IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Pankaj Sonawane
07-Jul-2024

# Outline

I. Executive Summary

II. Introduction

III. Methodology

IV. Insights drawn from EDA

V. Launch Sites – Proximity Analysis

VI. Building a Dashboard with Plotly Dash

VII. Predictive Analysis (Classification)

# 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 in Screenshots
  - Predictive Analytics

This project leverages data science techniques to address a critical aspect of space exploration economics.

By predicting the success of Falcon 9 rocket landings, we aim to provide actionable insights that can drive competitive strategies and enhance the cost-efficiency of space launches.

# Introduction

## Project background and context

- SpaceX advertises Falcon 9 rocket launches for $62 million each on its website.
- Other providers charge around $165 million for each launch. SpaceX saves money because it can reuse the first stage of the rocket.
- If we know whether the first stage will land successfully, we can figure out the cost of a launch. This information can help another company compete with SpaceX for a rocket launch contract.
- The goal of this project is to create a machine learning system to predict if the first stage will land successfully.

## Problems you want to find answers

- Identifying all factors that influence the landing outcome.
- The relationship between each variables and how it is affecting the outcome.
- The best condition needed to increase the probability of successful landing.

Section 1

# Methodology

# Methodology

## Methodology

- 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 data was collected using various methods

- 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

Get request for rocket launch data using API

Use json_normalize method to convert json result to dataframe

Performed data cleaning and filling the missing value

GitHub URL of the completed SpaceX API calls

[https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/jupyter-labs-spacex-data-collection-api.ipynb](https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/jupyter-labs-spacex-data-collection-api.ipynb)

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

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

Check the content of the response

In [8]:
```
print(response.content)
```

In [17]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
```

We should see that the request was successfull with the 200 status response code

In [18]:
```
response.status_code
```

Out[18]: 200

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

In [21]:
```
# Use json_normalize meethod to convert the json result into a dataframe
static_json_df = response.json()
data = pd.json_normalize(static_json_df)
```

In [24]:
```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, ar
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket bo
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leavir
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

# Data Collection - Scraping

Request the Falcon9 Launch Wiki page from url

Create a BeautifulSoup from the HTML response

Extract all column/variable names from the HTML header

GitHub URL for the process

[https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/jupyter-labs-webscraping.ipynb](https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/jupyter-labs-webscraping.ipynb)

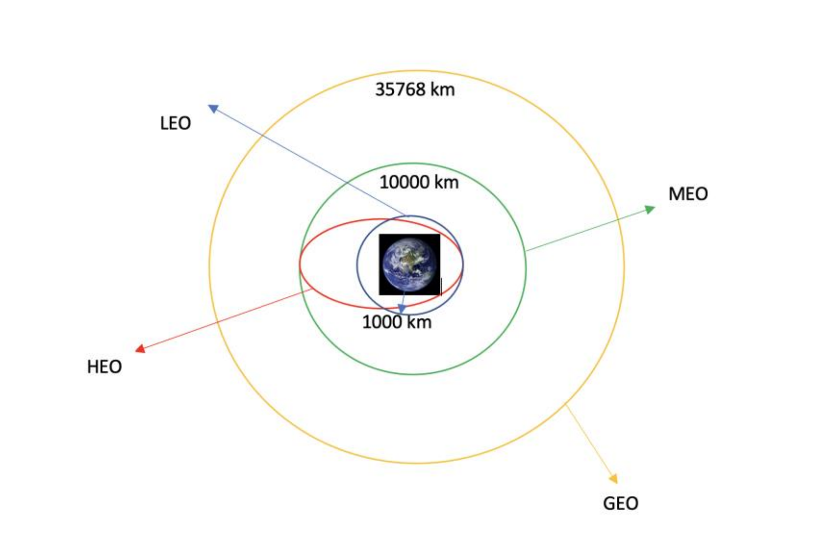
```
In [6]:
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code

Out[6]: 200
```

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

```
In [7]:
# Use BeautifulSoup() to create a BeautifulSoup object from a
soup = BeautifulSoup(html_data.text, 'html.parser')
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
```

# 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.

| | FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome | Flights | GridFins | Reused | Legs | LandingPad |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2010-06-04 | Falcon 9 | 6104.959412 | LEO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN |
| 1 | 2 | 2012-05-22 | Falcon 9 | 525.000000 | LEO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN |
| 2 | 3 | 2013-03-01 | Falcon 9 | 677.000000 | ISS | CCAFS SLC 40 | None None | 1 | False | False | False | NaN |
| 3 | 4 | 2013-09-29 | Falcon 9 | 500.000000 | PO | VAFB SLC 4E | False Ocean | 1 | False | False | False | NaN |
| 4 | 5 | 2013-12-03 | Falcon 9 | 3170.000000 | GTO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN |

GitHub URL for Data Wrangling

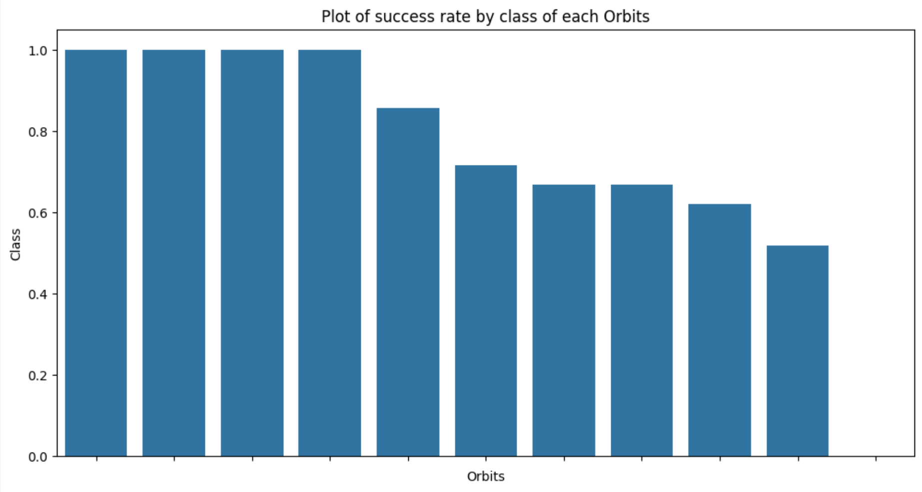
https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

