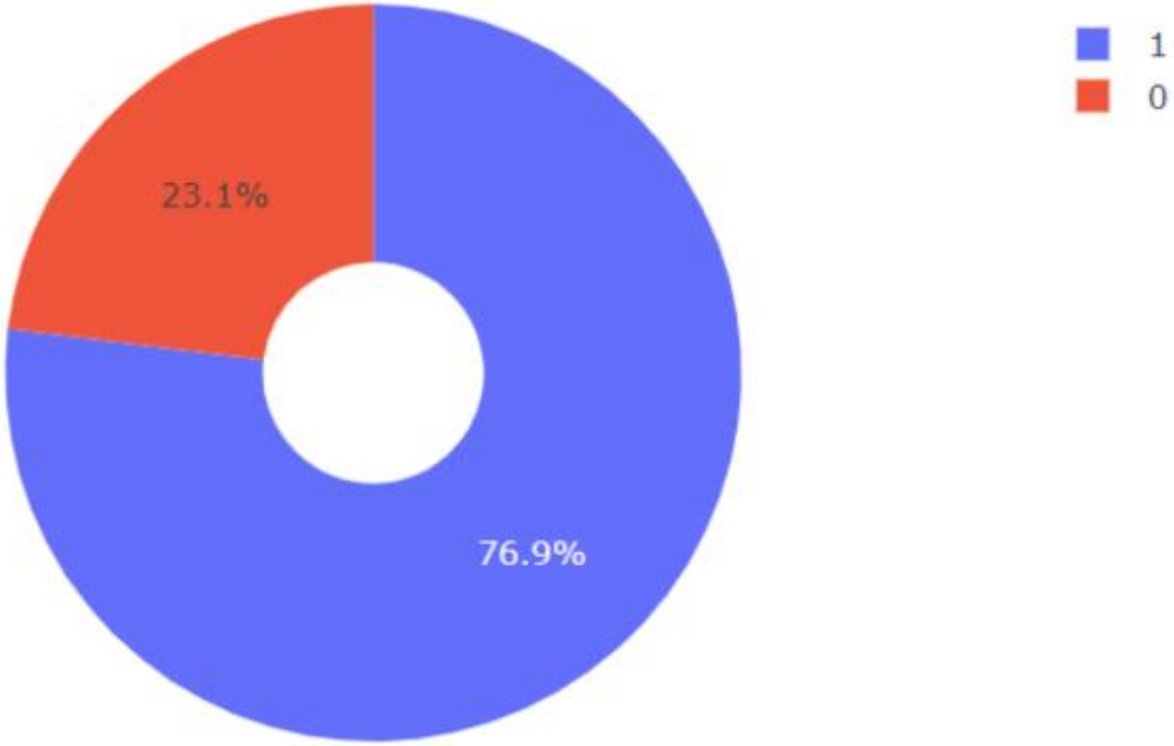
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



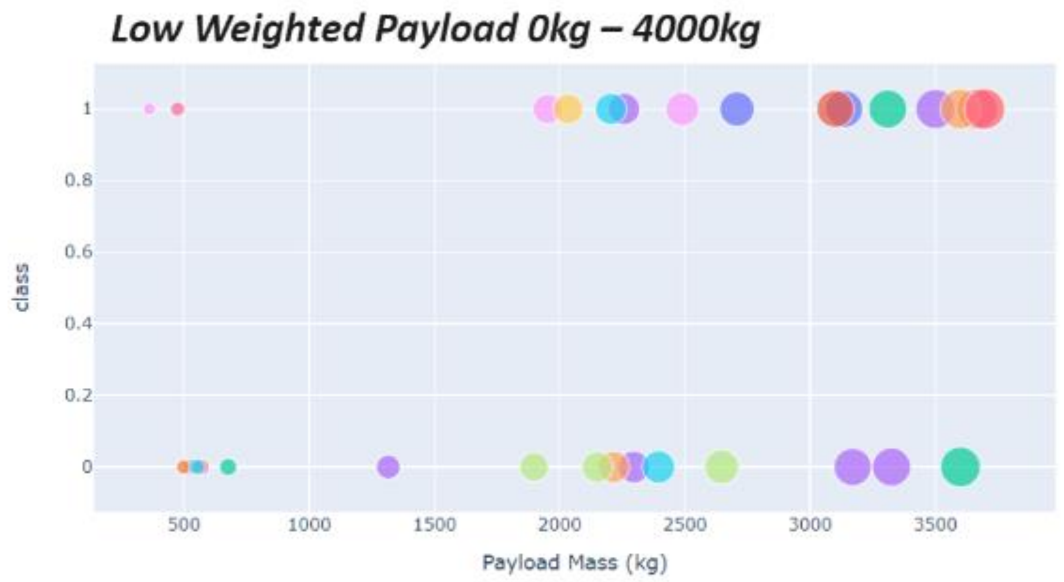
We can see that KSC LC-39A had the most successful launches from all the sites

