IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Alexis Turi
November 3, 2022

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data Collection with 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 Results
  - Interactive Analytics Results
  - Predictive Analytics Results

Introduction

- **Project background and context**

According to SpaceX, the Falcon 9 launches cost 62 million dollars, compared to about 165 million with other providers — all of which because SpaceX has reusable boosters (first stage). If we can know for sure if the first stage will land, then it is possible to know the cost of a launch. In this project, we aim to predict landing success of the first stage through a machine learning pipeline.

- **Problems you want to find answers to**

- What are the factors of a successful first stage landing?
- How are the different features and data interacting, and how do they impact the landing?
- What can we do to make sure the first stage will land successfully, every time?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - We collected data in two days: SpaceX API and web scraping from Wikipedia
- Perform data wrangling
  - We converted all landing outcomes into a binary class
- 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

- We primarily used the SpaceX API to collect data
- Response content was turned into a pandas data frame using .json_normalize()
- Data was cleaned (checking for missing values, filling them when necessary)
- We collected additional data through web scraping with BeautifulSoup
- To that end, web data was collected as an HTML table first, then parsed and converted into a pandas dataframe

Data Collection – SpaceX API

- Parsed launch data using GET request, filtered the dataframe to select Falcon 9 launches, and dealt with missing values

- https://github.com/AlexisTuri/ibm-data-science-capstone-spacex/blob/main/Data_Collection_API.ipynb

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

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

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

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

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data_falcon9 = launch_data.loc[launch_data['BoosterVersion']!="Falcon 1"]
```

Now that we have removed some values we should reset the FlgihtNumber column

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
# Calculate the mean value of PayloadMass column
mean = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(mean)

data_falcon9.isnull().sum()
```

Data Collection - Scraping

- Web data was collected as an HTML table with BeautifulSoup, then parsed and converted into a pandas dataframe

- https://github.com/AlexisTuri/ibm-data-science-capstone-spacex/blob/main/Data_Collection_Web_Scraping.ipynb

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.

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

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

```
# 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

```
# Use soup.title attribute
soup.title
```

```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

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

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
# 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.

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

Data Wrangling

- The main goal was to convert all landing outcomes into a binary class to be able to build ML models later on

- https://github.com/AlexisTuri/ibm-data-science-capstone-spacex/blob/main/Data_Wrangling.ipynb

```python
# Apply value_counts() on column LaunchSite
df["LaunchSite"].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

