IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Arjun Nair

4th March 2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through 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 result
  - Interactive analytics in screenshots
  - Predictive Analytics result

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. The main objective of the project is to create an ML model that is able to predict if the first stage of the rocket launches will land successfully.

- Problems you want to find answers

  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

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 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 datasets were collected using various methods:
  - Data was collected by sending a get request to the SpaceX API.
  - The response content was decoded as a Json object using .json() function call and turned into a pandas dataframe using .json_normalize().
  - The data is cleaned and checked for missing values. These missing values are then filled when required.
  - Web scraping is performed on the Wikipedia page for Falcon 9 launch records using the BeautifulSoup package.
  - The objective was to extract the launch records as an HTML table then parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- A get request was to the SpaceX API to collect and clean the requested data. Some data wrangling and formatting was required to perform data analysis.
- The link to the notebook is https://github.com/nujrarian/IBM_DataScience_Capstone/blob/main/1-jupyter-labs-spacex-data-collection-api.ipynb

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

- Web scrapping is used on the Falcon 9 launch records with BeautifulSoup
- Link: https://github.com/nujrarian/IBM_DataScience_Capstone/blob/main/1.5-jupyter-labs-webscraping.ipynb

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

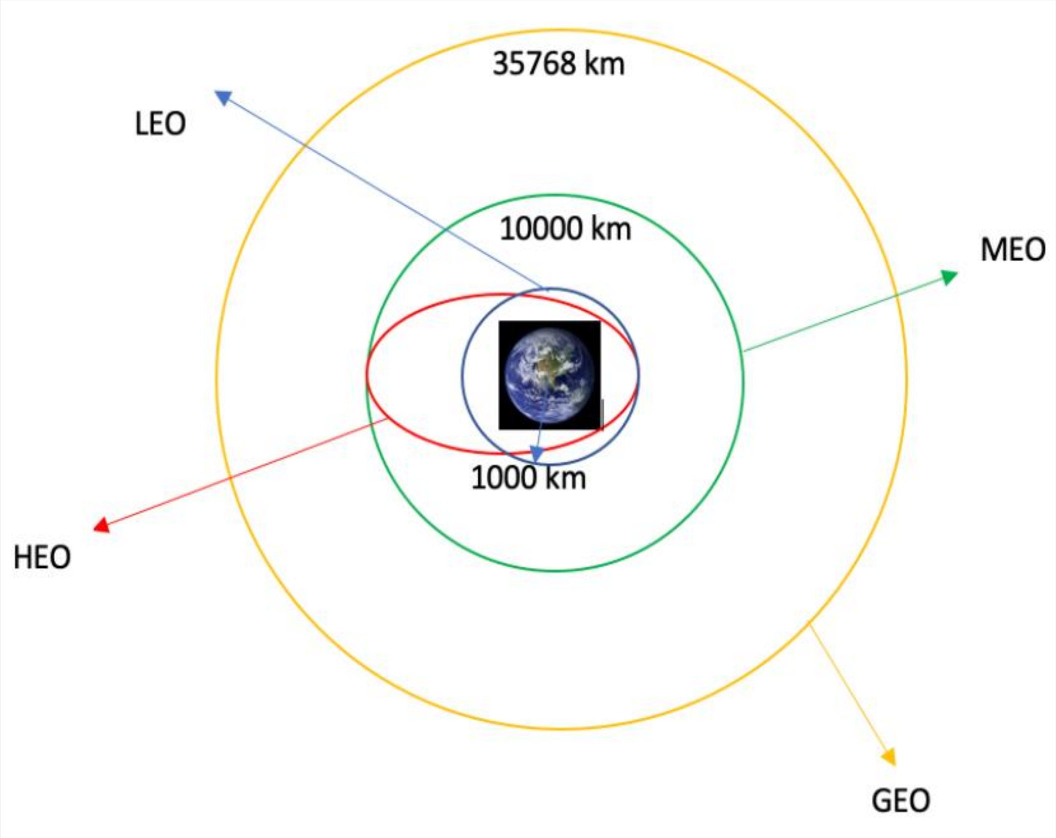
In [30]:
```
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

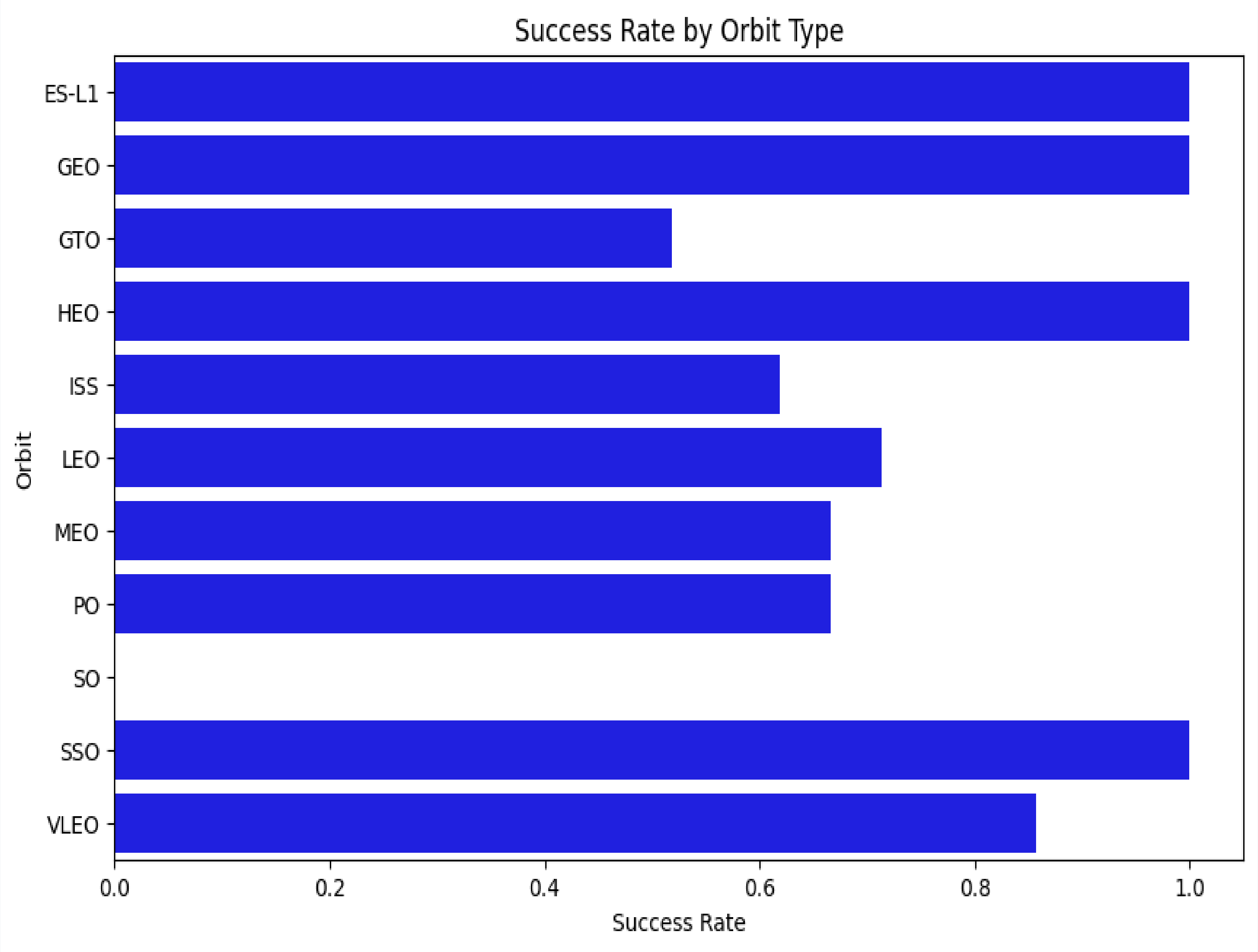
Data Wrangling

- Exploratory data analysis is performed to determine 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.
- https://github.com/nujrarian/IBM_DataScience_Capstone/blob/main/2-labs-jupyter-spacex-Data%20wrangling.ipynb



EDA with Data Visualization

- The data was explored 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.