# EDA with Data Visualization

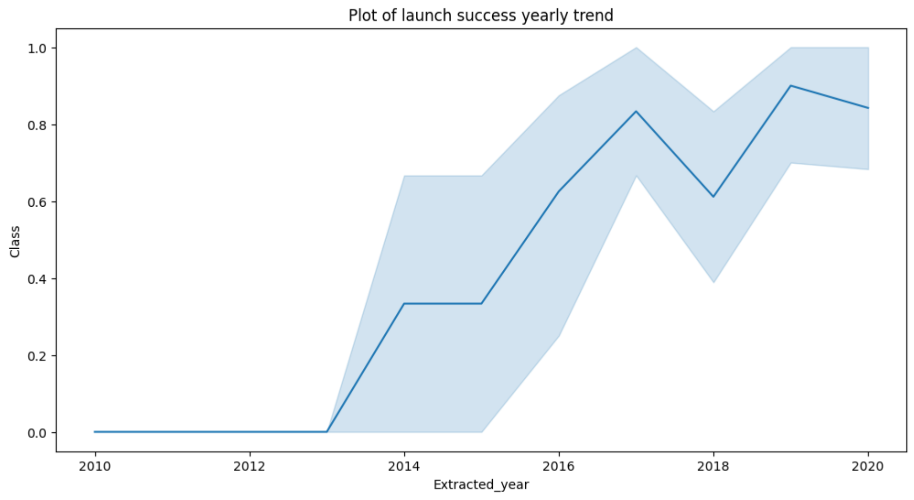
- Once we get a hint of the relationships using scatter plot. We will then use further visualization tools such as bar graph and line plots graph for further analysis.
- Bar graphs is one of the easiest way to interpret the relationship between the attributes. In this case, we will use the bar graph to determine which orbits have the highest probability of success.
- We then use the line graph to show a trends or pattern of the attribute over time which in this case, is used for see the launch success yearly trend.
- We then use Feature Engineering to be used in success prediction in the future module by created the dummy variables to categorical columns.

GitHub URL for EDA with Data Visualization

https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/edadataviz.ipynb




ES-11. GE-0 HEO, SSO have high success rates




you can observe that the sucess rate since 2013 kept increasing till 2020

# EDA with SQL

**Using SQL, we had performed many queries to get better understanding of the dataset, Ex:**

- Displaying the names of the launch sites.
- Displaying 5 records where launch sites begin with the string 'CCA'.
- Displaying the total payload mass carried by booster launched by NASA (CRS).
- Displaying the average payload mass carried by booster version F9 v1.1.
- Listing the date when the first successful landing outcome in ground pad was achieved.
- Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
- Listing the total number of successful and failure mission outcomes.
- Listing the names of the booster_versions which have carried the maximum payload mass.
- Listing the failed landing_outcomes in drone ship, their booster versions, and launch sites names for in year 2015.
- Rank the count of landing outcomes or success between the date 2010-06-04 and 2017-03-20, in descending order.

**We loaded the SpaceX dataset into SQL database without leaving the Jupyter notebook**

GitHub URL for EDA with SQL

https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

# Build an Interactive Map with Folium

- To visualize the launch data into an interactive map. We took the latitude and longitude coordinates at each launch site and added a circle marker around each launch site with a label of the name of the launch site.
- We then assigned the dataframe launch_outcomes(failure,success) to classes 0 and 1 with Red and Green markers on the map in MarkerCluster().
- We then used the Haversine's formula to calculated the distance of the launch sites to various landmark to find answer to the questions of:
  - *How close the launch sites with railways, highways and coastlines?*
  - *How close the launch sites with nearby cities?*

GitHub URL for Map with Folium

https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/lab_jupyter_launch_site_location.ipynb

# Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash which allowing the user to play around with the data as they need.
- We plotted pie charts showing the total launches by a certain sites.
- We then plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

GitHub URL for Dashboard with Plotly Dash
https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/spacex_dash_app.py

# Predictive Analysis (Classification)

| Building the Model | Evaluating the Model | Improving the Model | Find the Best Model |
|---|---|---|---|
| - Load the dataset into NumPy and Pandas<br>- Transform the data and then split into training and test datasets<br>- Decide which type of ML to use<br>- Set the parameters and algorithms to GridSearchCV and fit it to dataset. | - Check the accuracy for each model<br>- Get tuned hyperparameters for each type of algorithms.<br>- Plot the confusion matrix. | - Use Feature Engineering and Algorithm Tuning | - The model with the best accuracy score will be the best performing model. |

GitHub URL for Predictive Analysis

https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

# Results

The results will be categorized to 3 main results which is:

01 Exploratory data analysis results

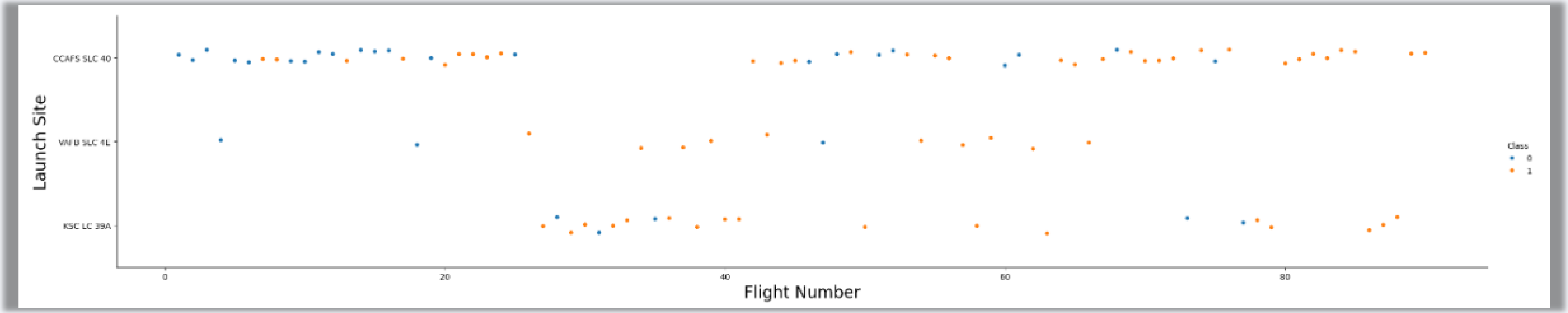
02 Interactive analytics demo in screenshots

