Winning Space Race with Data Science

Delbar Bi Gonah
15/04/2024

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data collection API
  - Data collection with web scraping
  - Data wrangling
  - EDA with SQL
  - EDA with visualisation
  - Interactive visial analytics and dashboard
  - Predictive analysis
- **Summary of all results**
  - EDA result
  - Interactive analytics
  - Predictive analysis

Introduction

- Project background and context
  - SpaceX 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 SpaceX 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 SpaceX for a rocket launch.
- Problems you want to find answers
  - The aim of this project is to predict if the Falcon 9 first stage will land successfully.

Section 1

Methodology

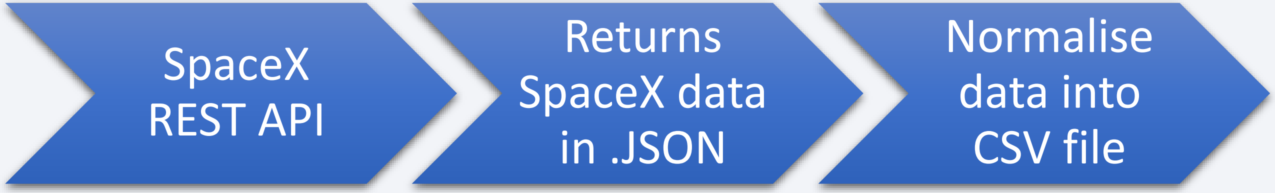
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - One-hot encoding data fields for Machine Learning / Cleaning irrelevant columns
- 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

- Data sets were collected by following methods:
  - ✓ SpaceX launch data were used to gather data from the SpaceX rest API.
  - ✓ Web scrapping were used to extract Falcon 9 launch data from Wikipedia website using BeautifulSoup
- Data collection process using flowcharts:
  - SpaceX API



  - Web Scrapping

Data Collection – SpaceX API

- Steps of data collection using SpaceX REST API

- GitHub URL of the completed SpaceX API calls notebook

1. Getting response from SpaceX API

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

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

2. Converting response to JSON file and turning it into a Pandas dataframe

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

3. Filter the dataframe to only include Falcon 9 launches

```
In [24]: # Hint data['BoosterVersion']!='Falcon 1'
         data_falcon9 = data[data.BoosterVersion == 'Falcon 9']
         data_falcon9
```

Out[24]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False
[illegible]	[illegible]	2013-	[illegible]	[illegible]	[illegible]	CCSFS SLC	None	[illegible]	[illegible]	[illegible]	[illegible]

4. Dealing with missing value

```
[30]: # Calculate the mean value of PayloadMass column
      PayloadMass_mean = data_falcon9.PayloadMass.mean()
      # Replace the np.nan values with its mean value
      data_falcon9['PayloadMass'].replace(np.nan, PayloadMass_mean, inplace=True)
```

Data Collection - Scraping

- Extract a Falcon 9 launch records HTML table from Wikipedia
- Parse the table and convert it into a Pandas data frame
- [GitHub URL]of the completed web scraping notebook

1. Request the Falcon9 launch Wiki page from its URL

```
[6]: # use requests.get() method with the provided static_url
     # assign the response to a object
     data = requests.get(static_url).text
```

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

```
[7]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
     soup = BeautifulSoup(data, 'html5lib')
```

