IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Karunesh Sehgal
07-02-2025

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
  - Interactive analytics (Dashboard)
  - Predictive Analytics

Introduction

- Project background and context

  Space X 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 Space X 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 space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

  1. What factors determine if the rocket will land successfully?
  2. The relation among various features that determine the success rate of a successful landing.
  3. 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

- 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.
- We applied web scrapping to webscrape Falcon 9 launch records with BeautifulSoup

GitHub URL: https://github.com/sehgal71/Data-Science-Capstone-SpaceX/blob/main/jupyter-labs-webscraping.ipynb

In [6]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_l
```

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 [49]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url).text
```

In [52]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response)
```

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

In [54]:
```
# Use soup.title attribute
soup.title
```

Out[54]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

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`, reference link towards the end of this lab

In [59]:
```
# 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')
```

Data Collection - Scraping

- We applied web scrapping to webscrape Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.

In [64]:

```
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

for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
```

Check the extracted column names

In [67]:

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

In [73]:

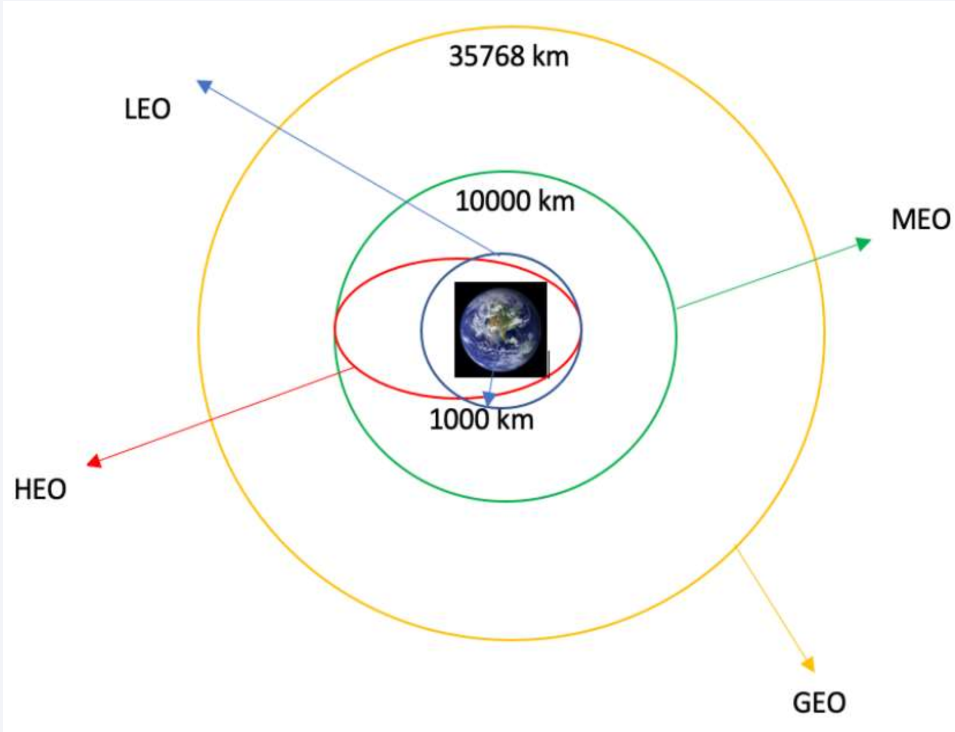
```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Data Wrangling

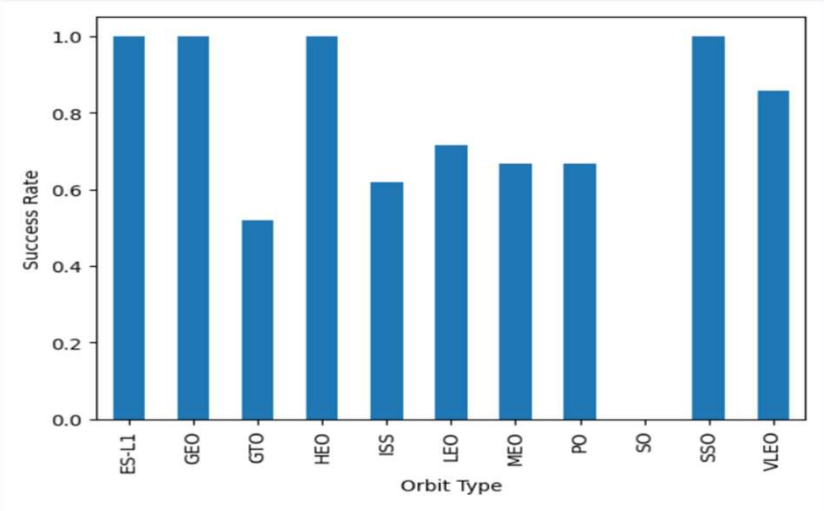
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.

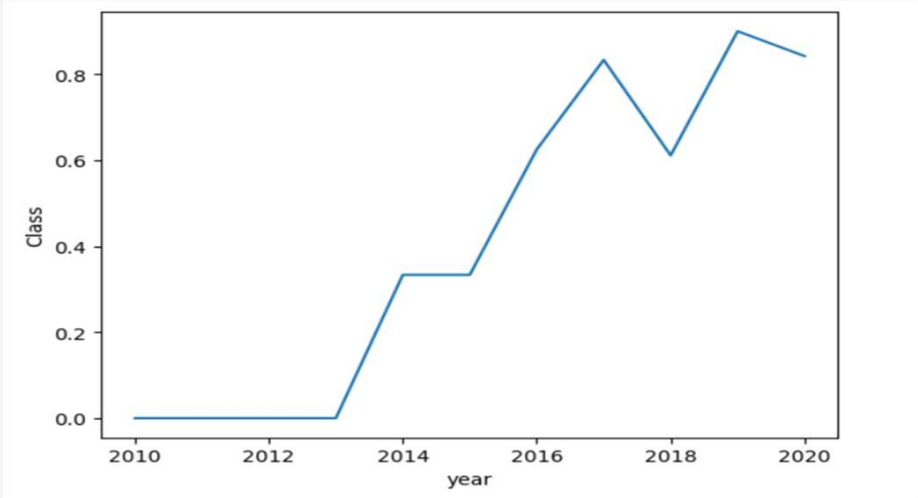


GitHub URL:
https://github.com/sehgal71/Data-Science-Capstone-SpaceX/blob/main/labs-jupyter-spacex-Data%20wrangling-v2.ipynb

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.





GitHub URL:
https://github.com/sehgal71/Data-Science-Capstone-SpaceX/blob/main/jupyter-labs-eda-dataviz-v2.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a SQL 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.

GitHub URL:

https://github.com/sehgal71/Data-Science-Capstone-SpaceX/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Interactive Map using 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. We answered some question for instance:
    - Are launch sites near railways, highways and coastlines.
    - Do launch sites keep certain distance away from cities.

