IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Shuvam Chakraborty
24.04.2025

Outline

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

Executive Summary

Data Collection Methodology:

- Collected data using **SpaceX API** and **Wikipedia web scraping**.

Data Wrangling:

- Applied **One-Hot Encoding** to categorical features.
- Cleaned data by handling missing values and duplicates.

Exploratory Data Analysis (EDA):

- Visualized trends using **histograms, bar charts, and scatter plots**.
- Used **SQL queries** to explore correlations and insights.

Interactive Visual Analytics:

- Created **interactive maps** with **Folium** to visualize launch sites.
- Developed a **Plotly Dash** dashboard for dynamic data exploration.

Predictive Analysis:

- Built **classification models** (Logistic Regression, Random Forest, SVM, KNN) to predict launch success.

Model Building & Evaluation:

- **GridSearchCV** for hyperparameter tuning.
- Evaluated models using **accuracy, precision, recall**, and **confusion matrix**.

Introduction

Project background and context

- **SpaceX Launch Data: The project revolves around analyzing SpaceX launch data to predict the success of future missions.**
- **Technologies Used: Utilized Python, Pandas, Scikit-learn, Plotly Dash, Folium, and SQL to analyze and visualize the data.**
- **Objective: To understand launch success factors, predict the outcome of SpaceX launches, and provide insights into the operations of launch sites.**

Problems you want to find answers

- **Launch Success Prediction**: What factors contribute to a SpaceX launch being successful or unsuccessful?
- **Impact of Launch Site**: Do different launch sites impact the success rate of launches?
- **Proximity to Infrastructure**: Are launch sites in close proximity to railways, highways, or coastlines?
- **Geographical Factors**: How do geographical features (latitude, longitude) influence launch success?
- **Performance of Models**: Which classification models (Logistic Regression, SVM, etc.) perform best for launch success prediction?

Section 1

Methodology

Methodology

Executive Summary

Data Collection Methodology:

- Data was collected using the SpaceX API and web scraping from Wikipedia.

Data Wrangling:

- Cleaned and transformed raw data for analysis.
- Handled missing data, outliers, and irrelevant features.

Exploratory Data Analysis (EDA):

- Used visualizations and SQL queries to explore patterns, trends, and relationships in the data.

Interactive Visual Analytics:

- Used Folium to create interactive maps and Plotly Dash for visualizing trends and metrics.

Predictive Analysis:

- Applied classification models to predict outcomes based on the available features.

Model Building, Tuning, and Evaluation:

- Built classification models such as Logistic Regression, SVM, Decision Trees, and K-Nearest Neighbors.
- Tuned hyperparameters using GridSearchCV.
- Evaluated model performance using metrics such as accuracy, precision, recall, and F1-score.

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/ShuvamChakraborty-B/DataScience-Capstone/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=requests.get(static_json_url)
```

In [11]:
```
response.status_code
```

Out[11]: 200

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

In [32]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data1=response.json()
data=pd.json_normalize(data1)
```

Using the dataframe `data` print the first 5 rows

In [33]:
```
# Get the head of the dataframe
data.head(5)
```

Out[33]:

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of	[]	[]	[]

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/ShuvamChakraborty-B/DataScience-Capstone/blob/main/jupyter-labs-webscraping.ipynb

In [5]:
```
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.

In [7]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response=requests.get(static_url)
if response.status_code==200:
    print("Request Sucessful")
else:
    print("Request Unscessful")
```

```
Request Sucessful
```

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

In [9]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup=BeautifulSoup(response.text,"html.parser")
```

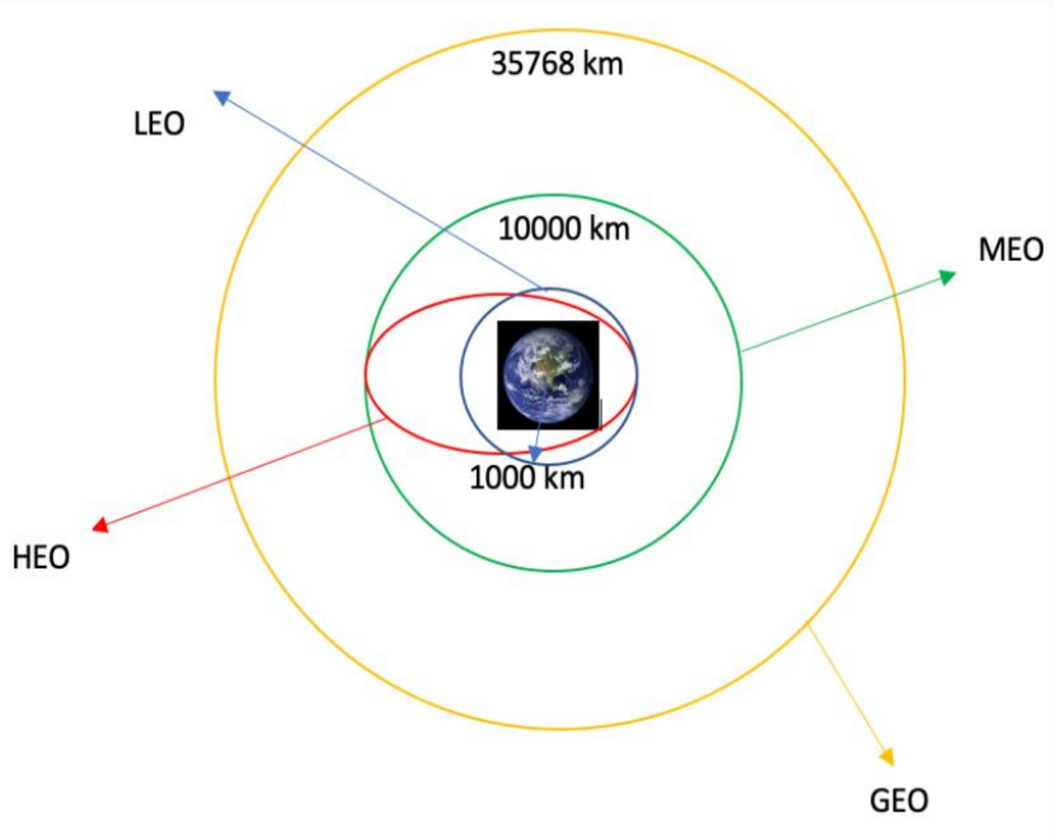
Print the page title to verify if the `BeautifulSoup` object was created properly

In [10]:
```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