```python
# landing_outcomes = values on Outcome column
landing_outcomes = df["Outcome"].value_counts()
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

```python
# Apply value_counts on Orbit column
df["Orbit"].value_counts()
```

```
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
```

```python
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for key,value in df["Outcome"].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

EDA with Data Visualization

- Data explored:
  - Flight number vs Launch Site
  - Payload vs Launch Site
  - Success Rate by Orbit Type
  - Flight Number vs Orbit Type
  - Payload vs Orbit Type
  - Launch Success Yearly Trend
- https://github.com/AlexisTuri/ibm-data-science-capstone-spacex/blob/main/EDA_Data_Viz.ipynb

EDA with SQL

- We performed a number of queries to find out:
  - The names of the unique launch sites
  - The total payload mass carried by NASA (CRS) boosters
  - The average payload mass carried by booster F9 v1.1
  - The date of the first ground pad successful landing
  - The total number of successful and failure mission outcomes
  - And more!
- https://github.com/AlexisTuri/ibm-data-science-capstone-spacex/blob/main/EDA_SQL.ipynb

Build an Interactive Map with Folium

- We marked all launch sites and added markers for all success/failed launches in each site
- We calculated the distance between a given launch site and noteworthy spots around it (coastline, highway, railway, nearest city, etc.)
- https://github.com/AlexisTuri/ibm-data-science-capstone-spacex/blob/main/Folium_Data_Viz.ipynb

Build a Dashboard with Plotly Dash

- We built a Plotly Dash interactive dashboard, using dropdowns, charts, sliders and callbacks