2. Extract all column/variable names from the HTML table header

```
[9]: # Use the find_all function in the BeautifulSoup object, with element type `table`
     # Assign the result to a list called `html_tables`
     html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

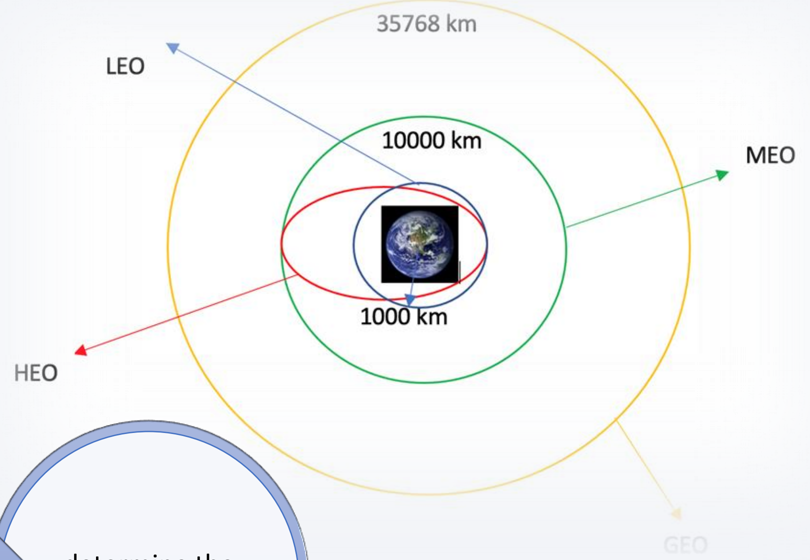
```
[10]: # Let's print the third table and check its content
      first_launch_table = html_tables[2]
      print(first_launch_table)
```

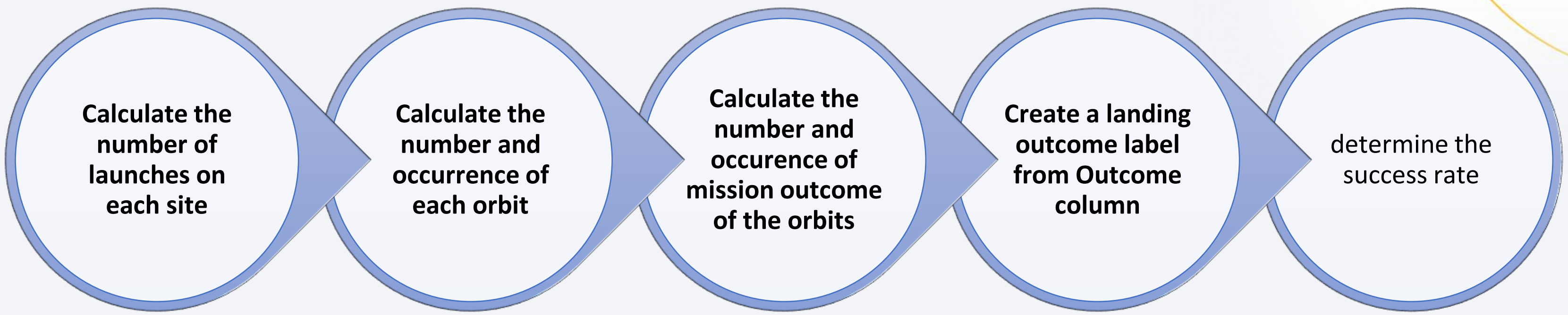
3. Create a data frame by parsing the launch HTML

We created an empty dictionary with keys from the extracted column names in the previous task. Then we converted this dictionary into a Pandas dataframe.

Data Wrangling

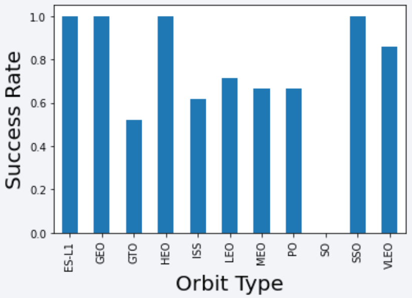
- Perform Exploratory Data Analysis EDA on dataset
- GitHub URL of completed data wrangling notebook





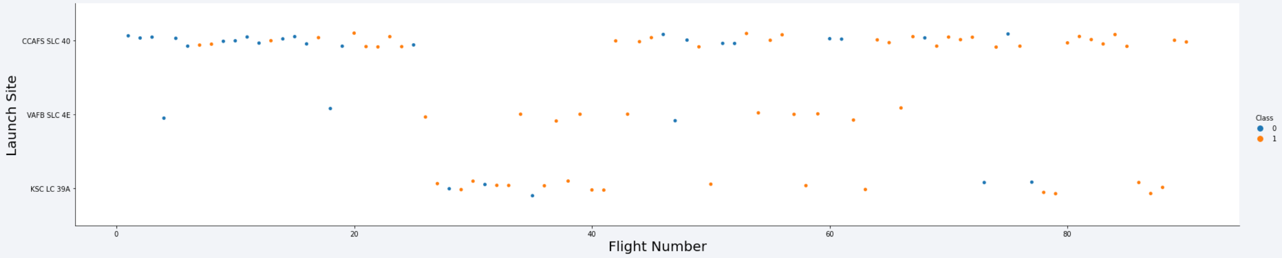
EDA with Data Visualization

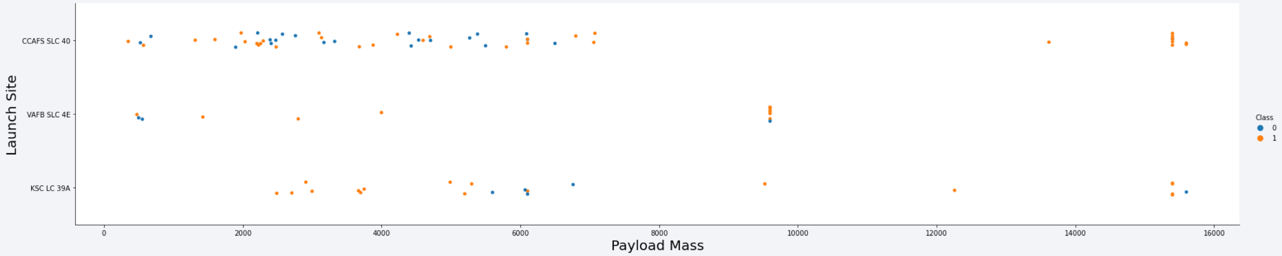
- Flight Number VS Launch Site
- Payload VS Launch Site
- Success rate of each orbit type

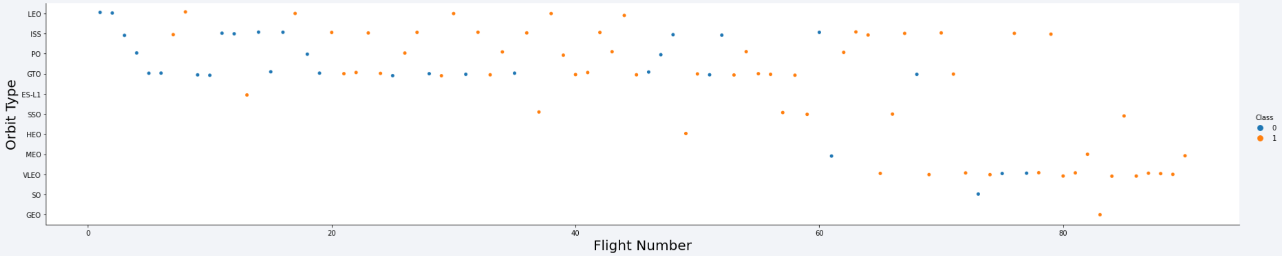


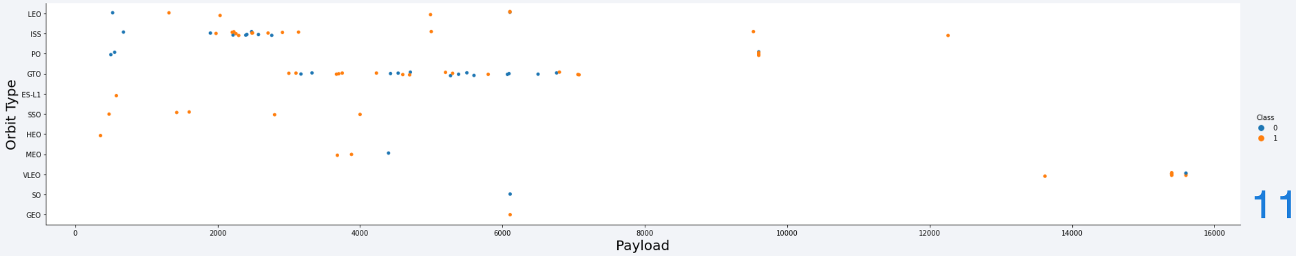
- FlightNumber and Orbit type
- Payload VS Orbit type

➢ [GitHub URL]()









EDA with SQL

- Performed SQL queries:
  - Display the names of the unique launch sites in the space mission
  - Display 5 records where launch sites begin with the string 'CCA'
  - 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 date when the first succesful landing outcome in ground pad was acheived.