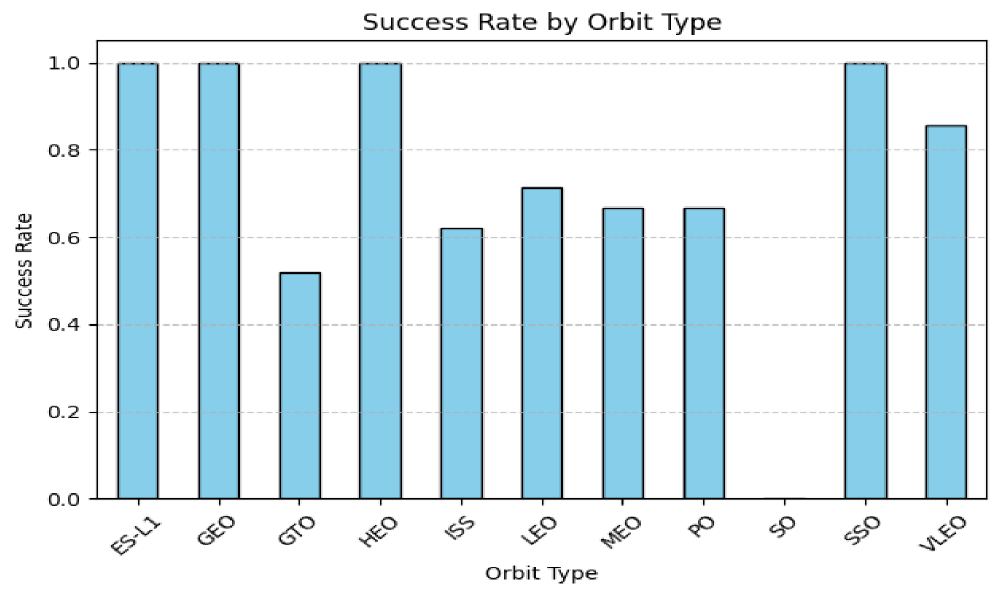
Data Wrangling

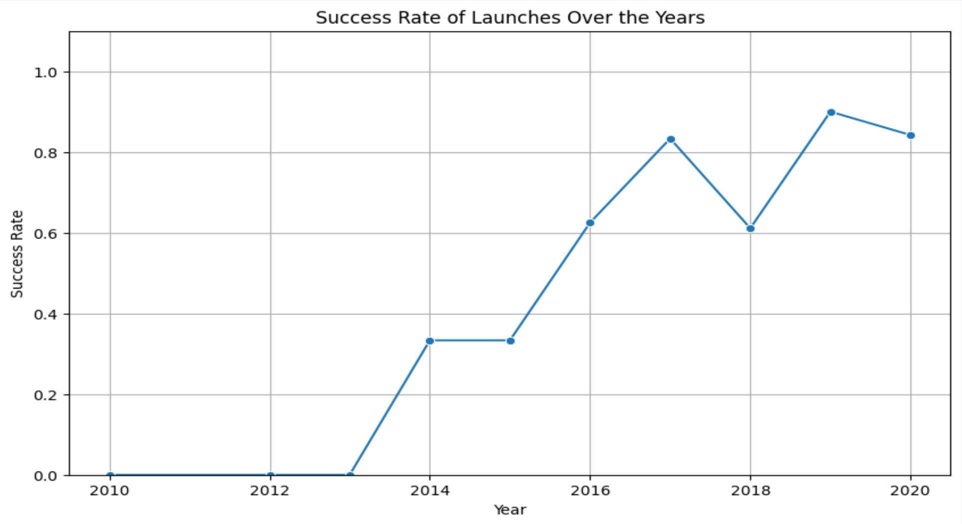
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/ShuvamChakraborty-B/DataScience-Capstone/blob/main/labs-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/ShuvamChakraborty-B/DataScience-Capstone/blob/main/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.
- The link to the notebook is https://github.com/ShuvamChakraborty-B/DataScience-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(2).ipynb

Build an Interactive Map with Folium

Key Features:

•All SpaceX launch sites were plotted using folium map markers folium map markers..

•Launch outcomes were labeled with color-coded markers:

- •**Green** → Success (1)
- •**Red** → Failure (0)

•Marker clusters helped visualize which launch sites had relatively higher success rates.

Proximity & Location Analysis:

•Measured distances from each launch site to nearby:

- 🚆 Railways
- 🛣️ Highways
- 🌊 Coastlines
- 🏙️ Cities

Answered key questions:

•Are launch sites located close to railways, highways, or coastlines?

•Are launch sites purposefully distant from populated urban areas?

Objective Achieved:

•Combined interactive maps with analytical reasoning to better understand the **geospatial factors** influencing launch success.

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/ShuvamChakraborty-B/DataScience-Capstone/blob/main/dash.py

Predictive Analysis (Classification)

Model Development Process:

•**Data Preparation:**
Preprocessed dataset using **One-Hot Encoding** to convert categorical variables (e.g., launch site, orbit, booster version) into numerical format.

•**Model Selection:**
Applied and compared several classification models:
- •Logistic Regression
- •Support Vector Machine (SVM)
- •Decision Tree
- •K-Nearest Neighbors (KNN)

•**Model Tuning:**
- •Used GridSearchCV with **10-fold cross-validation** to find best hyperparameters.
- •Tuned parameters such as C, kernel, max_depth, and n_neighbors.