- https://github.com/AlexisTuri/ibm-data-science-capstone-spacex/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Data was loaded using numpy and pandas, and then split into training and test data sets
- Different ML models were tried, including logistic regression, SVM, decision tree and k-nearest neighbors
- We calculated accuracy to find the best performing algorithm
- https://github.com/AlexisTuri/ibm-data-science-capstone-spacex/blob/main/Machine_Learning_Prediction.ipynb

Results

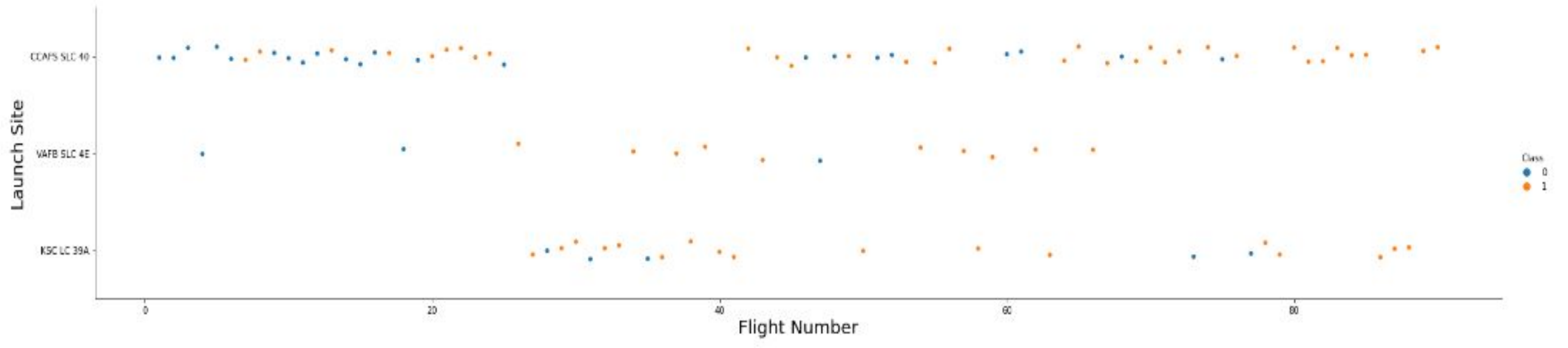
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

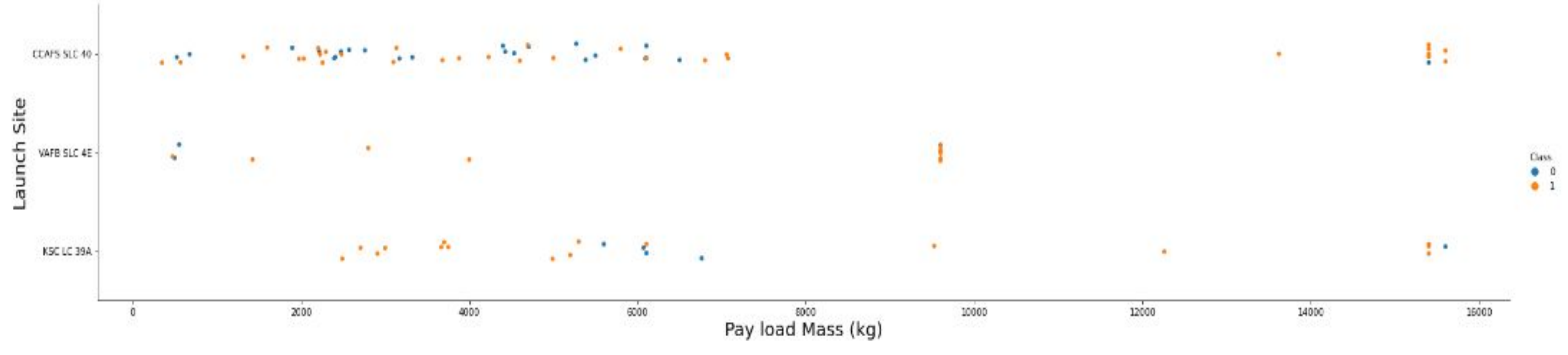
Flight Number vs. Launch Site

- For all launch sites, the success rate improves as the flight number increases



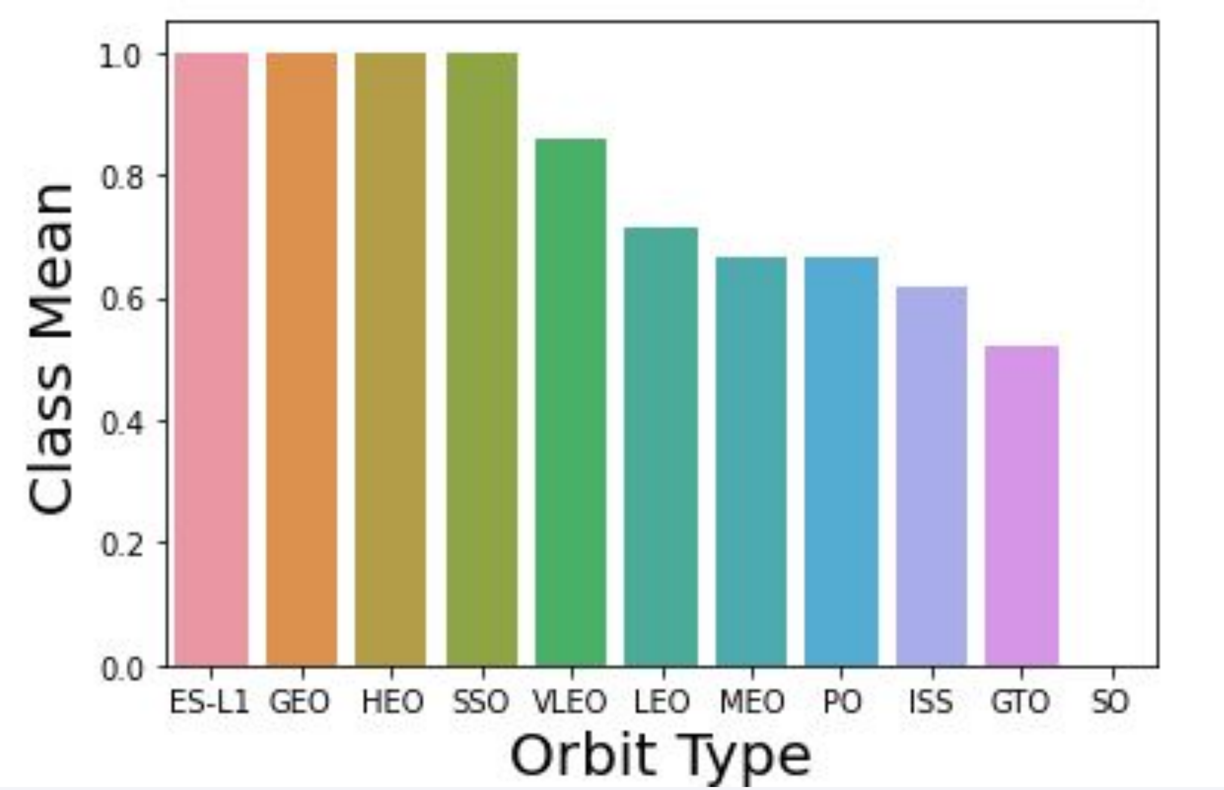
Payload vs. Launch Site

- We can see the success rate is higher with heavier payloads - we also notice that for the VAFB-SLC launch site, payloads don’t go over 10 000 kg



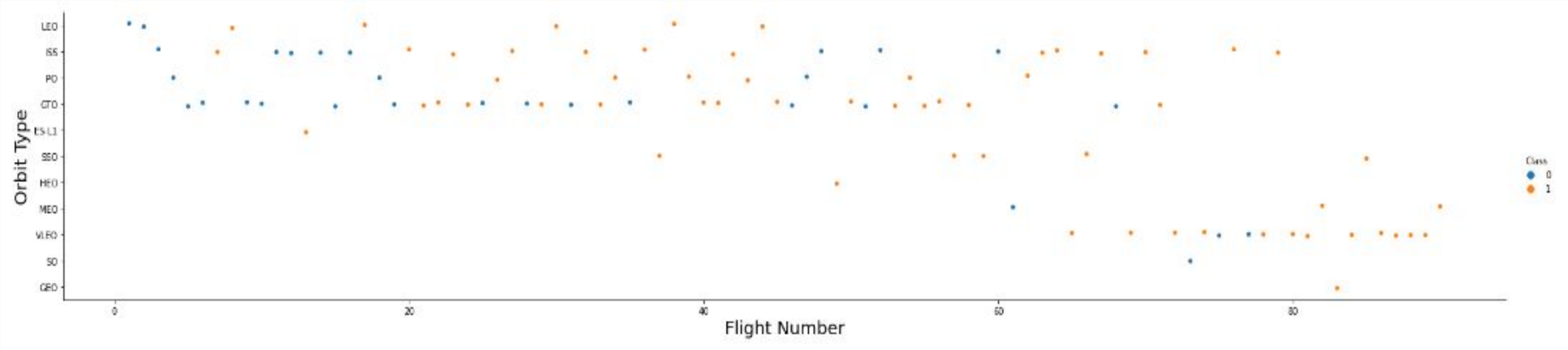
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO and SSO have the highest success rates



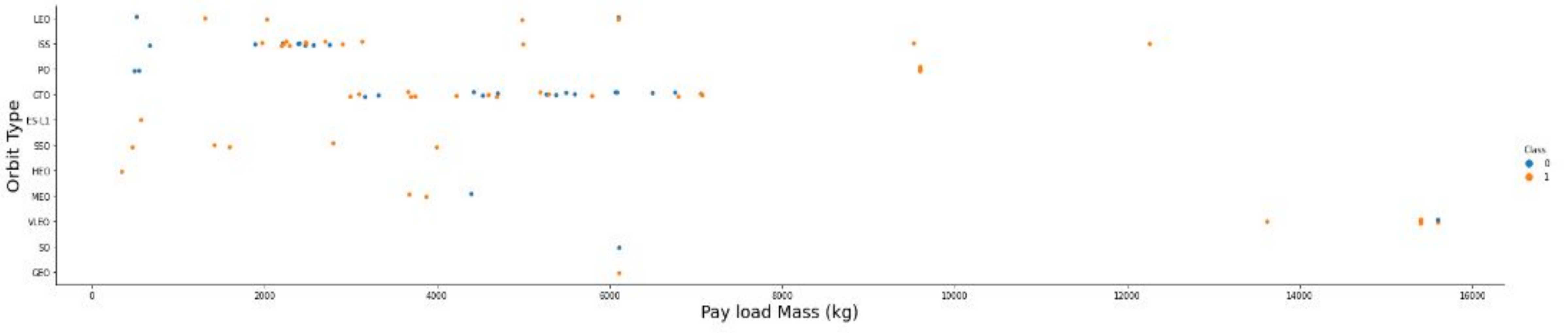
Flight Number vs. Orbit Type

- In the LEO orbit, success appears to be related to the number of flights, but in the GTO orbit, there seems to be no correlation



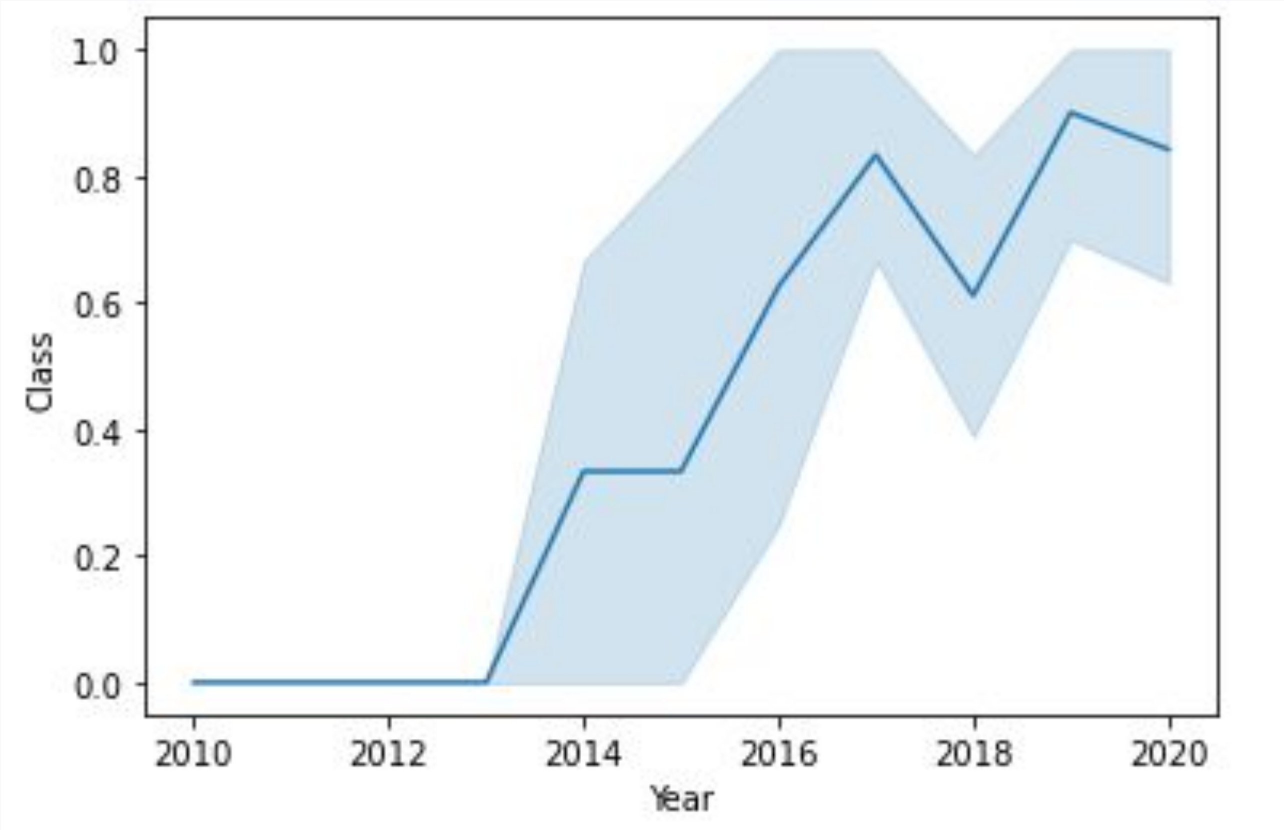
Payload vs. Orbit Type

- Heavy payloads seem to favor the landings for Polar, LEO and ISS orbits, but for GTO it is hard to tell whether there is any correlation



Launch Success Yearly Trend

- We can clearly see a positive trend from 2013 through 2020



All Launch Site Names

- Unique launch sites names were selected using DISTINCT

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

In [6]:
```
%sql SELECT DISTINCT(LAUNCH_SITE) FROM SPACEX
```

