IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

DEVAPRASAD JAYAKUMAR
03-06-2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API and 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

  This capstone project focuses on predicting the successful landing of the Falcon 9 first stage. SpaceX offers cost-effective rocket launches by reusing the first stage, resulting in significant cost savings compared to other providers. The project aims to analyze data on Falcon 9 landings, build interactive visualizations with Plotly Dash, create an interactive map using Folium, and employ machine learning algorithms to determine landing success. The insights derived will aid in estimating launch costs and supporting decision-making for companies bidding against SpaceX.

- Problems you want to find answers

  1. Can we predict the successful landing of the Falcon 9 first stage?
  2. How can the prediction of landing success be utilized to estimate launch costs?
  3. What insights can be gained to support companies bidding against SpaceX for rocket launches?

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
- 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 collection process involved multiple steps to gather Falcon 9 launch data from different sources. Here is a breakdown of the process:

1.RESTful API: We utilized a GET request to access the SpaceX API, which provided us with relevant launch data. The response content was decoded as JSON using the **.json()** function, allowing us to extract the required information.
2.Data Transformation: To convert the JSON response into a structured format suitable for analysis, we used the **.json_normalize()** function in pandas. This transformed the data into a pandas DataFrame, enabling us to manipulate and analyze it effectively.
3.Data Cleaning: After creating the DataFrame, we performed data cleaning procedures. This involved checking for missing values and filling them in where necessary. By ensuring data integrity and completeness, we prepared the data for further analysis.
4.Web Scraping: In addition to the API, we conducted web scraping from Wikipedia to obtain Falcon 9 launch records. Using BeautifulSoup, we extracted the launch records from the HTML table present on the Wikipedia page. We then parsed the table data and converted it into another pandas DataFrame for future analysis.

By combining data collection from the SpaceX API and web scraping from Wikipedia, we gathered comprehensive Falcon 9 launch data. This allowed us to have a rich dataset for our analysis and predictive modeling.

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.

- [Data-Science-Capstone/spacex-data-collection-api.ipynb at main · Devaprasad-coursera/Data-Science-Capstone (github.com)]()

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.
- [Data-Science-Capstone/data-collection-webscraping.ipynb at main · Devaprasad-coursera/Data-Science-Capstone · GitHub]

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

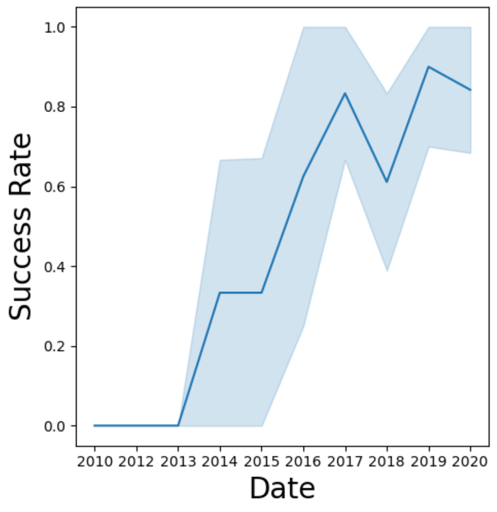
- Performed exploratory data analysis and determined the training labels.
- Calculated the number of launches at each site, and the number and occurrence of each orbits.
- Created landing outcome label from outcome column and exported the results to csv.

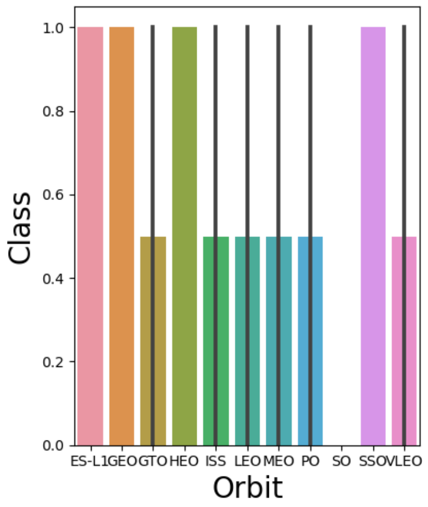
- Data-Science-Capstone/spacex-data-wrangling.ipynb at main · Devaprasad-coursera/Data-Science-Capstone · GitHub

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.

- [Data-Science-Capstone/eda-data-visualization.ipynb at main · Devaprasad-coursera/Data-Science-Capstone · GitHub]()





EDA with SQL

- We loaded the SpaceX dataset into a Db2 database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out:
  - **Display the names of the unique launch sites in the space mission**
  - **Display the total payload mass carried by boosters launched by NASA (CRS)**
  - **Display average payload mass carried by booster version F9 v1.1**
  - **List the total number of successful and failure mission outcomes**
  - **List the names of the booster_versions which have carried the maximum payload mass**
- Data-Science-Capstone/eda-sql.ipynb at main · Devaprasad-coursera/Data-Science-Capstone · GitHub

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.

- Data-Science-Capstone/Interactive Visual Analytics with Folium.ipynb at main · Devaprasad-coursera/Data-Science-Capstone · GitHub

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 between payload and launch success.

- Data-Science-Capstone/spacex_dash_app.py at main · Devaprasad-coursera/Data-Science-Capstone · GitHub

Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- Built different machine learning models and tune different hyperparameters using GridSearchCV.
- Used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- Found the best performing classification model.