- Link: https://github.com/nujrarian/IBM_DataScience_Capstone/blob/main/3-jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb



EDA with SQL

- The SpaceX dataset is loaded into a PostgreSQL database.
- EDA with SQL is performed to get insight from the data. Queries were used to find out:
  - 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/nujrarian/IBM_DataScience_Capstone/blob/main/2.5-jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- All launch sites were marked, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- The feature launch outcomes (failure or success) were set to 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, the launch sites having relatively high success rate were identified.
- Calculated the distances between a launch site to its proximities. The following questions were answered:
    - Are launch sites near railways, highways and coastlines.
    - Do launch sites keep certain distance away from cities

Build a Dashboard with Plotly Dash

- Built an interactive dashboard with Plotly dash.

- Plotted pie charts showing the total launches by a certain sites.

- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

- Link: https://github.com/nujrarian/IBM_DataScience_Capstone/blob/main/app.py

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 the model, improved the model using feature engineering and algorithm tuning.
- The best performing classification model were found.
- Link: https://github.com/nujrarian/IBM_DataScience_Capstone/blob/main/5-SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

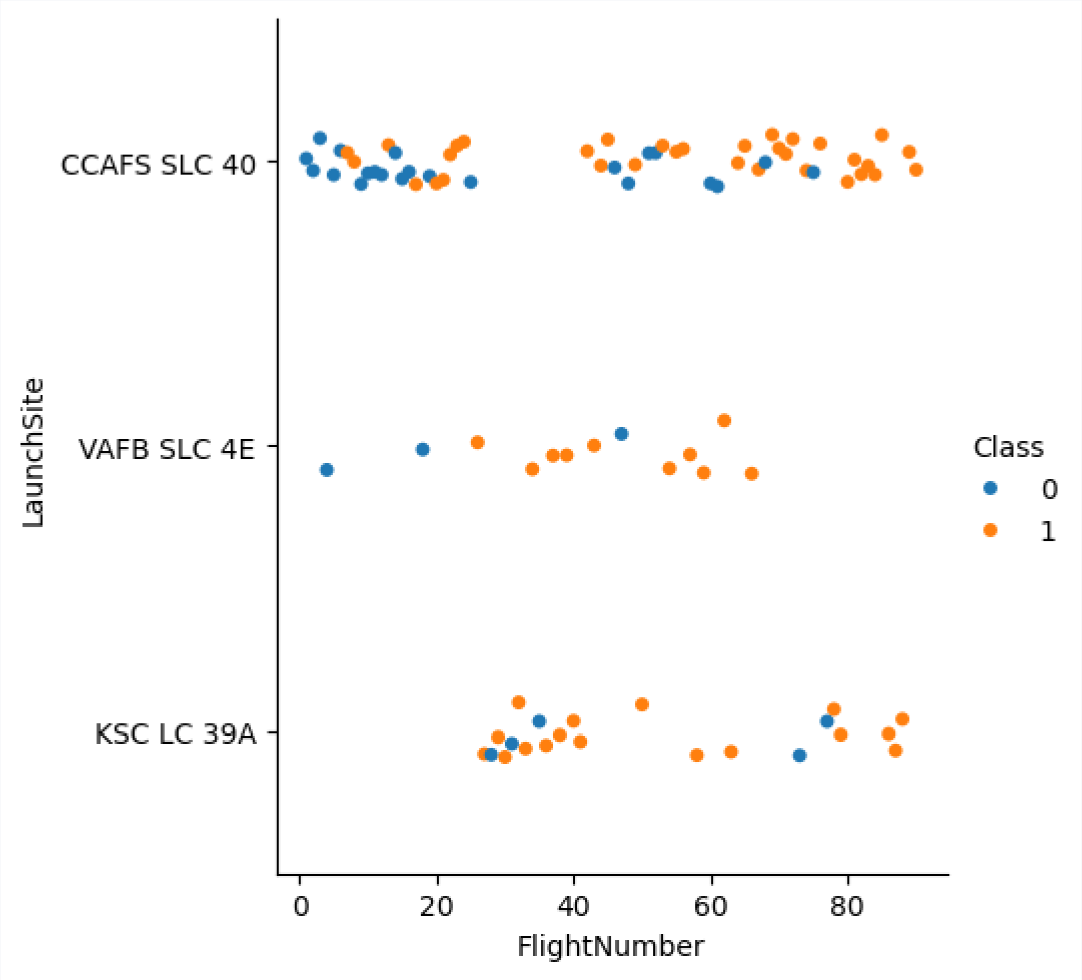
Results

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

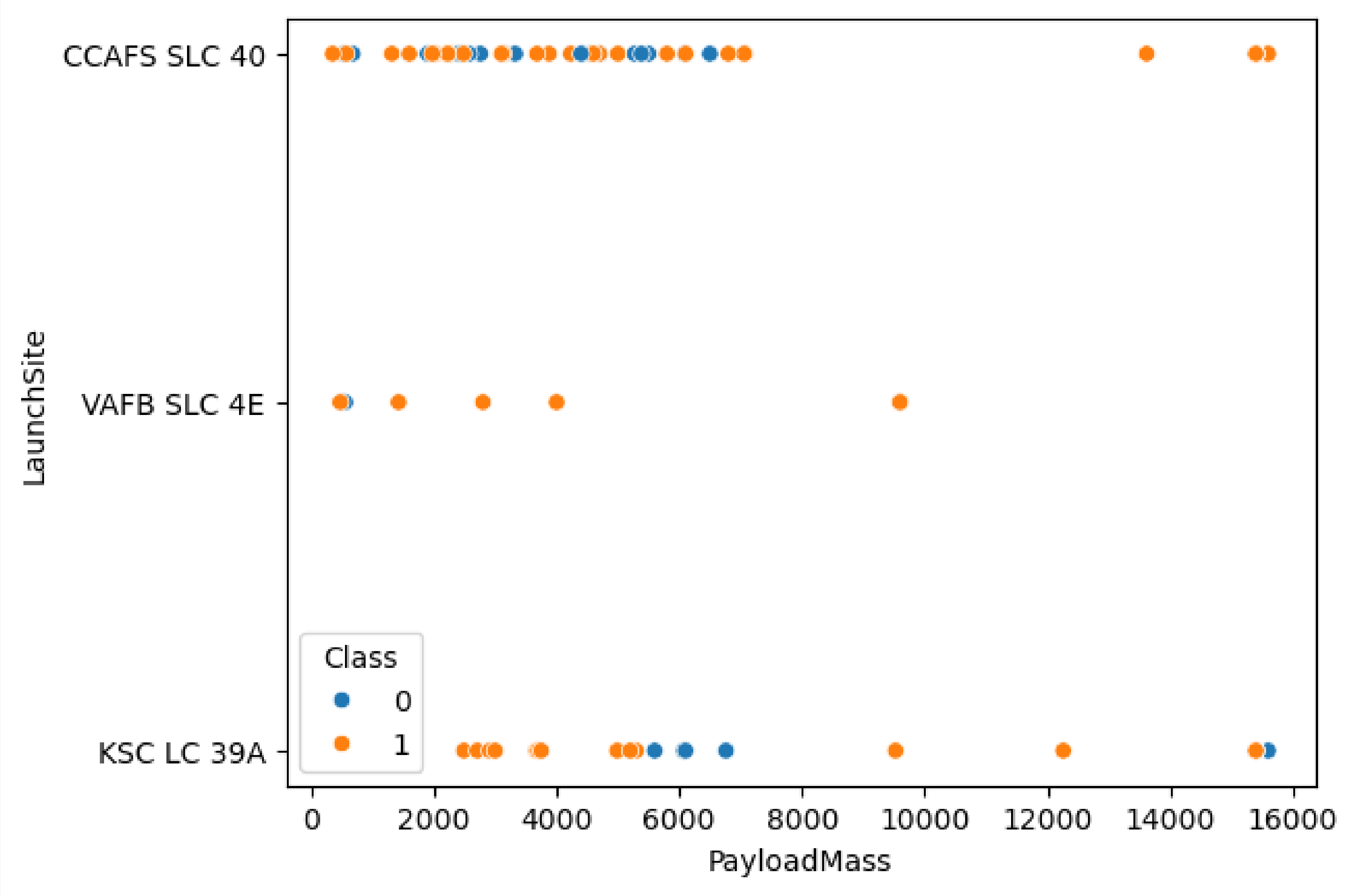
Section 2

Insights drawn from EDA

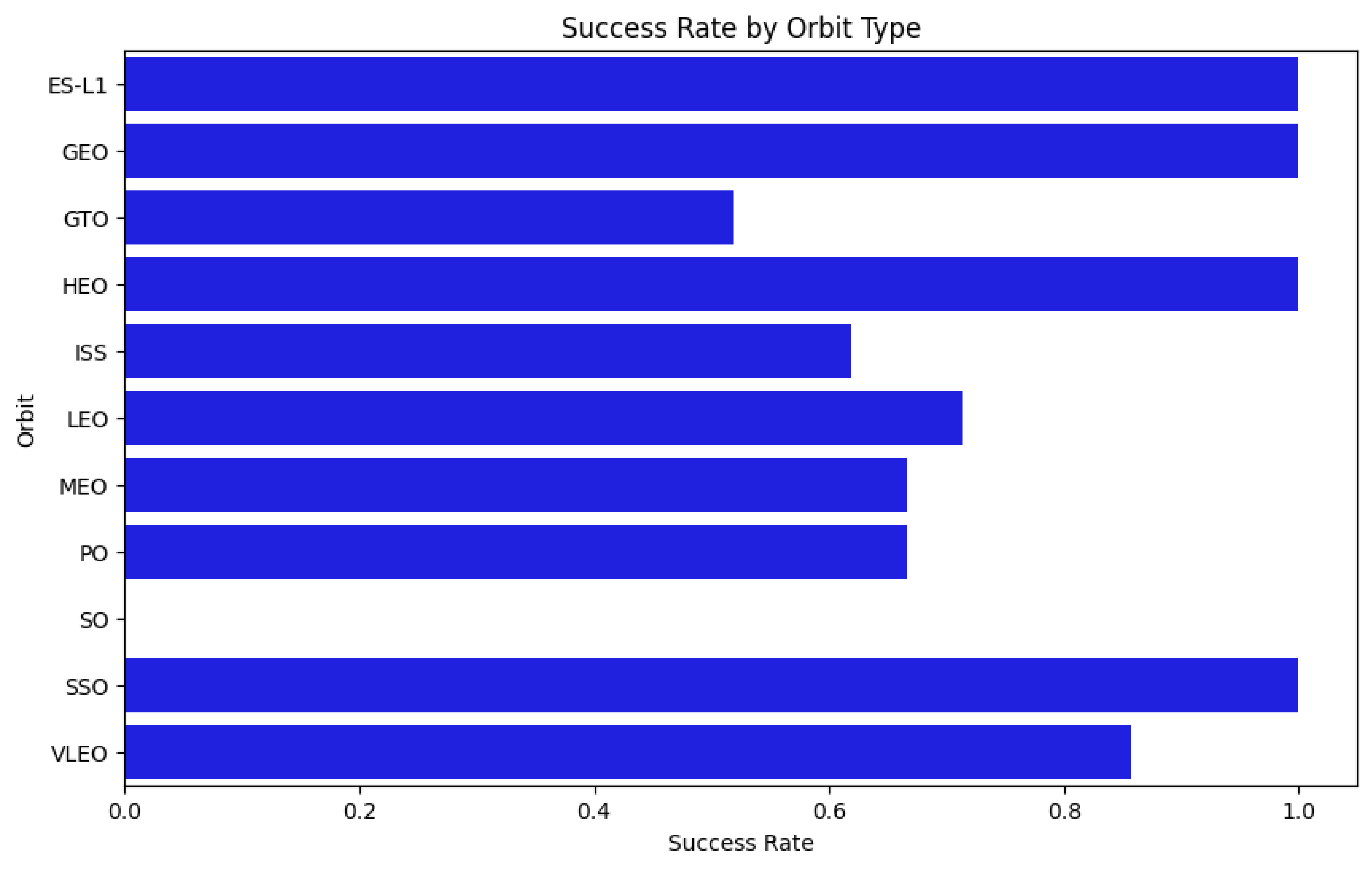
Flight Number vs. Launch Site



Payload vs. Launch Site

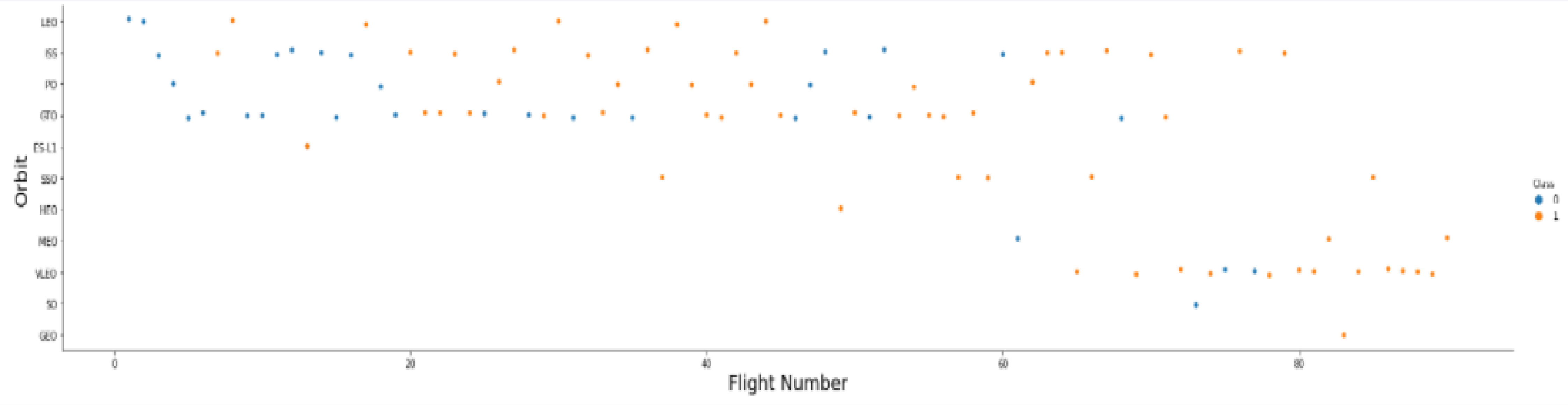


Success Rate vs. Orbit Type

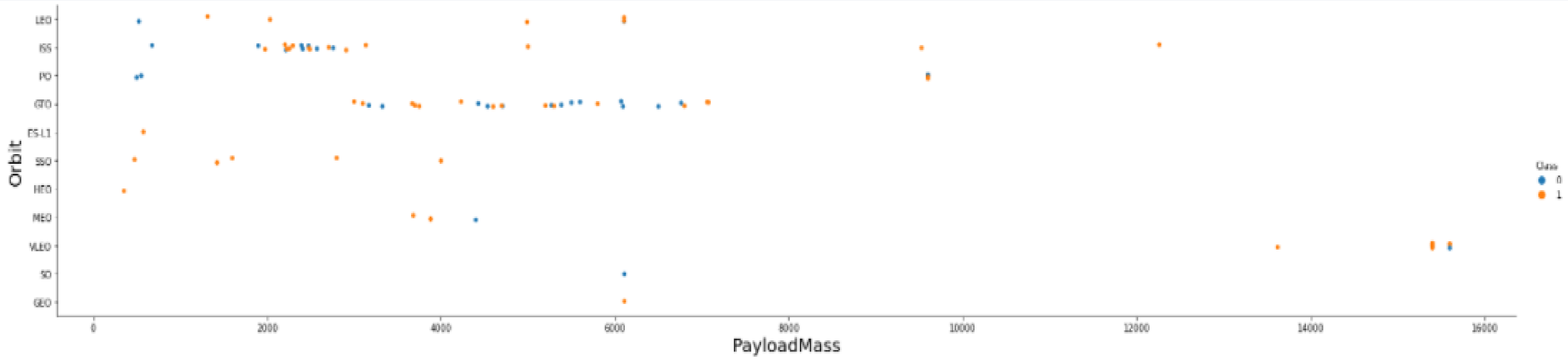


Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type.

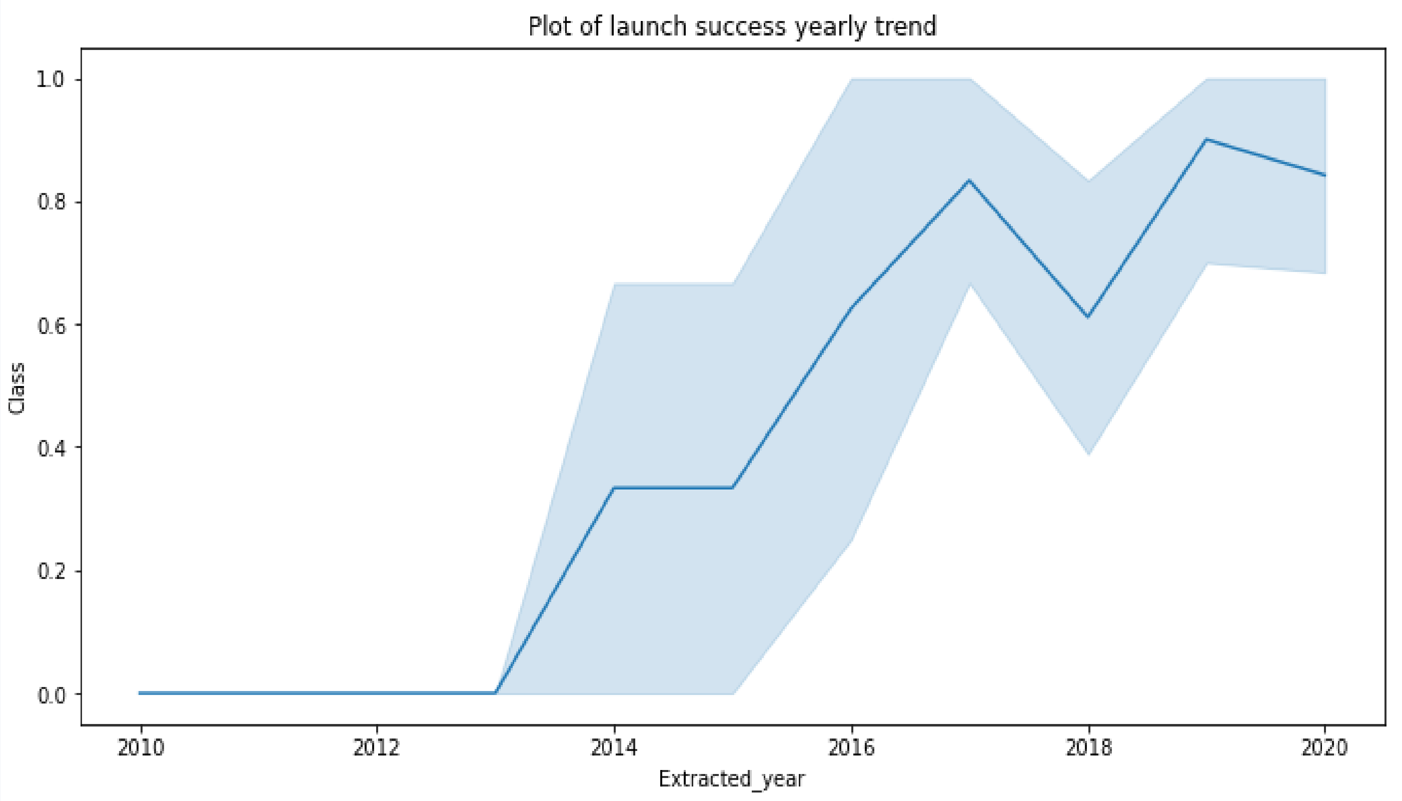


Payload vs. Orbit Type



Launch Success Yearly Trend

- From the plot, we can observe that success rate kept increasing from 2013 till 2020.



All Launch Site Names

- The key word **DISTINCT** was used 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'

- Used the query shown to display 5 records where launch sites begin with `CCA`

