IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Roye Bustan
11/15/23

Outline

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

Executive Summary

- Summary of methodologies
    - Collecting Data through:
        - API call to Space X Data
        - Web Scraping (beautiful soup)
    - Data Wrangling/Organization
    - Exploratory Data Analysis using:
        - SQL
        - Data Visualization
    - Interactive Visual Analytics using folium
    - Machine Learning Predictions on Associated Dataframes.
- Summary of all results
    - Exploratory Data Analysis Results
    - Interactive Analytics in Jupyter Notebooks
    - Predictive Analytics result from ML

Introduction

- Project background and context

In this capstone project, our objective is to predict the successful landing of the Falcon 9 first stage. SpaceX prominently features Falcon 9 rocket launches on its website, advertising a cost of 62 million dollars. In comparison, other providers charge upwards of 165 million dollars for each launch. The substantial cost savings with SpaceX can be attributed to their innovative approach of reusing the first stage of the rocket. Consequently, determining the likelihood of a successful first stage landing becomes crucial in assessing the overall cost of a launch.

- Problems you want to find answers
  - What factors determine when a rocket lands successfully?
  - What conditions need to be in place to ensure a successful landing program?
  - Can these features be used to not only analyze trends but predict future landings?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using the SpaceX API and Web Scraping from wikipedia page.
- 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 the following:
  - Data collection was done using get request to the SpaceX API.
  - Next, we formatted the response 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 filled in missing values with averages or N/A's.
  - We performed web scraping from Wikipedia as well for Falcon 9 launch records with BeautifulSoup; taking information as HTML table, parse the table and convert it to a pandas dataframe.

Data Collection – SpaceX API

- Used a simple get request along with a response status code. Also normalized the data to get a cleaner dataframe.

- GitHub Notebook URL: https://github.com/Rbustan0/Rbustan-Coursera-IBM-Data-Science-Capstone/blob/main/WEEK1_%20DATACOLLECTION_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-DS0321
```

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 [16]:
```
# Use json_normalize meethod to convert the json result into a dataframe
#ROYE:

dataforframe = response.json()

data = pd.json_normalize(data)

#df.columns
```

Using the dataframe `data` print the first 5 rows

In [17]:
```
# Get the head of the dataframe
# Roye:
data.head(5)
```

Data Collection - Scraping

- Requested Falcon9 Launch Wiki page from its URL, then extracted all column/variable names from the HTML table header, and finally created a dataframe by parsing the launch HTML tables.
- Github Notebook Link: https://github.com/Rbustan0/Rbustan-Coursera-IBM-Data-Science-Capstone/blob/main/WEEK1_DATACOLLECTION_SOUP_jupyter-labs-webscraping.ipynb

```
# use requests.get() method with the provided static_url
# assign the response to a object

#ROYE:
response = requests.get(static_url)
response.status_code
```

```
200
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
#ROYE:

soup = BeautifulSoup(response.text, 'html.parser')
```

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

```
# Use soup.title attribute
#ROYE:

print(soup.title.string)
```

Data Wrangling

- We took several cases where boosters did not land successfully and converted those values into an extra col of binary values, checked data was being passed through correctly by checking its type, calculated number of missing values from the scraped data, and replaced empty values with averages.

- We did this in conjunction of calculating number of orbit occurrences and used that to review our landing outcomes to determine success rates of certain landings.

- GitHub Notebook URL: https://github.com/Rbustan0/Rbustan-Coursera-IBM-Data-Science-Capstone/blob/main/WEEK1_Data%20wrangling.ipynb

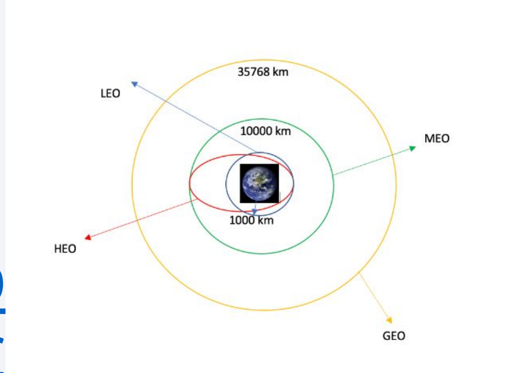
```
In [3]: df.isnull().sum()/len(df)*100

Out[3]: FlightNumber       0.000000
        Date               0.000000
        BoosterVersion     0.000000
        PayloadMass        0.000000
        Orbit              0.000000
        LaunchSite         0.000000
        Outcome            0.000000
        Flights            0.000000
        GridFins           0.000000
        Reused             0.000000
        Legs               0.000000
        LandingPad        28.888889
        Block              0.000000
        ReusedCount        0.000000
        Serial             0.000000
        Longitude          0.000000
        Latitude           0.000000
        dtype: float64
```

Identify which columns are numerical and categorical:

```
In [4]: df.dtypes

Out[4]: FlightNumber        int64
        Date               object
        BoosterVersion     object
        PayloadMass       float64
        Orbit              object
        LaunchSite         object
        Outcome            object
        Flights             int64
        GridFins             bool
        Reused               bool
        Legs                 bool
        LandingPad         object
        Block             float64
        ReusedCount         int64
        Serial             object
        Longitude         float64
        Latitude          float64
        dtype: object
```




```
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

1. Made SNS scatterplots/bubbleplots that visualizes the relationship between flight number and Launch Site, payload and launch site, Orbit success classifications based on flight numbers, and payload and orbit on class.
2. A SNS bar chart that visualizes the relationship between success rate of each orbit type.
3. A SNS line plot that visualizes success rate over the years.
4. Created dummy variables and casted numeric columns to float64 for features engineering Later on.

- GitHub Notebook URL: https://github.com/Rbustan0/Rbustan-Coursera-IBM-Data-Science-Capstone/blob/main/WEEK2_SNS_eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- Using Postgresql: Selected unique records of landing sites using SQL displaying/making averages on payloads generically and by booster versions.
- Also tracked and created queries that display unique records such as dates of successful landings and names of boosters associated with successful landings as well as their counts.