- Data-Science-Capstone/SpaceX-Machine-Learning-Prediction.ipynb at main · Devaprasad-coursera/Data-Science-Capstone · GitHub

Results

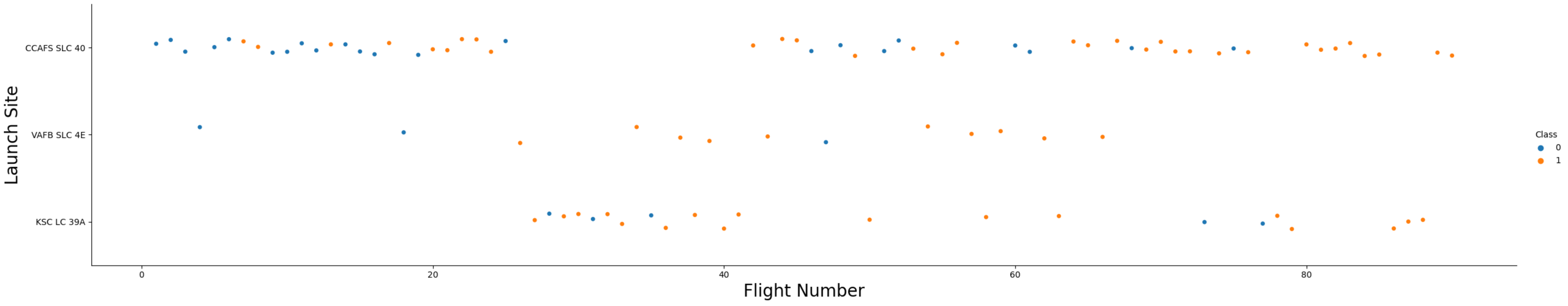
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

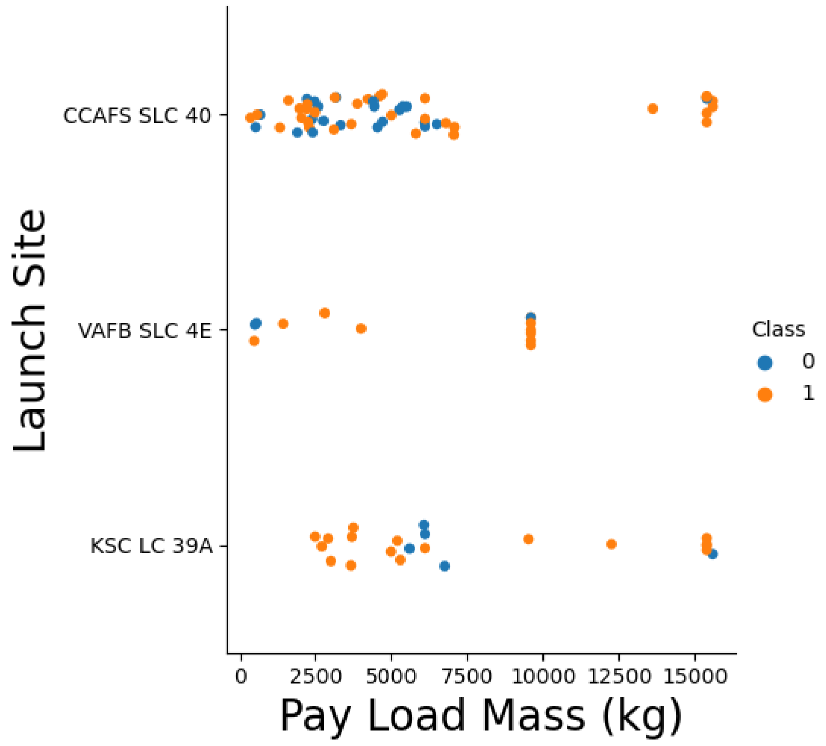
Insights drawn from EDA

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.