```
In [10]:
%sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5

 * sqlite:///my_data1.db
Done.
Out[10]:
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Cu
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	
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)	
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	

Total Payload Mass

- Calculated the total payload carried by boosters from NASA.

```
In [11]:
%sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "Customer" LIKE 'NASA
 * sqlite:///my_data1.db
Done.
Out[11]:
```

SUM("PAYLOAD_MASS__KG_")
45596

Average Payload Mass by F9 v1.1

- Calculated the average payload mass carried by booster version F9 v1.1

Task 4

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

In [12]:

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "Booster_Version" = 'F
```

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

Out[12]:

AVG("PAYLOAD_MASS__KG_")
2928.4

First Successful Ground Landing Date

- Observed that the dates of the first successful landing outcome on ground pad.

```
In [13]:

%sql SELECT MIN("Date") FROM SPACEXTABLE WHERE "Mission_Outcome" = 'Success'

 * sqlite:///my_data1.db
Done.
Out[13]:
```

MIN("Date")
2010-06-04

Successful Drone Ship Landing with Payload between 4000 and 6000

- 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.

In [14]:

```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "Mission_Outcome" = 'Success'
```

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

Out[14]:

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020

Total Number of Successful and Failure Mission Outcomes

- Calculated total number of MissionOutcome values that are a success or a failure.

```
In [15]:
%sql SELECT "Mission_Outcome", COUNT(*) AS "Total" FROM SPACEXTABLE GROUP BY "Miss
 * sqlite:///my_data1.db
Done.
Out[15]:
```

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

Boosters Carried Maximum Payload

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

```
In [21]:
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT M

 * sqlite:///my_data1.db
Done.
Out[21]:
```

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

2015 Launch Records

- 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

```
In [32]:
%sql SELECT substr(Date, 6,2) as "Month","Landing_Outcome","Booster_Version","Laun

 * sqlite:///my_data1.db
Done.
Out[32]:
```

Month	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

- 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.
- Applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