GitHub URL:

https://github.com/sehgal71/Data-Science-Capstone-SpaceX/blob/main/lab-jupyter-launch-site-location-v2.ipynb

Dashboard using 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 customizable using the range slider.

GitHub URL:

https://github.com/sehgal71/Data-Science-Capstone-SpaceX/blob/main/jupyter-labs-eda-dataviz-v2.ipynb

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.

GitHub URL:

https://github.com/sehgal71/Data-Science-Capstone-SpaceX/blob/main/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb

Results

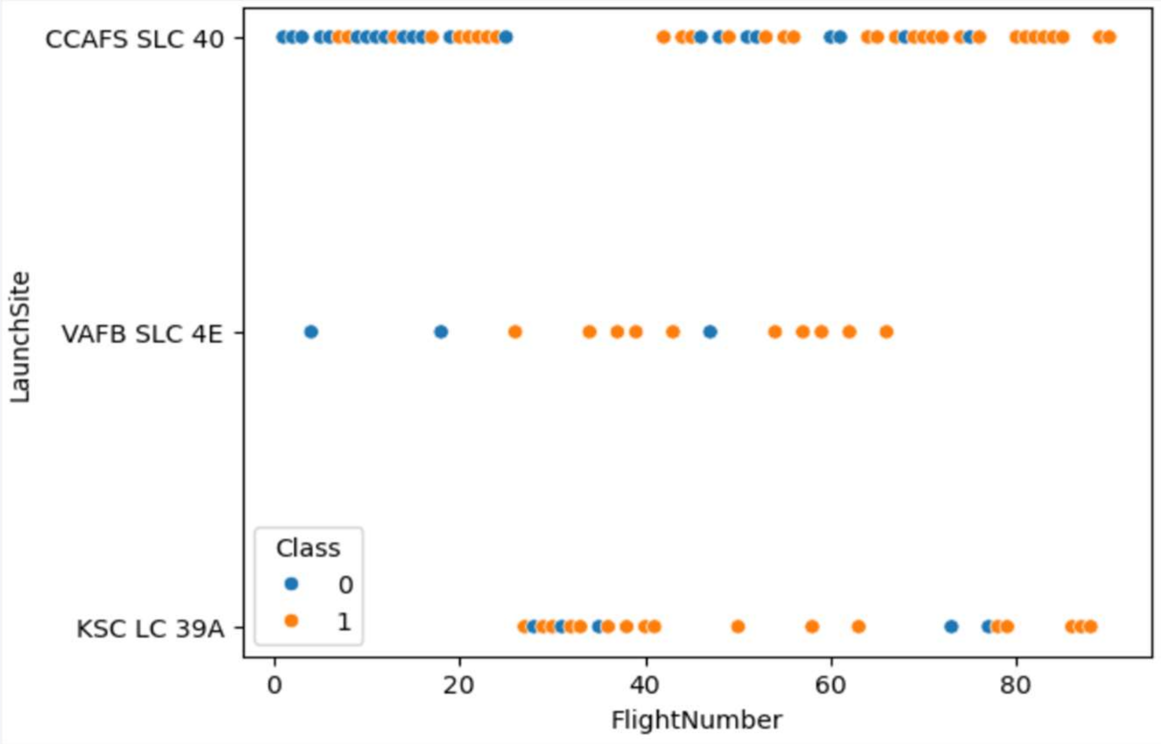
- Exploratory data analysis results
- Interactive analytics dashboard in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

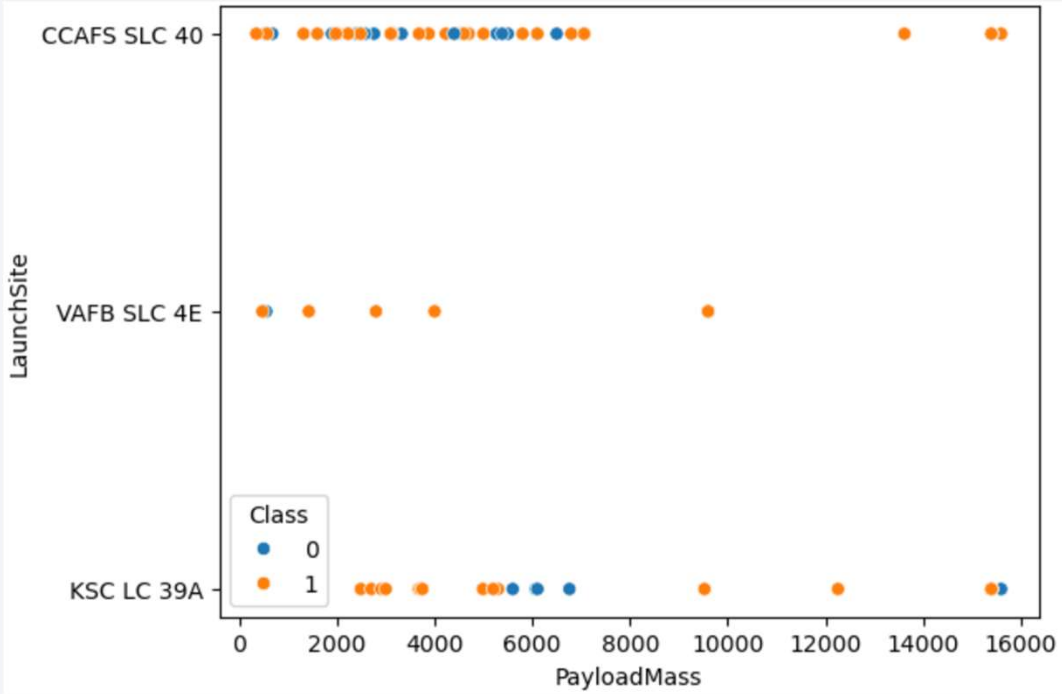
Flight Number vs. Launch Site

- The graph shows that higher amount of flight has better success rate at different sites.