```
 * ibm_db_sa://ncy38026:***@815fa4db-dc03-4c70-869a-a9cc13
Done.
```

Out[6]:

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

Launch Site Names Begin with 'CCA'

- To select launch sites beginning with CCA, we used LIKE, and then limited the records to 5 using LIMIT

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

In [7]:

```
%sql SELECT * FROM SPACEX WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://ncy38026:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[7]:

DATE	time_utc_	booster_version	launch_site	payload	payload_mass_kg_	orbit	customer	mission_outcome	landing_outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
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
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- We displayed the total payload mass using SUM, and WHERE to only get results from NASA(CRS)

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

```
In [8]:  %sql SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload_mass FROM SPACEX WHERE CUSTOMER = 'NASA (CRS)'

 * ibm_db_sa://ncy38026:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appd
Done.
```

Out[8]:

total_payload_mass
45596

Average Payload Mass by F9 v1.1

- We displayed the average payload mass using AVG coupled with WHERE to only get results from booster F9 v1.1

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

In [9]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEX WHERE BOOSTER_VERSION = 'F9 v1.1'
```

```
 * ibm_db_sa://ncy38026:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8l
Done.
```

Out[9]:

1
2928

First Successful Ground Landing Date

- In order to get the most recent date, we used MIN(DATE), and then specified the type of landing outcome with WHERE

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

Hint:Use min function

In [10]:

```
%sql SELECT MIN(DATE) AS Date FROM SPACEX WHERE LANDING__OUTCOME = 'Success (ground pad)'
```

```
 * ibm_db_sa://ncy38026:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databa
Done.
```

Out[10]:

DATE
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Here we combined two conditions, the type of landing outcome and the payload mass between two given values

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

In [11]:

```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE (LANDING__OUTCOME = 'Success (drone ship)') AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)
```

```
 * ibm_db_sa://ncy38026:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[11]:

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 displayed the total number of successful and failure missions using nested queries and the COUNT operator

List the total number of successful and failure mission outcomes

In [15]:
```
%sql SELECT (SELECT COUNT(MISSION_OUTCOME) FROM SPACEX WHERE MISSION_OUTCOME LIKE '%Success%') AS Success, (SELECT COUNT(MISSION_OUTCOME) FROM SPACEX
```

```
 * ibm_db_sa://ncy38026:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[15]:

success	failure
100	1

Boosters Carried Maximum Payload

- For that query, it was crucial to use GROUP BY and ORDER BY

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

In [22]:
```
%sql SELECT BOOSTER_VERSION, MAX(PAYLOAD_MASS__KG_) AS MAX_PAYLOAD FROM SPACEX GROUP BY BOOSTER_VERSION ORDER BY 2 DESC
```

```
 * ibm_db_sa://ncy38026:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[22]:

booster_version	max_payload
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600
F9 B5 B1060.3	15600

2015 Failed Launch Records

- Here we used the operator LIKE to only select failed launch outcomes, and extracted the year from the date to restrict results to 2015

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [28]:
```
%sql SELECT DATE, LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE (LANDING__OUTCOME LIKE '%Failure%') AND (YEAR(DATE) = '2015')
```

```
 * ibm_db_sa://ncy38026:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[28]:

DATE	landing__outcome	booster_version	launch_site
2015-01-10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
2015-04-14	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- For this query, we used COUNT, BETWEEN and GROUP BY

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

In [31]:

```
%sql SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS nb_landing_outcomes FROM SPACEX WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LAN
```

```
 * ibm_db_sa://ncy38026:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