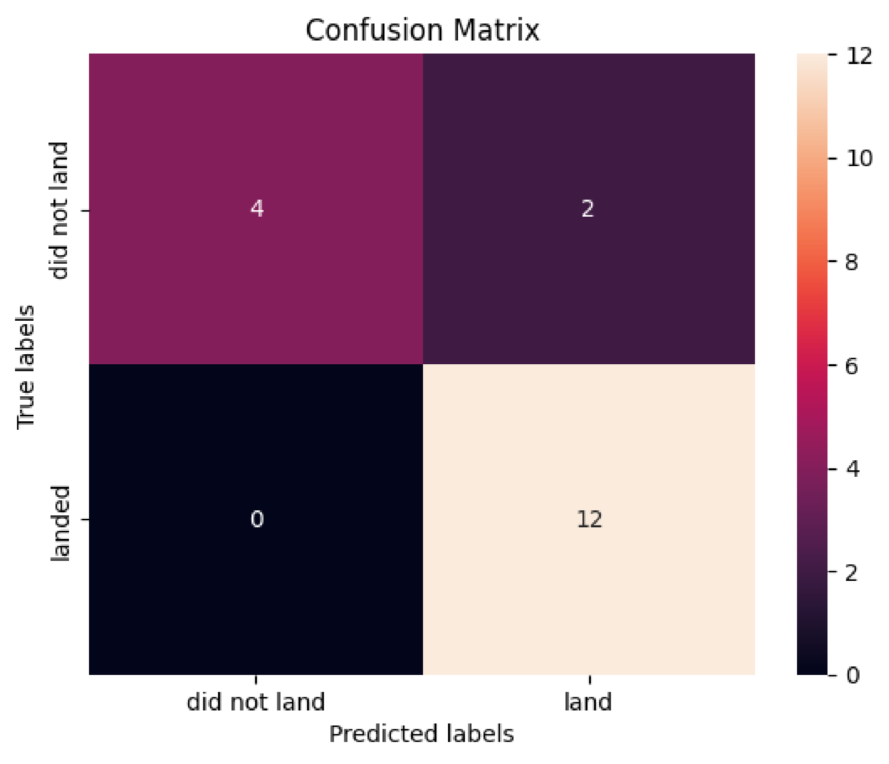
•**Model Evaluation:**
- •Evaluated each model using metrics:
  - ✔ Accuracy
  - ✔ Confusion Matrix
  - ✔ Precision & Recall
  - ✔ ROC Curve & AUC Score

•**Best Performing Model:**
Identified the model with **highest validation accuracy and balanced performance** on the test set, which is Decision Tree

The link to the notebook is https://github.com/ShuvamChakraborty-B/DataScience-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

📊 **Objective:**
Build and evaluate classification models to predict launch success (1) or failure (0).
⚙️ **Models Used:**
•Logistic Regression
•K-Nearest Neighbors (KNN)
•Support Vector Machine (SVM)
•Decision Tree
🔧 **Methodology:**
•Applied **GridSearchCV** for hyperparameter tuning
•Used **train-test split** (80-20) for evaluation
•Evaluated with **accuracy, confusion matrix, ROC curve**
🏆 **Best Performing Model:**
•**Decision Tree**
•Achieved **~88% accuracy** on test data