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - List the total number of successful and failure mission outcomes
  - List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
  - List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
  - Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.
- GitHub URL of completed EDA with SQL notebook

Build an Interactive Map with Folium

Steps:

- **Mark all launch sites on a map:** we added each site's location on a map using site's latitude and longitude coordinates. We add a circle for each launch site in data frame launch_sites. We used map objects such as markers, circles, lines to mark the success or failure of launches.
- **Mark the success/failed launches for each site on the map**
- **Calculate the distances between a launch site to its proximities:** first we added a MousePosition on the map to get coordinate for a mouse over a point on the map.

- GitHub URL of completed interactive map with Folium map

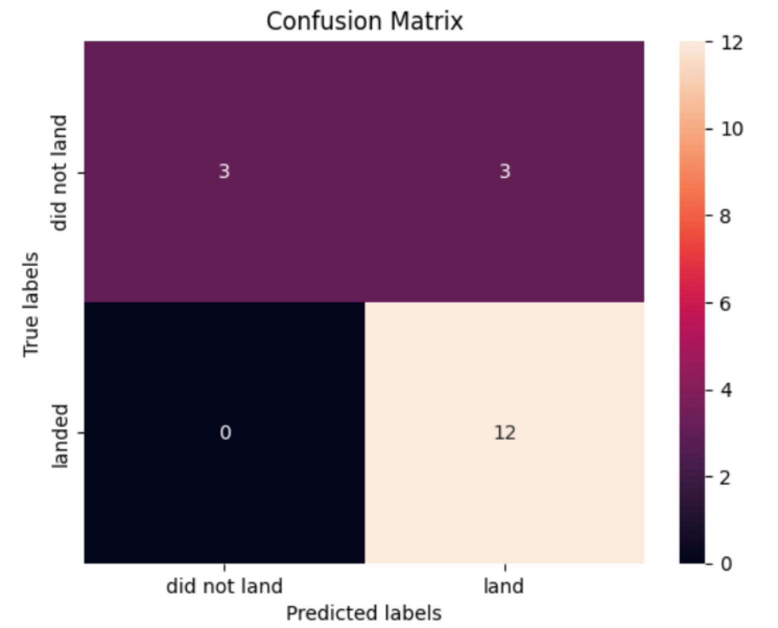
Build a Dashboard with Plotly Dash

- An interactive dashboard with Plotly dash was created
- Then a pie charts was created to show the total launches by a certain sites
- Finally, a scatter graph was created to show the relationship with Outcome and Payload Mass for the different booster version.

Predictive Analysis (Classification)

Summary:

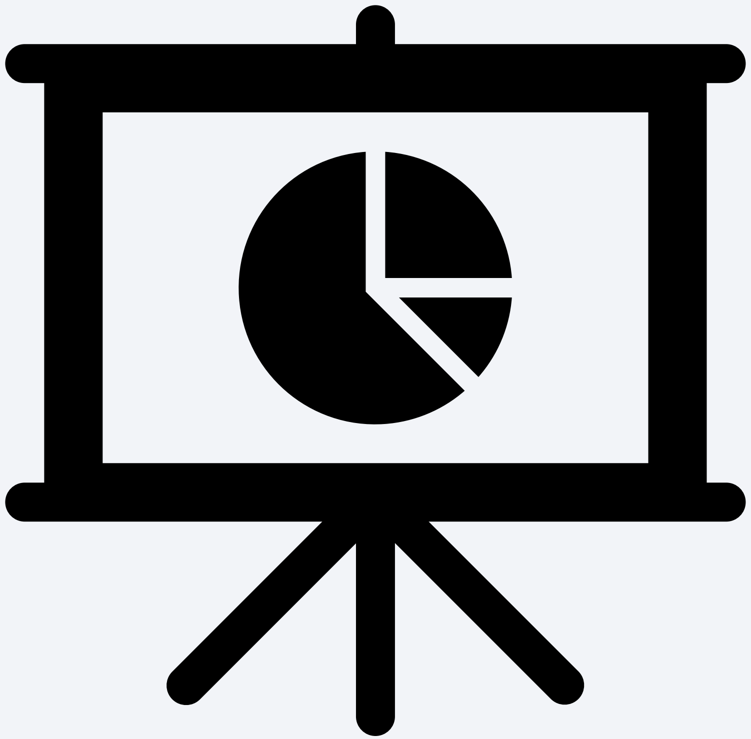
- Building Model
- Evaluating Model
- Improving Model
- Finding the best performing Classification Model



- GitHub URL of completed predictive analysis lab