03 Predictive analysis results

Section 2

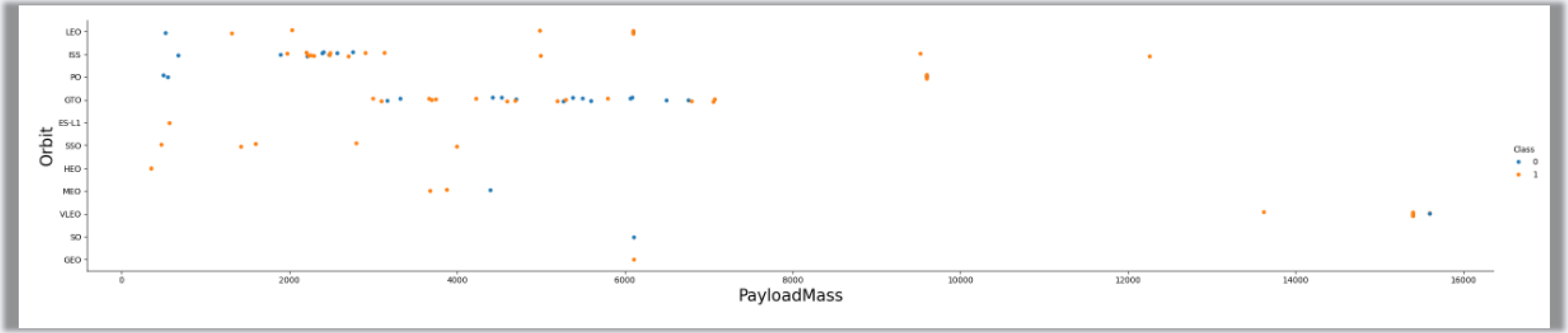
# Insights drawn from EDA

# Flight Number vs. Launch Site



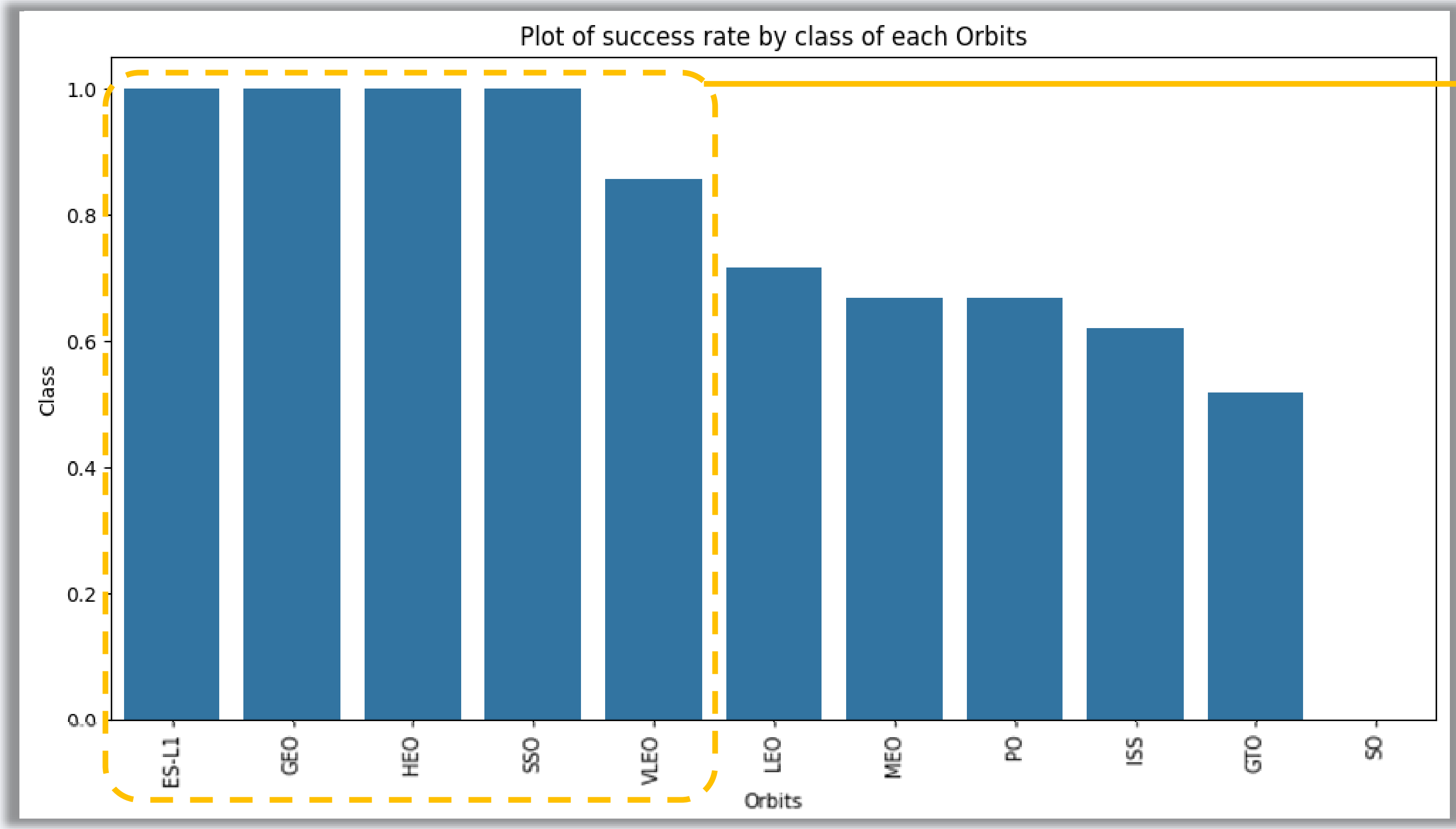

- This scatter plot shows that the larger the flights amount of the launch site, the greater the the success rate will be.
- However, site CCAFS SLC40 shows the least pattern of this.

# Payload vs. Launch Site



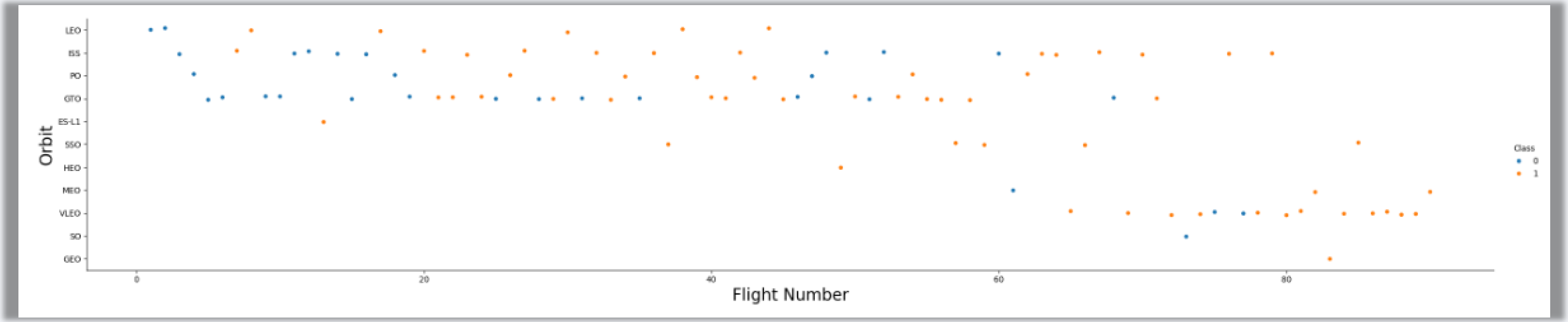

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- This scatter plot shows once the pay load mass is greater than 7000kg, the probability of the success rate will be highly increased.
- However, there is no clear pattern to say the launch site is dependent to the pay load mass for the success rate.

# Success Rate vs. Orbit Type



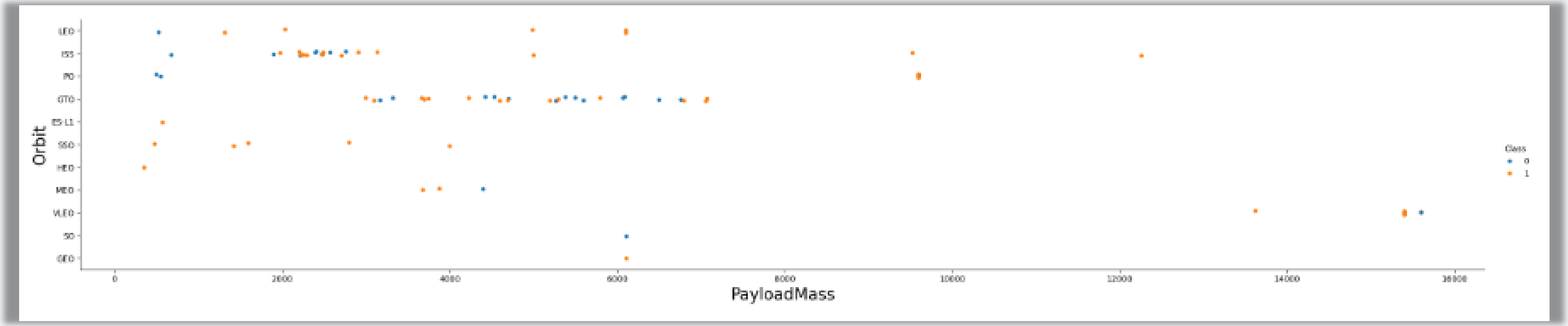

From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.