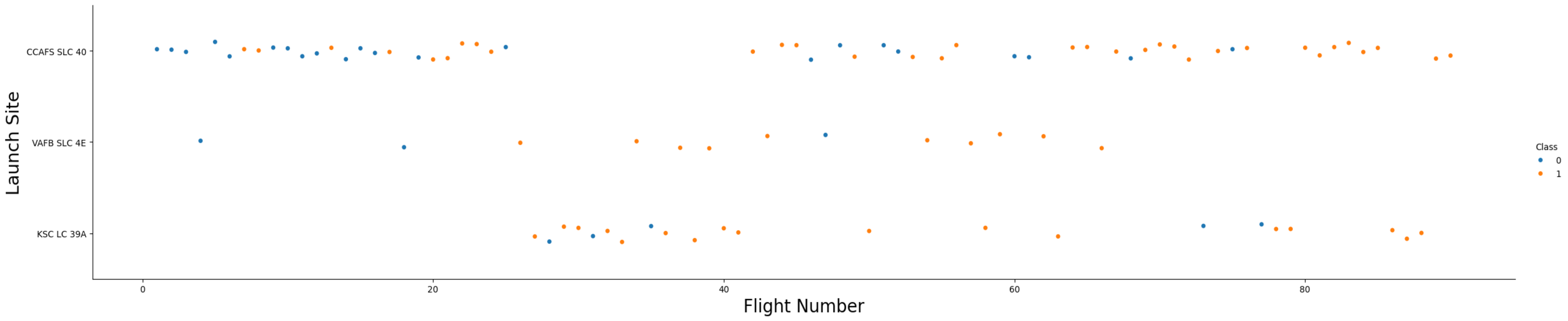
Matrix for Decision Tree

Section 2

Insights drawn from EDA

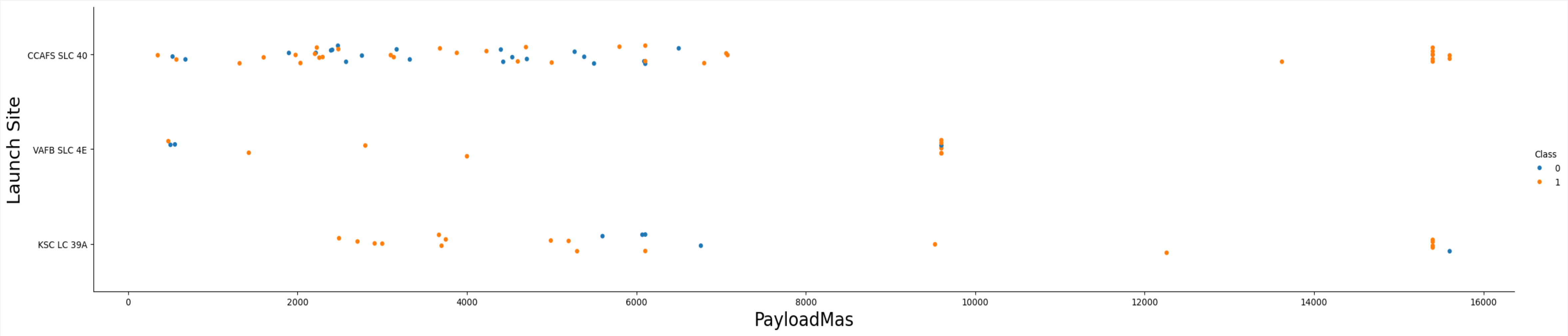
Flight Number vs. Launch Site

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



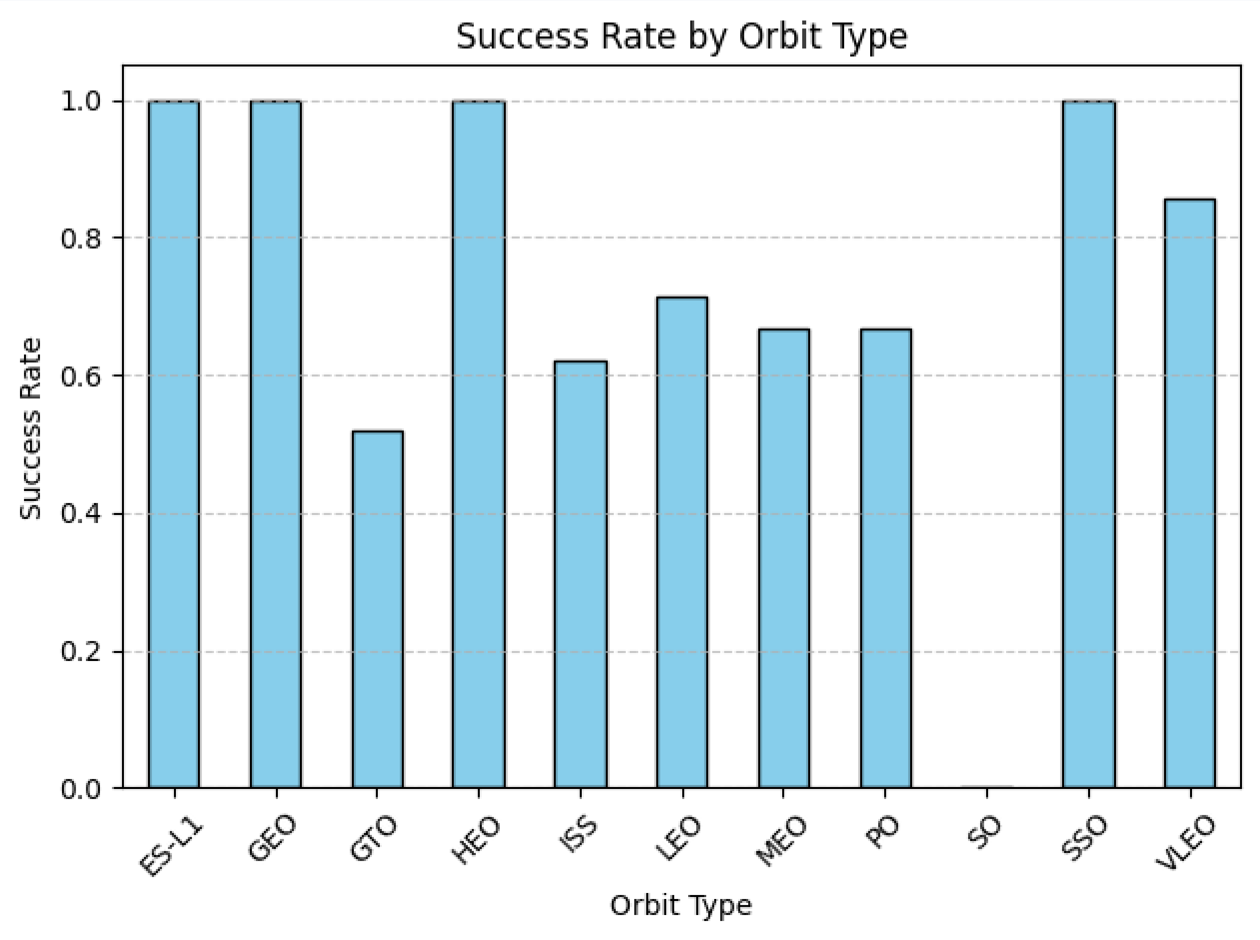
Payload vs. Launch Site

- The greater the payload, the higher the success.