```
In [33]:
%sql SELECT "Landing_Outcome", COUNT("Landing_Outcome") AS "Count" FROM SPACEXTABL
 * sqlite:///my_data1.db
Done.
Out[33]:
```

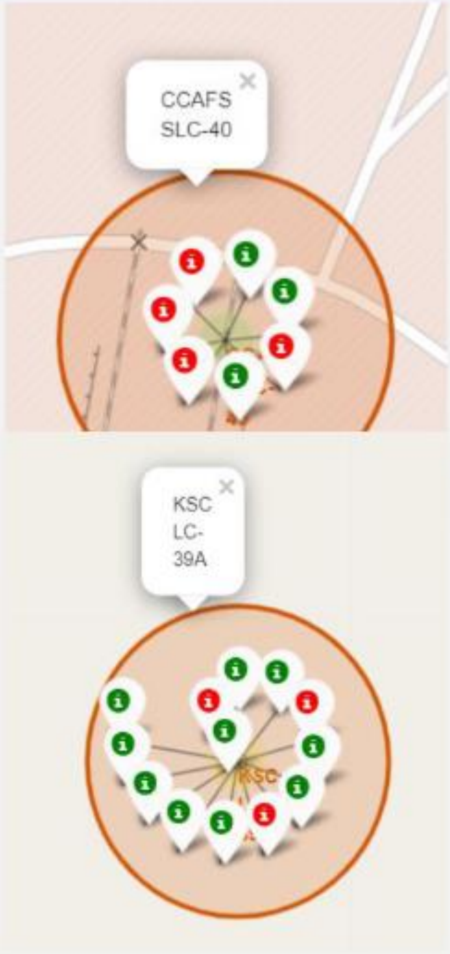
Landing_Outcome	Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

All launch sites global map markers

Markers showing launch sites with color labels



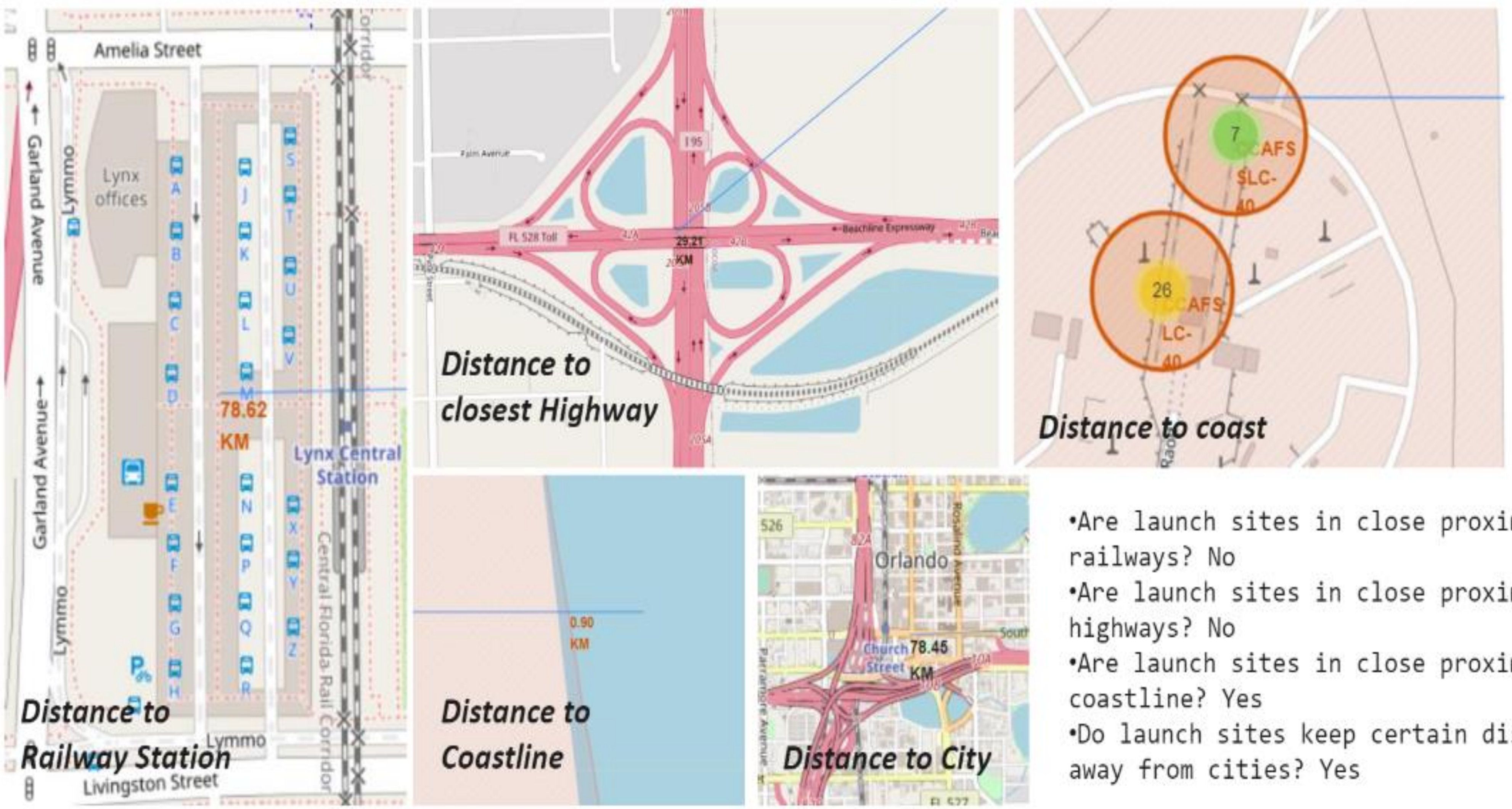
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

Launch Site distance to landmarks

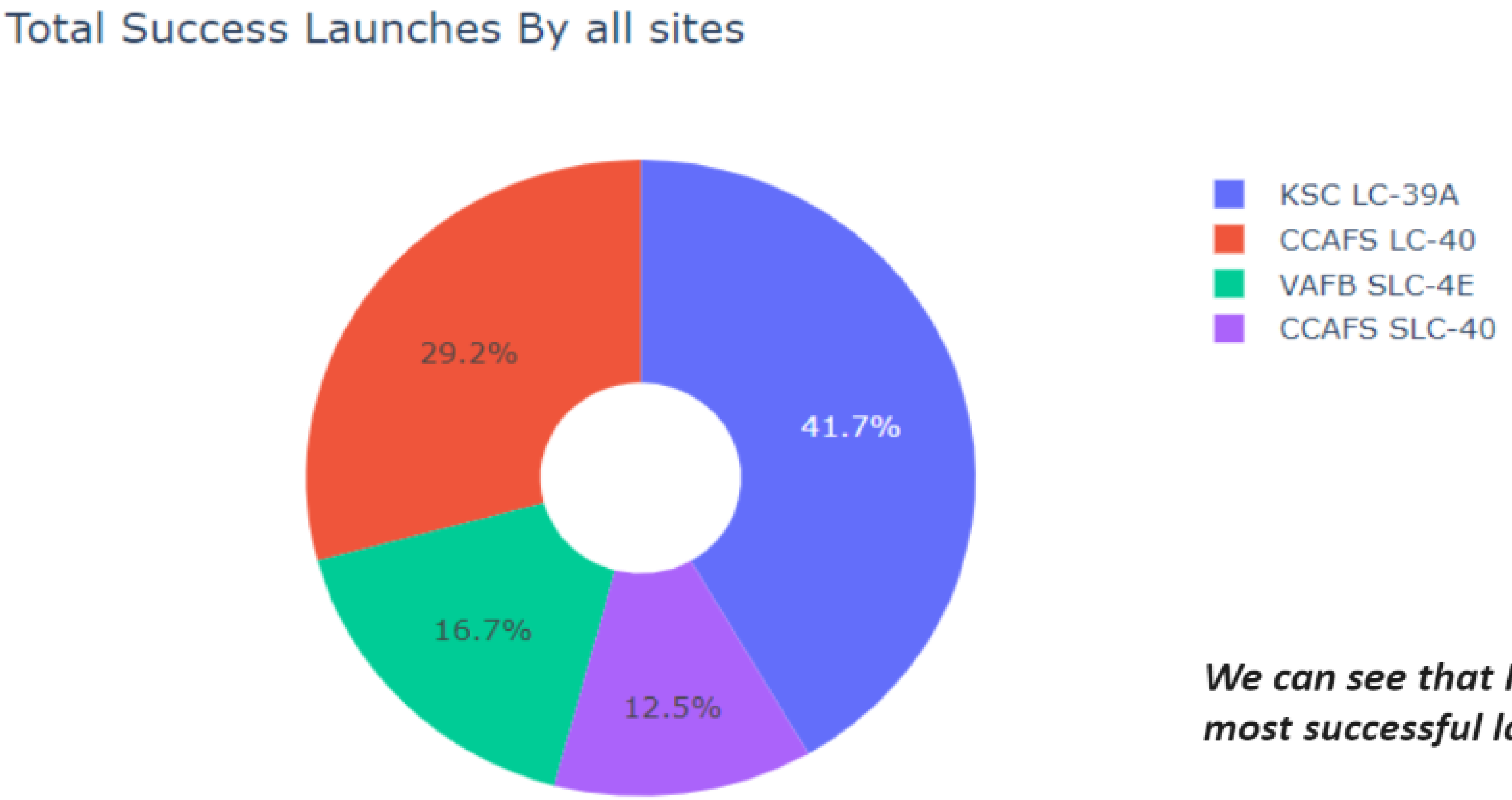