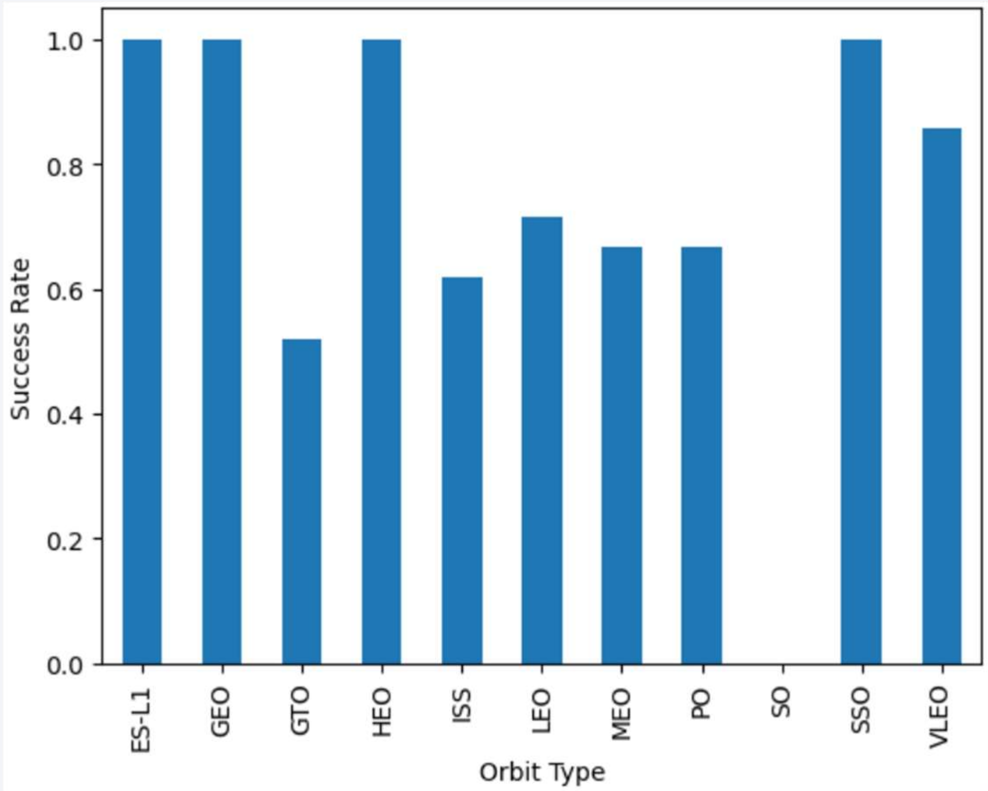
Payload vs. Launch Site

- Higher payload shows a higher success rate.



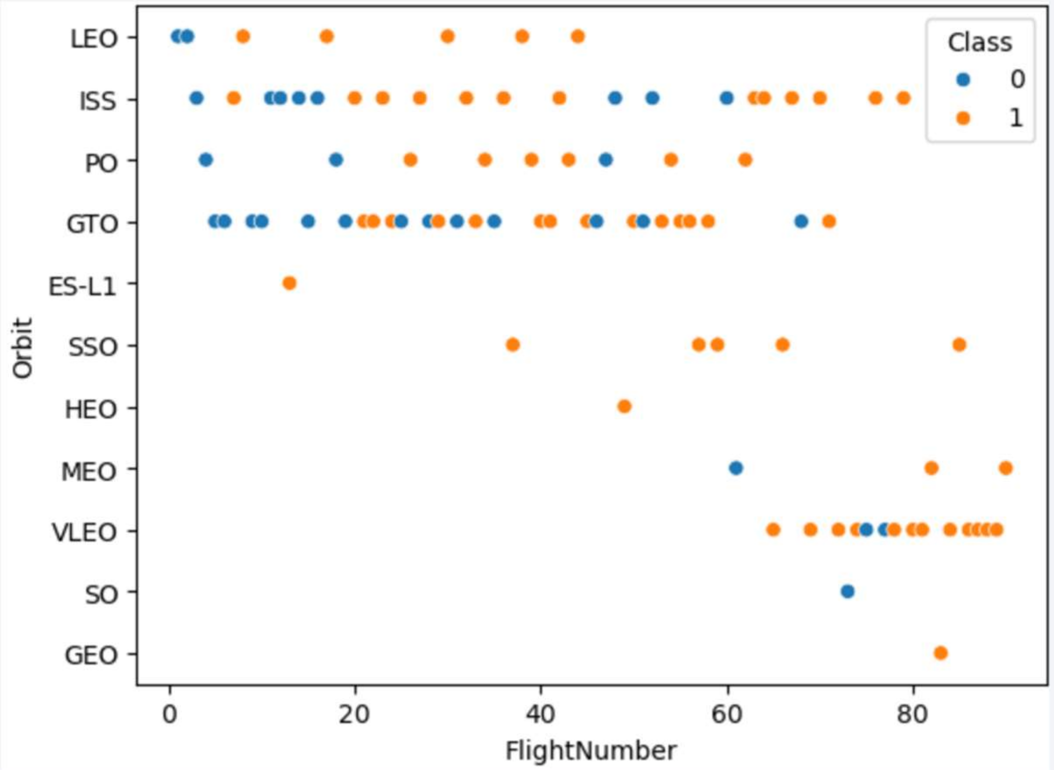
Success Rate vs. Orbit Type

- From the graph, it is noticed that ES-L1, GEO, HEO, SSO, VLEO had the highest success rate.



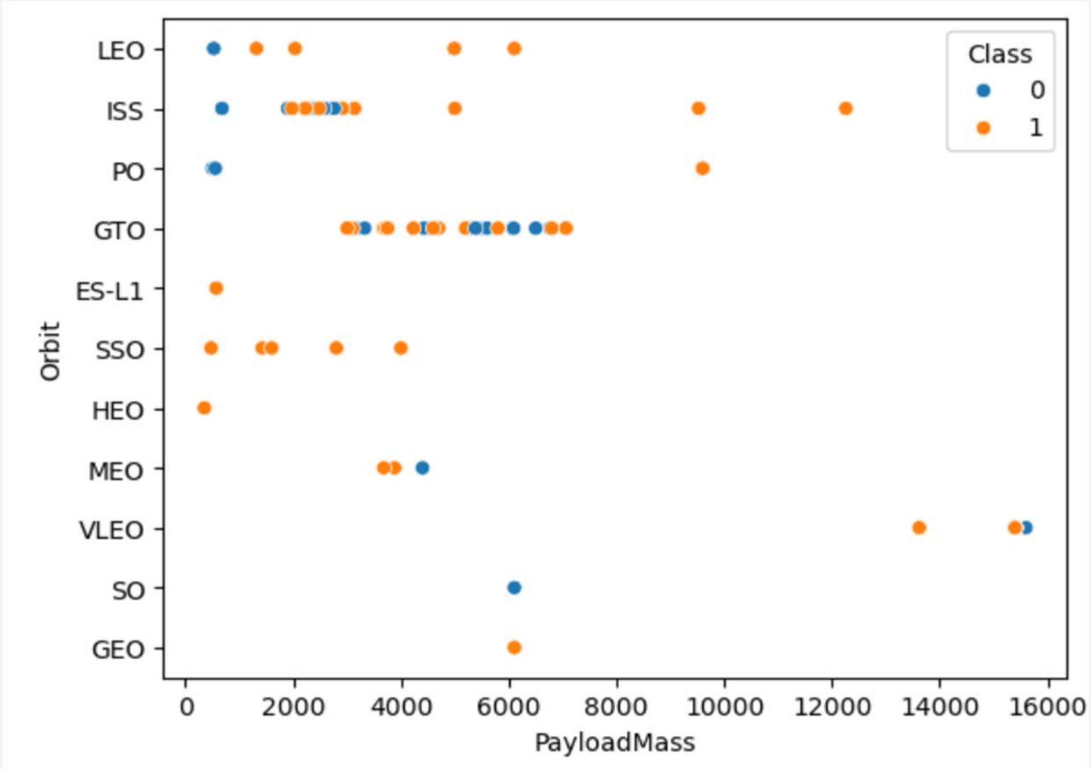
Flight Number vs. Orbit Type

- There is no relationship between Orbit and Flight Number.