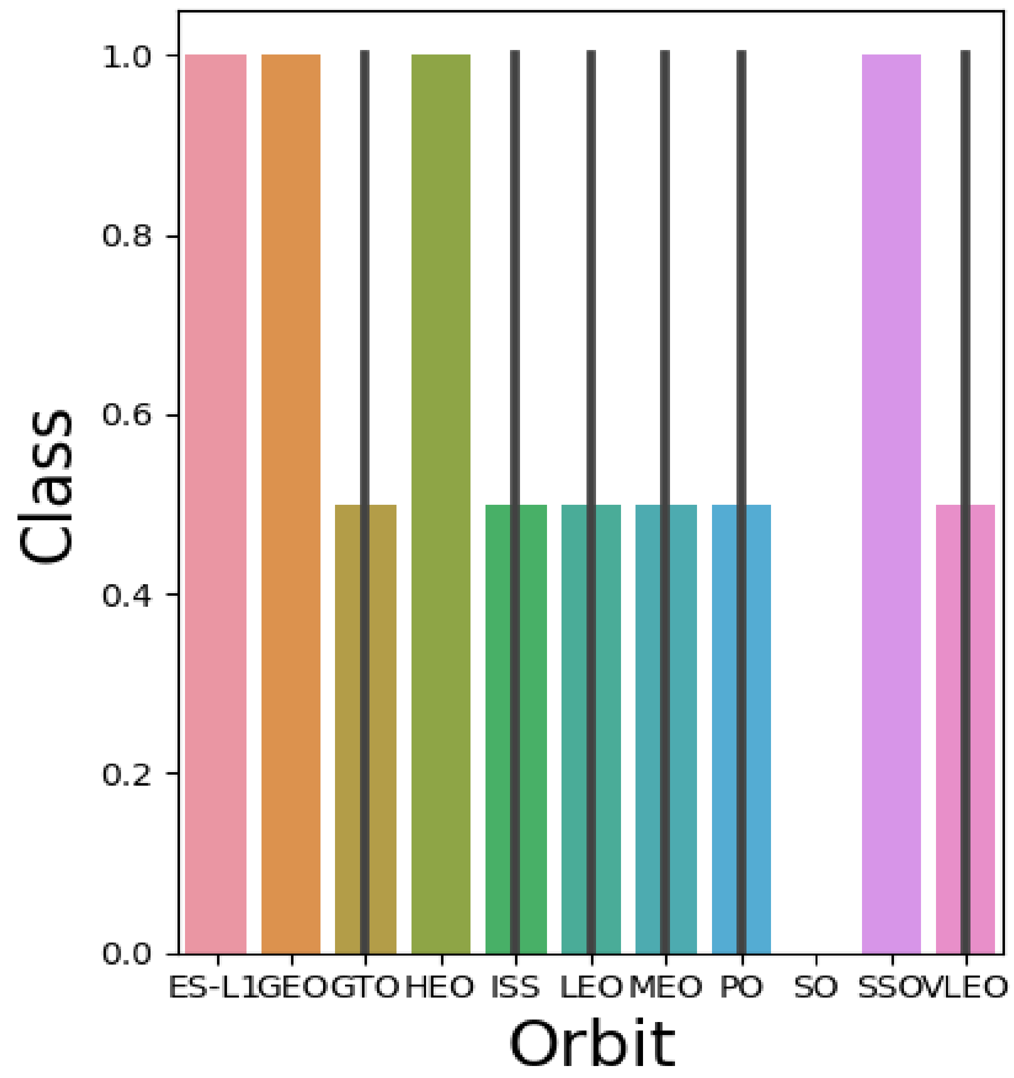


Payload vs. Launch Site



The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.