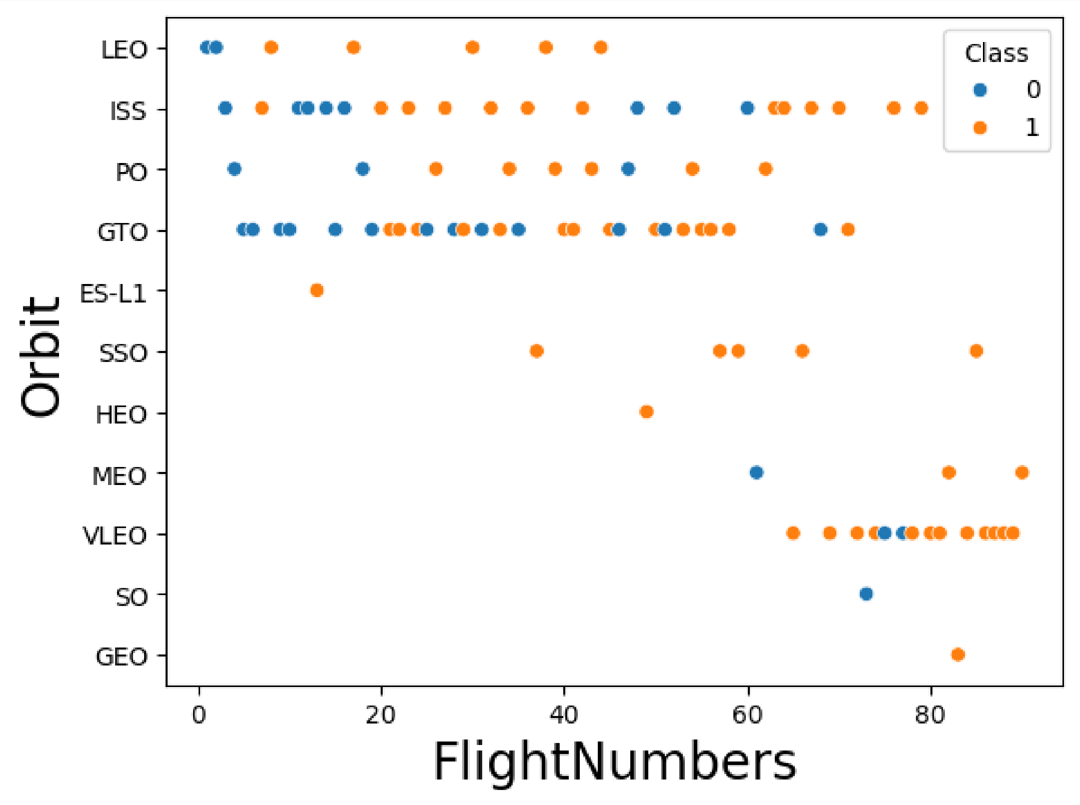
Success Rate vs. Orbit Type

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



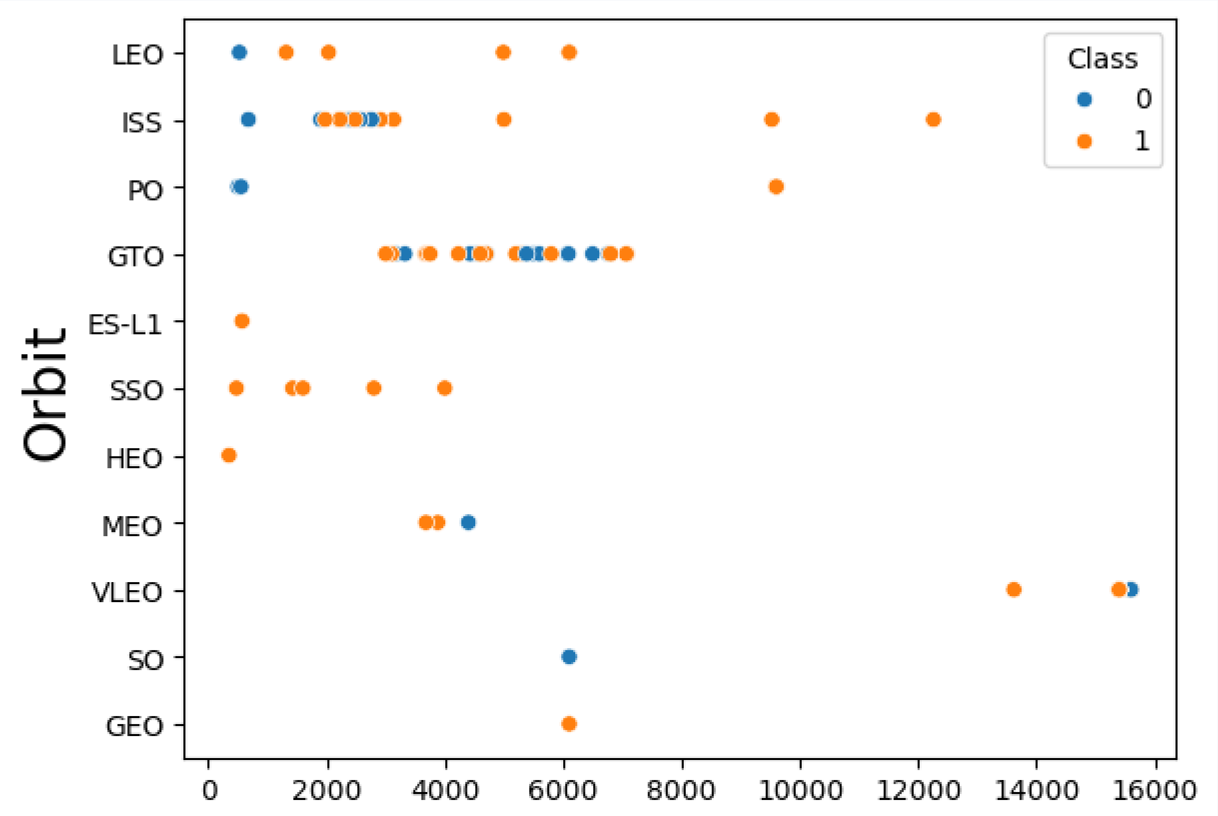
Flight Number vs. Orbit Type

- 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.