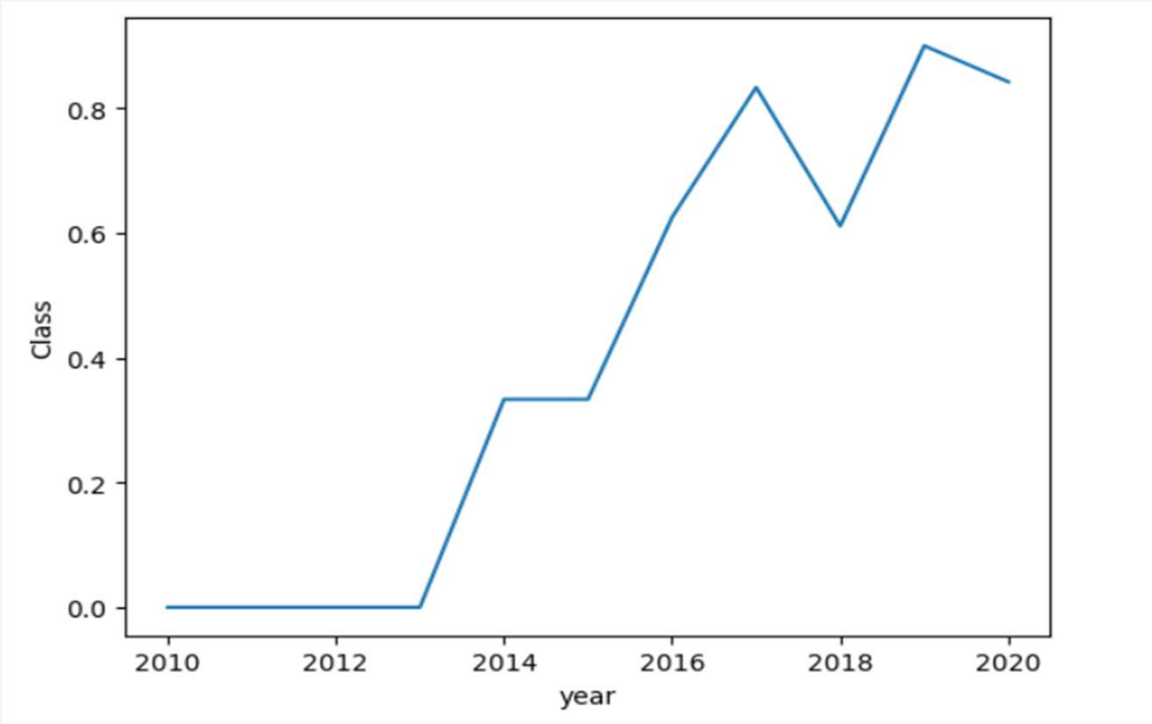
Payload vs. Orbit Type

- There is a positive outcome at high payloads.
- There is no strong relation between Orbit and Payload.



Launch Success Yearly Trend

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



All Launch Site Names

SELECT DISTINCT LAUNCH_SITE

FROM SPACEXTBL

In above query we are selecting launch_site from SPACEXTBL and distinct keyword in sql query is used to select only unique launch_site (no repetition). We used magic function %%sql to write sql command that functions in Jupyter notebook.

```
In [62]: %%sql
         SELECT DISTINCT LAUNCH_SITE
         FROM SPACEXTBL;

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

Out[62]:

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

Launch Site Names Begin with 'CCA'

SELECT *

FROM SPACEXTBL

WHERE LAUNCH_SITE LIKE 'CCA%'

LIMIT 5

In above query we are selecting 5 record where launch_site starts with CCA.

In [68]:
```
%%sql
SELECT LAUNCH_SITE
FROM SPACEXTBL
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 5;
```

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

Out[68]:

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

Total Payload Mass

SELECT SUM(PAYLOAD_MASS__KG_)

FROM SPACEXTBL

WHERE CUSTOMER = 'NASA (CRS)’

In above query we are calculating total payload mass carried by boosters launched by NASA (CRS).

In [70]:

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';
```

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

Out[70]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.0

SELECT AVG(PAYLOAD_MASS__KG_)

FROM SPACEXTBL

WHERE BOOSTER_VERSION LIKE 'F9 v1.0’

In above query we are calculating average payload mass for booster version 'F9 v1.0'.

```
In [72]: %%sql
         SELECT AVG(PAYLOAD_MASS__KG_)
         FROM SPACEXTBL
         WHERE Booster_Version LIKE 'F9 v1.0%';

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

Out[72]:

AVG(PAYLOAD_MASS__KG_)
340.4

First Successful Ground Landing Date

SELECT MIN(DATE)

FROM SPACEXTBL

WHERE LANDING__OUTCOME = 'Success%'

In above query we are finding the date where the landing was successful and we are selecting the very first date using MIN() function.

```
In [76]: %%sql
         SELECT MIN(Date)
         FROM SPACEXTBL
         WHERE Landing_Outcome = 'Success (ground pad)';

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

Out[76]:

MIN(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

SELECT BOOSTER_VERSION

FROM SPACEXTBL

WHERE LANDING__OUTCOME='Success (drone ship)’

AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000

In above query we are selecting booster_version from spacextbl which had success drone ship landing and payload mass in range 4000 to 6000.

In [80]:

```
%%sql
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE LANDING_OUTCOME = 'Success (drone ship)'
    AND 4000 < PAYLOAD_MASS__KG_ < 6000;
```

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

Out[80]:

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

SELECT MISSION_COUNT,

COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER

FROM SAPCEXTBL

GROUP BY MISSION_OUTCOME

In above query we are selecting the total number of successful and failure mission outcomes.

List the total number of successful and failure mission outcomes

In [84]:

```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

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

Out[84]:

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

Boosters Carried Maximum Payload

SELECT DISTINCT BOOSTER_VERSION

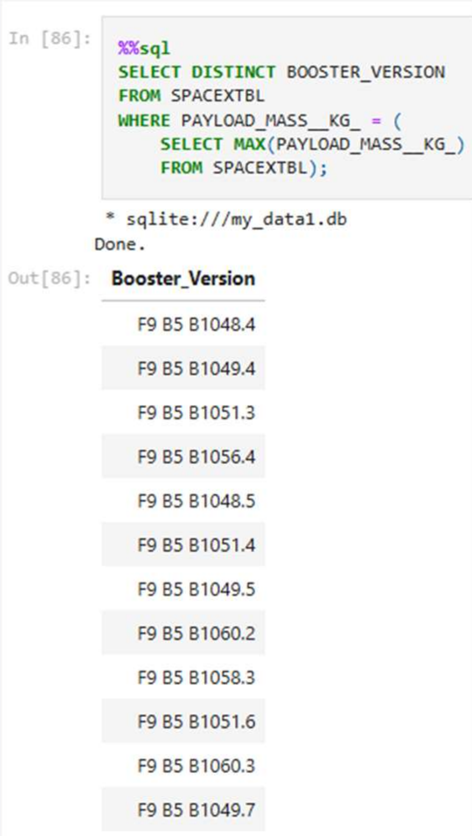
FROM SPACEXTBL

WHERE PAYLOAD_MASS__KG = (

SELECT MAX(PAYLOAD_MASS_KG_)

FROM SPACEXTBL)

In above query we are selecting the names of the booster_versions which have carried the maximum payload mass.



In [86]:

```
%%sql
SELECT DISTINCT BOOSTER_VERSION
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL);
```

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

Out[86]:

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

SELECT LANDING_OUTCOME , LAUNCH_SITE, BOOSTER_VERSION

FROM SPACEXTBL

WHERE LANDING_OUTCOME = 'Failure (drone ship)' AND DATE LIKE ‘2015%’

We used a combinations of the **WHERE** clause, **LIKE**, **AND** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

In [118…

```
%%sql
SELECT LANDING_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTBL
WHERE Landing_Outcome = 'Failure (drone ship)'
    AND DATE LIKE '2015%';
```

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

Out[118…

Landing_Outcome	Booster_Version	Launch_Site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME)

FROM SPACEXTBL

WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20’

GROUP BY LANDING__OUTCOME

ORDER BY TOTAL_NUMBER DESC

In above query we are selecting 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 [108…

```
%%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.
```

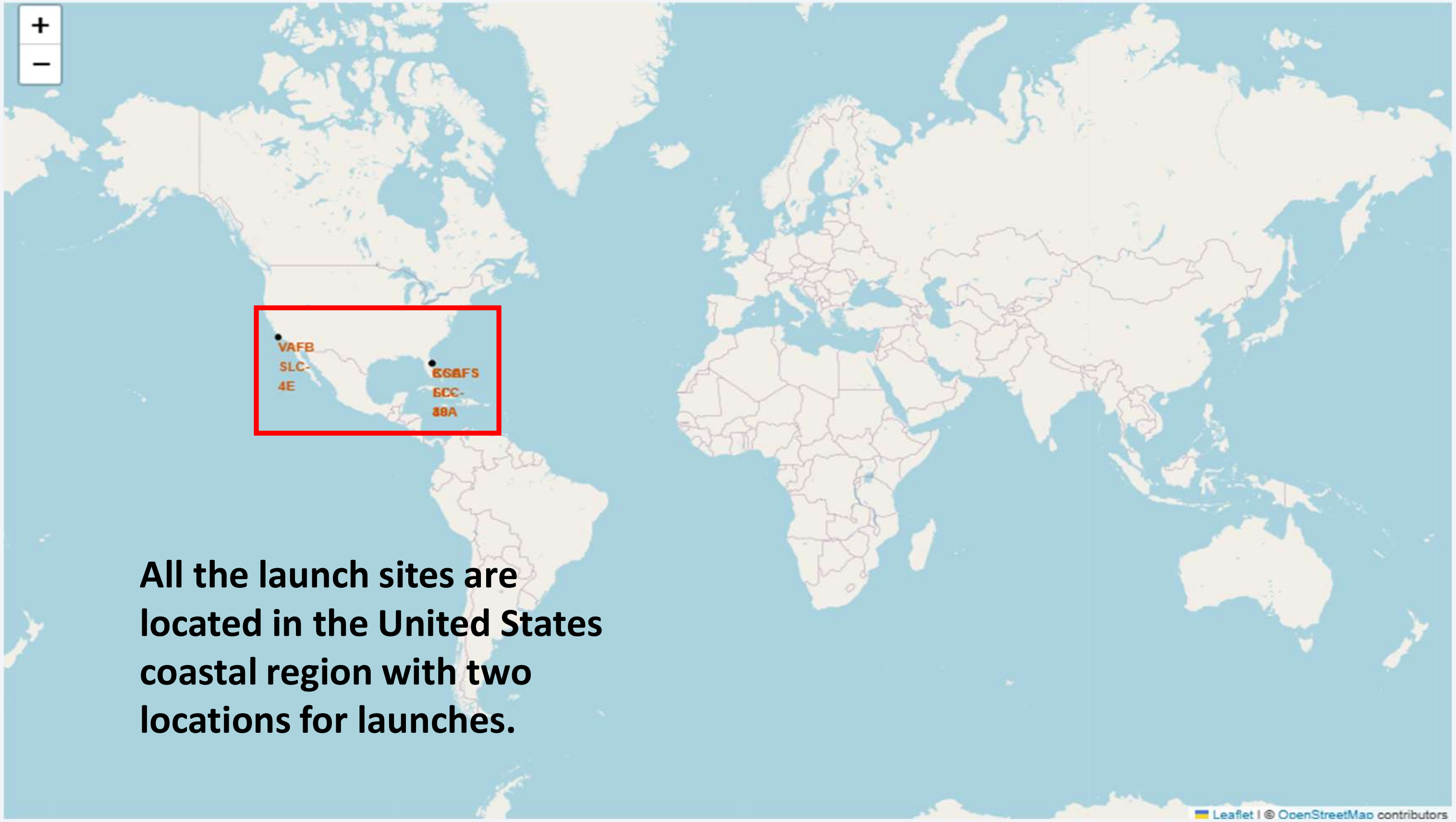
Out[108…

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

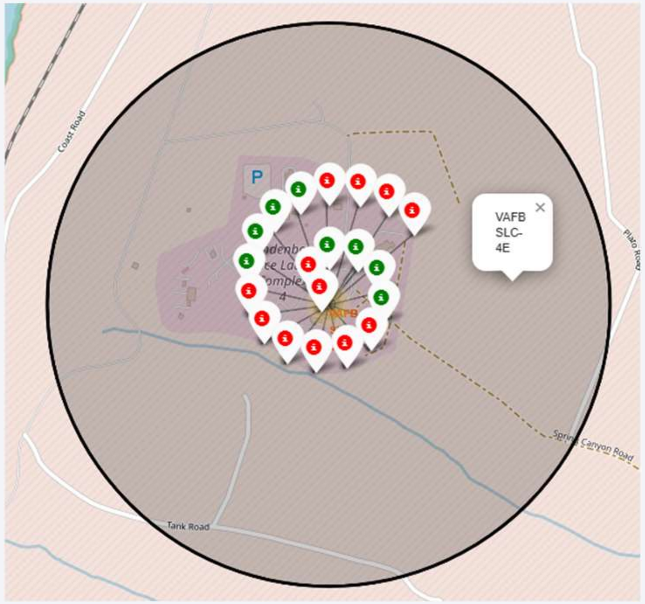
Launch Sites Proximities Analysis

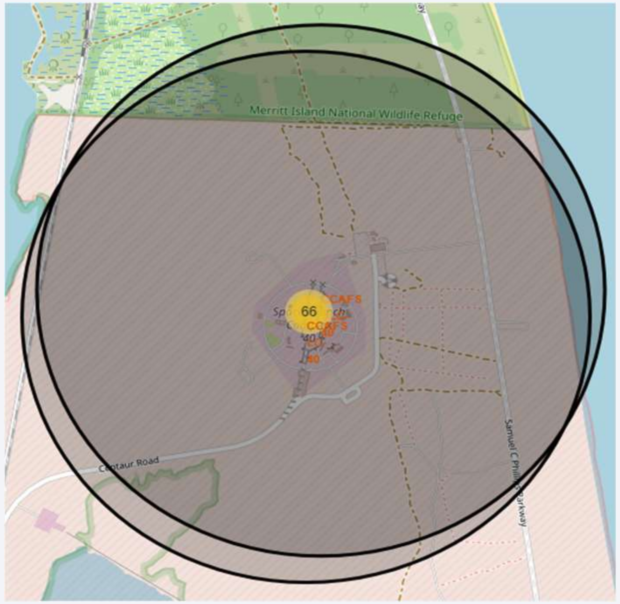
Launch Sites Map



All the launch sites are located in the United States coastal region with two locations for launches.

Success/Failed Launch for each site on map





California Launch Site
VAFB SLC-4E

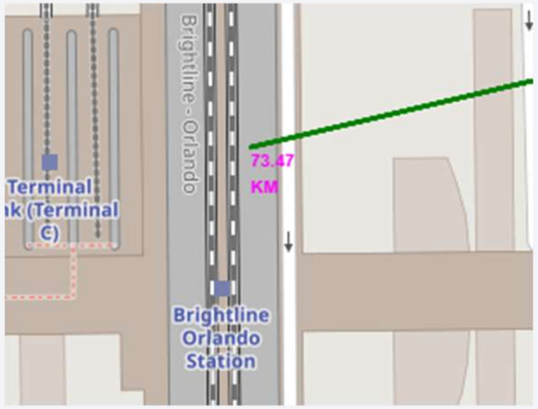
Florida Launch Sites:

KSC LC-39A

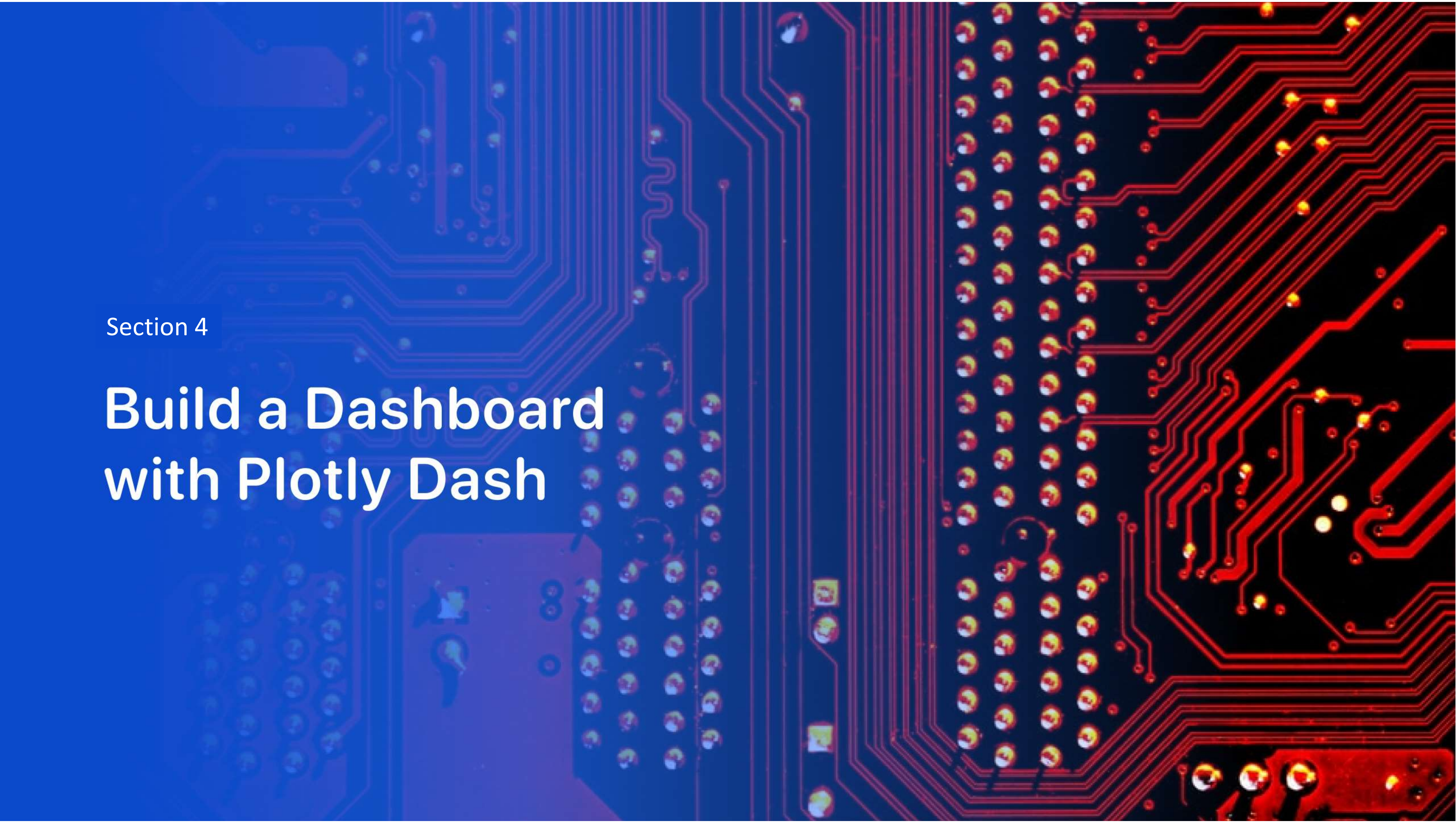
CCAFS LC-40
CCAFS SLC-40

Markers for all launch records. If a launch was successful (class=1), then we use a green marker and if a launch was failed, we use a red marker (class=0)

Distance between Launch Site and its Proximities



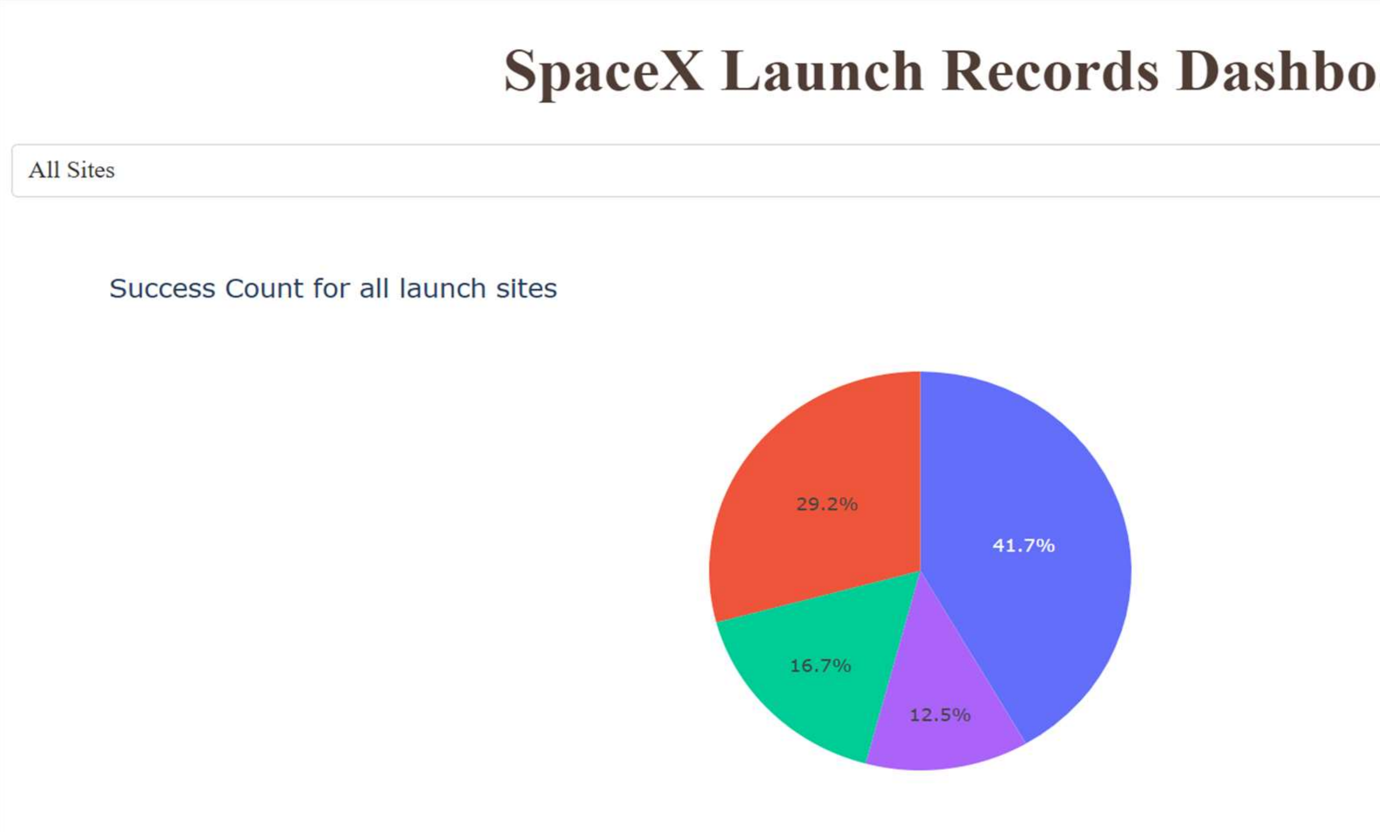
- Launch site is in close proximity to coastline. (0.89 KM)
- Launch sites are far away from city. (23.19KM)