# Flight Number vs. Orbit Type



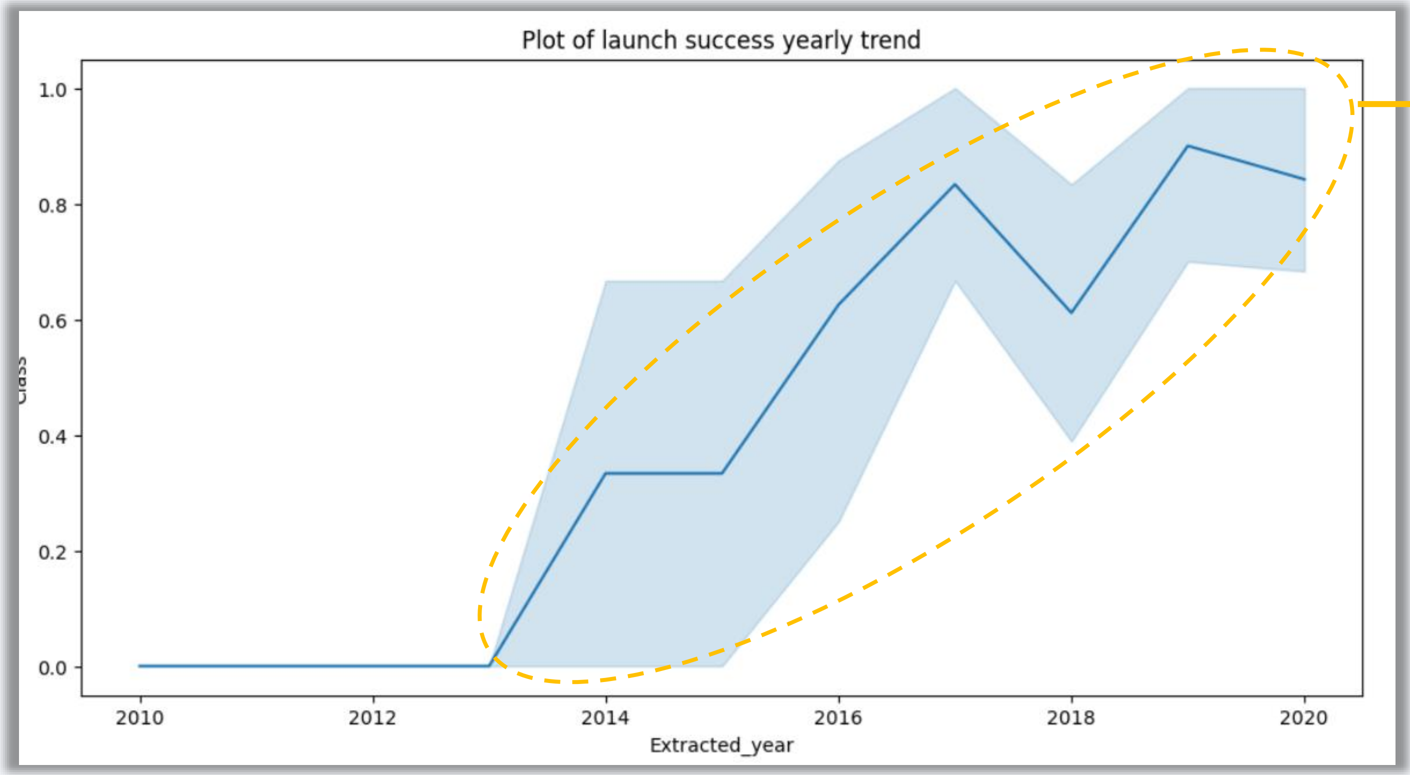

The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.

# 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

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

In [15]:
```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

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

Out[15]:

| 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'

In [16]:
```
%sql SELECT * FROM SpaceXtable WHERE Launch_Site LIKE 'CCA%'LIMIT 5
```

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

Out[16]:

| 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 | 7:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 0:35:00 | 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 |

We used the query above to display **5 records** where launch sites begin with `CCA`

# Total Payload Mass

```
In [21]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_PayloadMass FROM SpaceXtable WHERE Customer LIKE 'NASA (CRS)'

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

Out[21]:

| Total_PayloadMass |
|---|
| 45596 |

We calculated the total payload carried by boosters from NASA as **45,596** using the query below

# Average Payload Mass by F9 v1.1

```
In [23]: %sql SELECT AVG(Payload_Mass__KG_) AS Avg_PayloadMass FROM SpaceXtable WHERE Booster_Version = 'F9 v1.1'

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

Out[23]:

| Avg_PayloadMass |
|---|
| 2928.4 |

We calculated the average payload mass carried by booster version F9 v1.1 as **29,28.4**

# First Successful Ground Landing Date

```
In [24]: %sql SELECT MIN(Date) AS FirstSuccessfull_landing_date FROM SpaceXtable WHERE Landing_Outcome LIKE 'Success (ground pad)'

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

Out[24]:

| FirstSuccessfull_landing_date |
|---|
| 2015-12-22 |

We observed that the dates of the first successful landing outcome on ground pad was

**22nd December 2015**

# Successful Drone Ship Landing with Payload between 4000 and 6000

```
[25]: %sql SELECT Booster_Version FROM SpaceXtable WHERE Landing_Outcome = 'Success (drone ship)' AND Payload_Mass__KG_ > 4000  AND Payload_Mass__KG_ < 6000

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

[25]:

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

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

# Total Number of Successful and Failure Mission Outcomes

```
[33]: %sql SELECT COUNT(Mission_Outcome) AS SuccessOutcome FROM SpaceXtable WHERE Mission_Outcome LIKE 'Success%'

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

[33]:

| SuccessOutcome |
|---|
| 100 |

```
[34]: %sql SELECT COUNT(Mission_Outcome) AS FailureOutcome FROM SpaceXtable WHERE Mission_Outcome LIKE 'Failure%'

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

[34]:

| FailureOutcome |
|---|
| 1 |

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

# Boosters Carried Maximum Payload

```
[35]: %sql SELECT Booster_Version, Payload_Mass__KG_ FROM SpaceXtable WHERE Payload_Mass__KG_ = (SELECT MAX(Payload_Mass__KG_) FROM SpaceXtable) ORDER BY Booster_Version
```

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

[35]:

| Booster_Version | PAYLOAD_MASS__KG_ |
|---|---|
| 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 |

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

# 2015 Launch Records