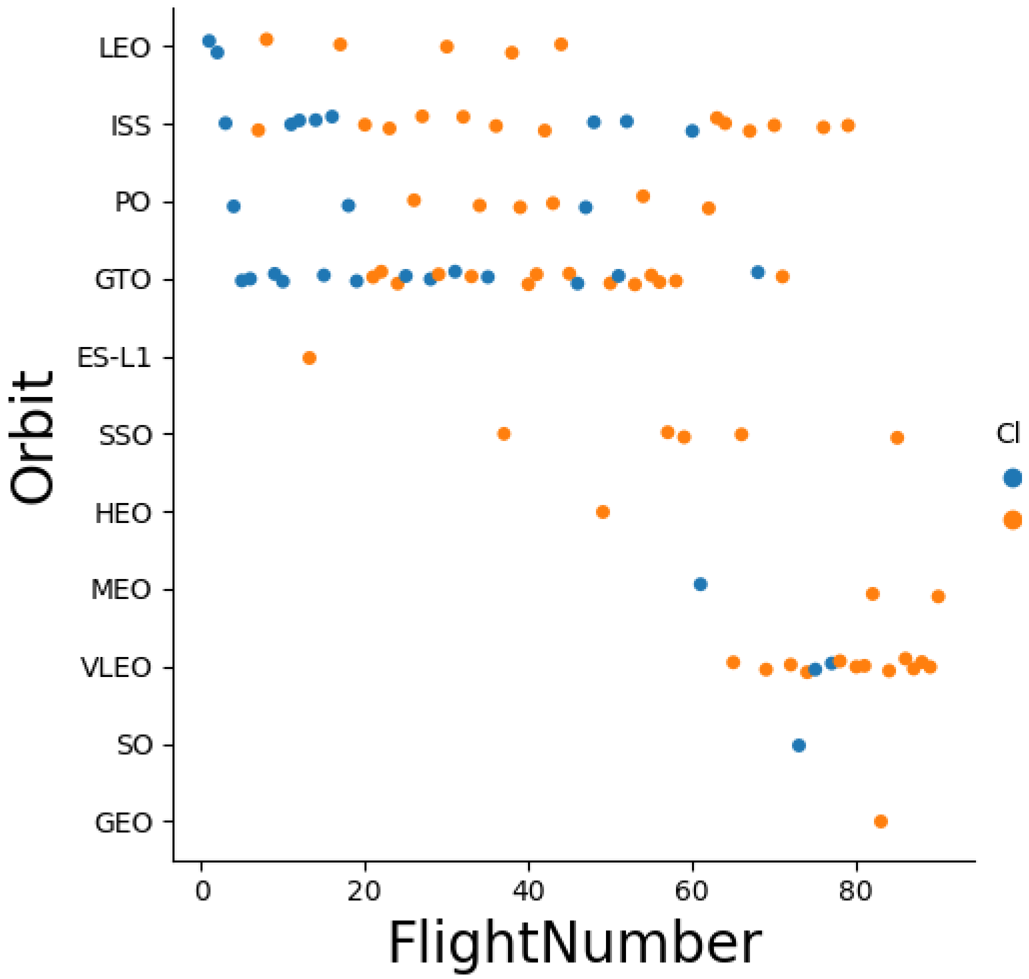
Success Rate vs. Orbit Type



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

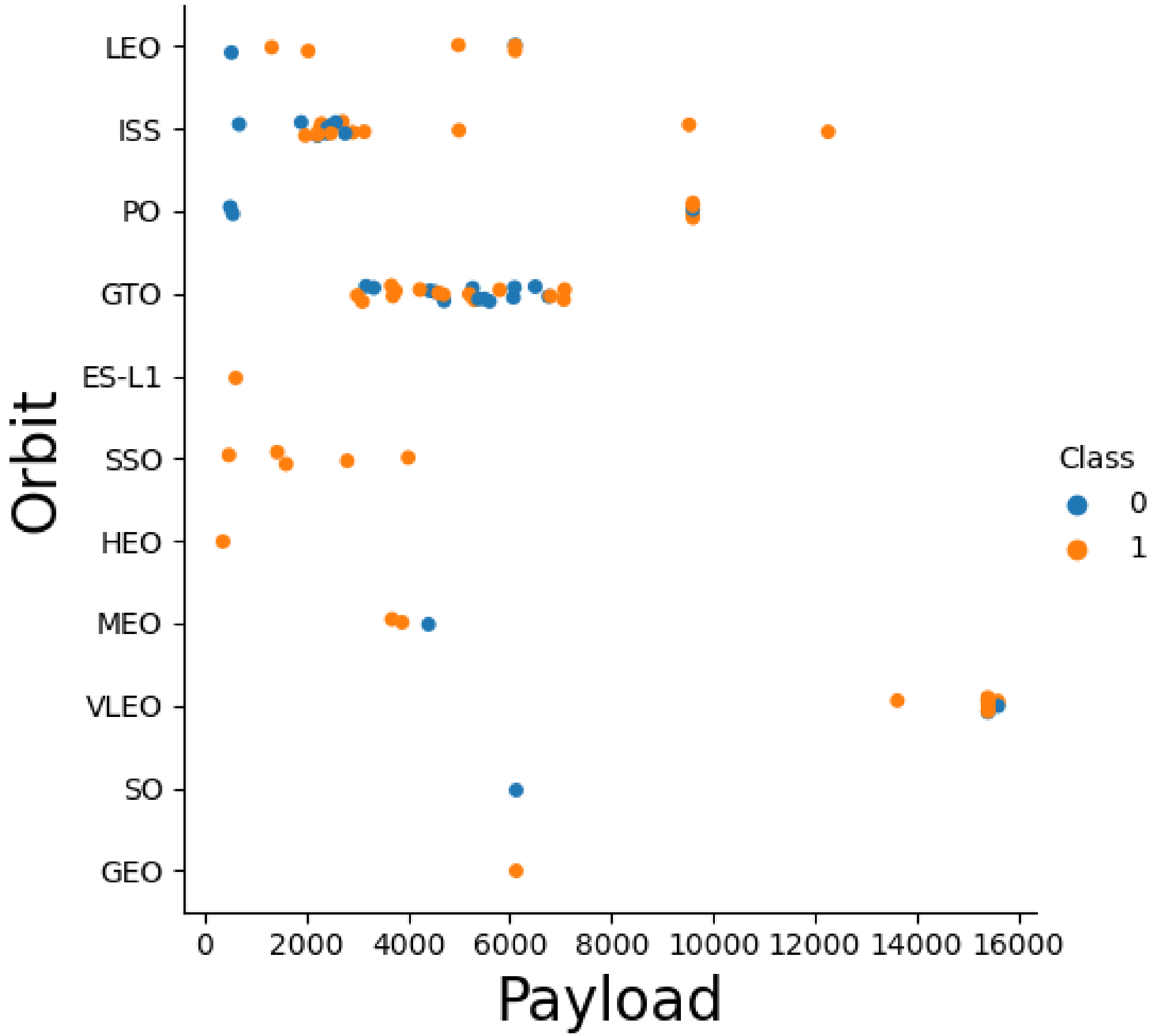
Flight Number vs. Orbit Type

- We observe that in the LEO orbit, success is related to the number of flights whereas in the GEO orbit, there is no relationship between flight number and the orbit.