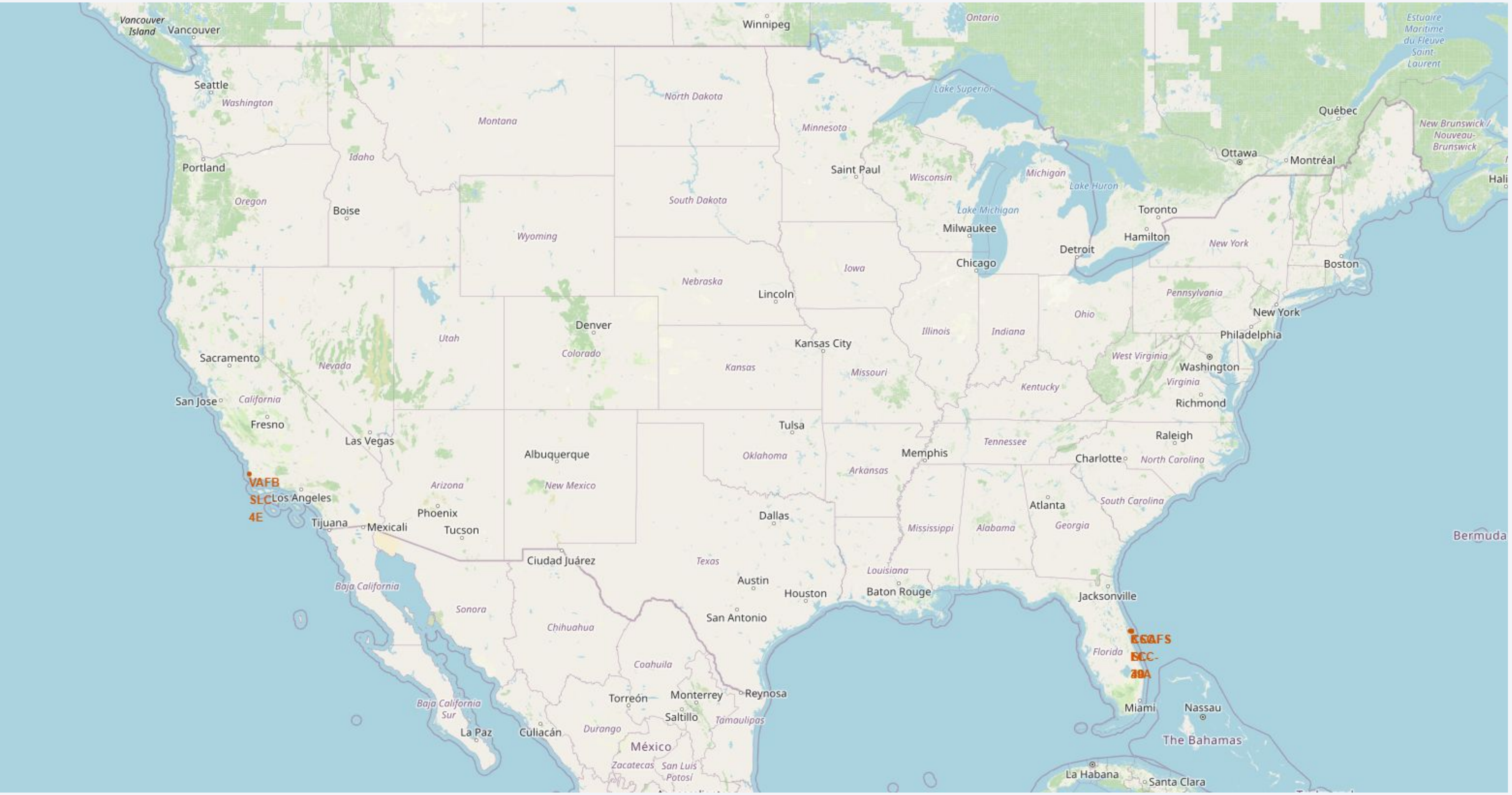
Out[31]:

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

Section 3

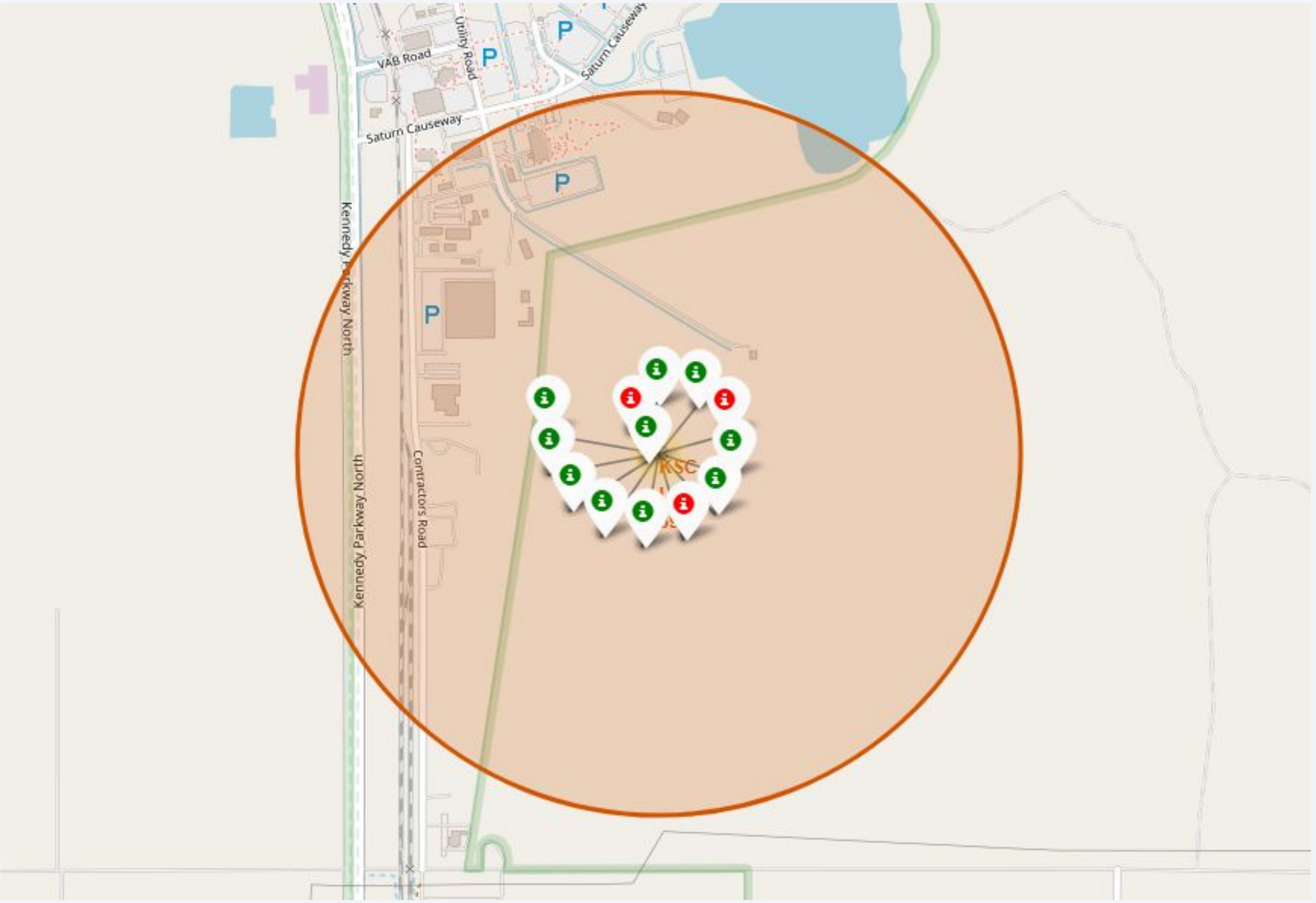
Launch Sites Proximities Analysis

All launch site location markers

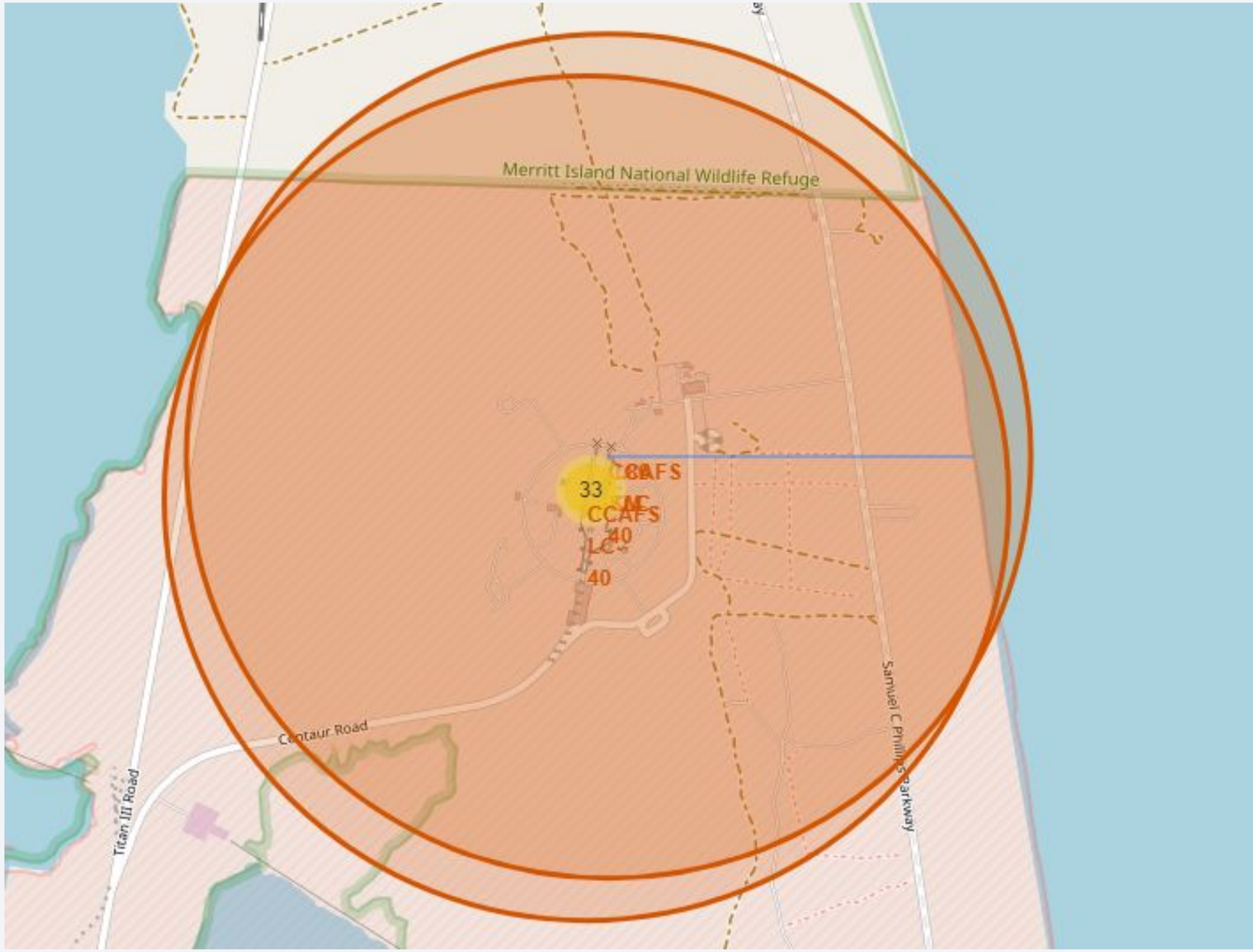


Launch outcomes in KSC LC-39A site

- Successful launches are in green, failed launches are in red



Distance from CCAFS SLC-40 to coastline

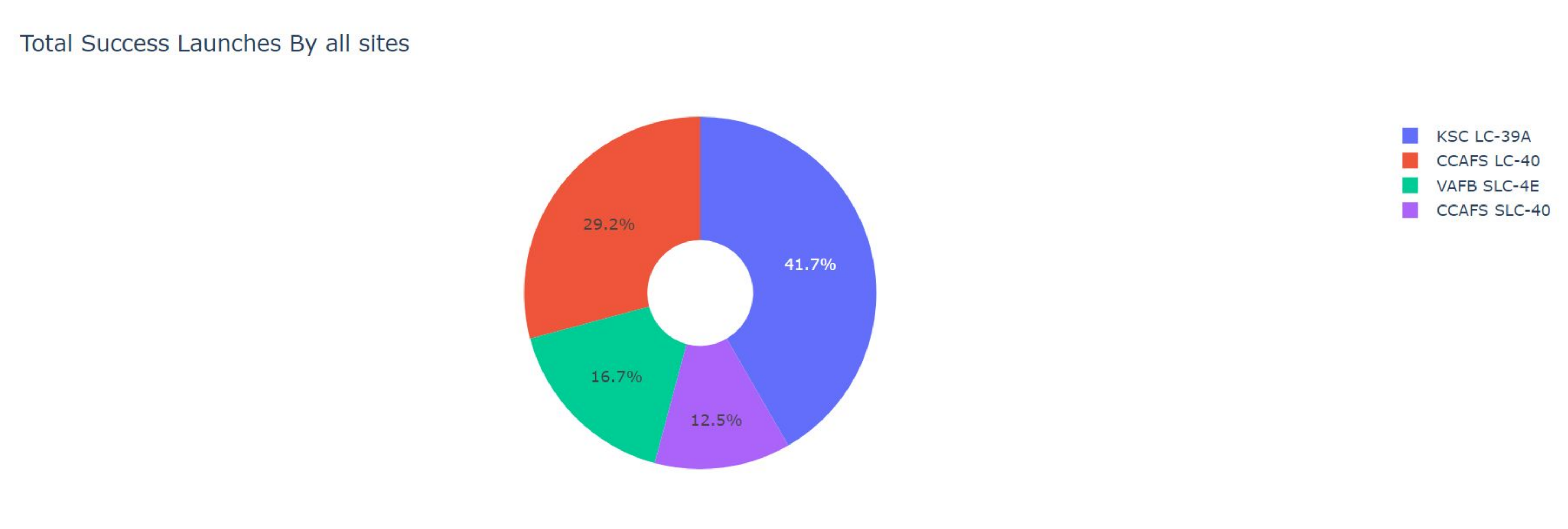


Section 4

Build a Dashboard with Plotly Dash

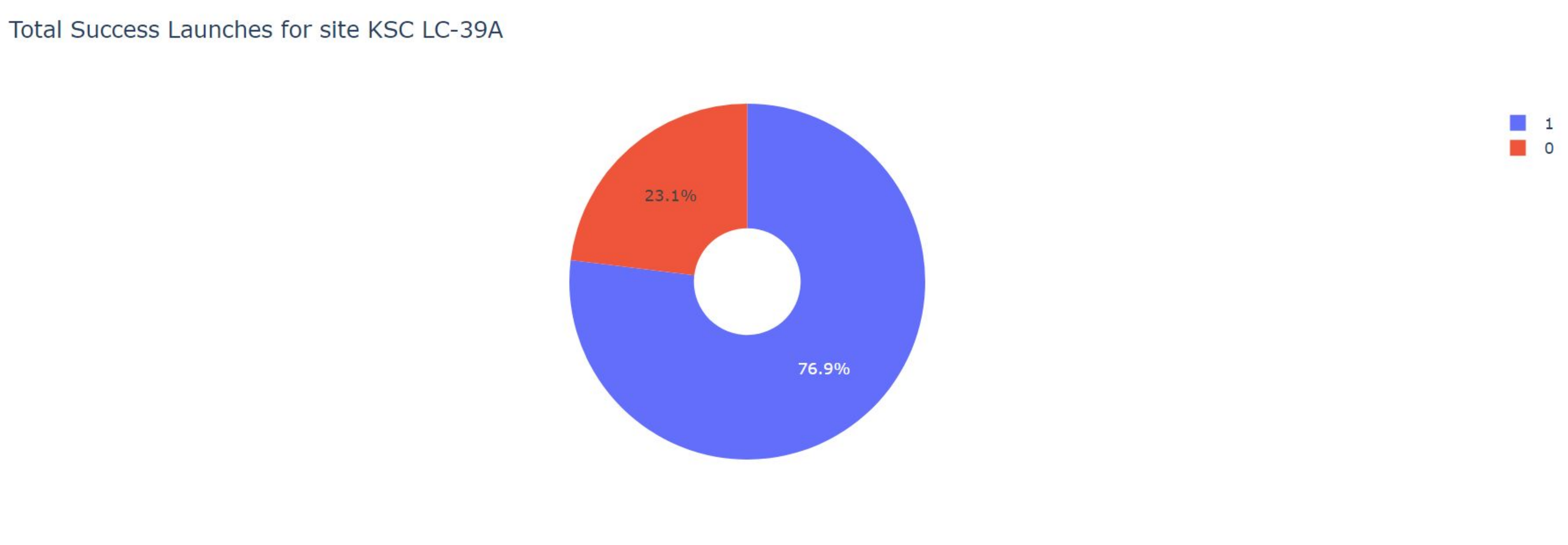
Launch success count for all sites

- We can see that the KSC LC-39A site has the highest success rate



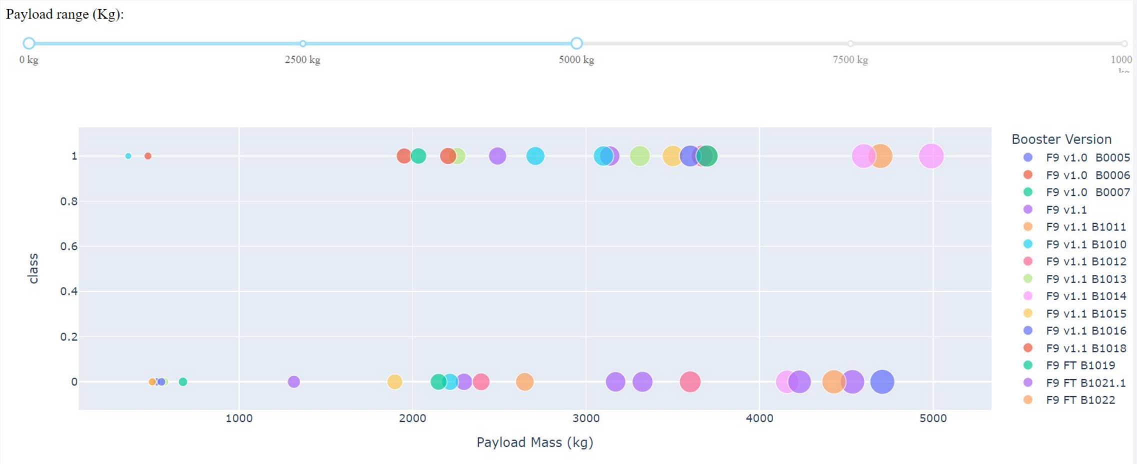
Launch site with highest launch success ratio