- GitHub Jupyter Notebook URL:

https://github.com/Rbustan0/Rbustan-Coursera-IBM-Data-Science-Capstone/blob/main/WEEK2_eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Added all launch sites on a folium geographic map using a blue Folium Circle object with marker objects.
- Assigned feature landing outcomes to true/false binary values
- Color labeled clusters were made as well.
- Distance between various objects and launch sites were measured as well such as railways, highways, and coastways.
- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map

- GitHub Jupyter Notebook URL: https://github.com/Rbustan0/Rbustan-Coursera-IBM-Data-Science-Capstone/blob/main/WEEK3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- Dashboard features (time adjustable):
  - Plotted pie charts with total launches by certain sites
  - scatter graphs that depict outcomes vs payload mass
- We wanted users to be able to compare total launches with their expected output in payload mass to show comparisons of outcomes and trends
- Github Plotly Dash Lab: https://github.com/Rbustan0/Rbustan-Coursera-IBM-Data-Science-Capstone/blob/main/WEEK3_spacex_launch_dash.csv

Predictive Analysis (Classification)

- Objective was to perform EDA to determine Training Labels:
    - created columns for classes
    - standardized the data
    - split into training data and test data
- Used and compared different machine learning models to tun hyperparameters (GridSearchCV).
- Tuned features (feature engineering) and algorithms to determine and weigh accuracy between different machine learning models stacked against each other in the lab.

- Results determined that our issue required a classification model.

- GitHub Jupyter Notebook URL: https://github.com/Rbustan0/Rbustan-Coursera-IBM-Data-Science-Capstone/blob/main/WEEK4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

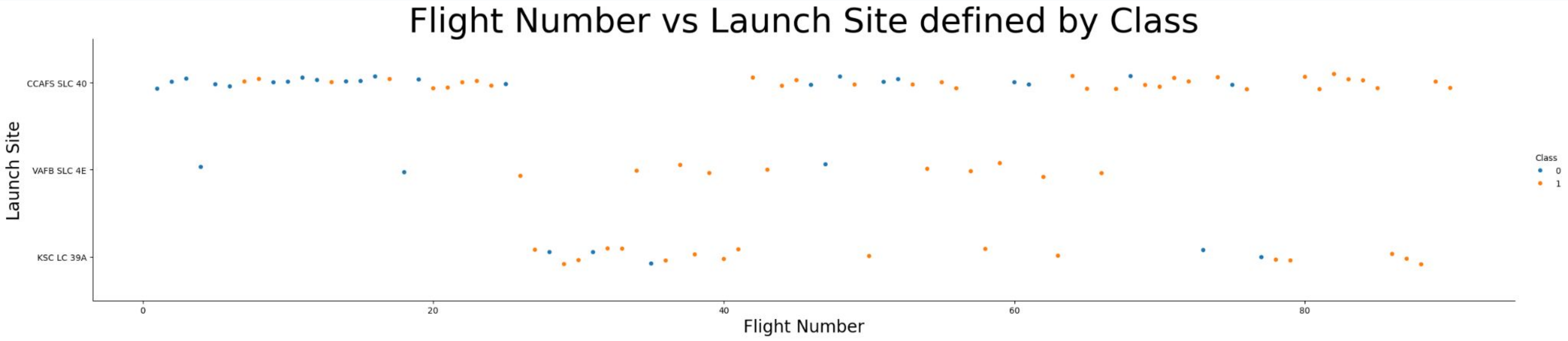
Results

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

Section 2

Insights drawn from EDA

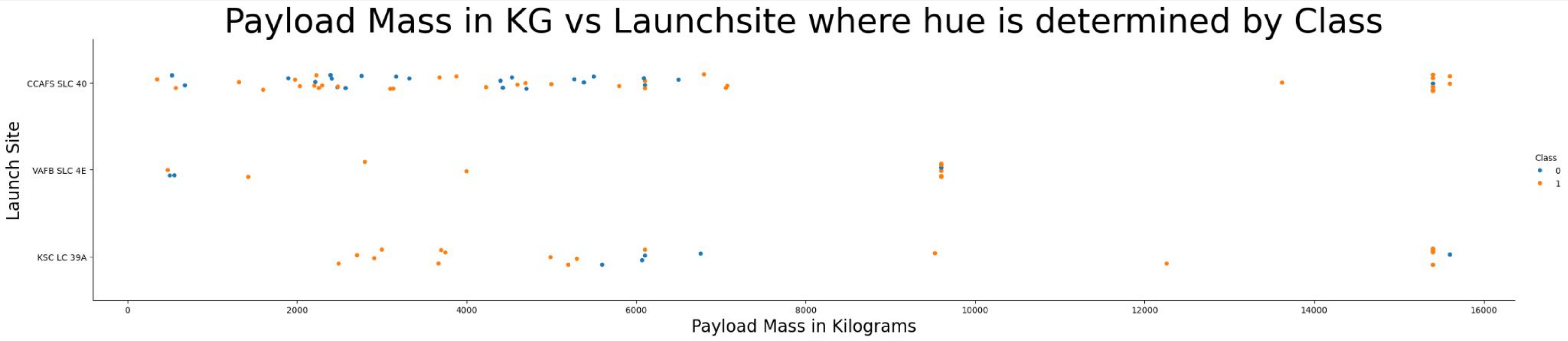
Flight Number vs. Launch Site



This study depicts that success return is higher when more flights are tallied at the same launch site.