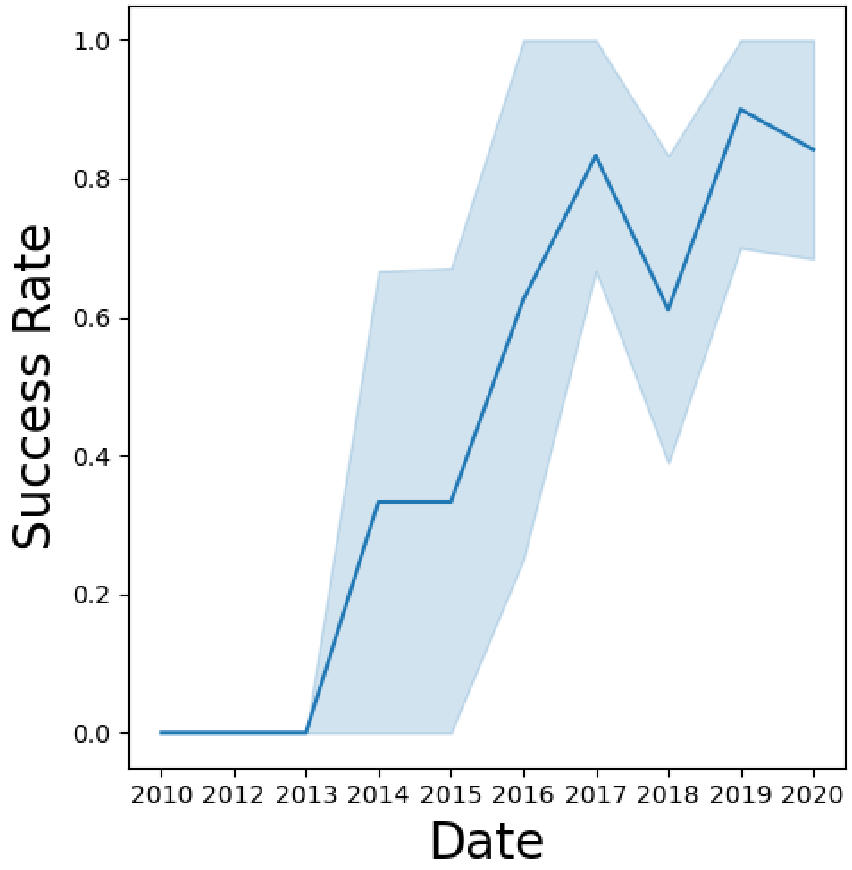
Payload vs. Orbit Type

- 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

Task 1

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

```
In [7]: %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

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

Launch Site Names Begin with 'CCA'

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

Task 2

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

In [8]:
```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```

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

Out[8]:

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)

In [9]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)'
```

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

Out[9]:

SUM(PAYLOAD_MASS__KG_)
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

Task 4

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

In [10]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION='F9 v1.1'
```

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

Out[10]:

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 1st August 2018

Task 5

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

Hint:Use min function

In [11]:
```
%sql SELECT min(DATE) FROM SPACEXTBL WHERE Landing_Outcome='Success (ground pad)'
```

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

Out[11]:

min(DATE)
01/08/2018

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

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

In [12]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ between 4000 and 6000 AND Landing_Outcome='Success (drone
```

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

Out[12]:

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

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

Task 7

List the total number of successful and failure mission outcomes

In [13]:
```
%sql SELECT COUNT(*) FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Success%' OR MISSION_OUTCOME LIKE '%Failure%'
```

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

Out[13]:

COUNT(*)
101

Boosters Carried Maximum Payload

Task 8

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

In [14]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

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

Out[14]:

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

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

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

- 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.

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]:

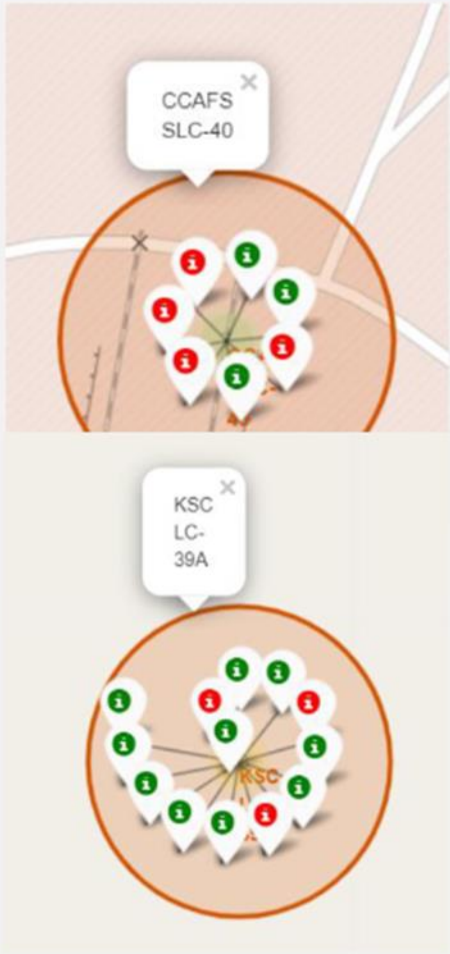
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

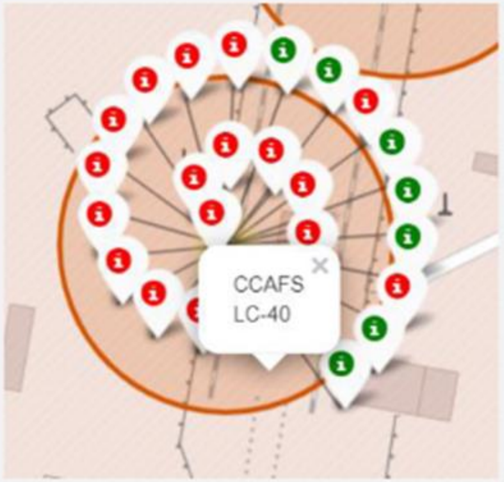
Section 3

Launch Sites Proximities Analysis

All launch sites

Markers showing launch sites with color labels





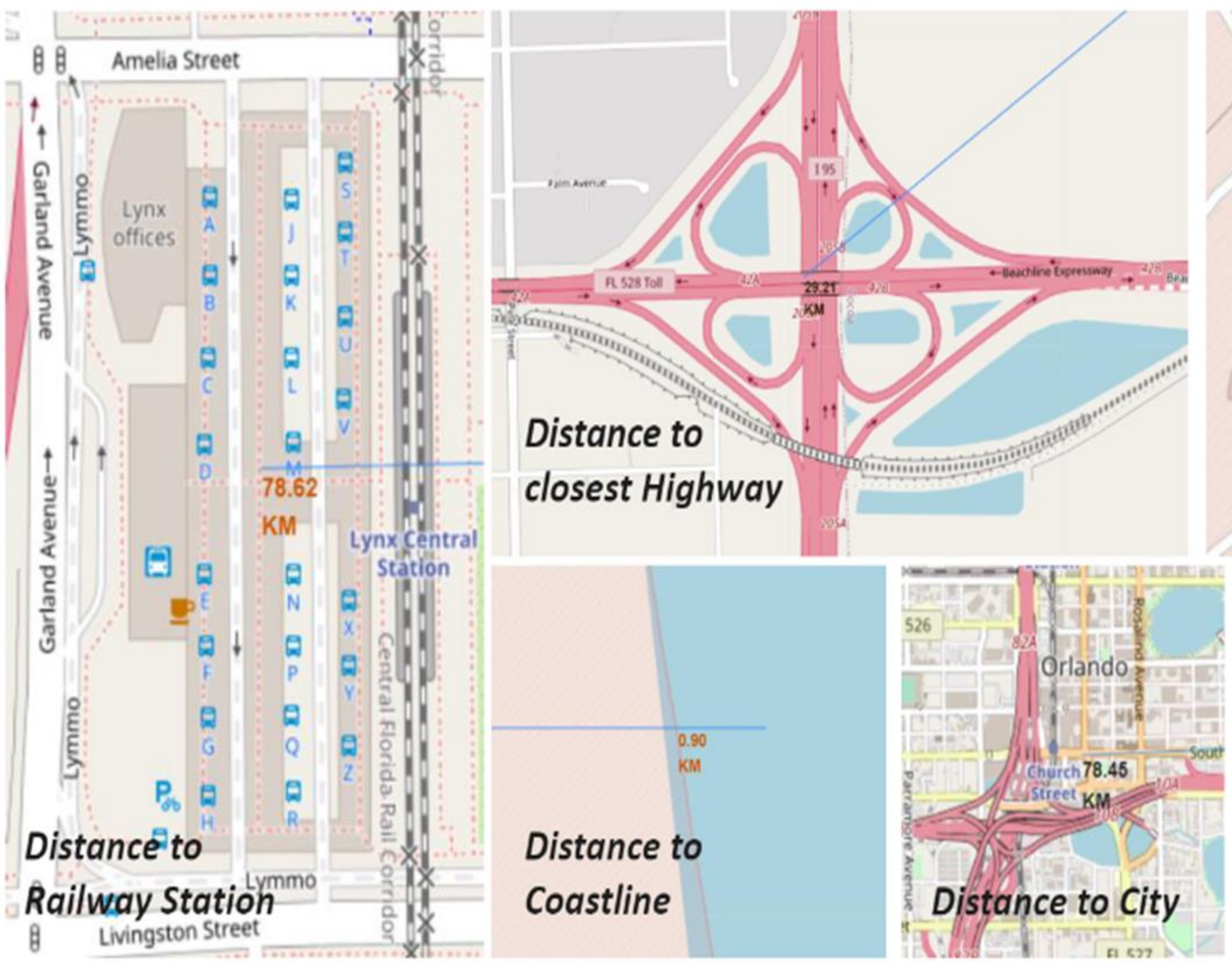
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

Launch Site distance to landmarks

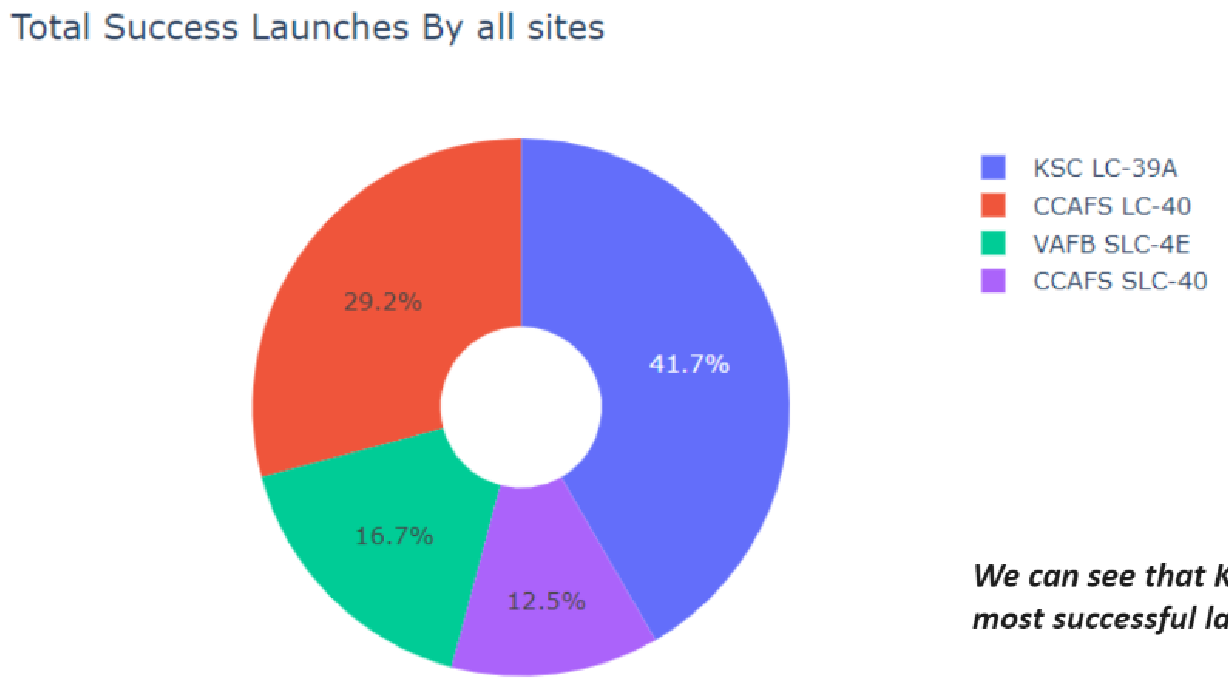