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

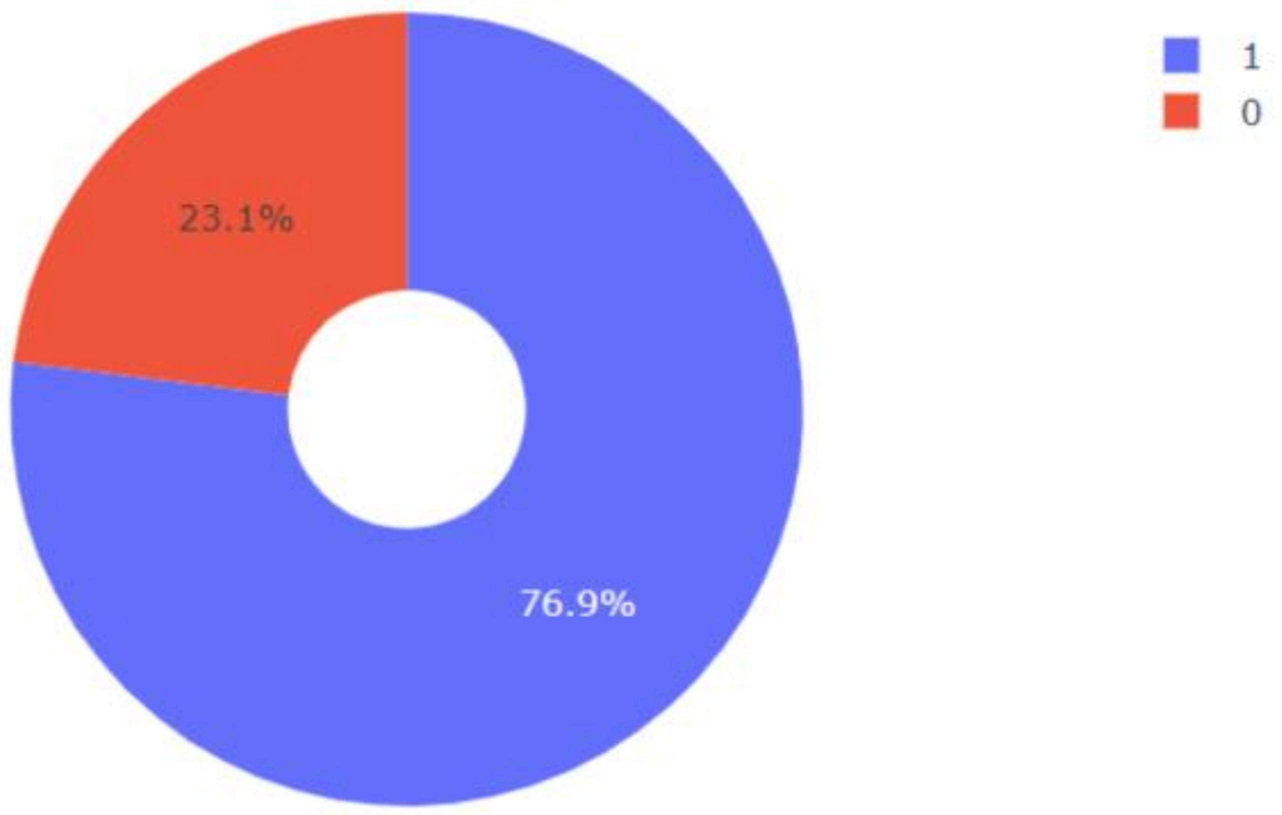
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



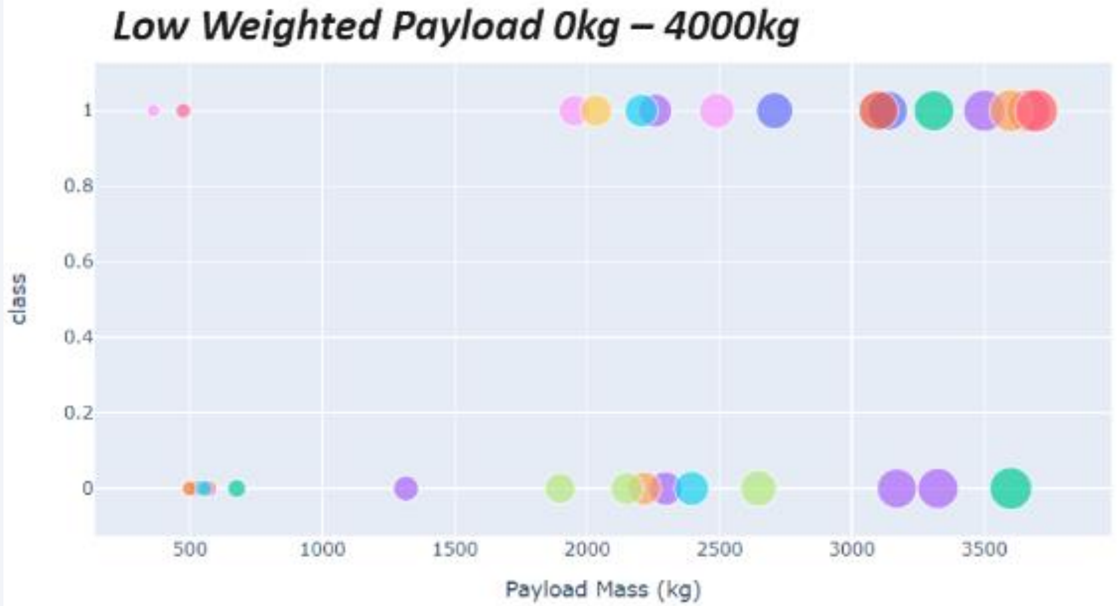
We can see that KSC LC-39A had the most successful launches from all the sites

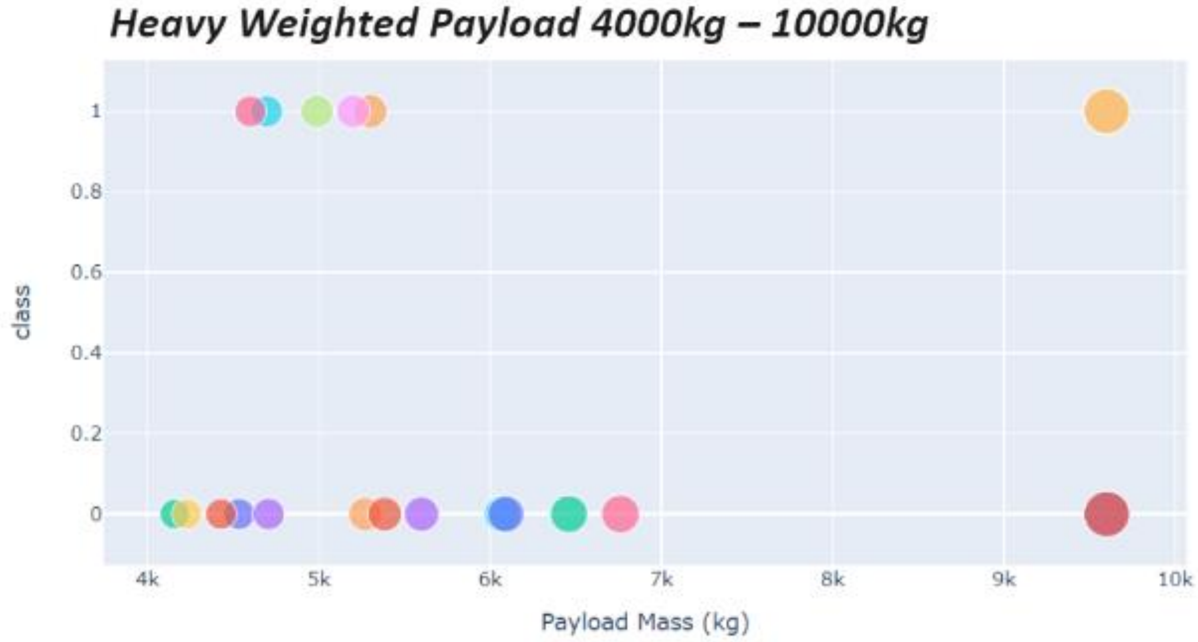
Pie chart showing the Launch site with the highest launch success ratio



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

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

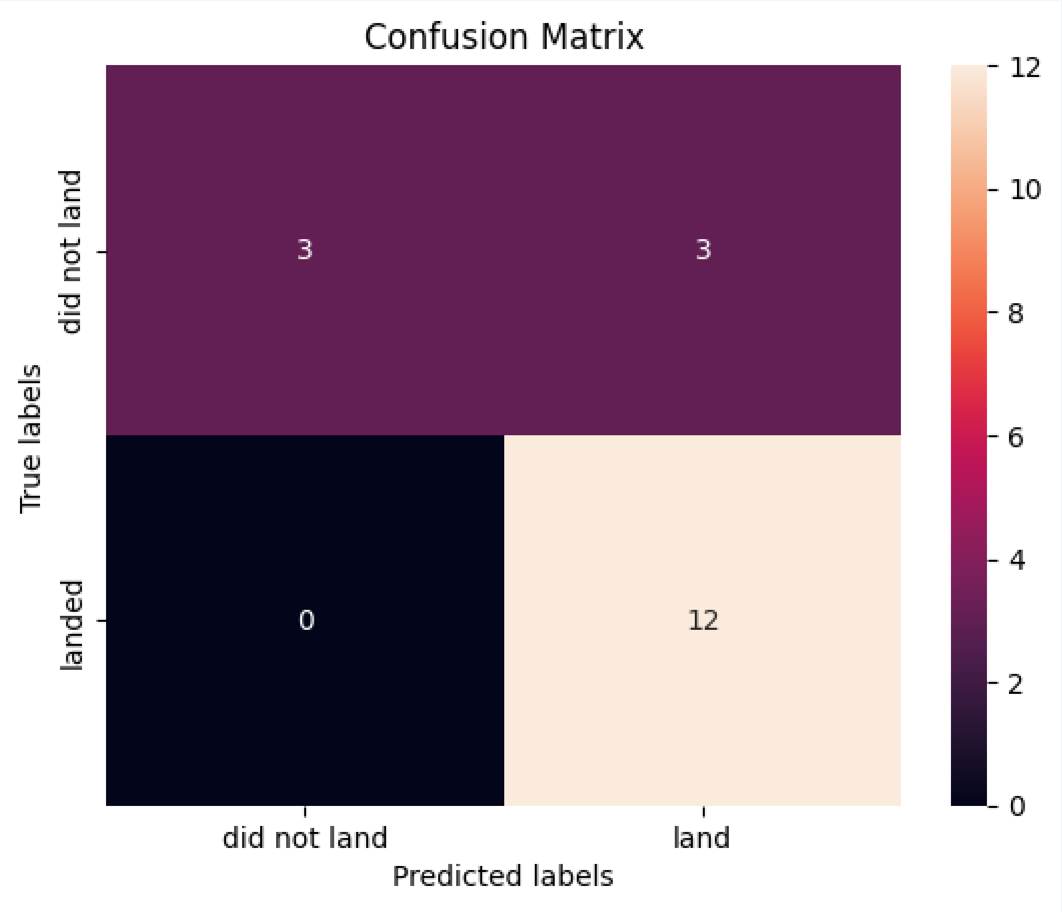
```
In [38]:
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('DT params:', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('KNN params:', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('LR params:', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('SVM params:', svm_cv.best_params_)

Best model is DecisionTree with a score of 0.8625
DT params: {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_s
amples_leaf': 2, '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.



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!