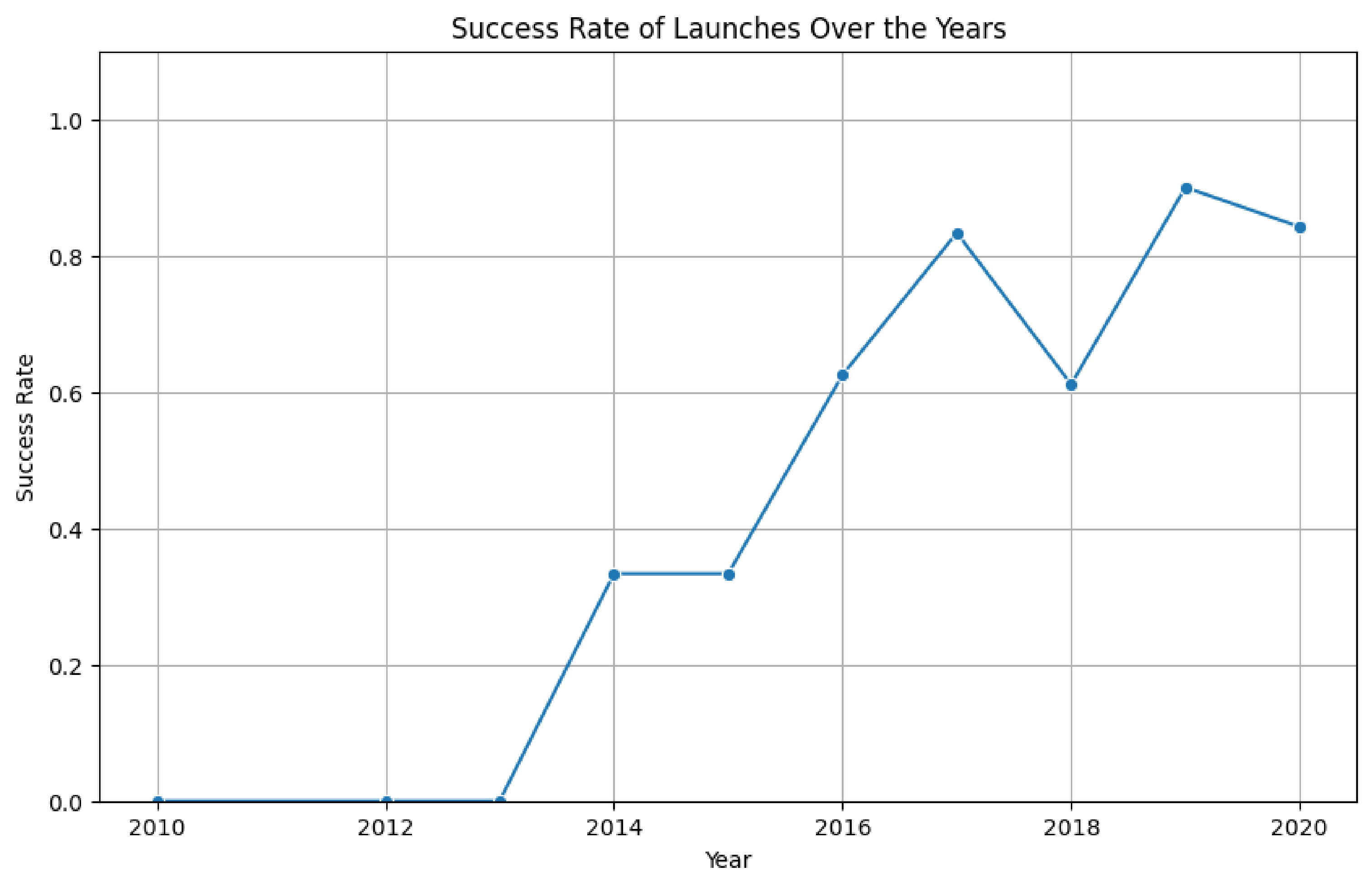
Payload vs. Orbit Type

- 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.

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

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

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

Launch Site Names Begin with 'CCA'

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

In [11]:

```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- 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 45596 using the query below

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

In [12]:

```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

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 [13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	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

In [14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

In [15]:

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
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

List the total number of successful and failure mission outcomes

In [16]:

```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

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

	successoutcome
0	100

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

Out[16]:

	failureoutcome
0	1

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

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

In [17]:

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
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
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	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 failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
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

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:

```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

Out[19]:

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

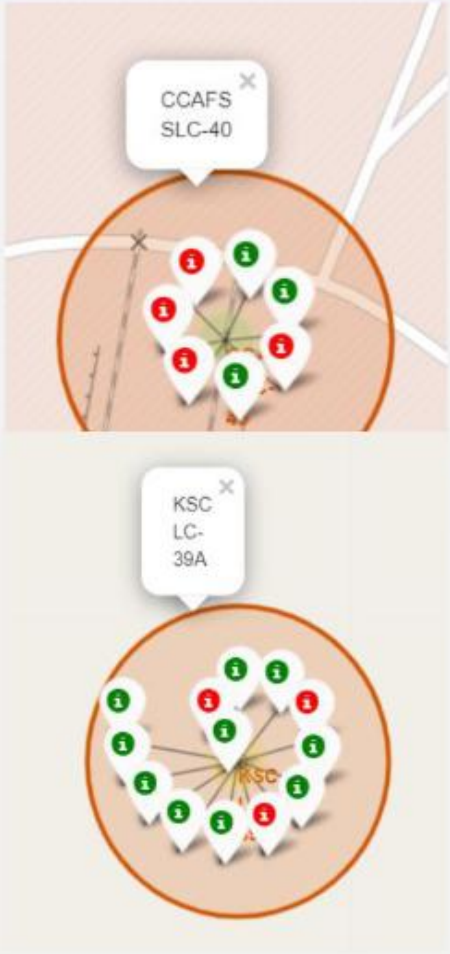
- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing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

Section 3

Launch Sites Proximities Analysis

All launch sites global map markers

Markers showing launch sites with color labels



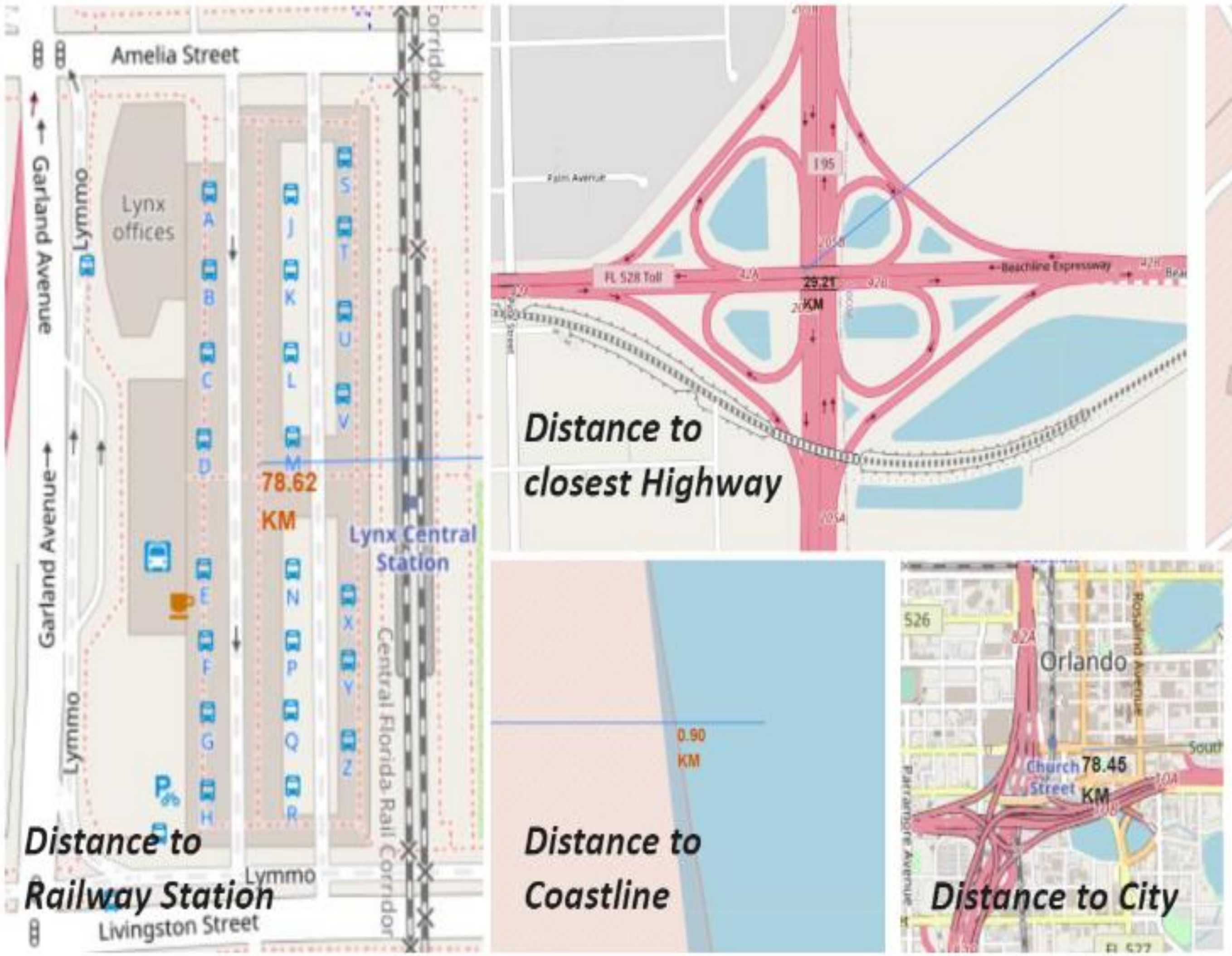
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