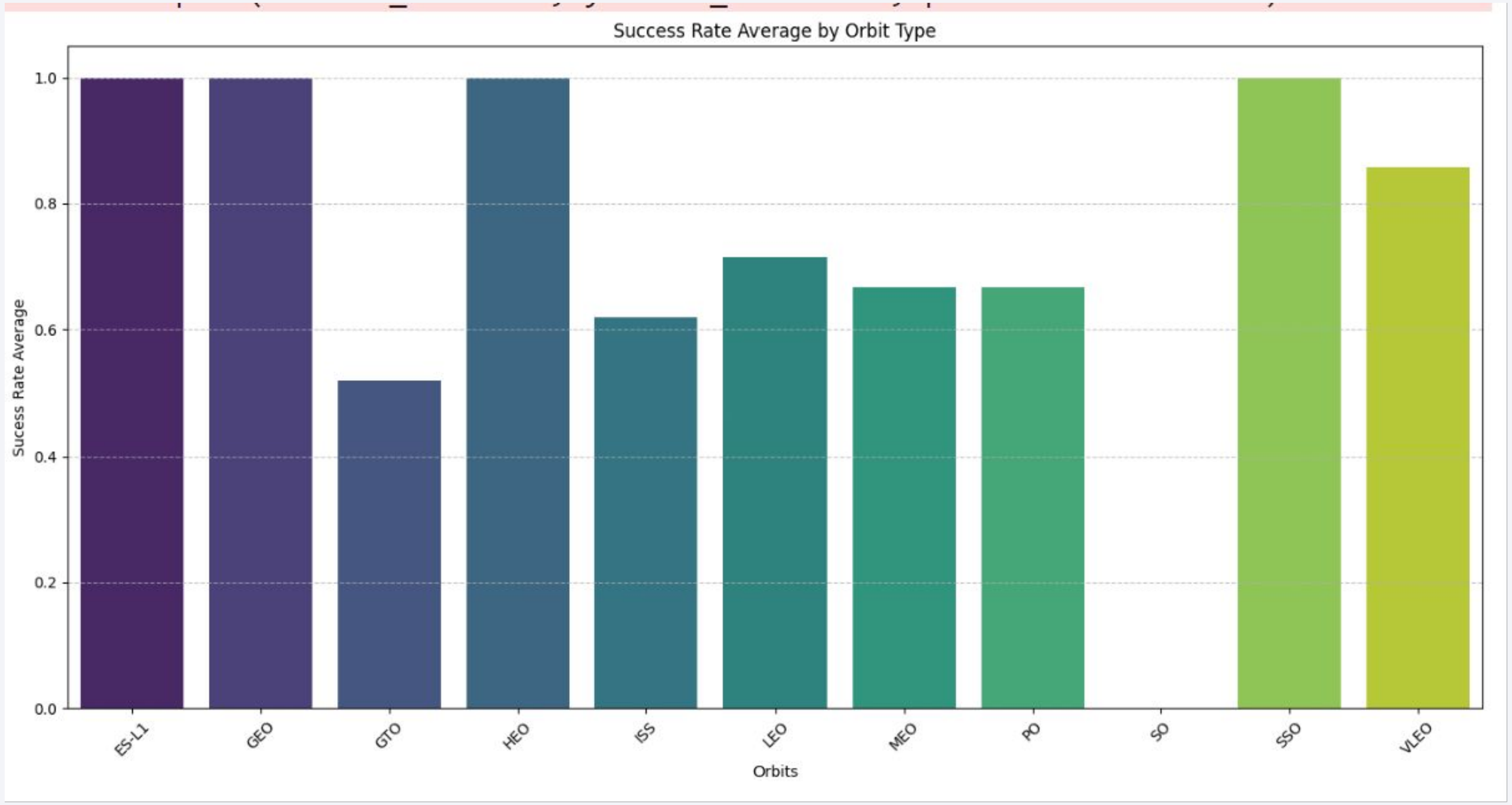
Payload vs. Launch Site

The greater the payload mass, the higher success rate, even at the outlier values.