Results

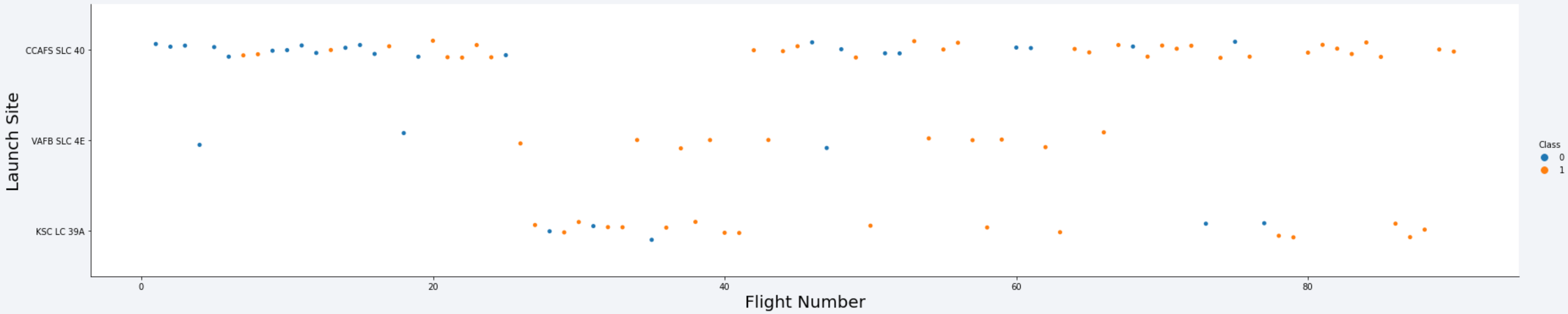
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

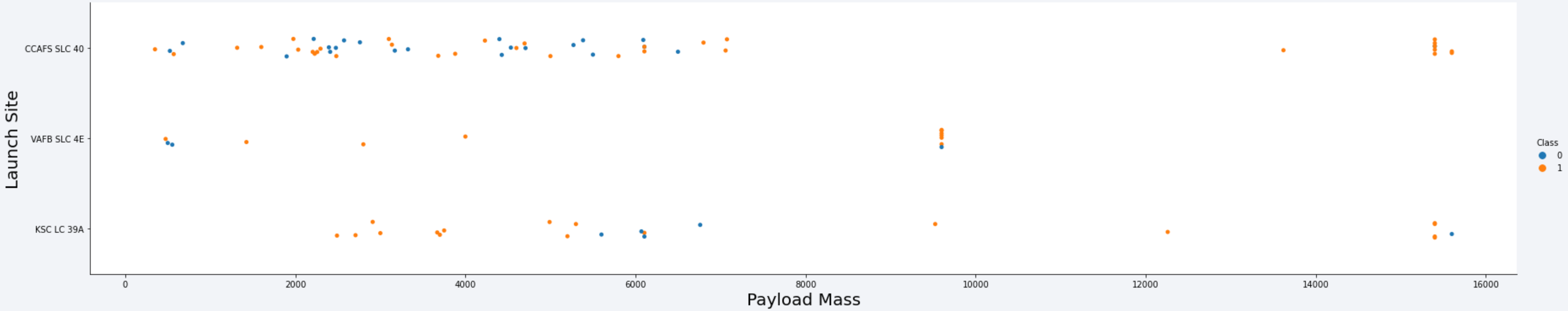
Insights drawn from EDA

Flight Number vs. Launch Site



- The graph shows that higher flight values at a launch site are associated with higher success rates.

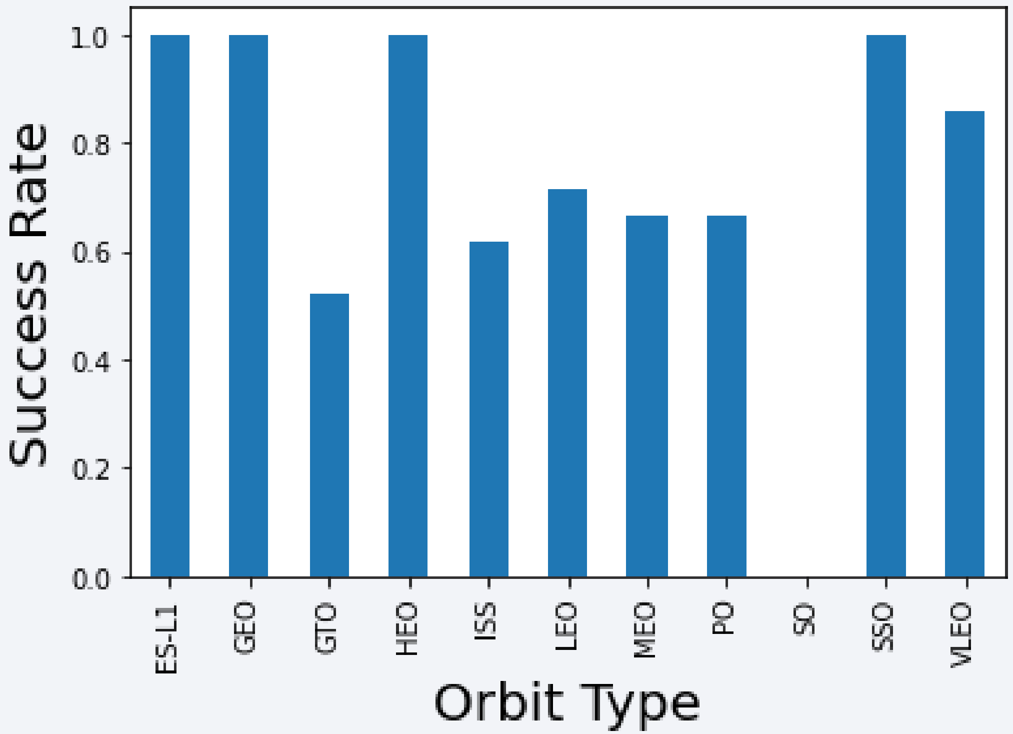
Payload vs. Launch Site



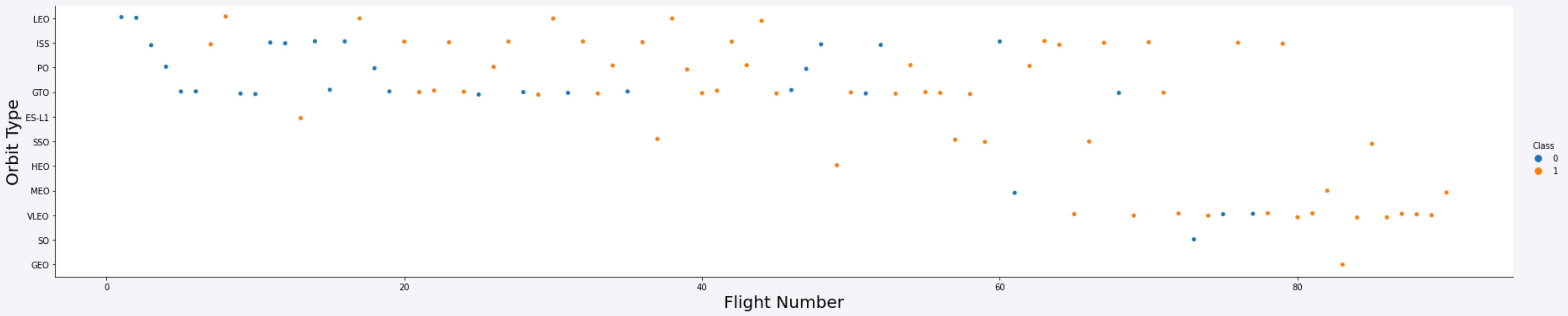
- The graph shows that the higher the Payload Mass for Launch Site CCAFS SLC 40, the higher the success rate for the rocket.

Success Rate vs. Orbit Type

- From the bar chart, we can see that the Orbit ES-L1, GEO, HEO, SSO, VLEO have the highest success rate.

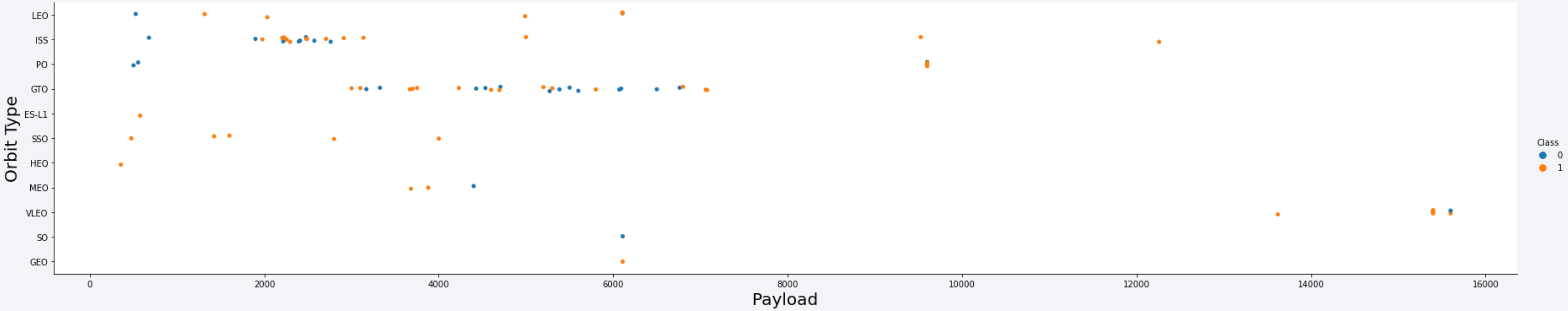


Flight Number vs. Orbit Type



- From the above scatter plot, It's noticeable that success in the LEO orbit seems to be associated with the number of flights, whereas in the GTO orbit, there doesn't appear to be any correlation between flight number and success.

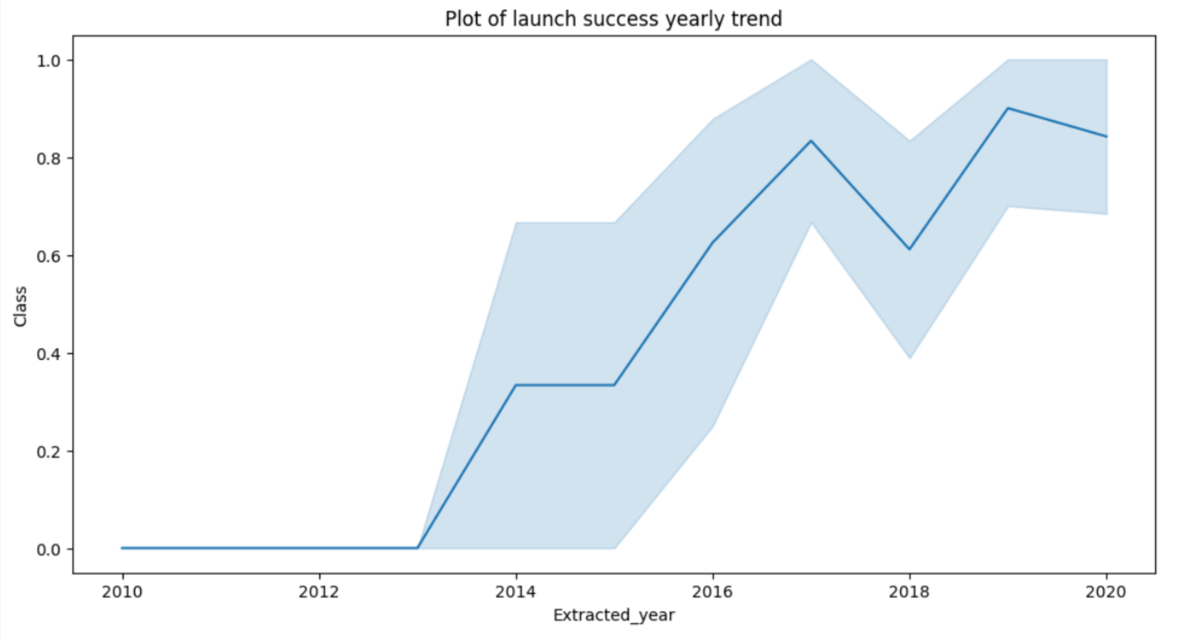
Payload vs. Orbit Type



- The above scatter plot shows that with heavier payloads, successful landings are more frequent for the PO, LEO, and ISS orbits.

Launch Success Yearly Trend

- Line chart is showing that the success rate is increasing from 2013



All Launch Site Names

- The key word **DISTINCT** is used to show only unique launch sites in the space mission.

Task 1

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

In [10]:
```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```