Launch Site distance to landmarks

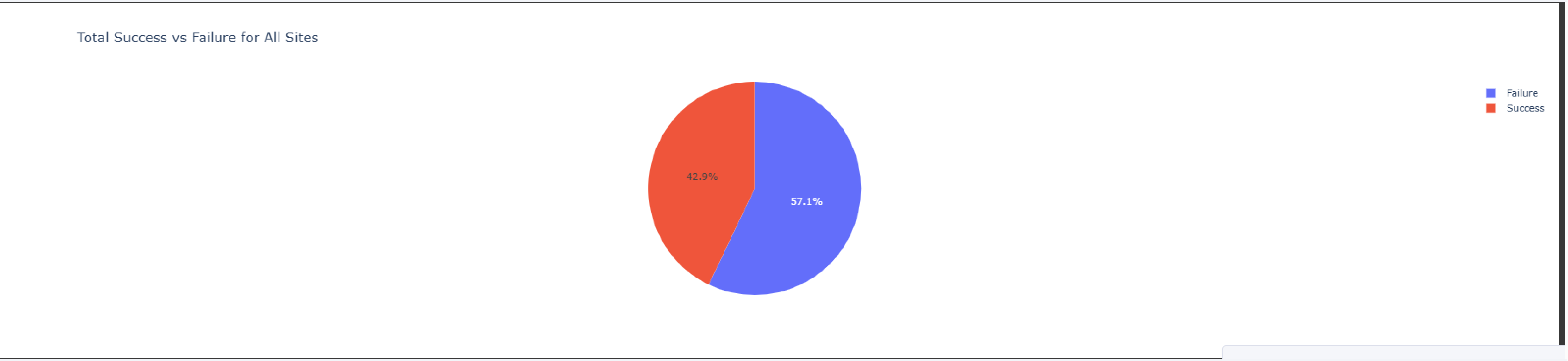


- •Are launch sites in close proximity to railways? No
- •Are launch sites in close proximity to highways? No
- •Are launch sites in close proximity to coastline? Yes
- •Do launch sites keep certain distance away from cities? Yes