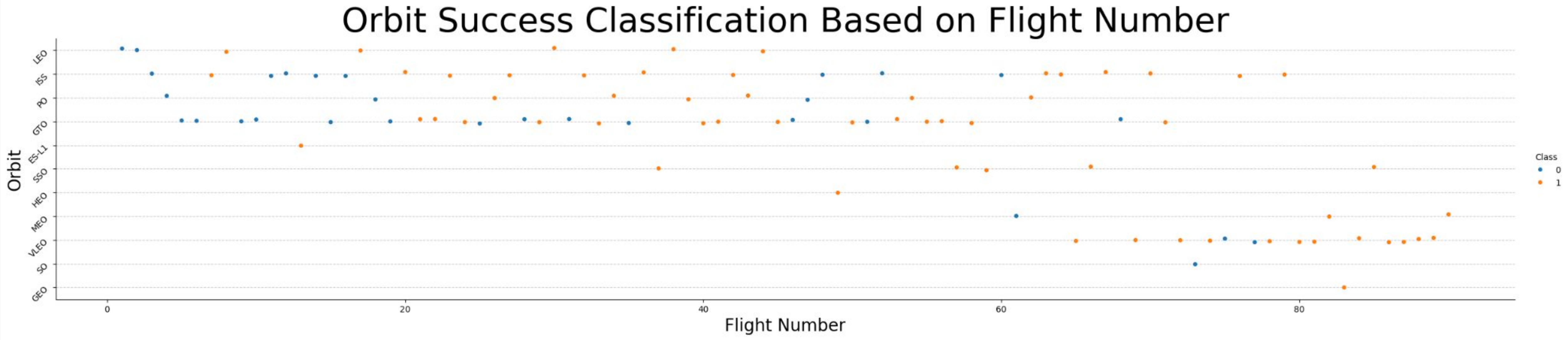
Success Rate vs. Orbit Type

- Most successful orbits were:
  - ES-L1
  - GEO
  - HEO
  - SSO
  - VLEO



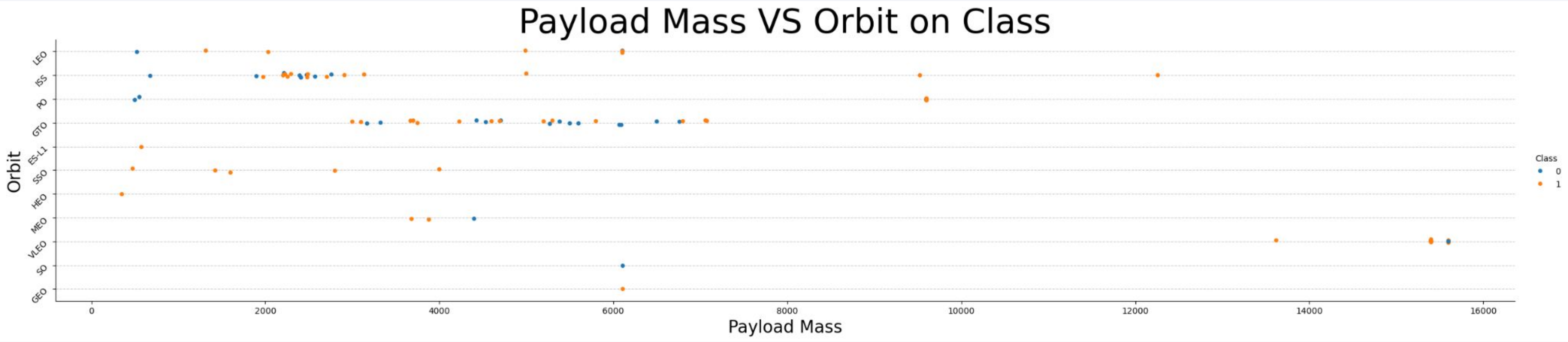
Flight Number vs. Orbit Type

Success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



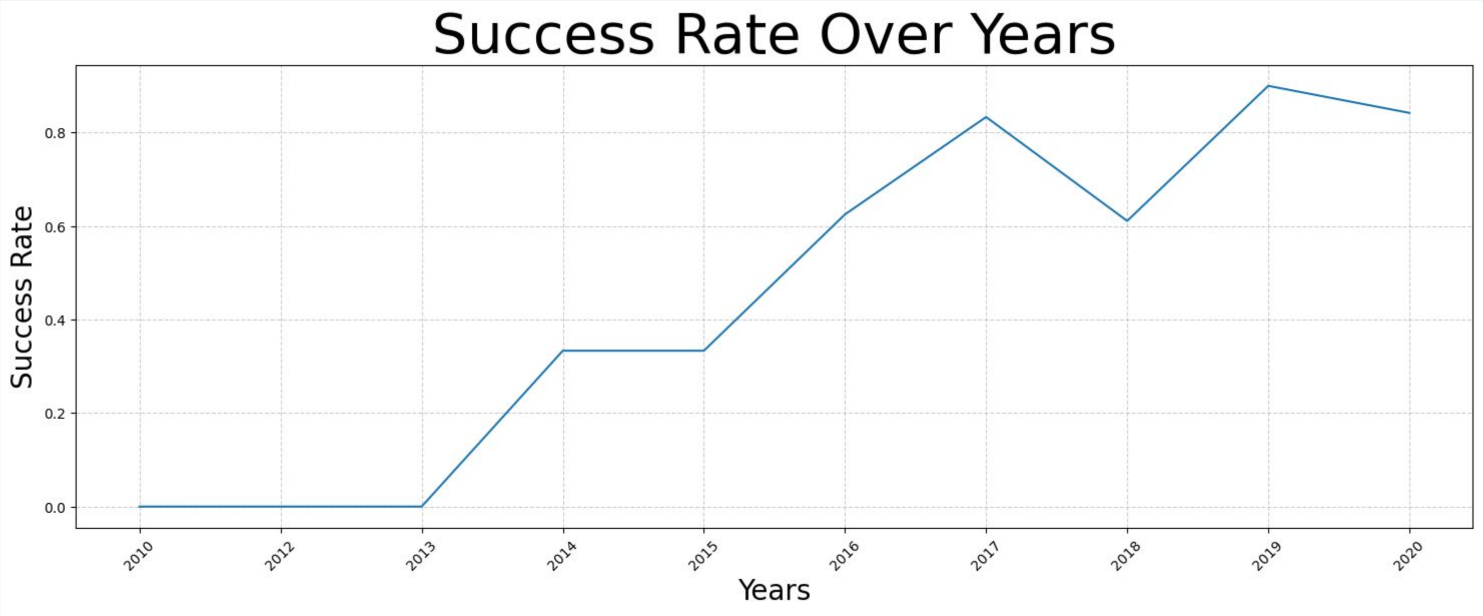
Payload vs. Orbit Type

Heavy Payloads, the successful landings are associated heavily with LEO, PO, and ISS orbits.



Launch Success Yearly Trend

- Success rate since 2013 kept increasing with a small crash in 2018 until 2020.



All Launch Site Names

- We used the keyword 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

```
12]:
#ROYE:
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE
```

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

12]:

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'

[13]:
```
#ROYE:
%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

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

t[13]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO
2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)

Used the simple query with a string lookup to display top five records with the matching string.

Total Payload Mass

Calculated total payload carried by boosters from NASA using the LIKE keyword.

Task 3

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

15]:
```
#ROYE:
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS 'Total Payload Mass in KG' FROM SPACEXTABL
```

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

15]:

Total Payload Mass in KG
619967

Average Payload Mass by F9 v1.1

- Used the avg method in sql to present the following result:

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

```
2]:
#ROYE:
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS 'Average of Payload Mass in KG of booster '
```

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

2]:

Average of Payload Mass in KG of booster version F9 v1.1
2928.4

First Successful Ground Landing Date

Use the Min() method in SQL to display the first Successful ground landing date from the edited dataframe:

Task 5

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

Hint:Use min function

In [26]:
```
#ROYE
%sql SELECT MIN(Date) AS "Date of First Successful Landing Outcome" FROM SPACEXT/
```

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

Out[26]:

Date of First Successful Landing Outcome
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Uses DISTINCT and WHERE keywords along with a subquery to determine success between certain payload sizes:

Task 6

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

6]:
```
#ROYE
%sql SELECT DISTINCT Booster_Version AS 'Booster Versions Successfully Landed in
```

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

6]:

Booster Versions Successfully Landed in Drone Ship
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
50]:
#ROYE
%sql SELECT Mission_Outcome AS 'Mission Outcome', COUNT(*) AS "Value" FROM SPACE
```

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

50]:

Mission Outcome	Value
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

Used wildcard '%' filtering WHERE keyword on success or failure:

Task 8

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

In [63]:

```
#ROYE
%sql SELECT Booster_Version as 'Booster Version' FROM SPACEXTABLE WHERE PAYLOAD_
```

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

Out[63]:

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

Created a substr with new dates to classify things through SQL to determine max payload through a subquery and WHERE/MAX fn's/keywords:

In [64]:

```
%%sql

SELECT
    CASE substr(Date, 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
    END AS Month_Name,
    Landing_Outcome,
    Booster_Version,
    Launch_Site
FROM
    SPACEXTABLE
WHERE
    substr(Date, 0, 5) = '2015'
    AND Landing_Outcome LIKE 'Failure%'
    AND Landing_Outcome LIKE '%drone ship%'
```

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

Out[64]:

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

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

Using COUNT, RANK() with and ORDER BY COUNT subquery: managed to rank outcomes:

In [66]:

```
%%sql
SELECT
    Landing_Outcome,
    COUNT(*) AS Outcome_Count,
    RANK() OVER (ORDER BY COUNT(*) DESC) AS Outcome_Rank
FROM
    SPACEXTABLE
WHERE
    Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY
    Landing_Outcome
ORDER BY
    Outcome_Rank;
```

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

Out[66]:

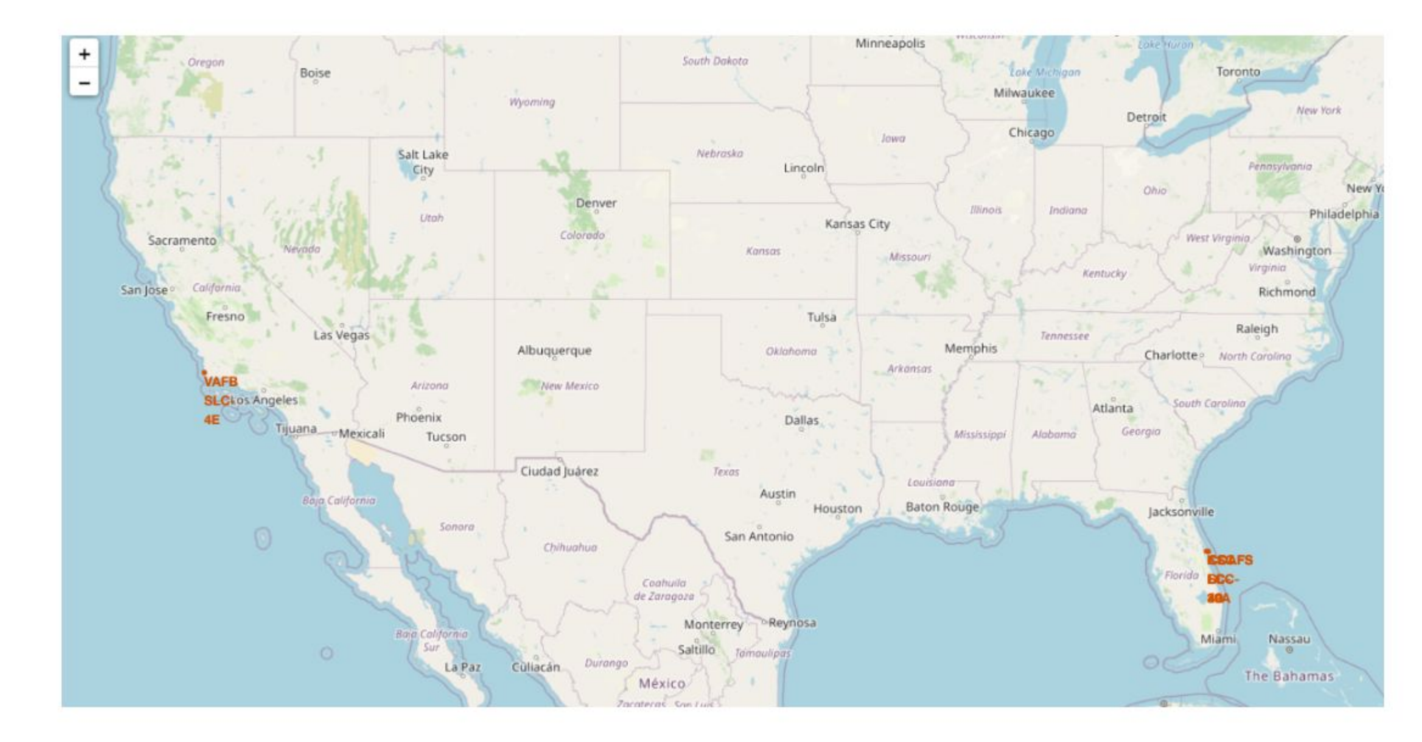
Landing_Outcome	Outcome_Count	Outcome_Rank
No attempt	10	1
Success (ground pad)	5	2
Success (drone ship)	5	2
Failure (drone ship)	5	2
Controlled (ocean)	3	5
Uncontrolled (ocean)	2	6
Precluded (drone ship)	1	7
Failure (parachute)	1	7

Section 3

Launch Sites Proximities Analysis

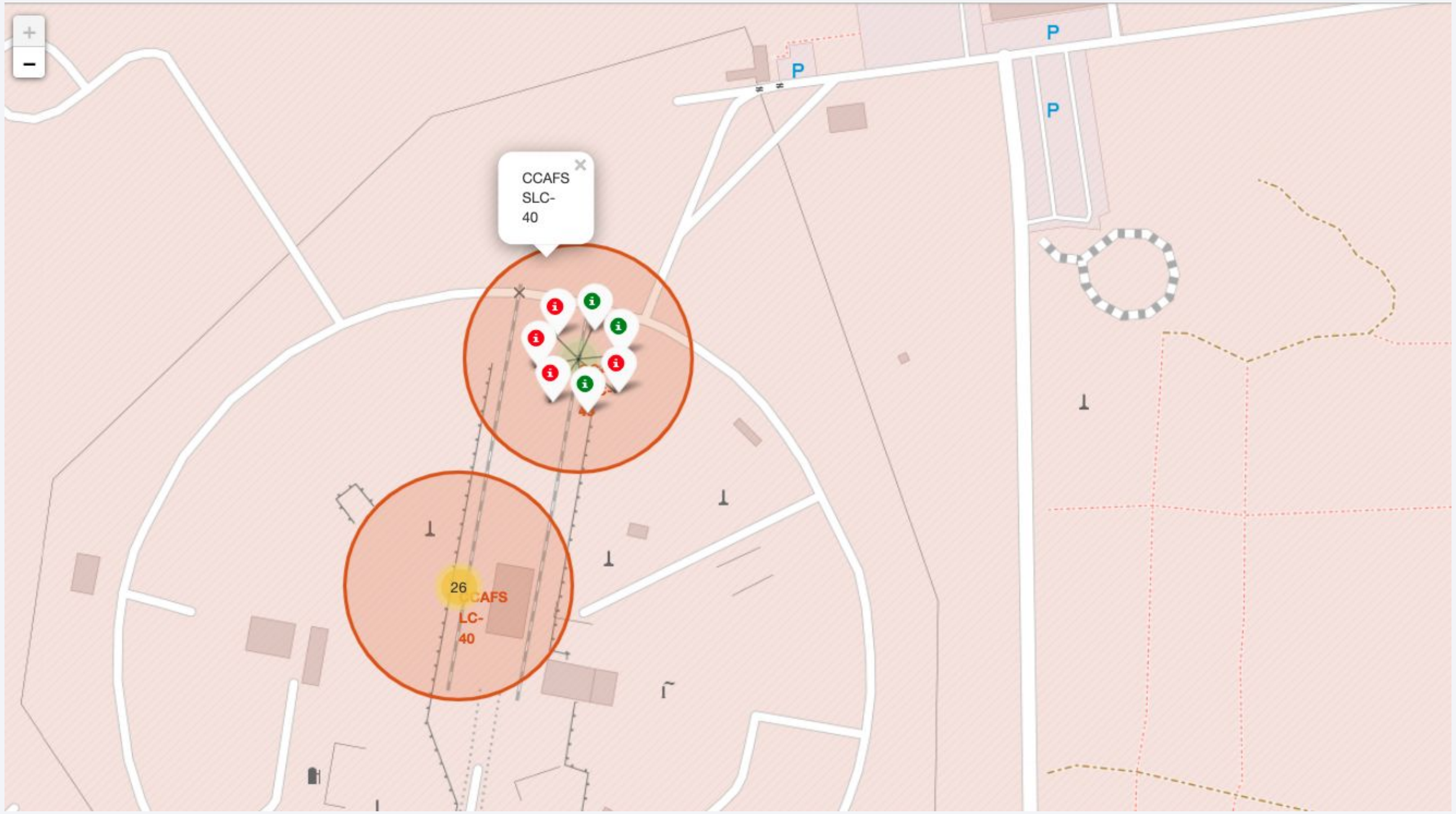
All National Map Markers

Shows a marker of where the space stations are (CA and FL)