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

Out[10]:

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

Launch Site Names Begin with 'CCA'

- LIKE ‘CCA%’ LIMIT 5; where used to select 5 records where Launch Site starting with ‘CCA’

Task 2

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

In [11]:
```
%sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

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

Out[11]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- The result of query below shows that the total payload mass carried by boosters launch by NASA is 45596 KG.

```
In [14]: %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';

 * sqlite:///my_data1.db
Done.

Out[14]:
```

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- AVG() function were used to calculate the average payload mass carried by booster version F9 V1.1
- Average = 340.4

```
In [15]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.0%';

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

Out[15]:

AVG(PAYLOAD_MASS__KG_)
340.4

First Successful Ground Landing Date

- MIN() function were used to find the dates of the first successful landing outcome on ground pad
- 2015-12-22

```
In [20]: %sql SELECT MIN(ate) FROM SPACEXTBL WHERE Landing__Outcome = 'Success (ground pad)';

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

Out[20]:

MIN(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Name of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

- query : %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE landing_outcome = 'Success (drone ship)' AND 4000 < PAYLOAD_MASS__KG_ < 6000;

Done.

Out[22]:

Booster_Version
F9 FT B1021.1
F9 FT B1022
F9 FT B1023.1
F9 FT B1026
F9 FT B1029.1
F9 FT B1021.2
F9 FT B1029.2
F9 FT B1036.1
F9 FT B1038.1
F9 B4 B1041.1
F9 FT B1031.2
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

Total Number of Successful and Failure Mission Outcomes

- COUNT() function were used to calculate the total number of successful and failure mission outcomes
- Below we can see the total number of success is 100 and there is only 1 failure mission outcome.

In [23]:
```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

Out[23]:

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

Boosters Carried Maximum Payload

- Below we can see a list of the booster names which have carried the maximum payload mass
- To achieve the result subquery were used. We used **WHERE** clause and the **MAX()** function.

In [24]:
```
%sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_)FROM SPACEXTBL);
```

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

Out[24]:

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

- Below we can see a list of the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015

- **Query:** %sql SELECT substr(Date, 6,2), LANDING_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Failure (drone ship)' AND substr(Date,0,5)='2015';

```
 * sqlite:///my_data1.db
Done.
Out[42]:
```

substr(Date, 6,2)	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

- The picture shows the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

- To achieve the result statements such as GROUP BY and ORDER BY were used.

In [43]:
```
%%sql
SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING_OUTCOME
ORDER BY TOTAL_NUMBER DESC
```

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

Out[43]:

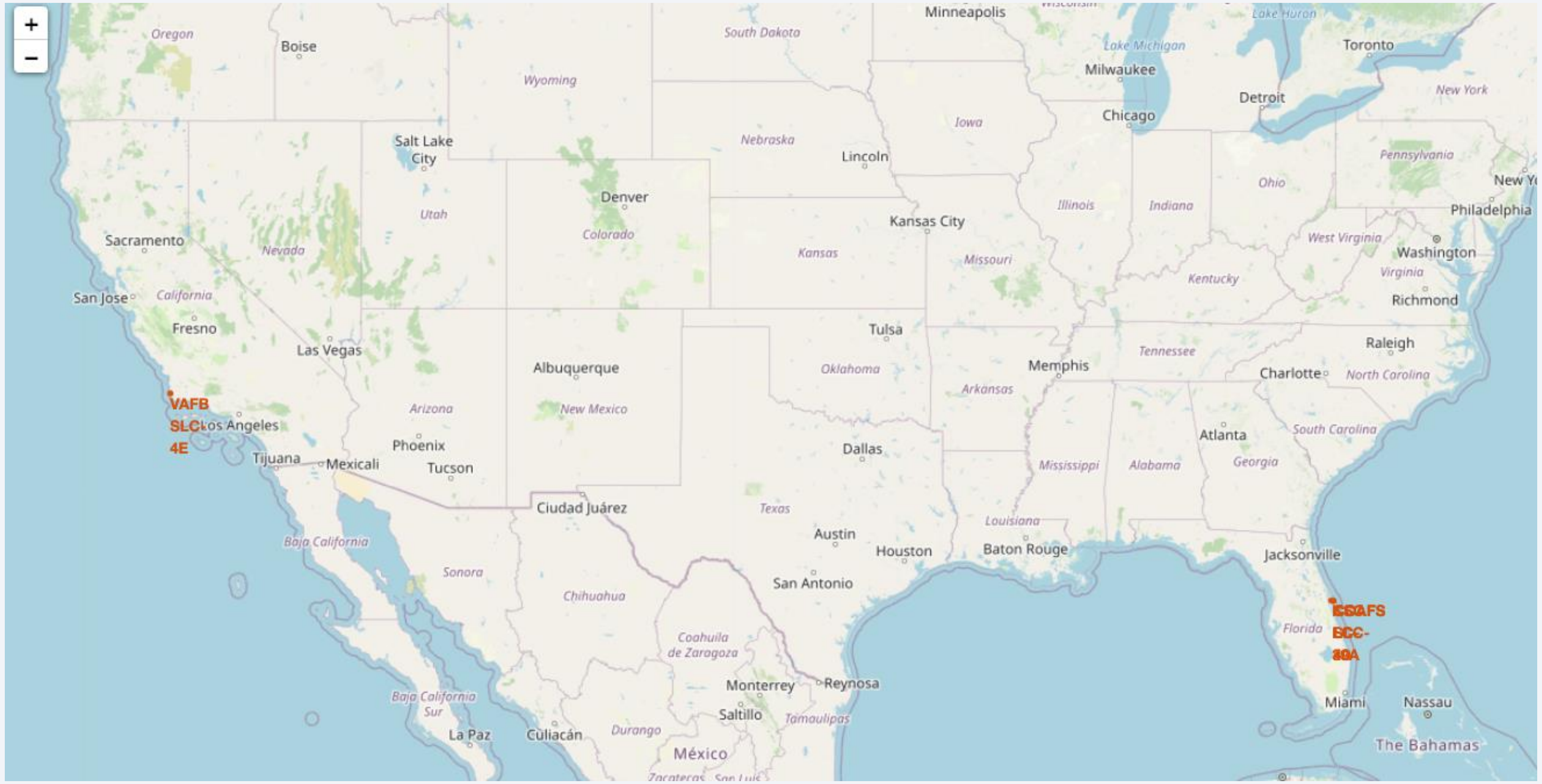
Landing_Outcome	TOTAL_NUMBER
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