- KSC LC-39A has the highest success ratio with 76,9%



Payload vs Launch Outcome for all sites

- We can see lighters payloads yield more successful launches than heavier payloads



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the best performing model, with an accuracy of 88.6%

Find the method performs best:

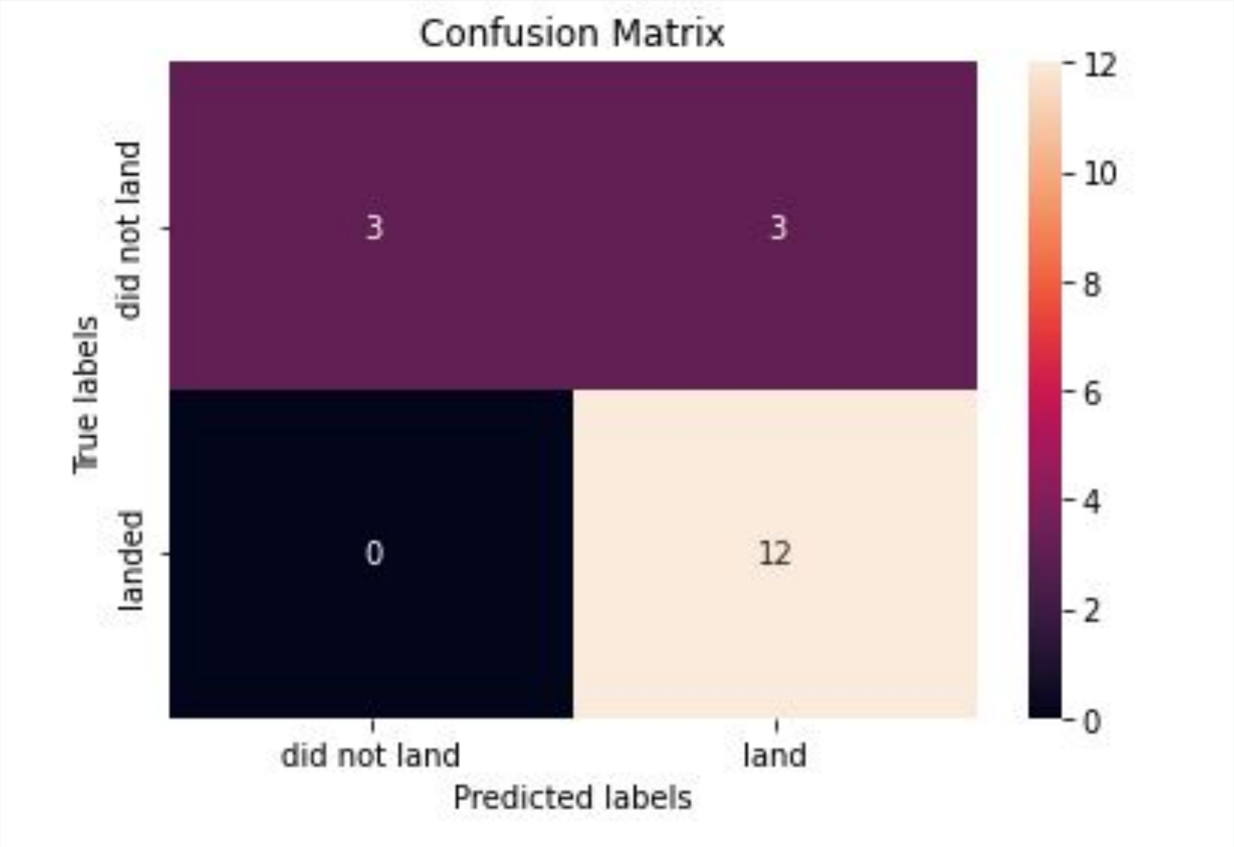
In [26]:

```
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 parameters are :', tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best parameters are :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best parameters are :', logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8857142857142858
Best parameters are : {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 10, 'splitter': 'b
est'}
```

Confusion Matrix

- Here's the confusion matrix for the decision tree classifier.
- It performs well except for unsuccessful launches being wrongly labeled as successful.



Conclusions

- For all launch sites, the success rate improves as the flight number increases
- The orbits ES-L1, GEO, HEO and SSO have the highest success rates
- Success rate soared from 2013 through 2020
- Out of all sites, KSC LC-39A is the most successful
- The best ML model to predict launch success is the decision tree classifier

Thank you!