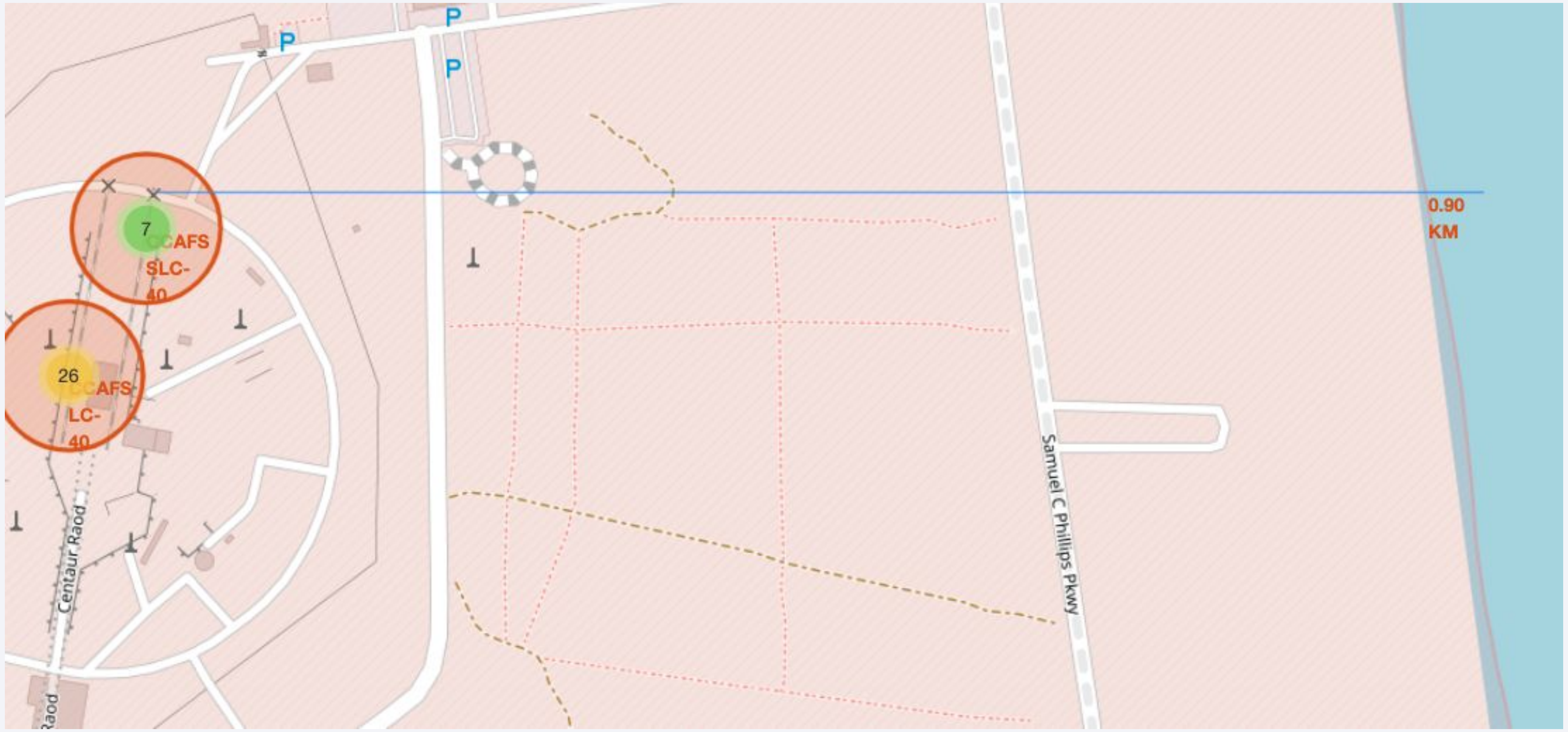
Markers Showing Launch Sites with Color Labels

Green Markers show successful launches and red shows failures



Launch Site Distance to Landmarks

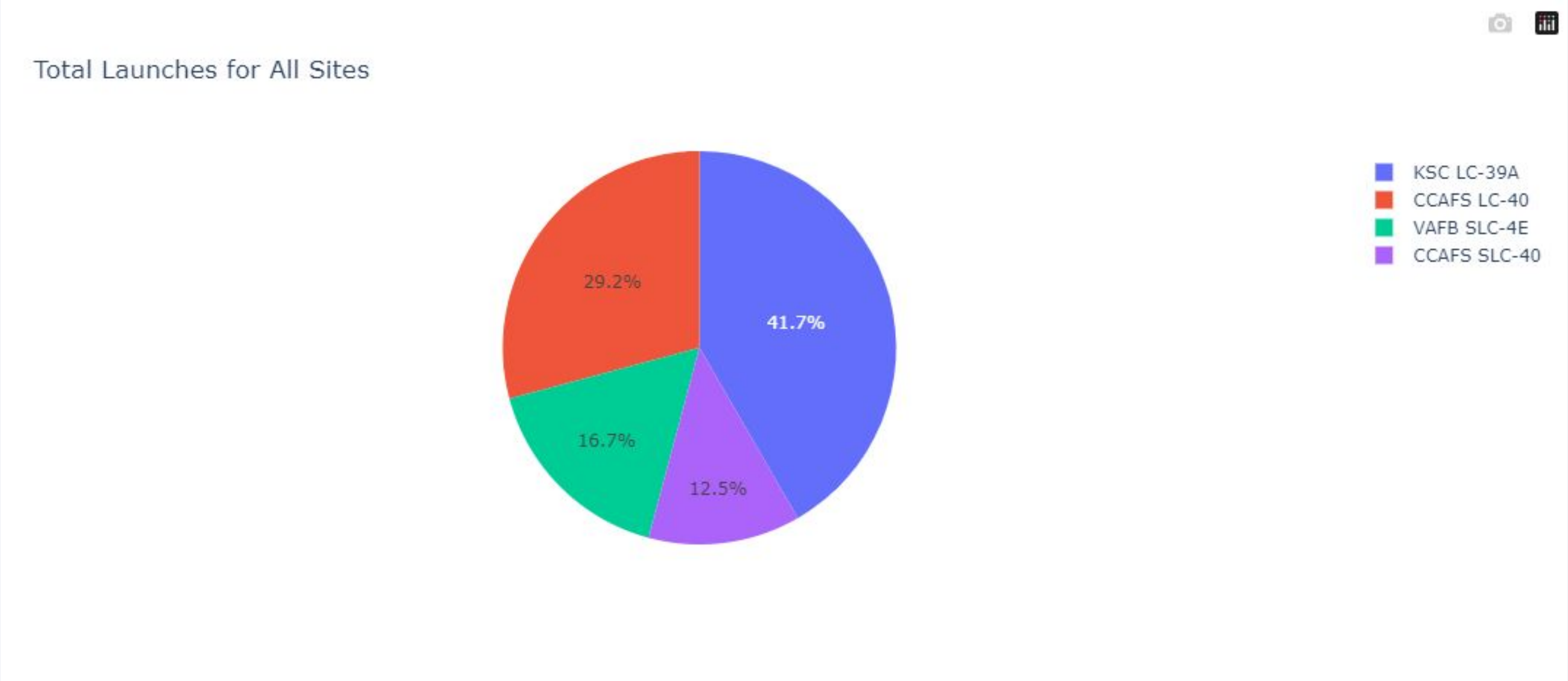
Also an example of lines drawn to show distance of coastline, landmarks, and highways in the lab:



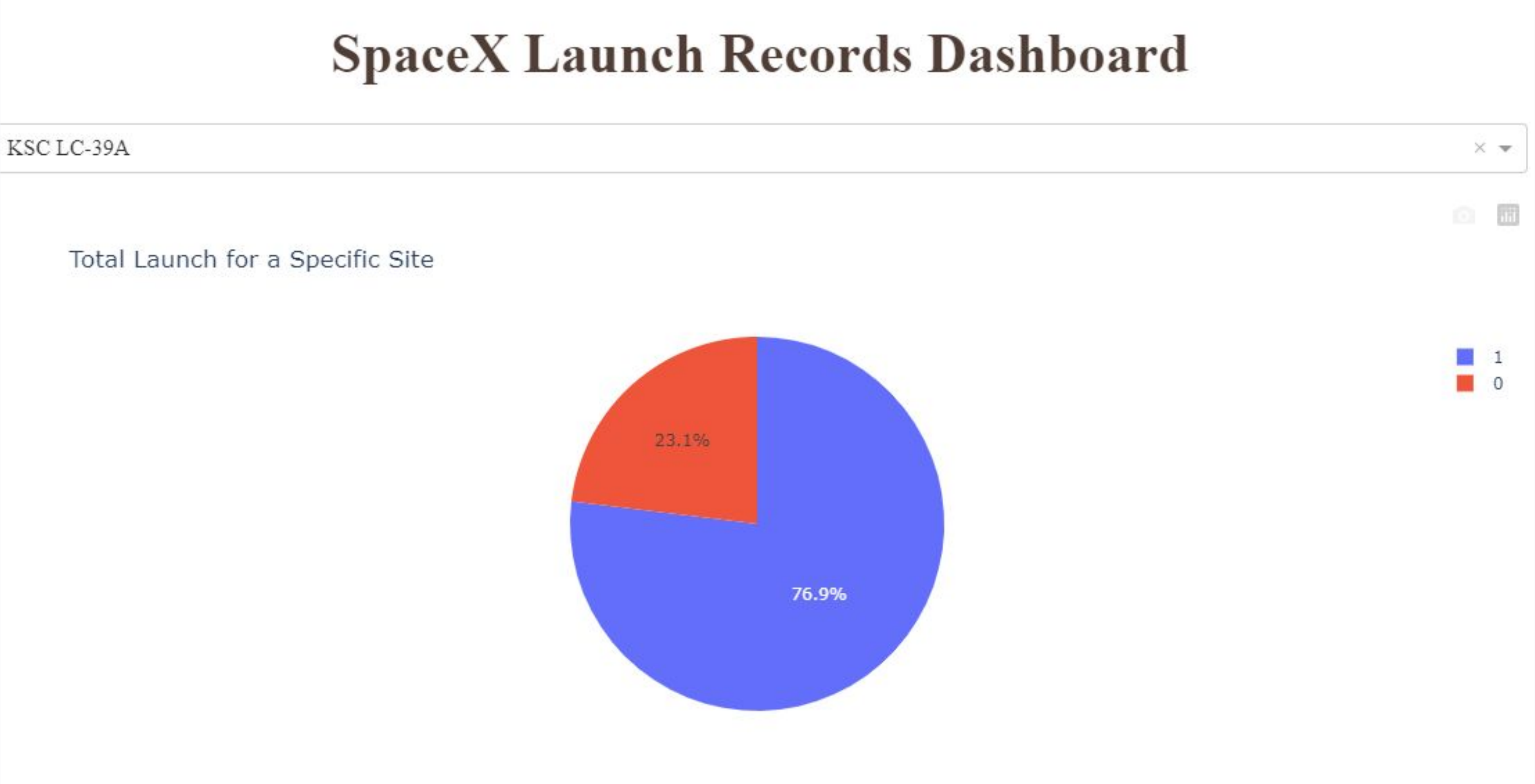
Section 4

Build a Dashboard with Plotly Dash

Pie Chart of Total Success Launches

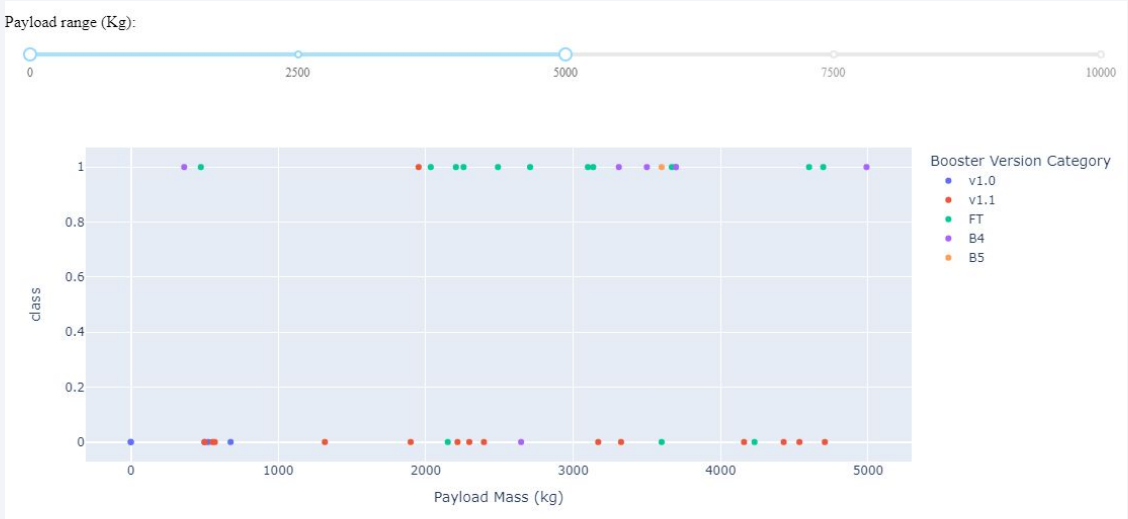


Launch Site with Highest Success Rate

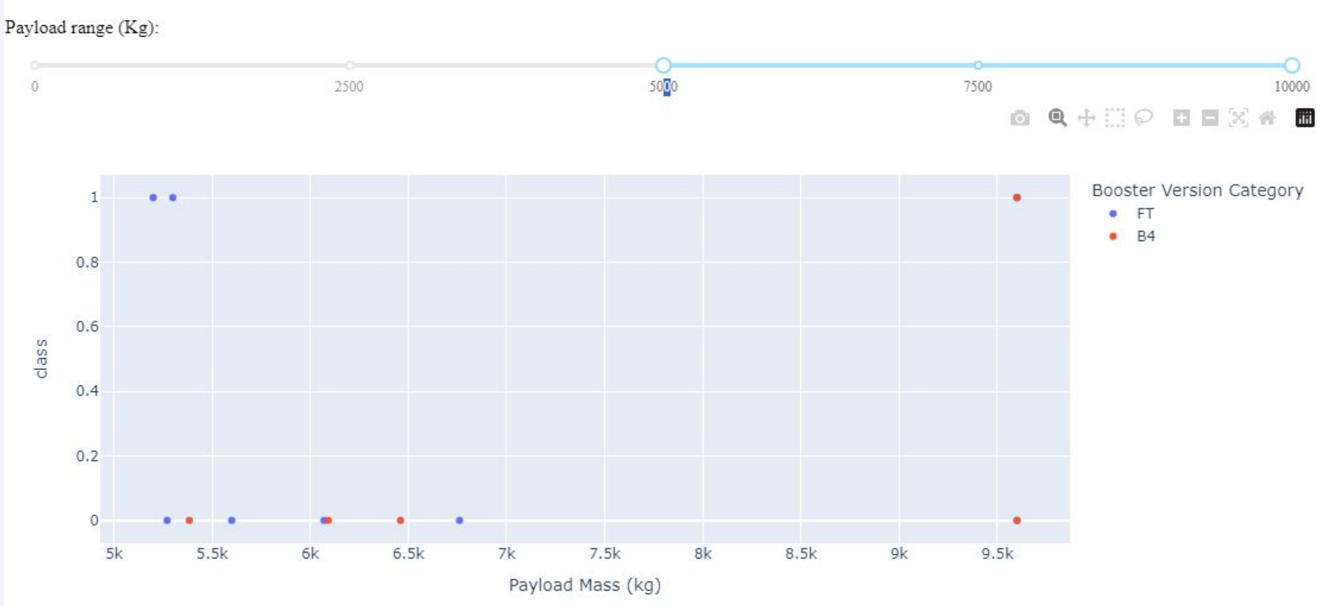


Payload Range of all Launch Outcomes using a Slider

1-5kg



5-10kg



Section 5

Predictive Analysis (Classification)

Classification Accuracy

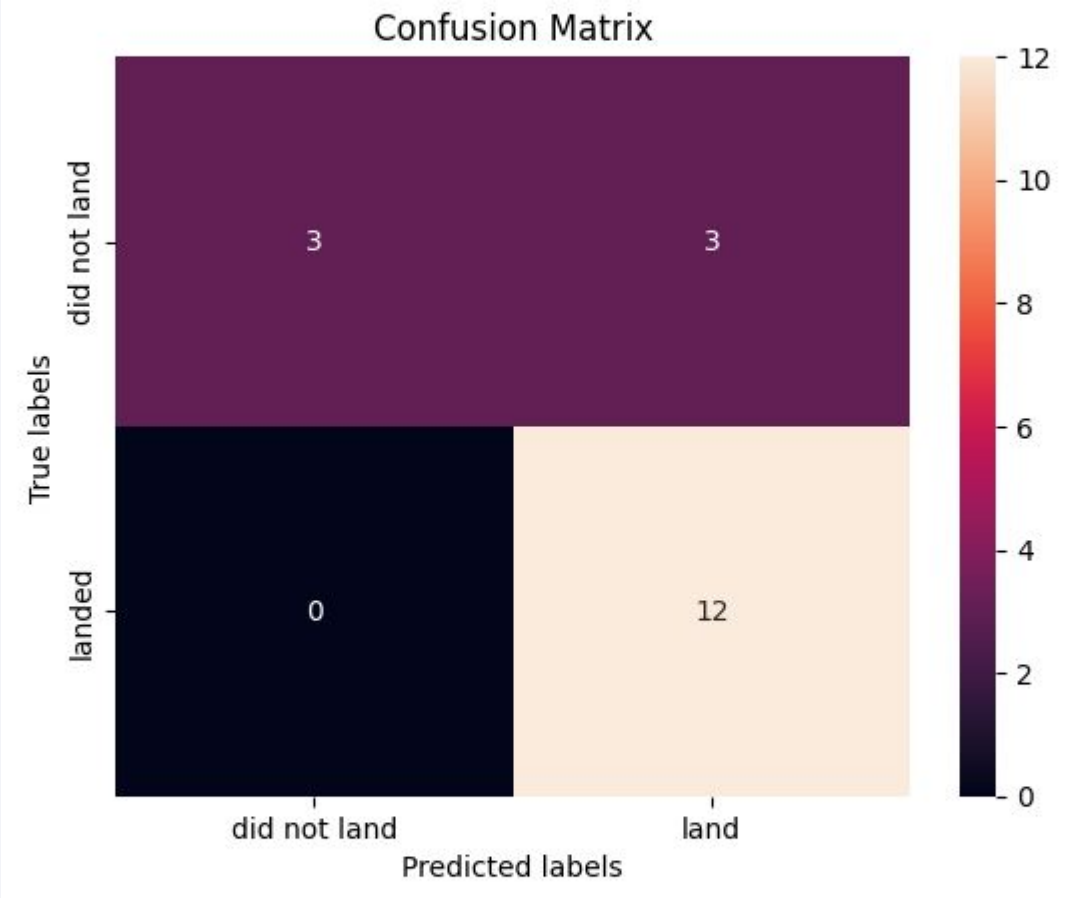
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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_)
```

```
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

- **Only major problem is false positives by the confusion Matrix. But it shows that the decision tree classifier can distinguish between different classes.**



Conclusions

- Larger the flight amount at the same sight, the better chances of it successfully landing due to improvements made
- Launch Success rate has overall been successful and has only increased from 2013-2020
- KSC LC-39A had the most successful launches.
- 5/9 orbits roughly had around the same success rate and were pretty high (ES-L1, GEO, HEO, SSO, VLEO)
- Decision Tree Classifier was the best machine learning algorithm for predictive behavior due to most of our observations and predictions being related to binary values (whether a launch was successful or not).

Thank you!