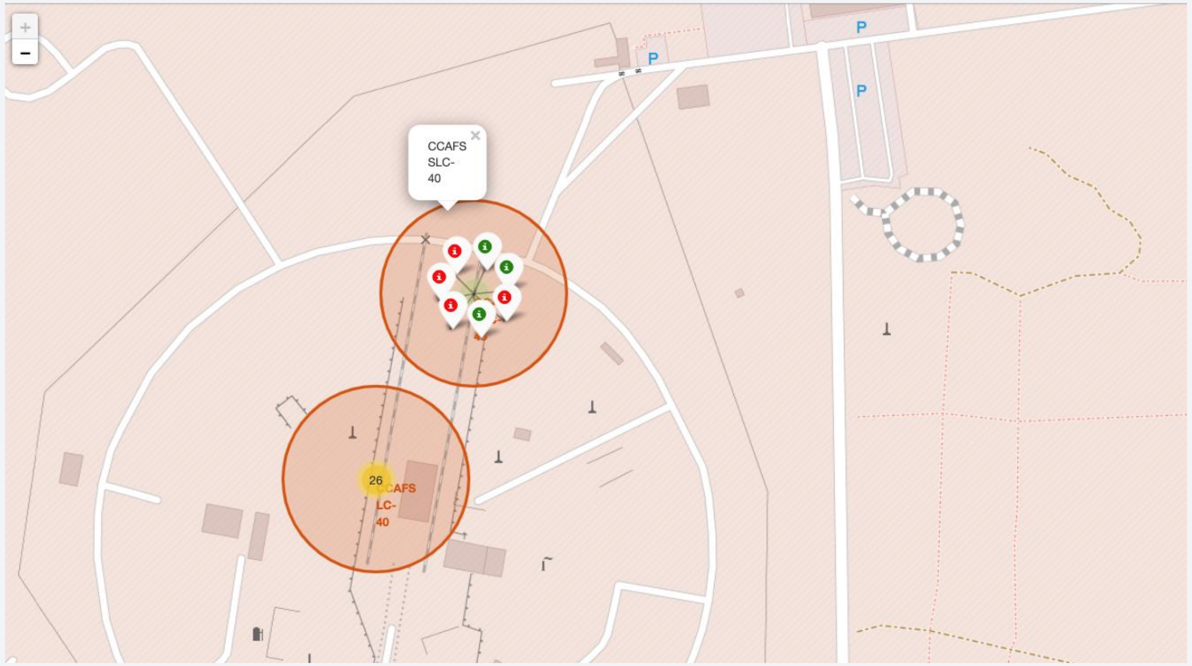
All launch sites global map markers

- Below, we can observe that the SpaceX launch sites are located on the coasts of Florida and California in the United States of America.