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

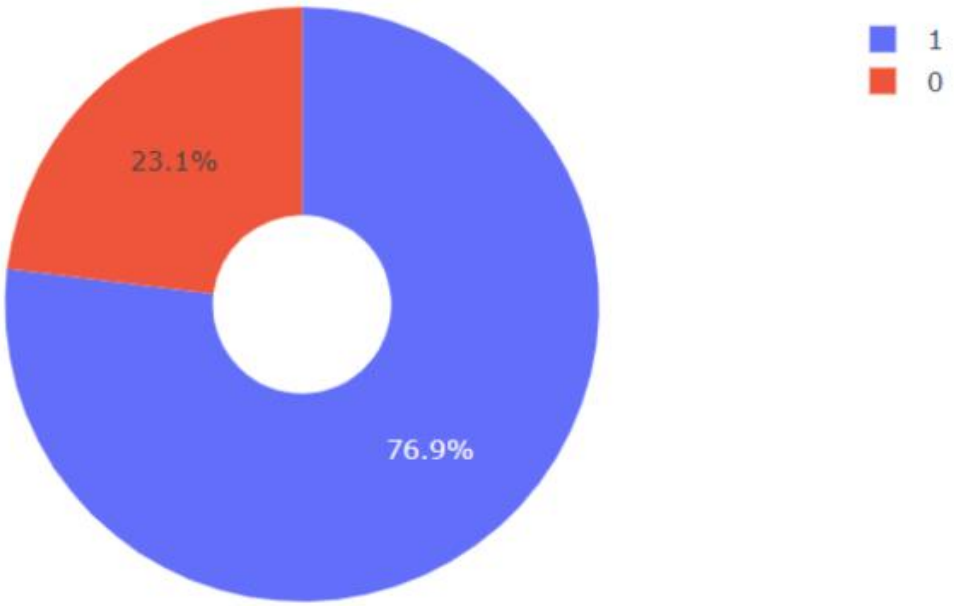
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



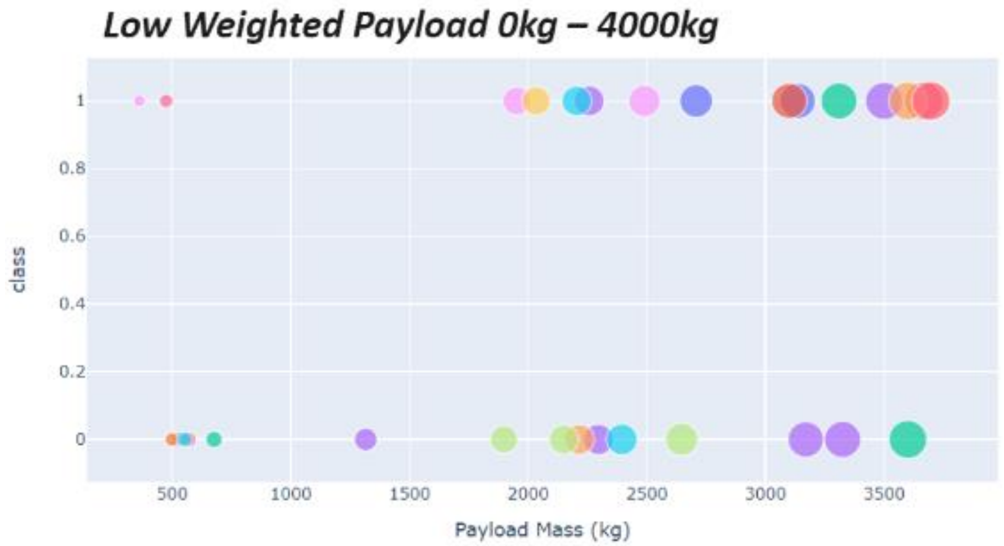
We can see that KSC LC-39A had the most successful launches from all the sites

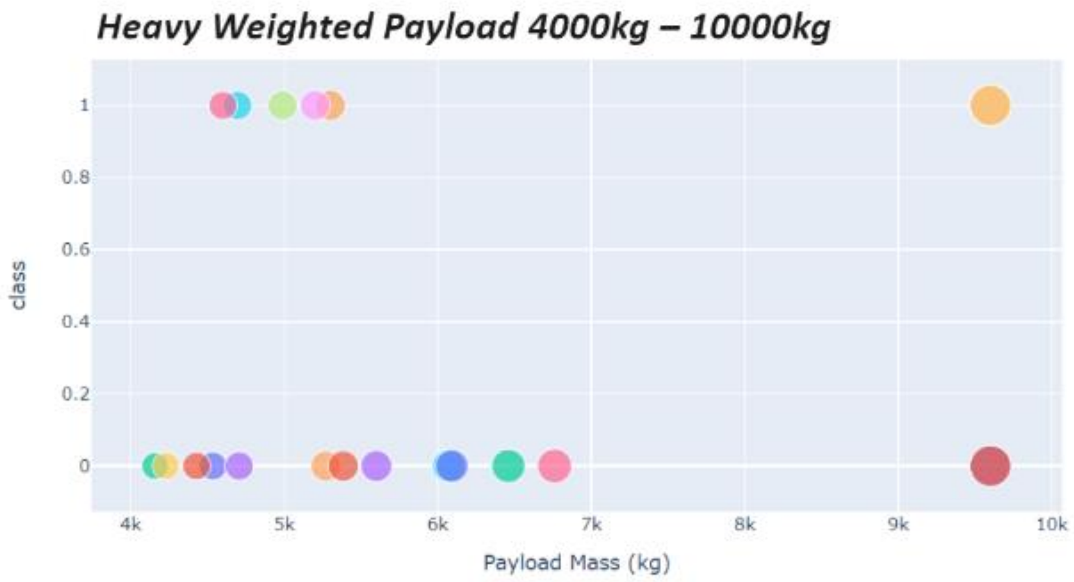
Pie chart showing the Launch site with the highest launch success ratio



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 5

Predictive Analysis (Classification)

Classification Accuracy

Find the method performs best:

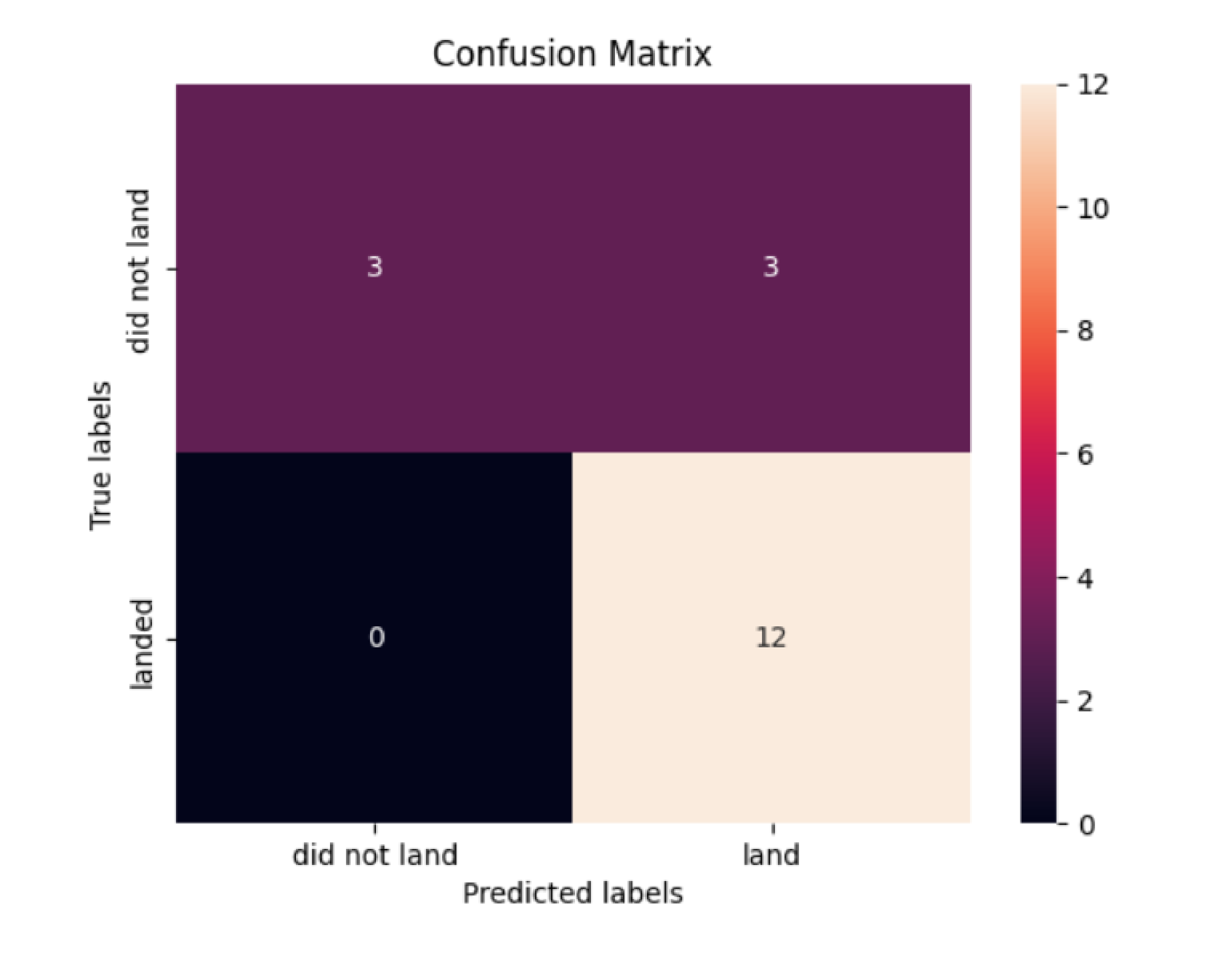
In [38]:

```
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 0.8857142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 10,
'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.

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!