- There is no launch site in close proximity to highways. (7.76 KM)
- There is no launch site in close proximity to railways. (73.47 KM)



Section 4

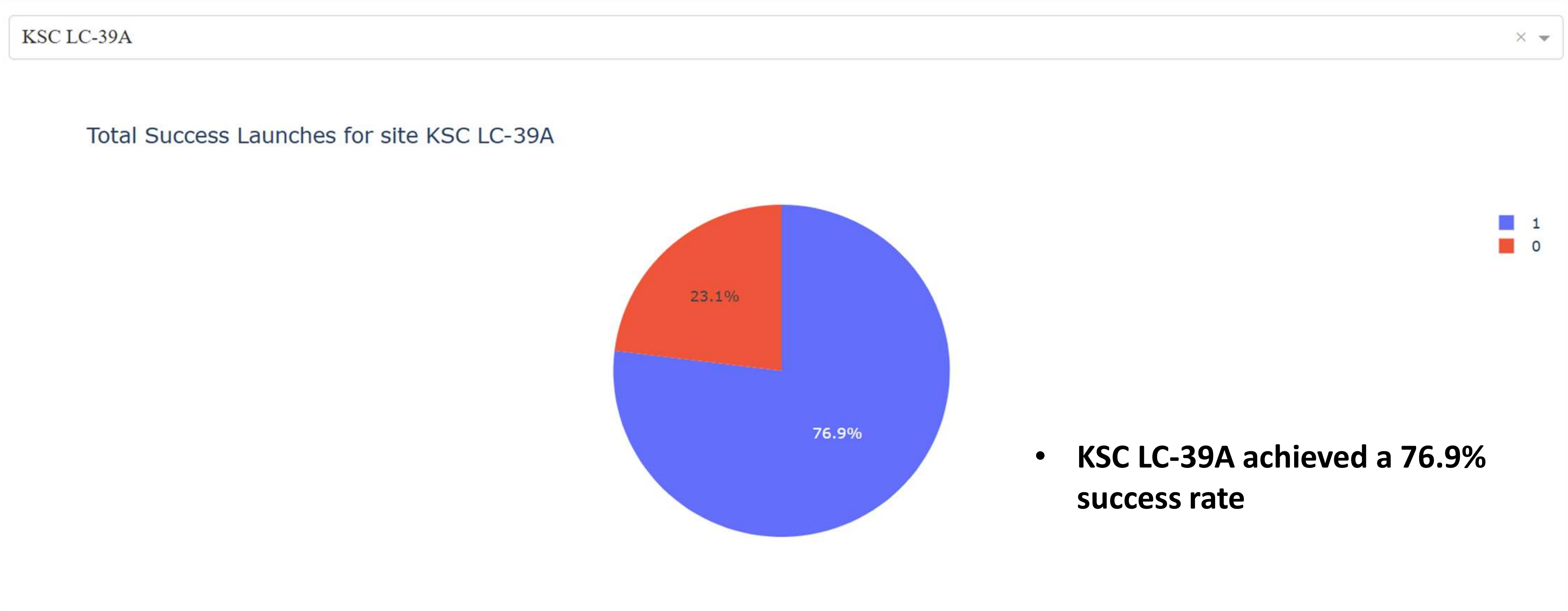
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



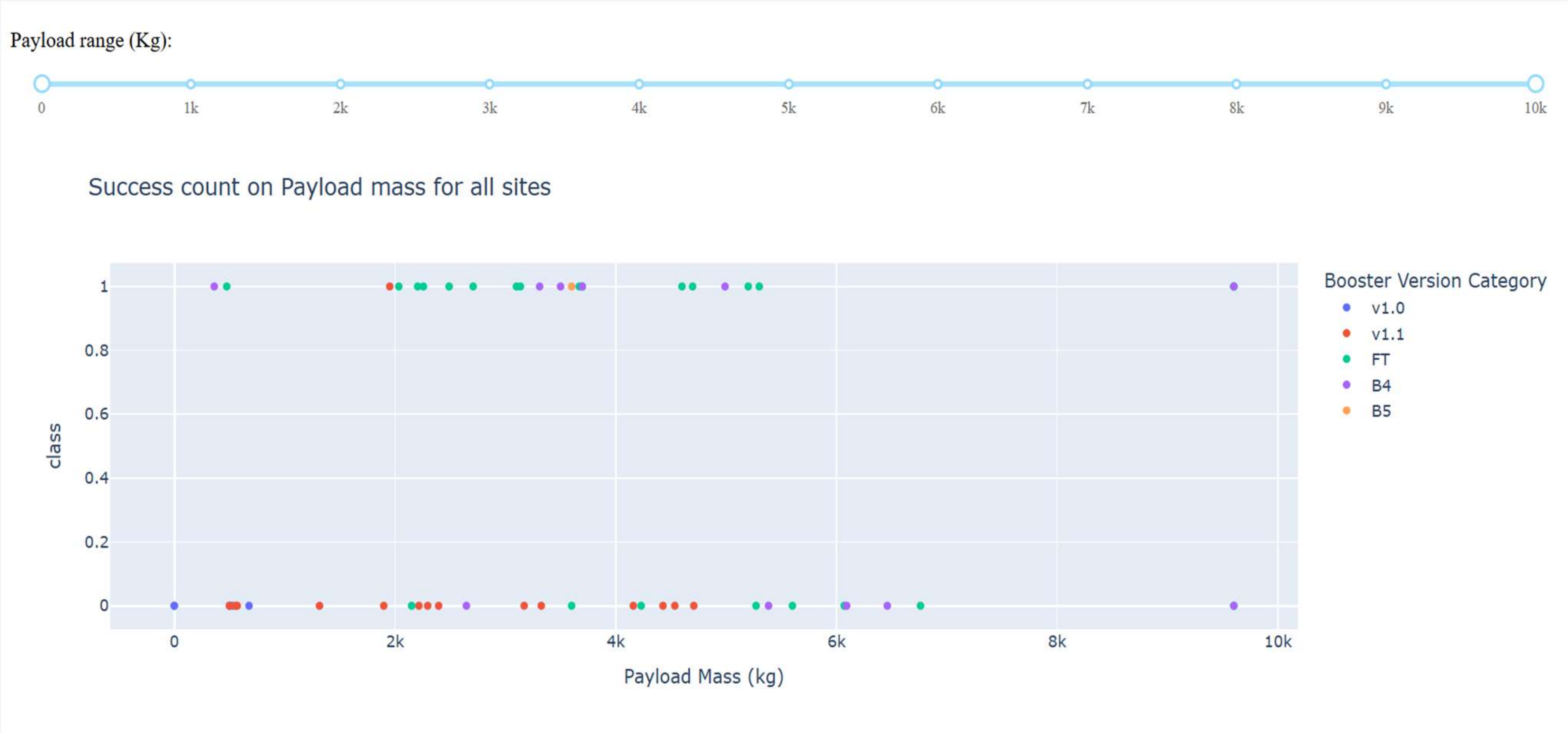
- **KCS LC-39A had the most successful launches from all the sites**

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



- **KSC LC-39A achieved a 76.9% success rate**

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

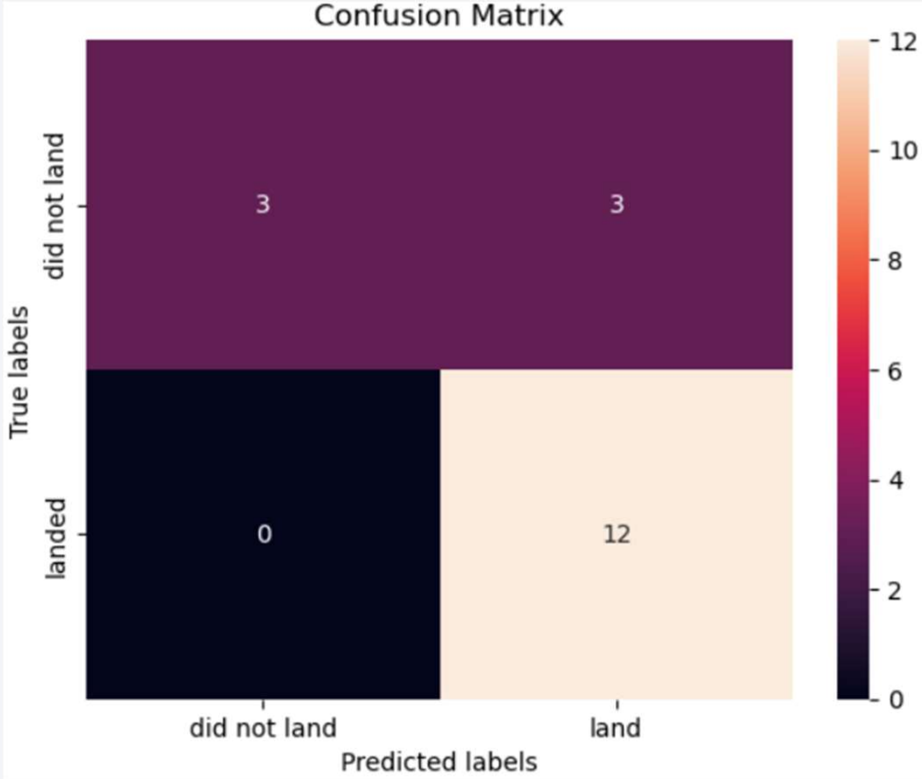
```
[177]: models = {
           'KNeighbors': knn_cv.best_score_,
           'DecisionTree': tree_cv.best_score_,
           'LogisticRegression': logreg_cv.best_score_,
           'SupportVector': svm_cv.best_score_
       }
       bestalgorithm = max(models, key=models.get)
       print(f'Best model 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)
```

	F1-score	Accuracy
Algorithm		
Logistic Regression	0.888889	0.833333
SVM	0.888889	0.833333
Decision Tree	0.916667	0.888889
K Nearest Neighbor	0.888889	0.833333

```
Best model is DecisionTree with a score of 0.8910714285714286
Best params is {'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, '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.
- We conclude that the Decision Tree was the best classifier which was able to predict the difference between two classes and also we have problem with false positive.



Conclusions

- Decision Tree Algorithm is the best classifier for this classification problem.
- Higher payload mass seems to have high success rate.
- **ES-L1, GEO, HEO, SSO** Orbit has the highest success rate.
- KSC LC-29A launch site has highest success rate.
- Launch success rate started to increase in 2013 till 2020.

Appendix

- Google Map to find nearest co-ordinate
- Python for Data Analysis (Jupyter Notebook)
- Plotly Docs
- Folium Docs

Entire source code and analysis notebook can be found on the given link :

Data Science Capstone Project GitHub Link

Thank you!