```
[42]: %sql SELECT substr(Date,6,2) as Month, substr(Date,0,5) as Year, Landing_Outcome FROM SpaceXtable WHERE Landing_Outcome LIKE 'Failure (drone ship)' AND Date BETWEEN '2015-01-01' AND '2015-12-31'
 * sqlite:///my_data1.db
Done.
```

[42]:

| Month | Year | Landing_Outcome |
|---|---|---|
| 01 | 2015 | Failure (drone ship) |
| 04 | 2015 | Failure (drone ship) |

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

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

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

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

[43]:

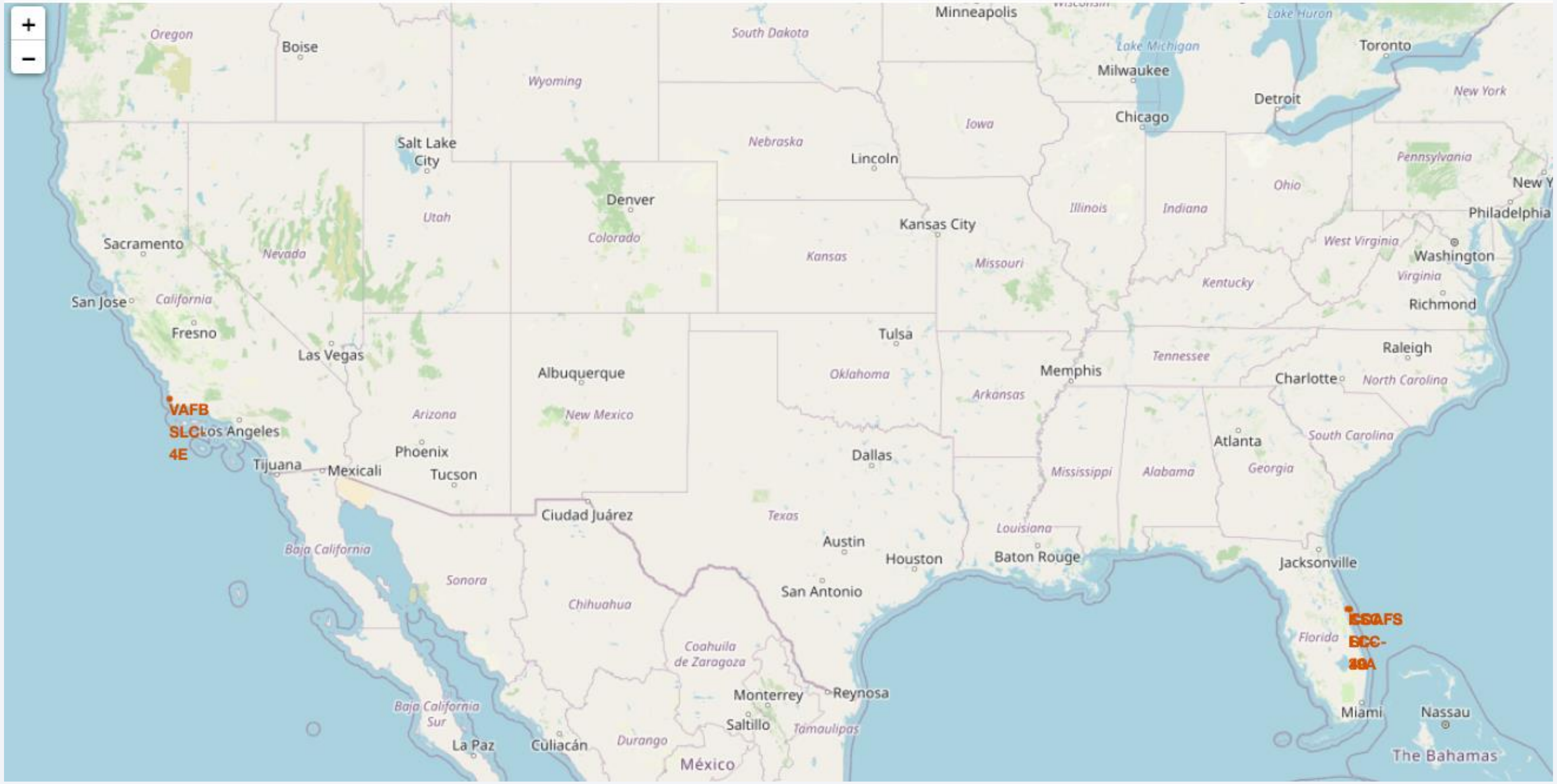
| Landing_Outcome | COUNT(Landing_Outcome) |
|---|---|
| No attempt | 10 |
| Success (drone ship) | 5 |
| Failure (drone ship) | 5 |
| Success (ground pad) | 3 |
| Controlled (ocean) | 3 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 2 |
| Precluded (drone ship) | 1 |

- 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.

Section 3

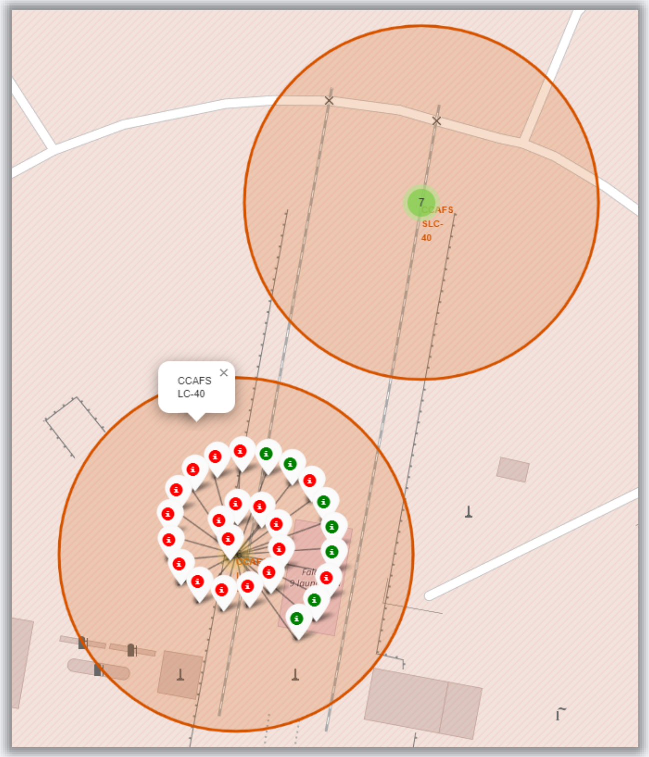