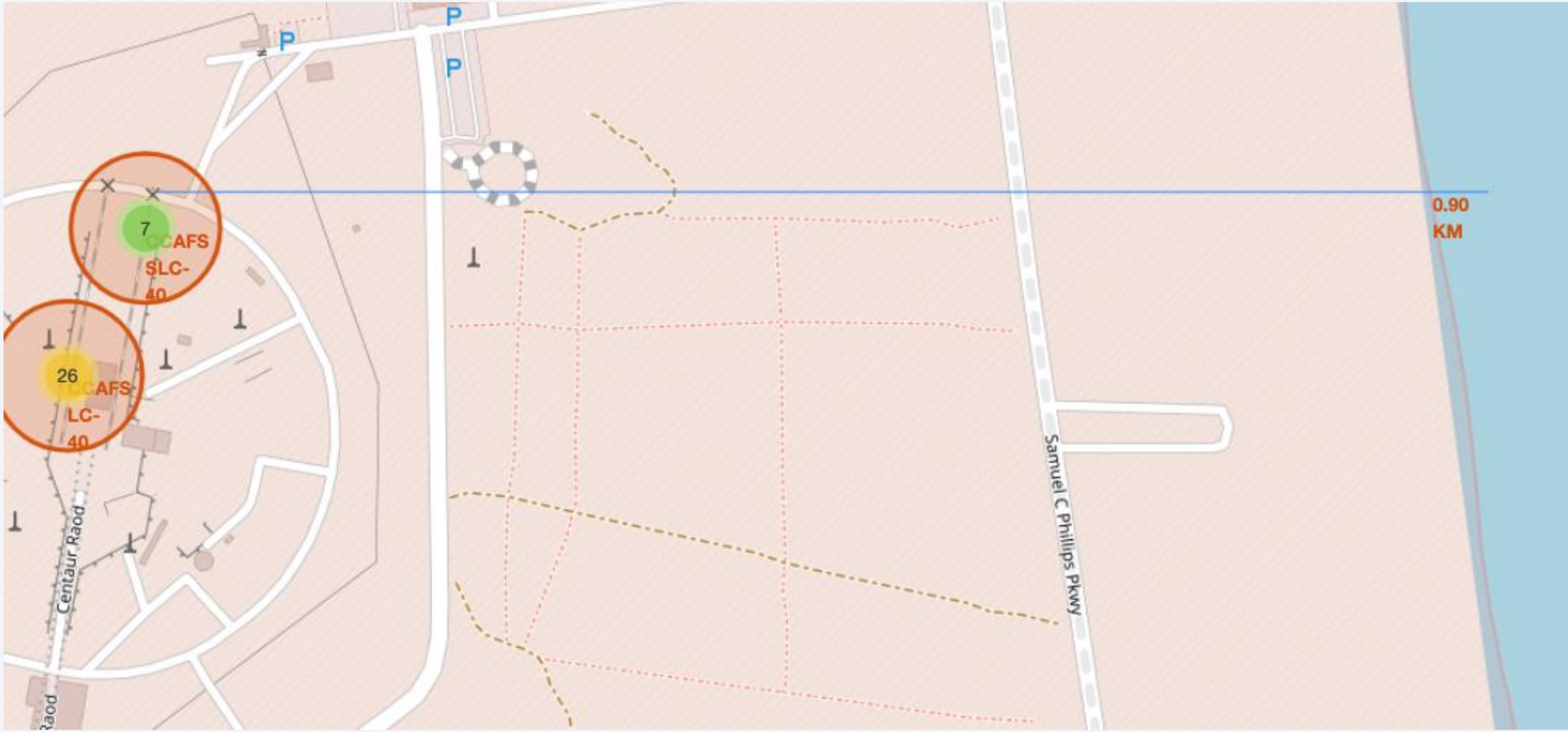


Launch sites with color labels

- Green markers shows successful launches
- Red markers shows failures



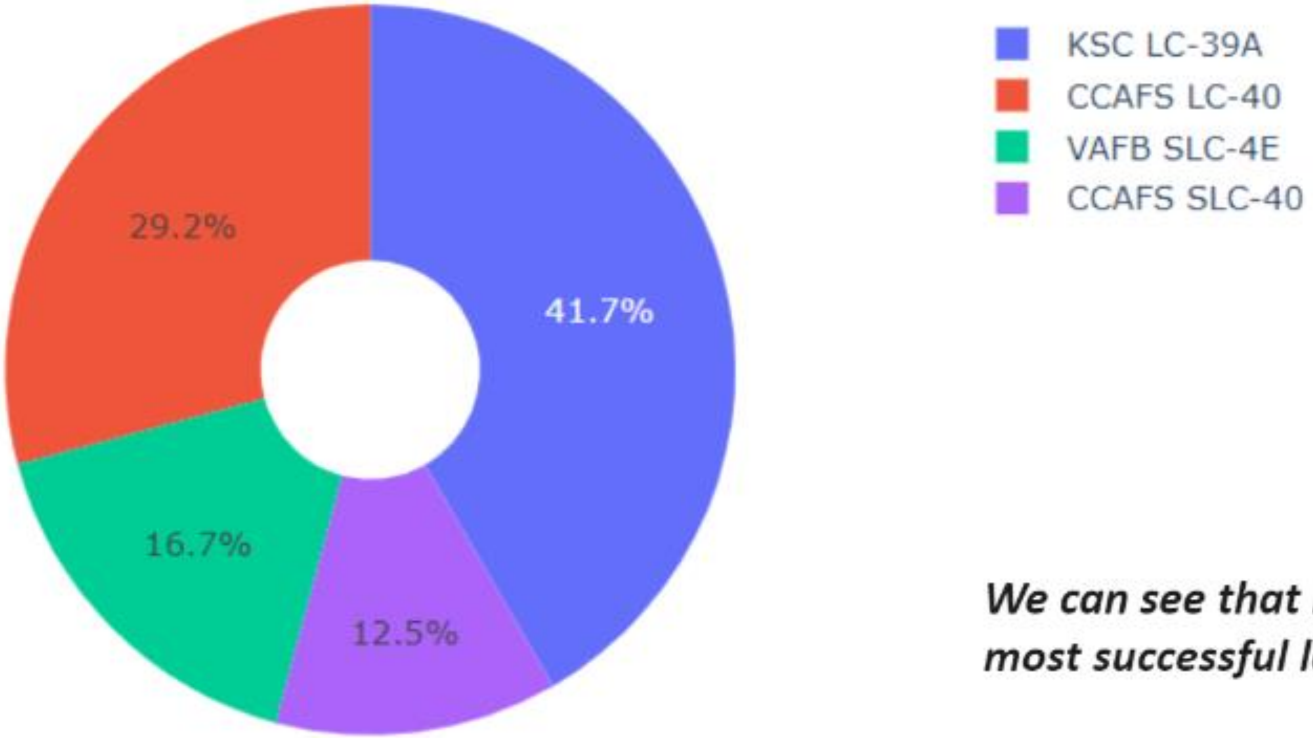
Distance between a launch site to its closest