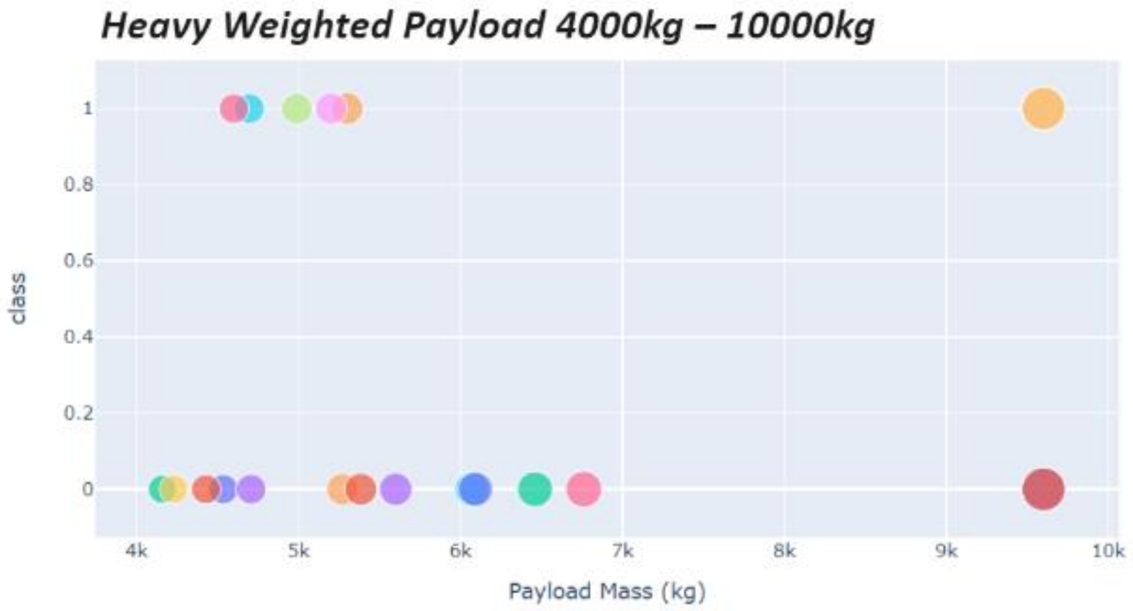
Pie chart showing the Launch site with the highest launch success ratio



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

Scatter plot of Payload vs Launch Outcome for 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 6

Predictive Analysis (Classification)

Classification Accuracy

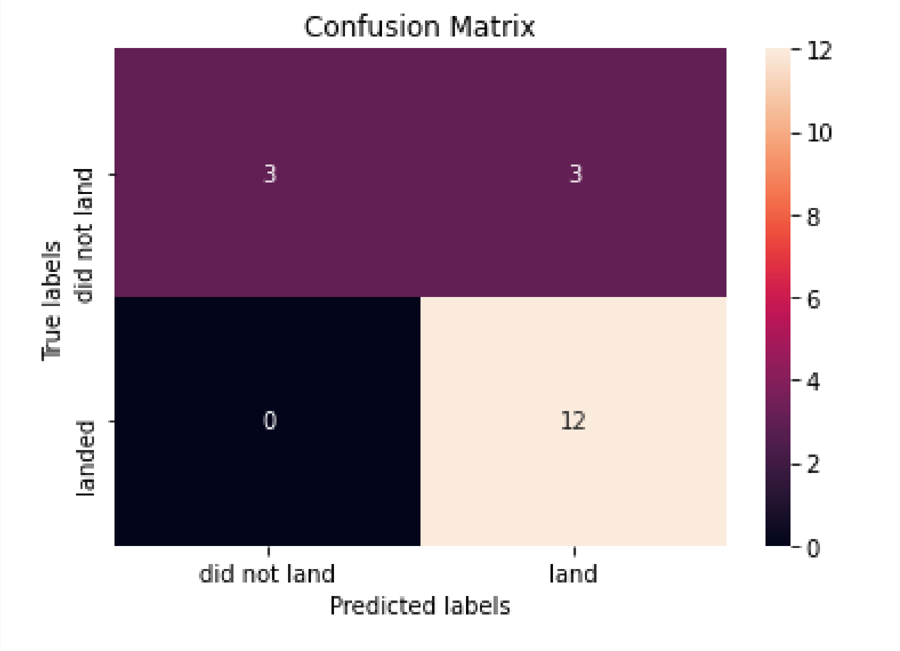
- All of the models (Logistic Regression, SVM, Decision Tree, KNN) have the same accuracy **0.83. If they have different accuracy that is not significantly affect** the model performance.

```
[56]: # Logistic Regression accuracy on test data
logreg_test_accuracy = logreg_cv.score(X_test, Y_test)

# Support Vector Machine accuracy on test data
svm_test_accuracy = svm_cv.score(X_test, Y_test)

# Decision Tree accuracy on test data
tree_test_accuracy = tree_cv.score(X_test, Y_test)

# K-Nearest Neighbors accuracy on test data
knn_test_accuracy = knn_cv.score(X_test, Y_test)

# Print the accuracy of each model
print("Logistic Regression test accuracy:", logreg_test_accuracy)
print("Support Vector Machine test accuracy:", svm_test_accuracy)
print("Decision Tree test accuracy:", tree_test_accuracy)
print("K-Nearest Neighbors test accuracy:", knn_test_accuracy)
```

```
Logistic Regression test accuracy: 0.8333333333333334
Support Vector Machine test accuracy: 0.8333333333333334
Decision Tree test accuracy: 0.8333333333333334
K-Nearest Neighbors test accuracy: 0.8333333333333334
```

Confusion Matrix

- The confusion matrix for the all models classifier we used, 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.
- All the models classifier we used, are the best machine learning algorithm for this task.

Thank you!