# Launch Sites Proximities Analysis

# All launch sites global map markers




We can see that all the SpaceX launch sites are located inside the United States

# Markers showing launch sites with color labels



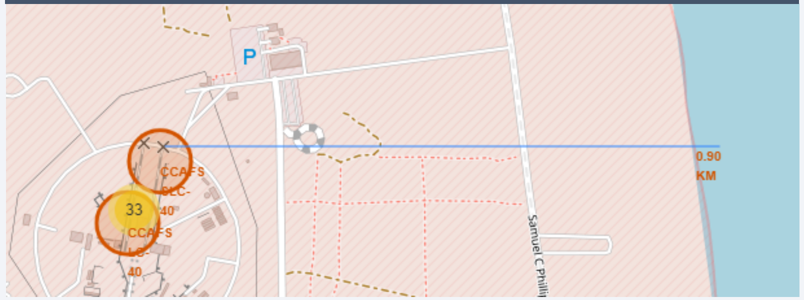

Florida Launch site


California Launch site

Green Markers show successful launches & Red Markers show Failures

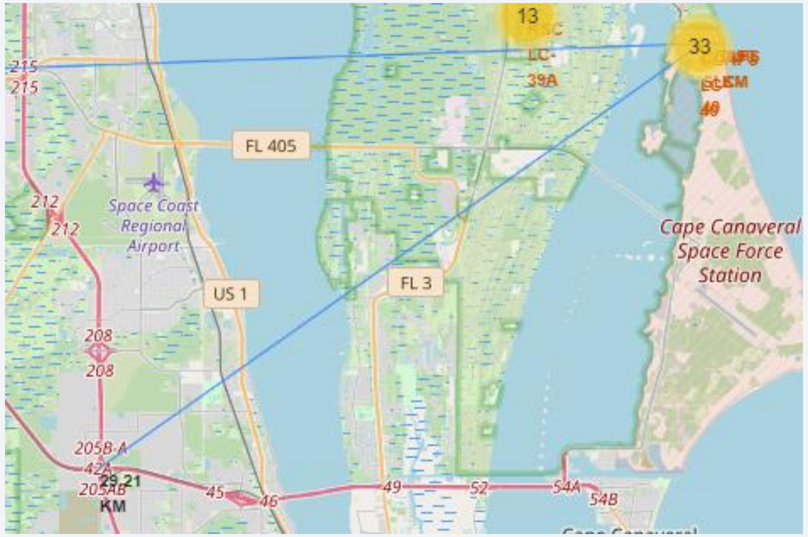
# Launch Sites Distance to Landmarks

## Distance to Coast




## Distance to Closest City


## Distance to Closest Highway




## Distance to Closest Railway Station

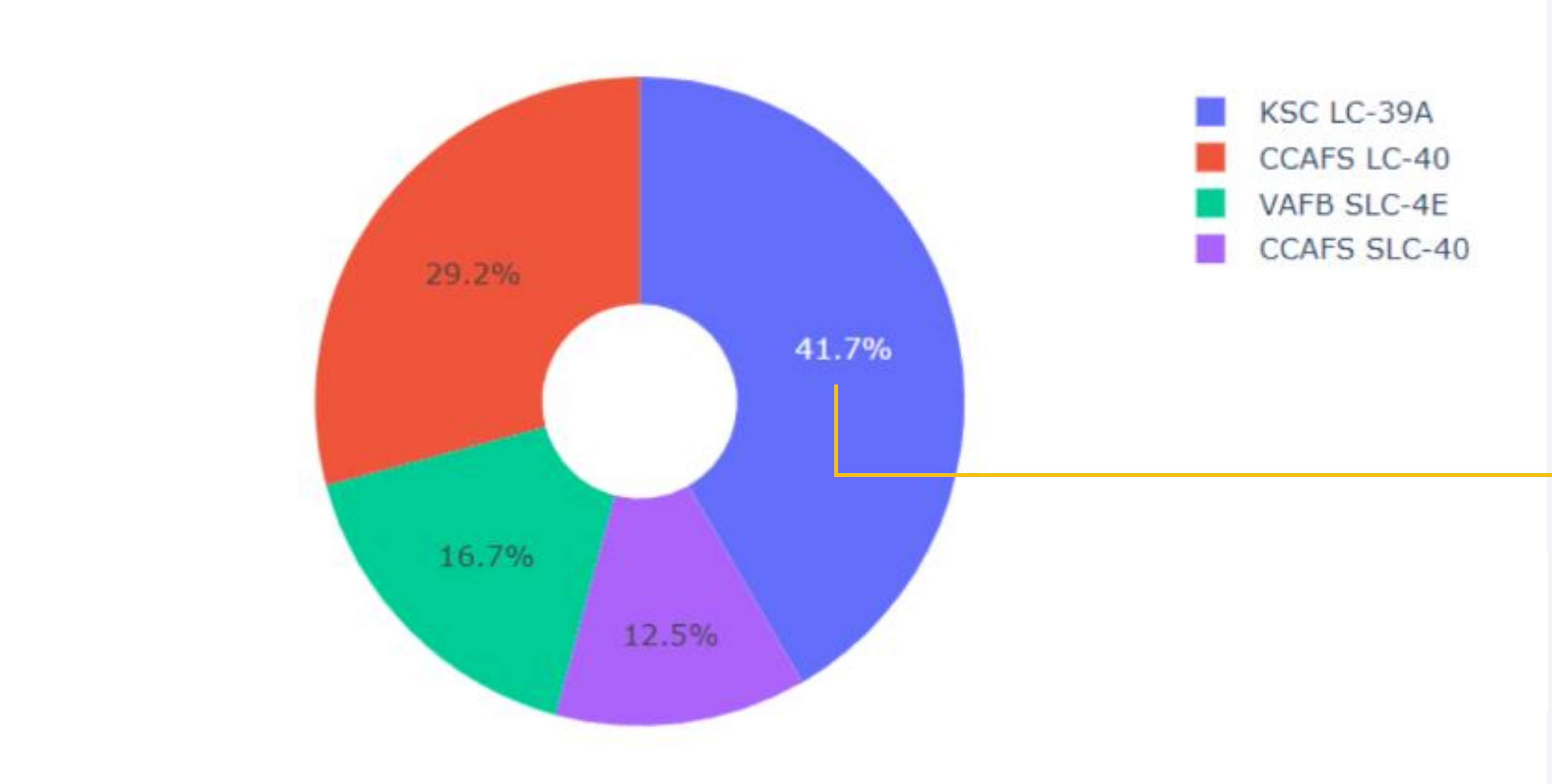

1. Are launch sites in close proximity to railways? **No**
2. Are launch sites in close proximity to highways? **No**
3. Are launch sites in close proximity to coastline? **Yes**
4. Do launch sites keep certain distance away from cities? **Yes**

Section 4

# Build a Dashboard with Plotly Dash

# The success percentage by each sites

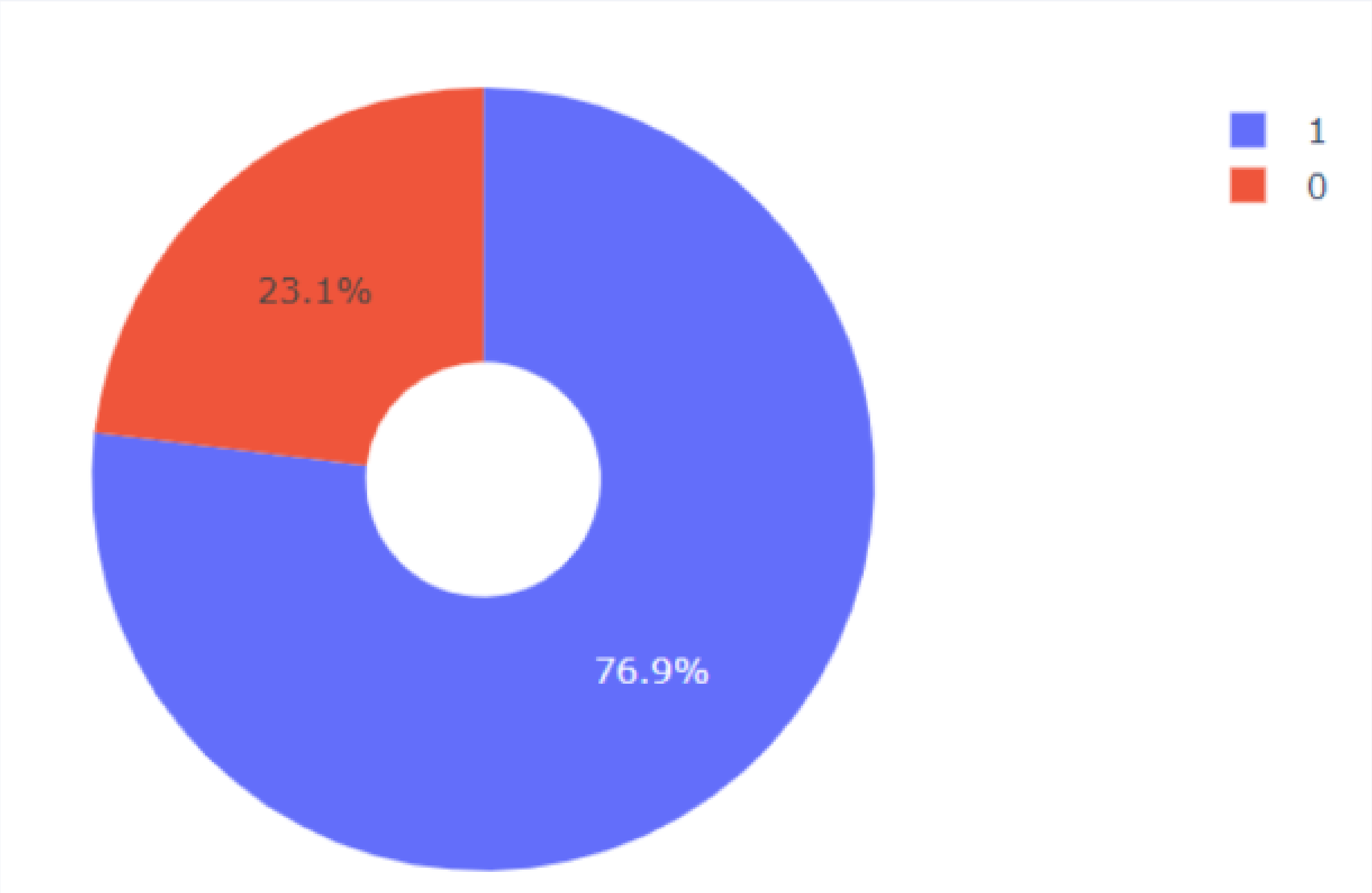
Total Success Launches by All sites




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