Section 4

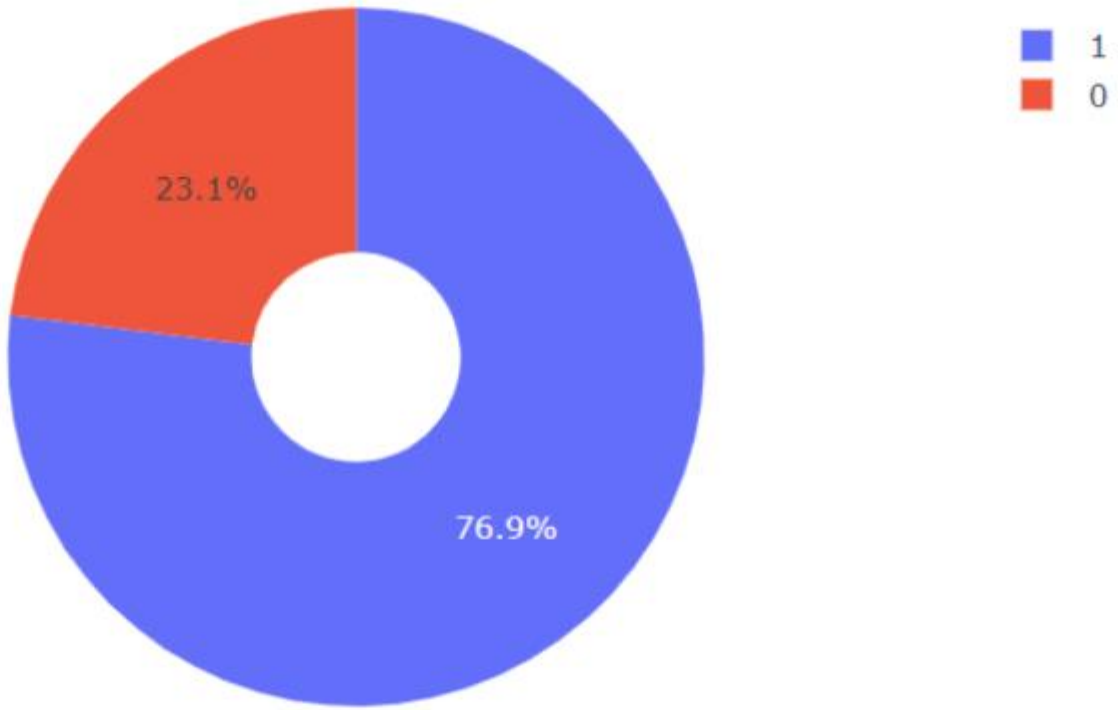
Build a Dashboard with Plotly Dash

Total success launches by all sites



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

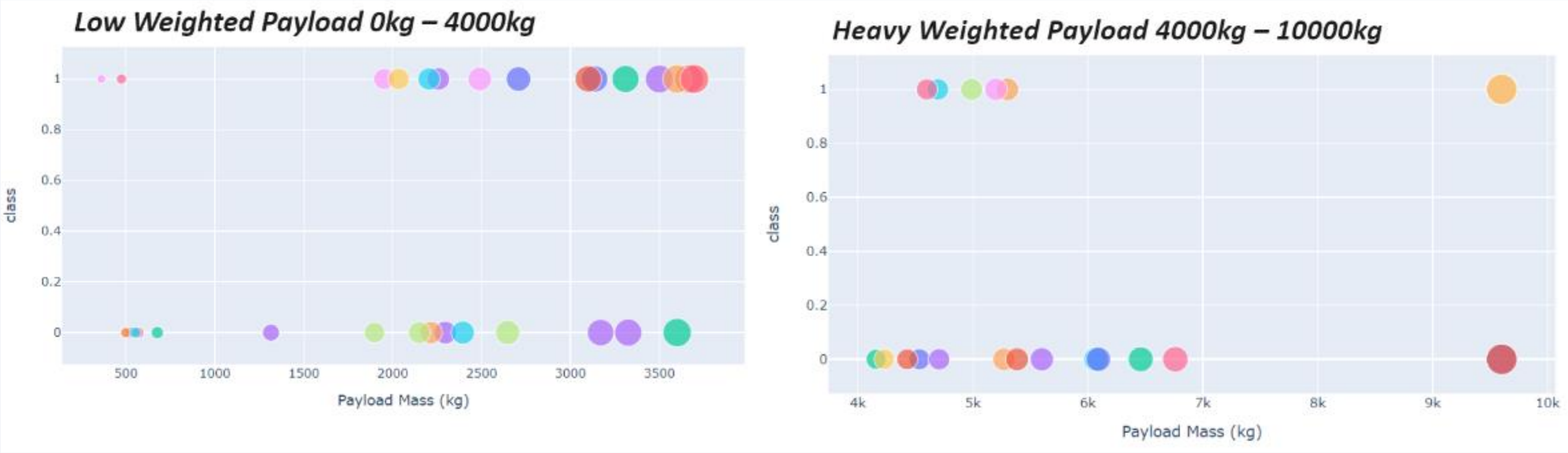
Success rate



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

Scatter plot of payload VS Launch outcome

- Below we can see the success rates for low weighted payloads is higher than the heavy weighted payloads



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Below we can see decision tree has the highest classification accuracy.

Find the method performs best:

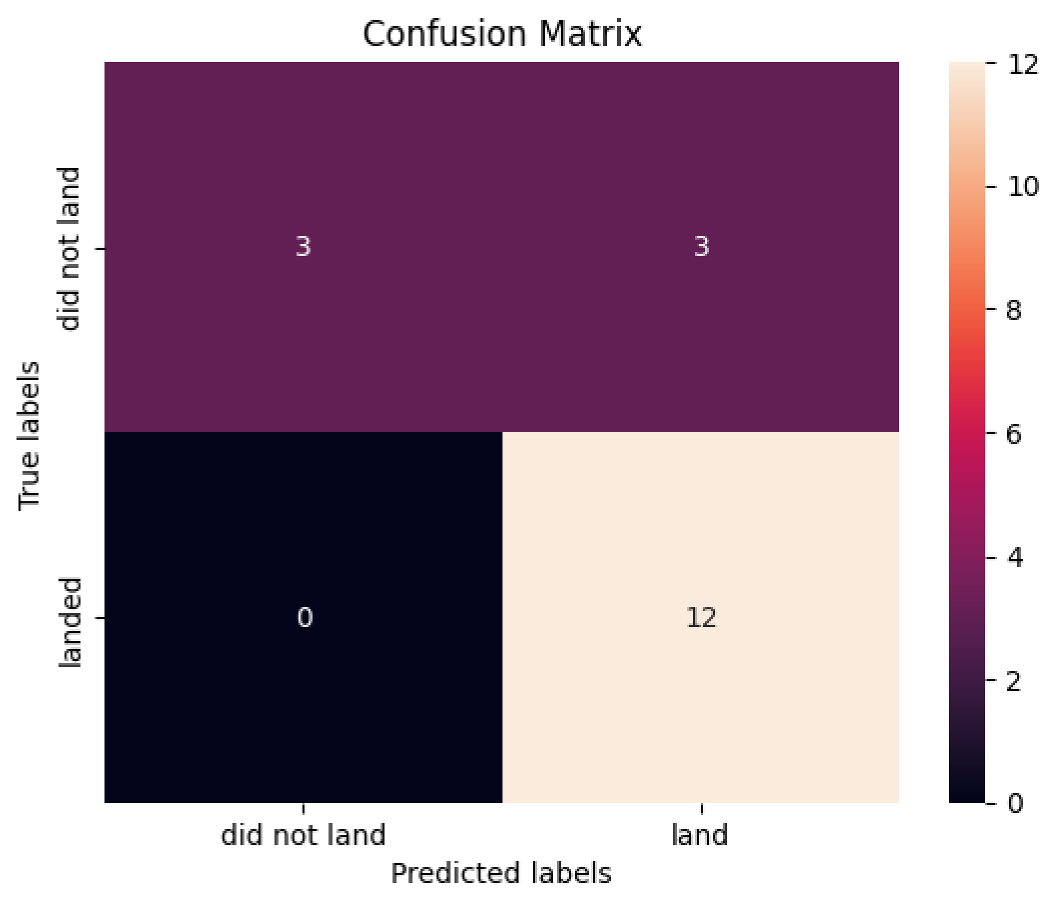
In [36]:

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8732142857142856
Best Params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5,
'splitter': 'random'}
```

Confusion Matrix

- Confusion matrix for the tree
- By examine the confusion matrix, we can clearly see that the main issue is in FP.



Conclusions

- The tree classifier algorithm is the best ML algorithm for this dataset.
- Low weighted payloads perform better than heavier payloads.
- The Orbit ES-L1, GEO, HEO, SSO, VLEO have the highest success rate.
- KSC LC-39A had the most successful launches.

Thank you!