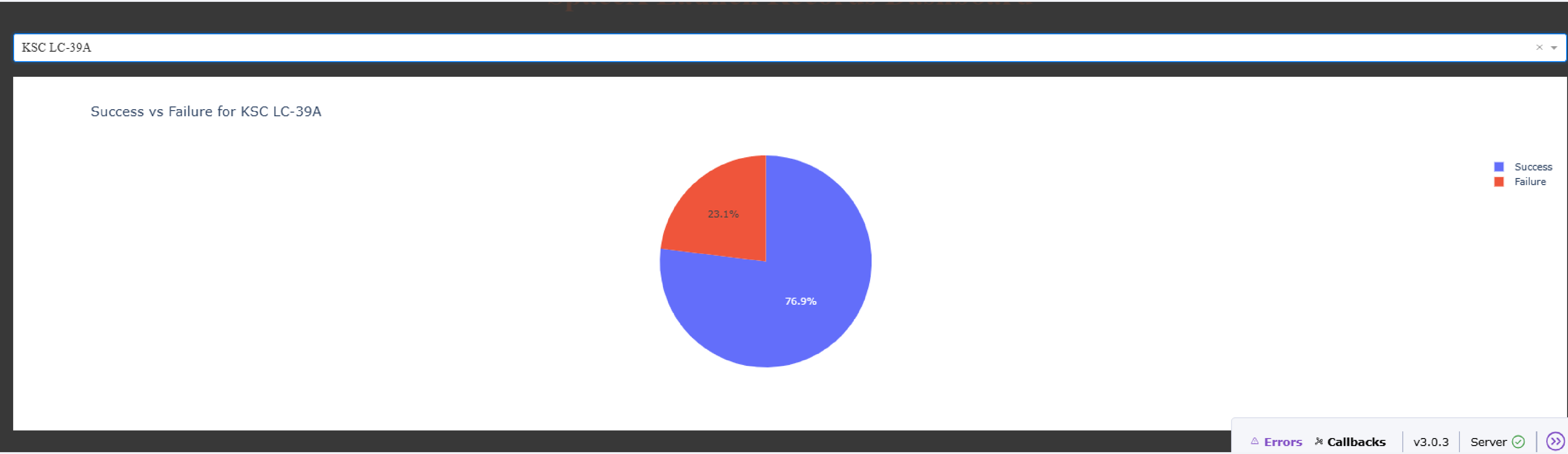
Section 4

Build a Dashboard with Plotly Dash

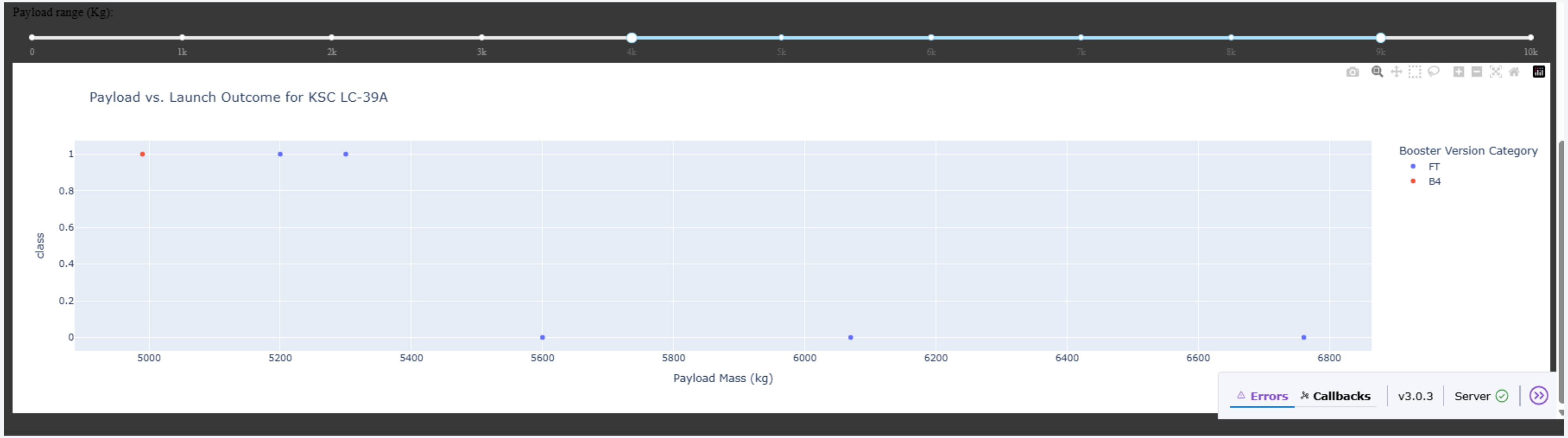
Total Success VS Failure of all sites

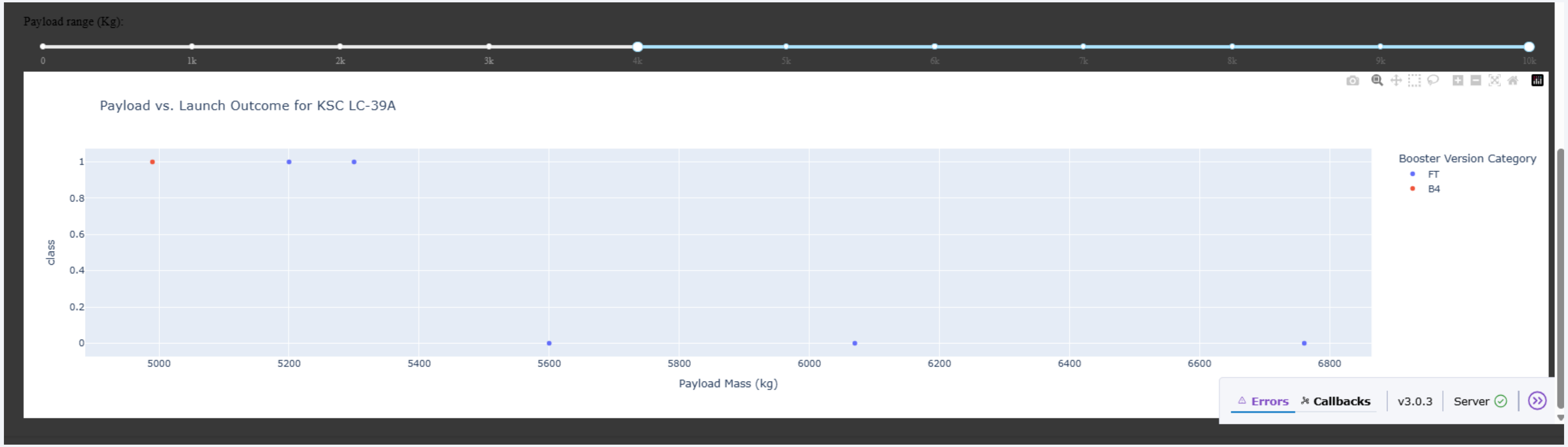


Pie chart showing the Launch site with the highest launch success ratio



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





Section 5

Predictive Analysis (Classification)

Classification Accuracy

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

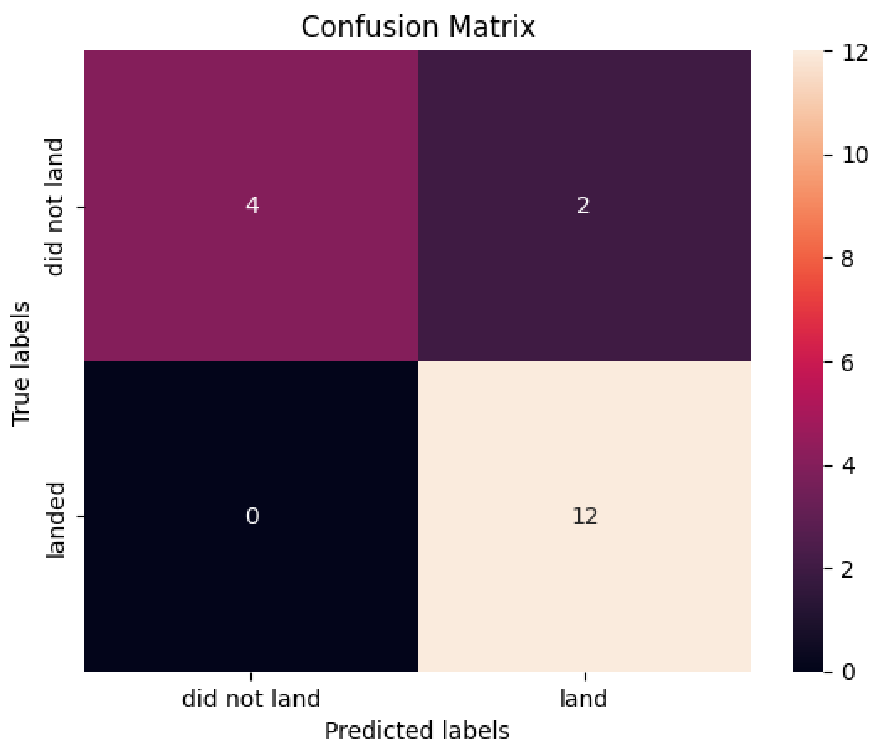
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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

- 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.



Matrix for Decision Tree

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!