# The highest launch-success ratio: KSC LC-39A

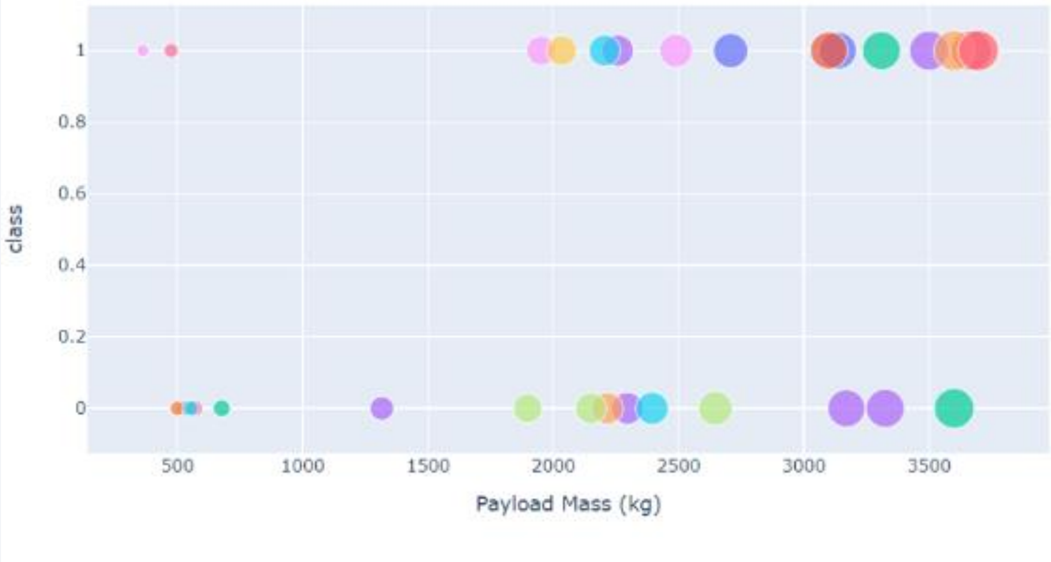
Highest launch-success ratio

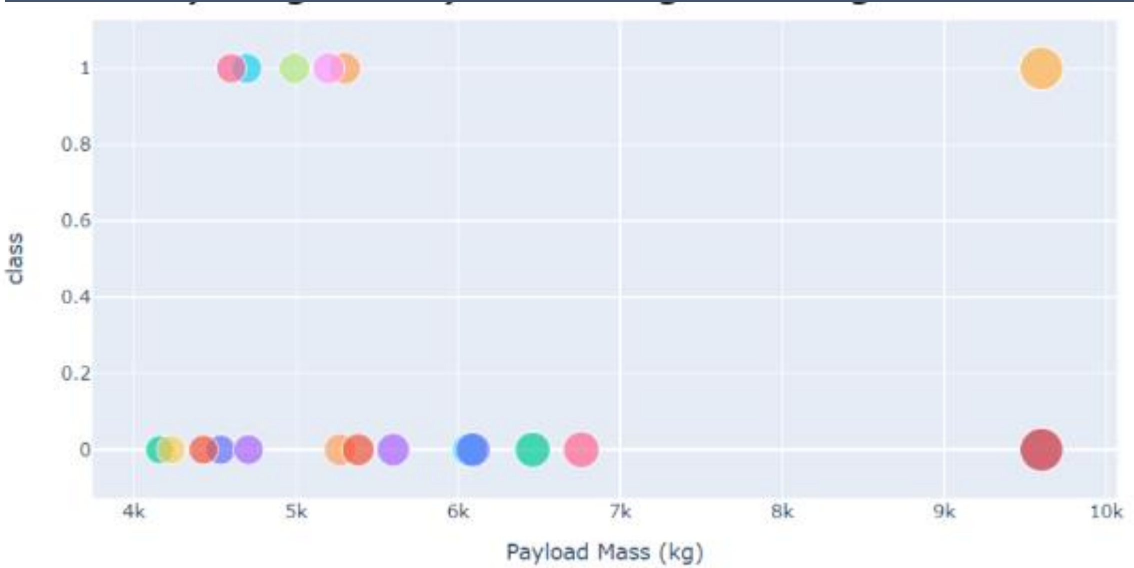



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

# Payload vs Launch Outcome Scatter Plot

| Low Weighted Payload 0kg – 4000kg | Heavy Weighted Payload 4000kg – 1000kg |
| --- | --- |




We can see that all the success rate for low weighted payload is higher than heavy weighted payload

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

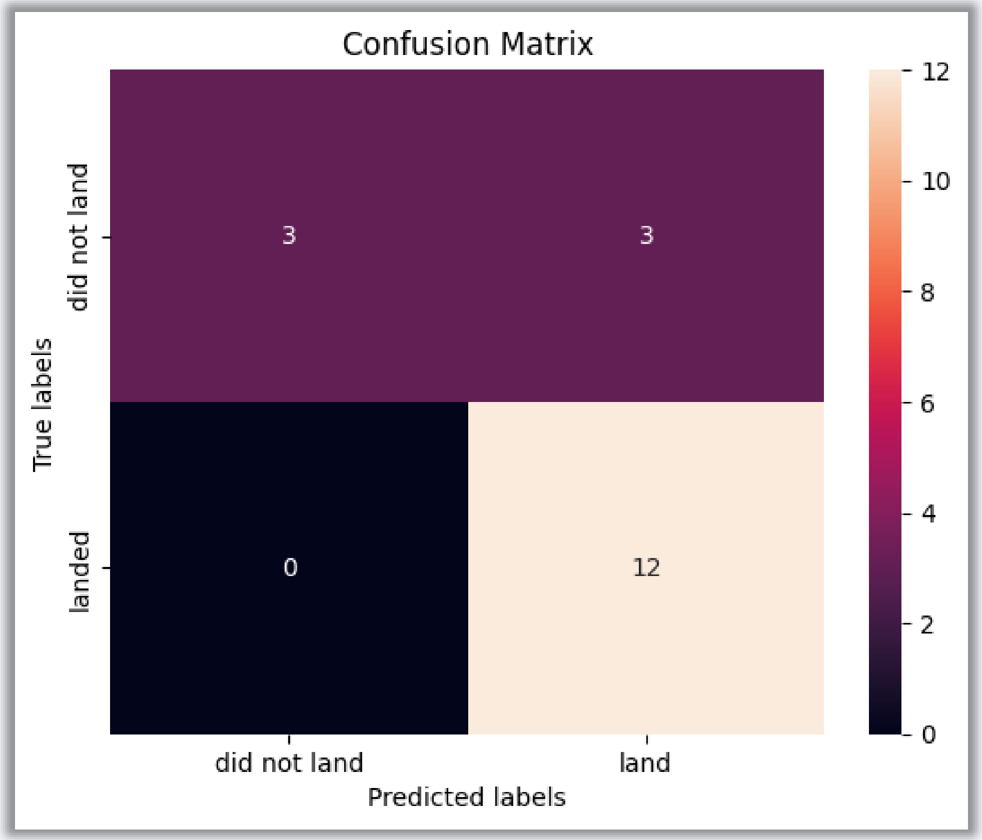
In [39]:

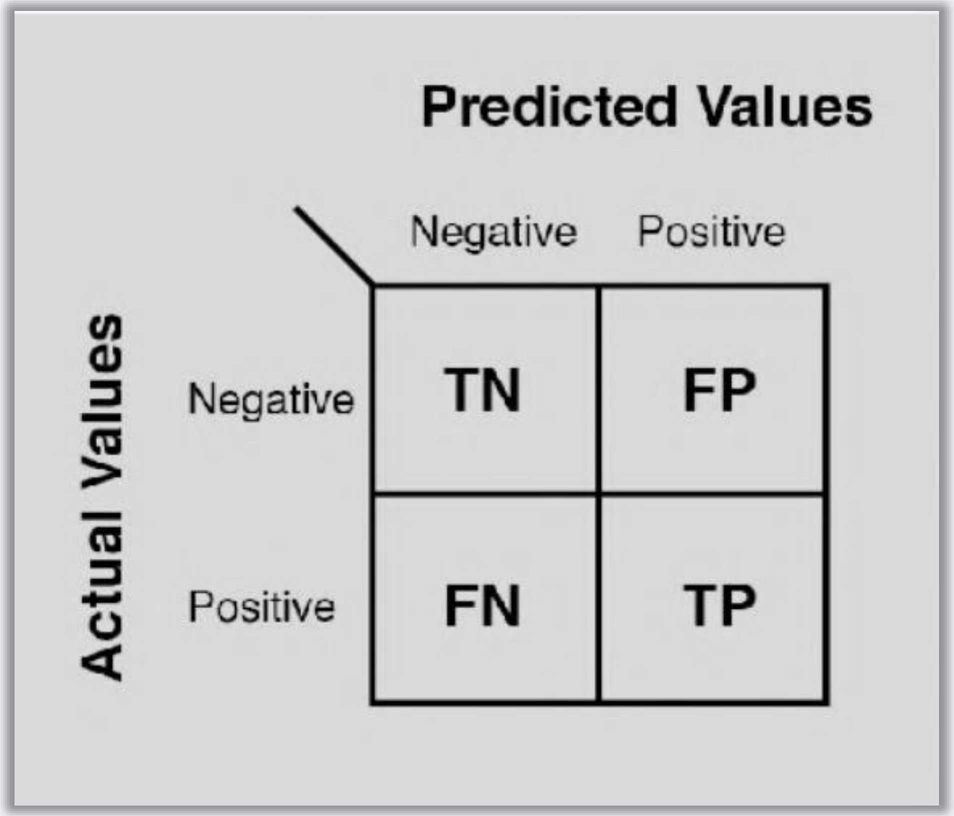
```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params 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 0.8625
Best Params is : {'criterion': 'gini', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split':
2, 'splitter': 'random'}
```

As we can see, by using the code as above: we could identify that the best algorithm to be the Tree Algorithm which have the highest classification accuracy.

# 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

Starting from the year 2013, the success rate for SpaceX launches is increased, directly proportional time in years to 2020, which it will eventually perfect the launches in the future.

KSC LC-39A have the most successful launches of any sites; 76.9%

Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate

SSO orbit have the most success rate; 100% and more than 1 occurrence.

The low weighted payloads (which define as 4000kg and below) performed better than the heavy weighted payloads.

The Tree Classifier Algorithm is the best Machine Learning approach for this dataset.

# Appendix

Code and Datasets to download

https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/dataset_part_1.csv

https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/dataset_part_2.csv

https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/dataset_part_3.csv

https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/my_data1.db

https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/spacex_dash_app.py

https://github.com/pankajsonawane2711/DS-CapstoneProject-SpaceX/blob/main/spacex_launch_dash.csv

# Thank you!